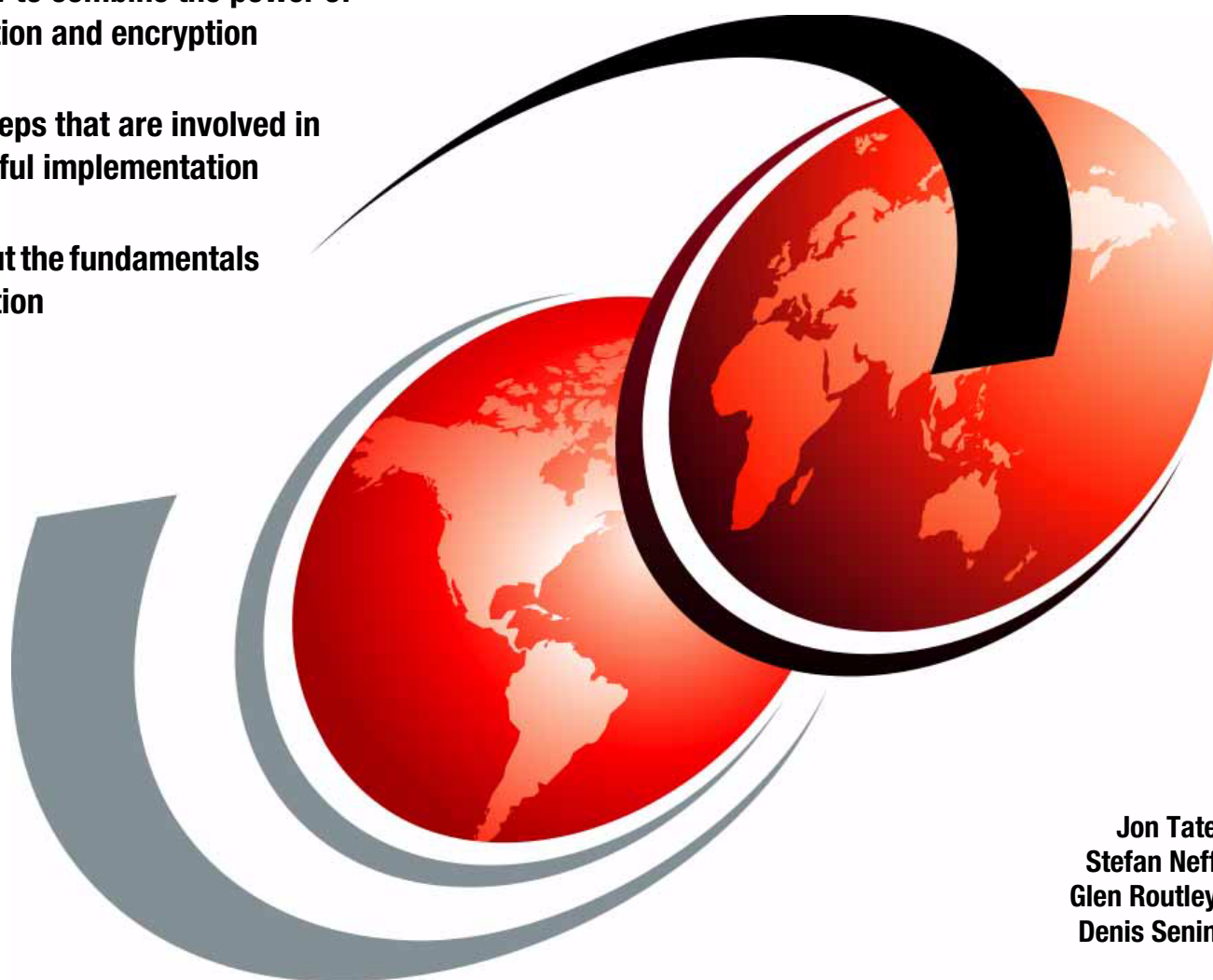


Implementing the Storwize V7000 and the IBM System Storage SAN32B-E4 Encryption Switch

Learn how to combine the power of virtualization and encryption

See the steps that are involved in a successful implementation

Read about the fundamentals of encryption



Jon Tate
Stefan Neff
Glen Routley
Denis Senin

Redbooks



International Technical Support Organization

**Implementing the Storwize V7000 and the IBM System
Storage SAN32B-E4 Encryption Switch**

February 2012

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

First Edition (February 2012)

This edition applies to IBM System Storage Storwize V7000 Version 6.1, IBM System Storage SAN32B-E4 Encryption Switch, Tivoli Key Lifecycle Manager V2.0, and Fabric Operating System 7.0.0a.

© Copyright International Business Machines Corporation 2012. 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

Notices	vii
Trademarks	viii
Preface	ix
The team who wrote this book	ix
Now you can become a published author, too!	xi
Comments welcome	xi
Stay connected to IBM Redbooks	xi
Chapter 1. Introduction to SAN encryption using the Storwize V7000 and the SAN32B-E4 Encryption Switch	1
1.1 Why we need SAN security and encryption solutions	2
1.1.1 Threats and security	2
1.2 Encryption basics	3
1.2.1 Symmetric key encryption	3
1.2.2 Asymmetric key encryption	4
1.2.3 Digital certificates	5
1.2.4 Encryption algorithms	5
1.2.5 Encryption challenges	6
1.3 IBM encryption products and software	7
1.3.1 IBM Tivoli Key Lifecycle Manager	7
1.3.2 SAN32B-E4 Encryption Switch	8
1.3.3 IBM Storwize V7000	9
1.4 SAN environment (pre-encryption)	10
1.5 Encrypted SAN environment	11
Chapter 2. Terminology and technology	13
2.1 Why terminology is important	14
2.2 Basic terminology	14
2.2.1 Data	14
2.2.2 LUN	14
2.2.3 Fabric and SAN	14
2.2.4 Management network	15
2.2.5 Private network	15
2.2.6 Key	15
2.2.7 Certificate	15
2.2.8 CryptoModule	15
2.2.9 Cleartext	16
2.2.10 Ciphertext	16
2.3 Elements of the encryption process	16
2.3.1 Encryption group	16
2.3.2 Group leader	16
2.3.3 Encryption engine	16
2.3.4 Encryption node	17
2.3.5 Encryption certificates	17
2.3.6 Key vault	17
2.3.7 Recovery cards	18
2.4 Terminology of the encryption process	18
2.4.1 Crypto Target Container	18

2.4.2	Data encryption key	19
2.4.3	Data encryption key lifecycle management.	20
2.4.4	First-time encryption	21
2.4.5	Rekeying operation	22
2.4.6	First-time encryption and rekey operation details	22
2.4.7	Frame redirection zone	24
2.4.8	Key encryption key	24
2.4.9	Master key	25
2.4.10	Virtual targets and virtual initiators	25
2.5	Cluster technology	27
2.5.1	High availability cluster configurations	27
2.5.2	Failover and fallback of the HA cluster	29
2.5.3	Data encryption key cluster.	32
Chapter 3. Initial setup for the IBM Tivoli Key Lifecycle Manager and the SAN32B-E4 Encryption Switch		
3.1	The need for a key management tool	36
3.1.1	Why IBM Tivoli Key Lifecycle Manager.	37
3.2	Tivoli Key Lifecycle Manager components and resources	39
3.3	Initial setup of Tivoli Key Lifecycle Manager and the Encryption Switch.	41
3.3.1	Basic installation of Tivoli Key Lifecycle Manager.	41
3.3.2	Multiple Tivoli Key Lifecycle Managers for redundancy	42
3.3.3	Installation of the Encryption Switch	46
3.3.4	Master key management	47
3.3.5	Considerations before the first TKLM and SAN32B-E4 Encryption Switch setup.	47
3.3.6	Setting up the SAN32B-E4 Encryption Switch and TKLM using the switch CLI .	50
3.3.7	Setting up the SAN32B-E4 Encryption Switch and TKLM using the DCFM GUI	71
Chapter 4. Implementation scenarios and recommendations for managing the SAN32B-E4 Encryption Switch		
4.1	Configuring encryption on a single fabric	90
4.1.1	Current configuration	90
4.2	General prerequisites for encryption	95
4.2.1	Setting the default zone	95
4.2.2	Synchronizing switch times.	96
4.2.3	Host requirements.	96
4.3	Defining the encryption configuration	97
4.3.1	Creating containers.	98
4.3.2	Adding a host initiator to a container.	98
4.3.3	Adding a target LUN to a container.	100
4.3.4	Displaying a container.	104
4.3.5	Enabling encryption	105
4.4	Data encryption key cluster config: Multiple path/dual fabric	110
4.4.1	Starting the configuration	110
4.4.2	Target configuration for a data encryption key cluster environment	111
4.4.3	Steps to extend a single fabric setup to a DEK cluster environment	112
4.4.4	Preparing the multipath/dual-fabric configuration	112
4.4.5	Verifying the current encryption group members	113
4.4.6	Adding a new host server Fibre Channel port to the existing LUNs	114
4.4.7	Adding and defining the new CTCs for the second fabric via DCFM	116
4.4.8	Adding the existing LUNs for the second fabric CTCs via DCFM.	122
4.4.9	Activating the DEK cluster CTC definitions.	126
4.4.10	Final activation/confirmation of the new paths (via a second fabric).	130
4.5	Adding a second Encryption Switch for high availability	131

4.6	Creating and adding a new LUN to an existing EG.	137
4.6.1	Creating and adding a new LUN in a single-fabric environment.	137
4.6.2	Creating a new LUN in the IBM Storwize V7000	138
4.6.3	Adding a newly created LUN into an existing encryption group	143
4.6.4	Creating and adding a new LUN in a dual-fabric environment	153
4.6.5	Creating a new LUN in the IBM Storwize V7000 (dual fabric)	155
4.6.6	Adding a new LUN into an existing encryption group (dual fabric)	155
4.7	Adding and removing a path to an initiator from a CTC	155
4.7.1	Adding host paths	155
4.7.2	Removing host paths	160
4.8	Changing a host HBA	163
Chapter 5. Advanced techniques and functionality		171
5.1	Frame redirection zone in detail	172
5.1.1	Name service	173
5.1.2	Crypto Target Container	174
5.1.3	Estimating the number of required CTCs	175
5.1.4	Estimating the multipathing requirements for the LUN	178
5.1.5	Adding LUNs to the CTCs.	179
5.1.6	Understanding the structure of the CTC	182
5.1.7	Changing the fabric configuration	185
5.2	First-time encryption and rekey operation in detail	186
5.2.1	Performance effect of the first-time encryption and rekey operations.	186
5.2.2	The nature of the first-time encryption and rekey operations	187
5.3	Designing the encryption solution	195
5.3.1	Performance effect of encryption operations	196
5.3.2	Defining the correct number of encryption engines.	196
5.3.3	Connecting the encryption engine	197
5.4	Copy services in the encryption environment	197
5.4.1	Metro/Global Mirror	204
5.5	External storage virtualization	206
5.5.1	Thin-provisioned volumes	208
Chapter 6. Maintenance and troubleshooting		209
6.1	Firmware upgrades	210
6.2	Master key maintenance	211
6.2.1	Backing up the master key to a file	212
6.2.2	Backing up the master key to the smart cards	213
6.2.3	Restoring the master key from a file	214
6.2.4	Restoring the master key from the smart cards	216
6.3	Configuration upload and download	217
6.3.1	Uploading the configuration	217
6.3.2	Configuration download	219
6.4	Adjusting heartbeat signaling values.	221
6.5	Removing and replacing the IBM SAN768/384 Encryption Blade	222
6.5.1	Encryption node removal and replacement in a multinode group.	225
6.6	Removing stale rekey information for a LUN	229
6.7	Troubleshooting	229
6.7.1	Errors when adding a switch to an existing group.	229
6.7.2	Errors related to adding a switch to a new group	230
6.7.3	LUN policy troubleshooting.	233
Related publications		235
IBM Redbooks publications		235

Other publications	235
Online resources	236
Help from IBM	237
Index	239

Notices

This information was developed for products and services offered in the U.S.A.

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 give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: 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 states 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 supply in any way it believes appropriate without incurring any obligation to you.

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.

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 the names and addresses used by an actual business enterprise 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.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	POWER6®	Storwize®
DB2®	POWER7®	System Storage®
DS8000®	PowerVM®	System z®
FICON®	POWER®	Tivoli®
FlashCopy®	Redbooks®	WebSphere®
IBM®	Redbooks (logo)  ®	z/OS®

The following terms are trademarks of other companies:

Linear Tape-Open, LTO, Ultrium, the LTO Logo and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

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

Snapshot, and the NetApp logo are trademarks or registered trademarks of NetApp, Inc. in the U.S. and other countries.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Intel, Intel logo, Intel Inside, Intel UNIX, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

In this IBM® Redbooks® publication, we describe how these products can be combined to provide an encryption and virtualization solution:

- ▶ IBM System Storage® SAN32B-E4 Encryption Switch
- ▶ IBM Storwize® V7000
- ▶ IBM Tivoli® Key Lifecycle Manager

We describe the terminology that is used in an encrypted and virtualized environment, and we show how to implement these products to take advantage of their strengths.

This book is intended for anyone who needs to understand and implement the IBM System Storage SAN32B-E4 Encryption Switch, IBM Storwize V7000, IBM Tivoli Key Lifecycle Manager, and encryption.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center. Figure 1 (Left to Right) includes Glen Routley, Stefan Neff, Jon Tate, and Denis Senin.



Figure 1 Glen Routley, Stefan Neff, Jon Tate, and Denis Senin

Jon Tate is a Project Manager for IBM System Storage SAN Solutions at the International Technical Support Organization, San Jose Center. Before joining the ITSO in 1999, he worked in the IBM Technical Support Center, providing Level 2 support for IBM storage products. Jon has 26 years of experience in storage software and management, services, and support, and is both an IBM Certified IT Specialist and an IBM SAN Certified Specialist. He is also the UK Chairman of the Storage Networking Industry Association.

Stefan Neff is a Leading Technical Sales Professional and Architect for IBM Data Protection and Retention Solutions within the IBM Germany STG Technical Sales organization. Before joining the Technical Sales team in 2006, he worked for several years in the IBM Product Field Engineering (PFE) team in Mainz to provide Level 3 support in the area of Tape and Virtual Tape Systems (VTS) to the EMEA region. Stefan has 14 years of experience in backup and archive solutions (especially mainframe virtual tape systems), services, and support. Stefan is also a member of the worldwide IBM SWAT Team for storage solutions and one of the leading experts in middle Europe for IBM Tape Encryption solutions. He also focuses on software-based backup solutions using the IBM Tivoli Storage Manager. He is both a Level 2 IBM Certified IT Specialist and an IBM High-End-Tape V5/V6 Certified Specialist. Stefan is also the chairman of the German Technical Focus Group (TFG) "Data Protection & Retention". Stefan also holds patents in the area of storage virtualization (1st Plateau inventor).

Glen Routley is a SAN and Storage Virtualization specialist in IBM Australia. He has 20 years of experience in the IBM System z® field and 11 years of experience supporting the IBM SAN portfolio of products. He leads the SAN and SAN Volume Controller (SVC) Storage Central team in A/NZ and has supported SVC and V7000 since their initial releases. He has worked at IBM for 27 years and co-authored numerous IBM Redbooks publications, including *Implementing an IBM b-type SAN with 8 Gbps Directors and Switches*, SG24-6116-10, and *IBM System Storage b-type Multiprotocol Routing: An Introduction and Implementation*, SG24-7544-03. In this book, he has written extensively about the configuration of the V7000 and the SAN32B-E4 Encryption Switch.

Denis Senin is an IT Specialist in IBM Russia. He has 10 years of experience in the IT industry and has worked at IBM for six years. Denis holds a Masters degree of Design-Engineering of Computer Systems from the Moscow State University of Radiotechnics, Electronics and Automatics. His current areas of expertise include open systems high-performing and disaster recovery storage solutions.

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

Tayfun Arli
Chris Canto
Peter Eccles
John Fairhurst
Carlos Fuente
Howard Greaves
Geoff Lane
Alex Howell
Andrew Martin
Cameron McAllister
Paul Merrison
Lucy Harris
Matt Smith
Barry Whyte
Muhammad Zuba
IBM Hursley

Chris Saul
IBM San Jose

Axel Melber
IBM Germany

A special thanks also goes to Brocade for hosting the authors in the Brocade headquarters building in San Jose, California:

Yong Choi
Silviano Gaona
Brian Steffler
Marcus Thordal
Steven Tong
Brocade Communications Systems

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

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- Send your comments in an email to:

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?0openForm>

- Stay current on recent Redbooks publications with RSS Feeds:

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



Introduction to SAN encryption using the Storwize V7000 and the SAN32B-E4 Encryption Switch

In this chapter, we discuss several reasons why you might want to use SAN encryption for your IBM Storwize V7000 and outline the products that we used. We describe our existing host to Storwize V7000 SAN connection, and the environment after the encryption solution is implemented.

IBM Network Advisor: In this book, we used Data Center Fabric Manager (DCFM) to implement our solution. This product has been rebranded as IBM Network Advisor. The functionality is exactly the same. There are no significant differences between using IBM Network Advisor and DCFM.

1.1 Why we need SAN security and encryption solutions

In the following topics, we describe several of the reasons for implementing security solutions.

1.1.1 Threats and security

A Fibre Channel threat in its simplest form is anything that can harm your SAN. An IT security threat is anything that can harm your IT assets. There are various types of threats to your IT assets:

- ▶ Disasters: Earthquake, flood, hurricane, thunderstorm, tornado, fire, terrorism, war, and so on
- ▶ Technology: Viruses, trojans, worms, spyware, botnets, rootkits, spam, denial of service attacks, and so on
- ▶ Human: Malicious hackers, industrial espionage, cyber terrorists, criminals, disgruntled employees, carelessness, lack of training, and lack of security

There are many more threats in addition to these threats that must be considered, so you have to protect against a variety of external and internal threats. Therefore, security is extremely important for your IT assets and especially for your stored and backed-up data.

Over the last few years, the SAN environment has grown dramatically. Although each organization has its own unique perspective on the subject of SAN security, certain issues are common to all of these groups. Several misconceptions have developed from the early days of the SAN, which unfortunately have become integrated and accepted into normal IT business and are now perceived as fact, for example:

- ▶ SANs are inherently secure, because they are in a closed, physically protected environment.
- ▶ The Fibre Channel (FC) Protocol (FCP) is not well known by hackers and there are almost no avenues available to attack FC fabrics.
- ▶ You cannot “sniff” optical fiber without cutting it first and causing disruption.
- ▶ Even if it were possible to sniff fiber-optic cables, there are so many protocol layers, file systems, and database formats that the data will be illegible in any case.
- ▶ Even if it were possible to sniff fiber-optic cables, the amount of data to capture is simply too large to capture realistically and requires expensive equipment to do so.
- ▶ If the switches already come with built-in security features, why do I need to be concerned with implementing security features in the SAN?

There is always a risk that unpredictable things can happen to your SAN, so you need to consider how to protect your SAN and your data.

Also, look at the storage subsystem perspective, for example, a disk storage subsystem, with respect to the various threats and security. A big concern is often when disk drives leave the company premises, which usually happens when a disk drive fails and it is replaced with a new drive. Perhaps the drive is not really damaged and data can still be accessed. Of course, IBM has a procedure to delete all data on the drive.

However, this task is no longer under the control of the client. Certain clients buy the defective drives back and destroy the drives themselves, which can be quite expensive. Another concern is when the entire disk storage subsystem is going to be returned to the vendor, or to the used market. The vendor, for example, IBM, will erase all the data, but this action is not sufficient for certain clients. IBM offers a service (IBM Certified Secure Data Overwrite) to

erase all data (using several passes) in compliance with the American Department of Defense regulations (DoD 5220.20-M). But, do other disk subsystem vendors have such processes and services in place?

However, all these concerns become redundant when the data on the disk drives is encrypted, because without the decryption key the data is unreadable.

1.2 Encryption basics

In this section, we describe basic encryption, cryptographic terms, and several ideas about how you can protect your data. Encryption is one of the simple ways to secure your data. If the data is stolen, lost, or acquired in any way, it cannot be read without the encryption key.

Encryption has been used to exchange information in a secure and confidential way for many centuries. Encryption transforms data that is unprotected (plain or clear text) into encrypted data, or *ciphertext*, by using a key. It is difficult to “break” ciphertext to change it back to clear text without the associated encryption key. Earlier in history, people used a code-book or a secret alphabet as a “key” to encrypt and decrypt a certain kind of data, which was most likely written text.

Computer technology has enabled increasingly sophisticated encryption algorithms for digitally stored data. Working with the U.S. Government National Institute of Standards and Technology (NIST), IBM invented one of the first computer-based algorithms, Data Encryption Standard (DES), in 1974. Today, several widely used encryption algorithms exist, including triple DES (TDES) and Advanced Encryption Standard (AES).

Today, there are generally two general methods of encryption available: symmetric key encryption and asymmetric key encryption.

1.2.1 Symmetric key encryption

Symmetric key encryption uses identical keys, or keys that can be related through a simple transformation, for encryption and decryption. This encryption method is the oldest and best-known technique. Similar to the key of your house, you have *one* key to lock and unlock the main door.

Symmetric key encryption algorithms are significantly faster than asymmetric encryption algorithms, which makes symmetric encryption an ideal candidate for encrypting large amounts of data or dynamically encrypting data while processing the data. Depending on the key sizes (128, 160, 192, 224, and 256 bits), you can make the symmetric key encryption safer. The most popular and respected algorithms are AES, Blowfish, CAST5, IDEA, RC4, Serpent, TDES, and Twofish.

Benefits: Speed and short key length are the advantages of symmetric encryption.

Figure 1-1 on page 4 shows an example of how symmetric encryption and decryption works in detail. In this scenario, both Jan and Jonas are aware of the private key that they use to perform encryption and decryption. However, if anyone else gains knowledge of this key, they will be able to transform the ciphertext back to plain text. If Jan and Jonas want to preserve confidentiality, they must highly protect the symmetric key and keep it in an extremely safe place.

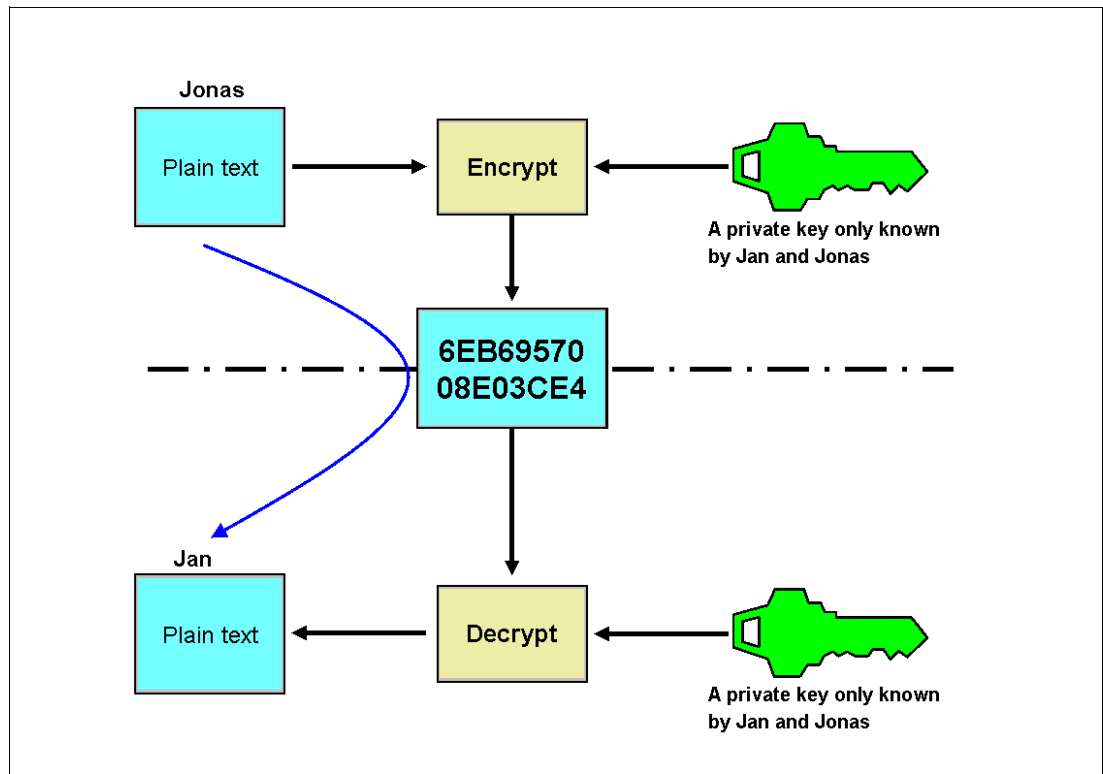


Figure 1-1 Symmetric key encryption

1.2.2 Asymmetric key encryption

The asymmetric key encryption method uses a *key pair* for encrypting and decrypting data. One key is used to encrypt the data, and the other key is used to decrypt the data. Because the key that is used for encrypting plain text cannot be used for decrypting it, this key does not have to be kept a secret. It can be widely shared and is therefore called a *public key*. Anyone who wants to send secure data to a person can use this public key. The receiving person then uses its related *private key* to decrypt back to the plain text. The private key is the corresponding half of the public-private key pair and must always be kept as a secret. This process is also called *public-private key encryption* or *public key encryption*.

Well-known examples of asymmetric key algorithms are RSA, Diffie-Hellman, Elliptic curve cryptography (ECC), and ElGamal. Currently, the Rivest-Shamir-Adleman (RSA) algorithm is the most widely used public key technique on the market.

Figure 1-2 on page 5 gives you an idea of how asymmetric key encryption works. Jonas is aware of Jan's public key and can encrypt his plain text with his public key. He can send the encrypted data over to Jan, who is able to decrypt back to plain text using his secret private key.

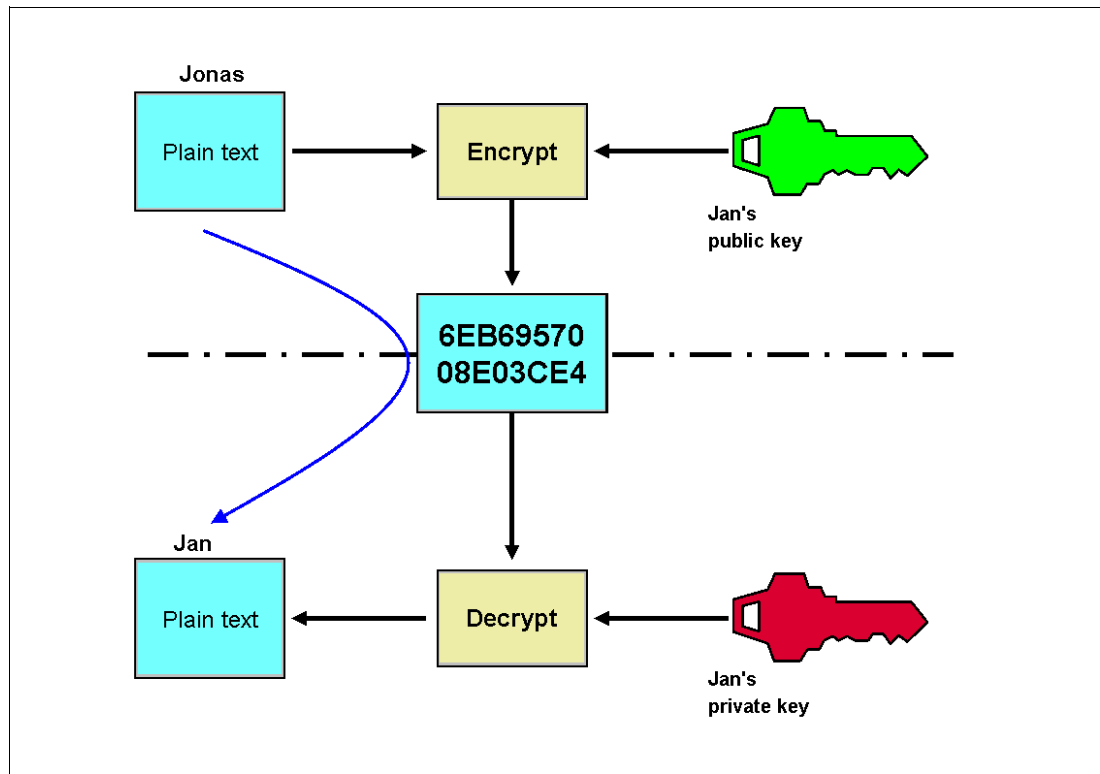


Figure 1-2 Asymmetric key encryption

1.2.3 Digital certificates

If you use one of these encryption methods, you must also be certain that the person or machine to whom or which you are sending and receiving the key is the correct one. When you initially receive someone's public key for the first time, how do you know that this individual is really the person they claim to be? If "spoofing" someone's identity is so easy, how do you knowingly exchange public keys?

The answer is to use a digital certificate. A *digital certificate* is a digital document that is issued by a trusted institution that vouches for the identity and key ownership of an individual: it guarantees authenticity and integrity.

There are trusted institutions all over the world that generate trusted certificates. We will use this kind of mechanism also. In the first instance, we use a certificate that has been generated by our switch and Tivoli Key Lifecycle Manager. For more details, go to Chapter 3, "Initial setup for the IBM Tivoli Key Lifecycle Manager and the SAN32B-E4 Encryption Switch" on page 35.

1.2.4 Encryption algorithms

Understand that there are several encryption schemes and algorithms in the field. The following encryption algorithms are the most popular encryption algorithms in use today:

- ▶ 3DES
- ▶ DES
- ▶ AES
- ▶ RSA
- ▶ ECC

- ▶ Diffie-Hellman
- ▶ DSA
- ▶ SHA

AES256: The SAN32B-E4 Encryption Switch and the FC Encryption Blades use AES256. AES256 uses a key of 256 bits for encryption. The National Institute of Standards and Technology (NIST) announced the AES256 standard.

To get more information about the details of IBM System Storage Data Encryption, refer to *IBM System Storage Data Encryption*, SG24-7797.

1.2.5 Encryption challenges

Encryption depends on encryption keys. Those keys have to be, at the same time, kept secure and available, and responsibilities have to be divided:

- ▶ Key security

To preserve the security of encryption keys, make sure that no one individual (system or person) has access to all the information that is required to determine the encryption key. In a system-based solution, the encryption data keys are encrypted with a *wrapping key*, which is another key to encrypt and decrypt the data keys, typically, an asymmetric key. This wrapped key method is used with the SAN32B-E4 Encryption Switch when a data key for a LUN is created or exchanged from the disk storage subsystem with the key server.

- ▶ Key availability

Ensure that more than one individual (person or system) has access to any single piece of information that is necessary to determine the encryption key. In a system-based solution, redundancy is provided by having multiple isolated key servers. Additionally, backups of the key server's data are maintained.

- ▶ Separation of responsibilities

The SAN32B-E4 Encryption Switch and the FC Encryption Blade offer a master key (MK), which can be stored for recovery purposes. To prevent one person from gaining access to the data, the handling of a recovery key requires two people (separate roles): a security administrator and a storage administrator. This setup can be achieved with the optional features for the SAN32B-E4 Encryption Switch and Blade family by implementing a Smartcard Reader and using a Smartcard. For more details, refer to the *Fabric OS Administrator's Guide Supporting Fabric OS v7.0.0*, 53-1002148-02, and *Fabric OS Encryption Administrator's Guide Supporting Tivoli Key Lifecycle Manager (TKLM) Environments Supporting Fabric OS v7.0.0*, 53-1002162-02.

The sensitivity of possessing and maintaining encryption keys and the complexity of managing the number of encryption keys in a typical environment result in a client requirement for a key server. A key server is integrated with encrypting storage products to resolve most of the security and usability issues that are associated with key management for encrypted storage.

IBM Tivoli Key Lifecycle Manager: IBM offers an enterprise-scale key management infrastructure through IBM Tivoli Key Lifecycle Manager (TKLM) and lifecycle management tools to help organizations efficiently deploy, back up, restore, and delete keys and certificates in a secure and consistent fashion.

However, the client must be sufficiently aware of how these products interact to be able to provide the appropriate management of the IT environment. Even with a key server, generally at least one encryption key (the overall key that manages access to all other encryption keys, or a key that encrypts the data that is used by the key server) or a recovery key must be maintained manually.

One critical consideration with a key server implementation is that all code and data objects that are required to make the key server operational must *not* be stored on storage that depends on any key server to be accessed. A situation where all key servers cannot become operational because there is data or code that cannot be accessed without an operational key server is referred to as an *encryption deadlock*. It is analogous to having a bank vault that is unlocked by using a combination lock and the only copy of the combination lock is locked inside the vault.

1.3 IBM encryption products and software

In this section, we review the products that we used in our lab environment for this encryption implementation. Make sure you check the Interoperability Matrix to ensure that your hosts are supported before beginning your implementation.

<http://www.ibm.com/storage/support/san/>

1.3.1 IBM Tivoli Key Lifecycle Manager

The IBM approach to key management revolves around IBM Tivoli Key Lifecycle Manager (TKLM), a product that was announced in 2008 and that will be enhanced in phases. From an initial focus on key management for tape and disk encryption, IBM plans to expand TKLM into a centralized key management facility for managing encryption across a range of deployments. The latest version, V2, supports key exchange with the SAN32B-E4 Encryption Switch and FC Blades and also delivers keys for a SAN encryption solution.

A large number of symmetric keys, asymmetric keys, and certificates can exist in an enterprise. You must manage all these keys and certificates. You can use TKLM to handle the management of these keys and certificates.

TKLM is an application that performs key management tasks for IBM encryption-enabled hardware, such as the IBM System Storage DS8000® and DS5000 series family, the IBM encryption-enabled tape drives (TS11x0 series and TS10x0 series), and of course, the SAN32B-E4 Encryption Switch and FC Blade. TKLM provides, protects, stores, and maintains the encryption keys that are used to encrypt information being written to, and decrypt information being read from, an encryption-enabled device.

TKLM is designed to be a shared resource that is deployed in several locations within an enterprise. It is capable of serving numerous IBM encryption-enabled hardware devices, regardless of where those devices reside. The typical communication between TKLM and the storage subsystem or device is based on the IP protocol.

The sole task of TKLM is to handle the serving of keys to the encrypting drives. TKLM does not perform any cryptographic operations, such as generating encryption keys, and it does not provide storage for keys and certificates. To perform these tasks, TKLM has to rely on external components. In Chapter 3, “Initial setup for the IBM Tivoli Key Lifecycle Manager and the SAN32B-E4 Encryption Switch” on page 35, we describe the TKLM components and the resources that are used by TKLM.

In addition to the key-serving function, TKLM also offers the following additional functions over the previous product Encryption Key Manager (EKM):

- ▶ Lifecycle functions:
 - Notification of certificate expiration through the Tivoli Integrated Portal (TIP in Figure 1-3)
 - Automated rotation of certificates
 - Automated rotation of groups of keys
- ▶ Usability enhancements of the previous EKM software:
 - Provides a graphical user interface (GUI)
 - Initial configuration wizards
 - Migration wizards
- ▶ Integrated backup and restore of TKLM file:
 - One button to create and restore a single backup that is packaged as a JAR file
- ▶ Tivoli Integrated Portal installation manager:
 - Simple to use installation for Windows, Linux, IBM AIX®, or Solaris
 - Can be a silent installation

Figure 1-3 shows the TKLM components and external resources.

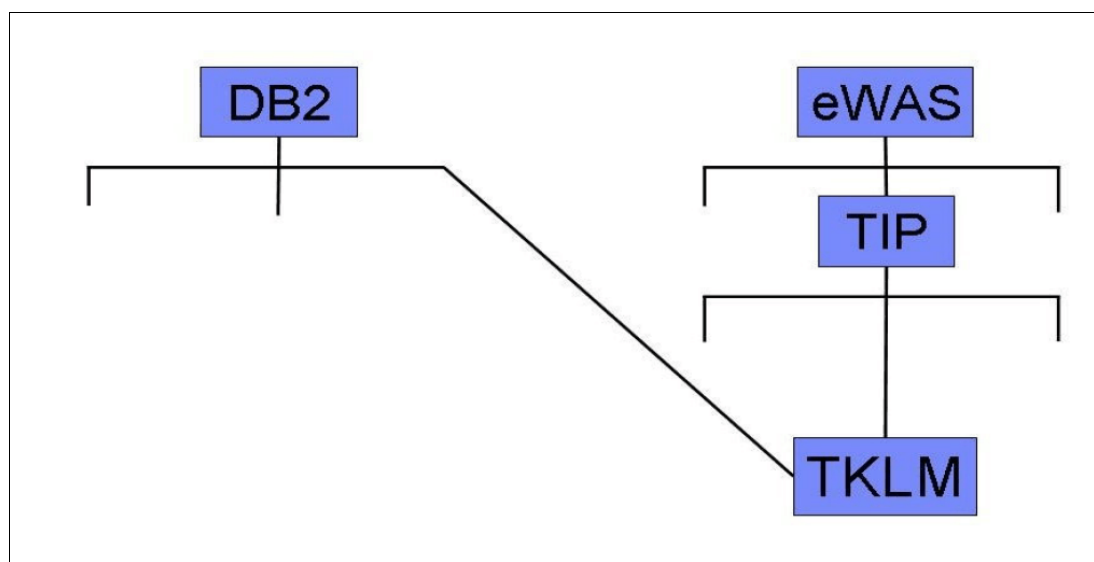


Figure 1-3 TKLM components and resources

1.3.2 SAN32B-E4 Encryption Switch

The SAN32B-E4 Encryption Switch, which is shown in Figure 1-4 on page 9, provides a high-performance encryption solution (96 Gbps crypto for disk and 48 Gbps for tape). It has 32 x 8 Gbps high-speed FC ports and hot-swappable, redundant power supplies and fans that provide for high availability. You can use the SAN32B-E4 for its AES 256-bit encryption and compression features on data-at-rest that is on supported disk LUNs and tape drives.



Figure 1-4 IBM SAN32B-E4

The SAN384B and SAN768B Encryption Blade, which is shown in Figure 1-5, provides the same encryption and compression features as the SAN32B-E4 switch, although it has 16 x 8 Gbps FC ports on the Encryption Blade. However, this number is not a limitation, because any FC port on another Encryption Blade within the director can be used in the same way.

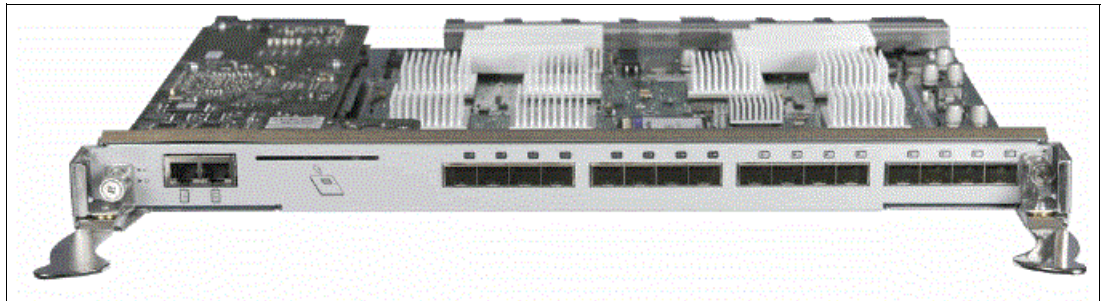


Figure 1-5 Encryption Blade for SAN384B or SAN768 directors

1.3.3 IBM Storwize V7000

The IBM Storwize V7000 is a virtualizing storage system that provides virtualized LUNs to hosts from storage pools and offers great flexibility and performance. It is based on the well-proven IBM SAN Volume Controller platform, and it combines IBM DS8000 enterprise storage array technology to provide a highly reliable mid-range storage platform.

Figure 1-6 shows a single enclosure Storwize V7000.



Figure 1-6 IBM Storwize V7000 model 124

By using the IBM Storwize V7000, you can encrypt target LUNs on the internal arrays. And, you can encrypt any targets on external arrays that are supported by the Storwize V7000, even though they might not be listed on the encryption support matrix.

1.4 SAN environment (pre-encryption)

We implemented an encryption solution on a fairly typical SAN environment. Figure 1-7 shows our initial lab environment. We have two stand-alone hosts, both running Microsoft Windows Server 2008. These hosts have dual-FC paths via redundant b-type fabrics to target LUNs that are provisioned on the IBM Storwize V7000.

For the purpose of our encryption implementation, each host has access to a separate LUN on the Storwize V7000, and neither host can access the other host's LUN.

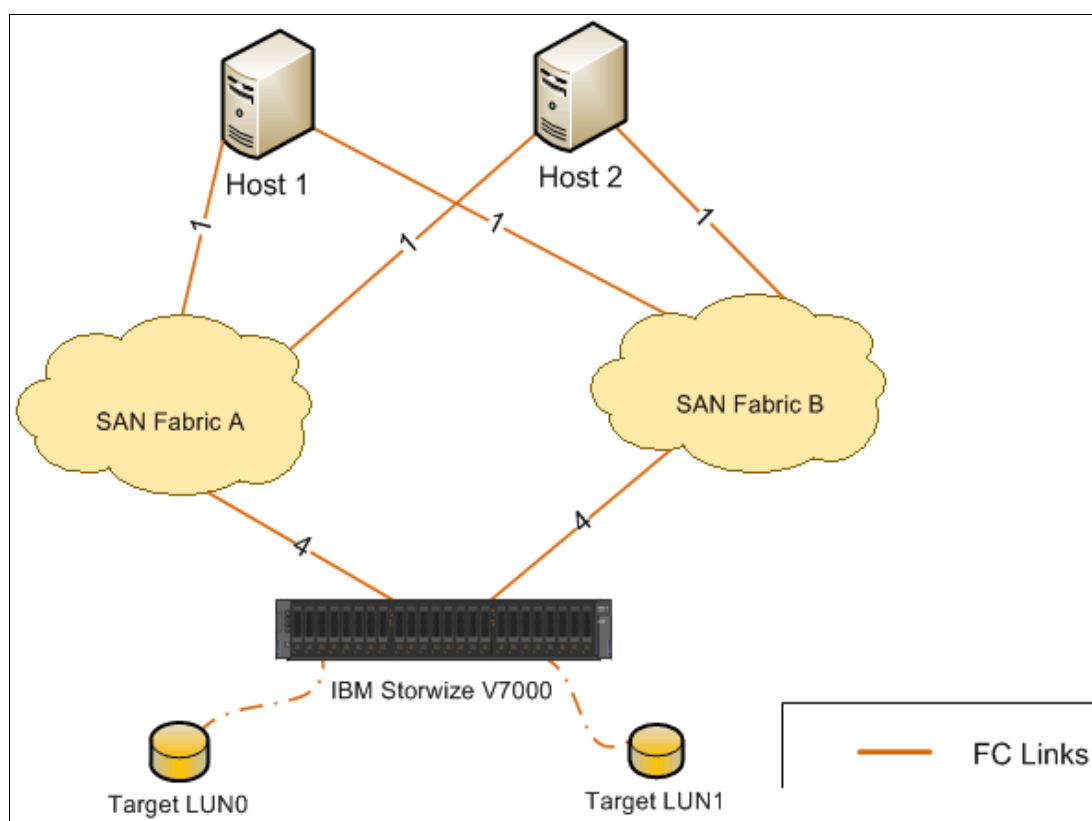


Figure 1-7 Standard dual-fabric SAN

The fabrics are IBM b-type 8 Gbps switches (the specific model of switch is irrelevant). We used the SAN06B-R for Fabric A and SAN384B for Fabric B. Both fabrics run Fabric Operating System (FOS) 7.0.0a.

The SAN06B-R fabric is zoned for access, as shown in Example 1-1. We do not show the zoning for Fabric B, but it is zoned identically.

Example 1-1 Existing fabric zone configuration

```
IBM_2498_R06:admin> cfgshow
Defined configuration:
cfg: ITS0_SG247977
```



```

W2k8_1_to_V7000_1; W2k8_2_to_V7000_1
zone: W2k8_1_to_V7000_1
V7000_1_p3; V7000_1_p4; Win2k8_1
zone: W2k8_2_to_V7000_1
V7000_1_p3; V7000_1_p4; Win2k8_2
alias: V7000_1_p3
50:05:07:68:01:30:a7:be; 50:05:07:68:01:30:a7:fe
alias: V7000_1_p4
50:05:07:68:01:40:a7:be; 50:05:07:68:01:40:a7:fe
alias: Win2k8_1
10:00:00:05:1e:c7:6b:8a
alias: Win2k8_2
10:00:00:05:1e:c7:6b:a2

```

Effective configuration:

```

cfg: ITS0_SG247977
zone: W2k8_1_to_V7000_1
50:05:07:68:01:30:a7:be
50:05:07:68:01:30:a7:fe
50:05:07:68:01:40:a7:be
50:05:07:68:01:40:a7:fe
10:00:00:05:1e:c7:6b:8a
zone: W2k8_2_to_V7000_1
50:05:07:68:01:30:a7:be
50:05:07:68:01:30:a7:fe
50:05:07:68:01:40:a7:be
50:05:07:68:01:40:a7:fe
10:00:00:05:1e:c7:6b:a2

```

Our Storwize V7000 is a single control enclosure (2076-124) running at a 6.2.0.2 software level. A 30 GB LUN is provisioned to Host 1 and a 50 GB LUN is provisioned to Host 2, as shown in Figure 1-8. These LUNS have pre-existing data on them that must be accessible throughout the entire encryption implementation.

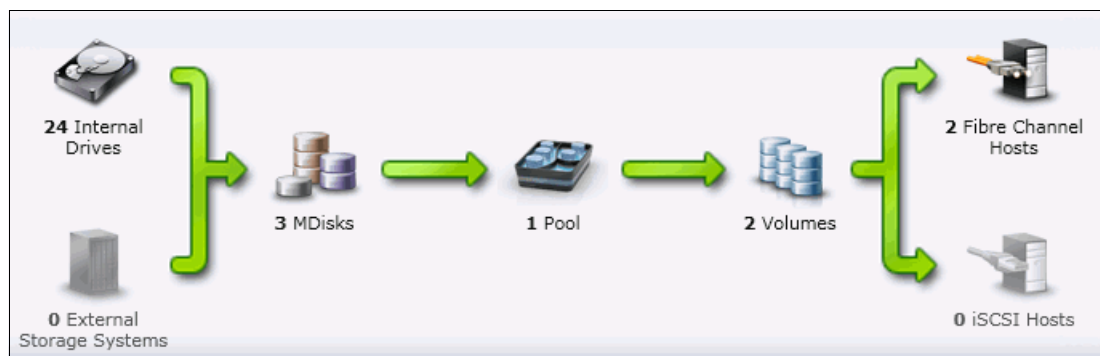


Figure 1-8 Storwize V7000 provisioning

1.5 Encrypted SAN environment

To provide an encryption solution for our high-availability, dual-fabric, multipath environment, we require at a minimum an encryption engine (EE) for each fabric. We use the SAN32B-E4 switch on Fabric A and the Encryption Blade in the DCX director on Fabric B. We use two

TKLM servers to provide a primary and secondary key vault function to store and manage the crypto keys for the encryption group. Figure 1-9 shows the resulting configuration.

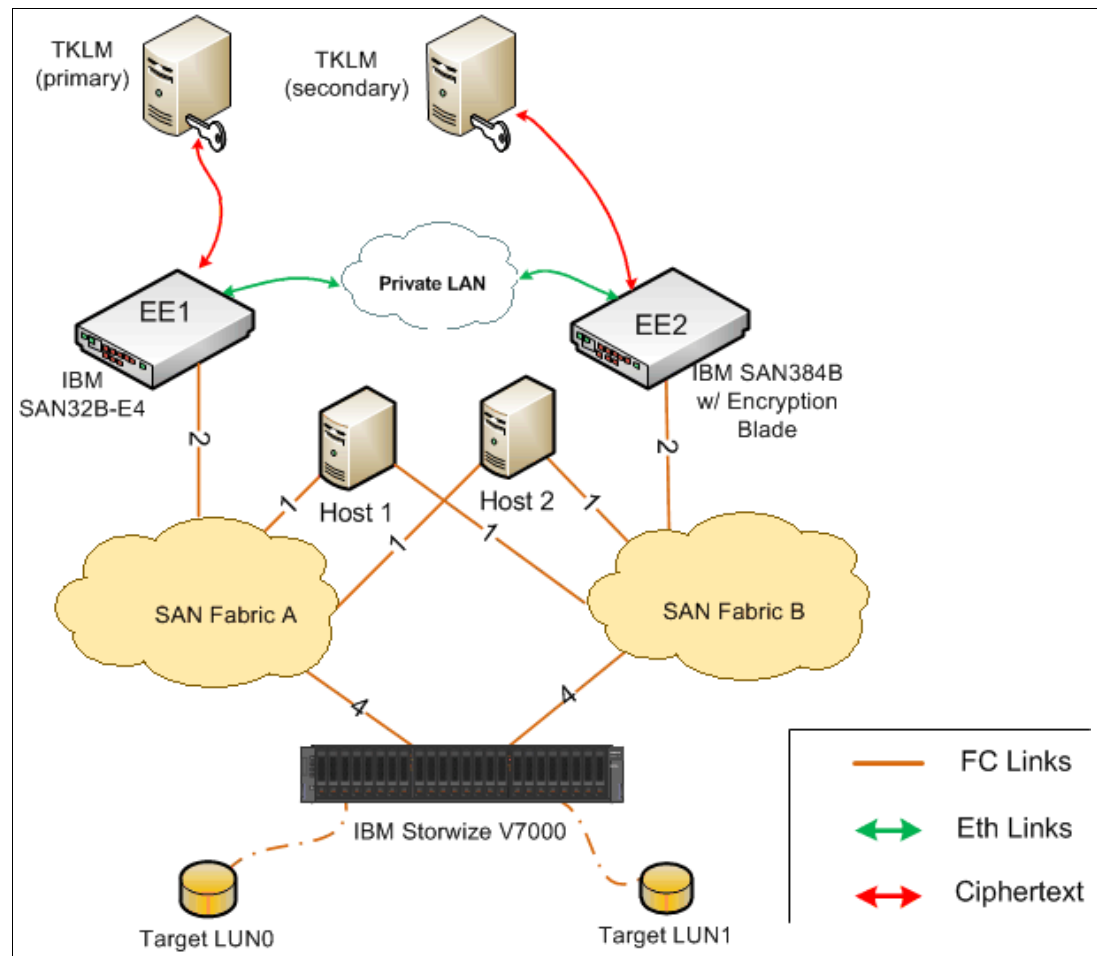


Figure 1-9 Encrypted SAN configuration

In this dual-fabric, multipath configuration, it is extremely important that key synchronization is maintained between the EEs on each fabric; otherwise, data corruption will occur. This synchronization occurs automatically between the EEs. The synchronization is achieved with the private LAN that is dedicated for this purpose only.



Terminology and technology

In this chapter, we describe and explain the terminology that is used in this book. You need to understand the terminology completely. Remember that everything is explained from the SAN32B-E4 Encryption Switch, IBM SAN384/768, IBM Tivoli Key Lifecycle Manager (TKLM), and IBM Storwize V7000 points of view and might differ from the meanings with which you are familiar. Also, we explain several of the key technological and architectural aspects in detail.

This chapter covers the following topics:

- ▶ Basic terminology
- ▶ Elements of encryption
- ▶ Encryption process terminology
- ▶ High-availability terminology and concepts

2.1 Why terminology is important

Encryption is one of many ways to protect data from unauthorized access. In contrast to other ways, which work with the metadata and attributes, encryption works with the data itself. Encryption changes data completely, preventing any untoward actions. Therefore, you must design encryption solutions well and implement them carefully. From the outset, it is important to have a clear understanding of encryption and why you implement it, which makes the terminology important. Having a clear and complete understanding of the terminology and technology helps enormously. In this chapter, our main purpose is to describe the well-known terms relating to the encryption process, and the specific hardware and software that we have used.

In terms of our audience, we assume that specialists reading this book have a basic knowledge of storage area networks (SANs) and are familiar with SAN terminology. If not, we recommend reading *Introduction to Storage Area Networks*, SG24-5470.

We describe the common terms, which have no strict relationship to the process, briefly. We explain in detail the main terms and technology and supplement the explanations with graphic illustrations, where appropriate, to help you to understand the material better.

2.2 Basic terminology

In the following sections, we define how we use several basic terms with which you probably are already familiar.

2.2.1 Data

Data here means that the data resides on the storage system and relates only to the user data that is placed on the logical unit number (LUN) to be encrypted/decrypted later. We are not referring to any configuration or management data that is used by the system for the background processes. Data can be on the LUN or in the disk system's cache, or it can be in the process of being transmitted, encrypted, decrypted, deleted, restored, or saved.

2.2.2 LUN

LUN is similar to the common meaning of the Small Computer System Interface (SCSI) term, the logical unit number. In this book, LUN is the central term that takes part in every movement that relates to the data, connection, and recovery purposes.

2.2.3 Fabric and SAN

Fabric and *SAN* have the common meaning of the Fibre Channel (FC) fabric and storage area network (SAN) in this book. This component is the only component to unite storage and servers and provide transport for the data to encrypt and store. We do not transmit any management, cluster, or configuration data over the SAN in this book.

2.2.4 Management network

We mostly use the term *management network* to define the network to which the management ports of the devices connect. This network is used for configuration purposes, monitoring, initializing, and following the key exchange process. It is also important to know that the key vault transfers keys to the encryption engines (EEs) using the management network. This type of communication is not fully secure in terms of the safe key and information exchange process, so every record that relates to the key or certificate is encrypted while being transmitted in this network. You can include the management network as part of an existing network.

2.2.5 Private network

We use the term *private network* to describe the network that is used to form a cluster of EEs and to provide intracluster communication between EEs, failover, and failback actions. All records, which are in transport in this network, are encrypted.

All switches in the planned EG must interconnect on a private LAN. We use this LAN to exchange security parameters and certificates and to synchronize EE operations. Both ports of each SAN32B-E4 Encryption Switch or Encryption Blade must connect to the same IP network and the same subnet. You must assign static IP addresses. Do *not* use VLANs or Dynamic Host Configuration Protocol (DHCP). These two ports are bonded together as a single virtual network interface to provide link layer redundancy.

You must interconnect all Encryption Switches and Blades in an EG by these links through a dedicated LAN before their EEs are enabled. Security parameters and certificates cannot be exchanged if these links are not configured and active.

2.2.6 Key

We use the term *key* as a label for the output of the work of the specific mathematical functions that are used for the purposes of encryption and the subsequent generation of keys. We use this term in combination with other terms, such as asymmetric and symmetric. We use the term key in many situations to define something that is used to encrypt/decrypt data. The key needs to be stored and kept in a safe place.

2.2.7 Certificate

A *certificate* is the result of the work of specific mathematical functions that are based on the key, or a passphrase, that is used to authenticate devices with each other for the communication session or for continuous interaction. There are many kinds of certificates mentioned in this book, and each one is described when appropriate.

2.2.8 CryptoModule

The *CryptoModule* is the secure part of an EE that is protected to the Federal Information Processing Standards (FIPS) 140-2 level 3 standard. The term CryptoModule is used primarily in the context of FIPS authentication. We do not distinguish this term from the EE definition.

2.2.9 Cleartext

Cleartext is simple plain text or unencrypted data that must be protected with encryption.

2.2.10 Ciphertext

Ciphertext is data or text that has been encrypted with the specific encryption algorithms in the IBM SAN32B-E4 switch or the IBM SAN768\384 Encryption Blade.

2.3 Elements of the encryption process

In the following topics, we describe the elements of the encryption process that are important for you to understand.

2.3.1 Encryption group

An *encryption group* (EG) is a collection of one or more data encryption key (DEK) clusters, high-availability (HA) clusters, or both, which share the same key vault and device configuration and are managed as a single group. EGs can span multiple encryption nodes and even multiple fabrics. The maximum number of encryption nodes in one EG is four. The maximum number of EEs is 16 (maximum of four per node) for one EG.

Clarification: It is important to understand that the EG does not form any kind of cluster itself. It is merely the intention to describe the fact that several EEs have the same set of rules and have access to the same key vault. All kinds of clusters are defined within the EG.

2.3.2 Group leader

A *group leader* is a special node within an EG that acts as a group and cluster manager. It manages and distributes all group-wide and cluster-wide configurations to all members of the group or cluster. The group leader performs tasks, such as configuration, key vault certificate distribution, and DEK creation, for the entire group. If the group leader node fails, another node becomes the group leader node. Nodes that will be included in the EG must be registered on the group leader node.

Group leader succession: The order in which member node registration is performed defines the group leader succession. At any given time, only one active group leader exists in an EG. The group leader succession list specifies the order in which group leadership is assumed if the current group leader is not available.

2.3.3 Encryption engine

The *encryption engine* is the entity within a node that performs encryption operations, including the generation of the data encryption keys. The EE is the workhorse that performs the algorithm computations. It is the logical entity within the encryption node. Each IBM SAN32B-E4 is a *single* EE, and each IBM SAN768/385 Encryption Blade is a *single* EE.

2.3.4 Encryption node

An *encryption node*, in the context of this book, is a switch or a SAN Director backbone through which users can manage an EE. Each IBM SAN32B-E4 switch is a *single* EE and a *single* encryption node at the same time. However, an IBM SAN768/384 with four installed Encryption Blades is a *single* encryption node with *four* EEs.

Figure 2-1 shows a diagram of an EG, the encryption nodes, and the EEs.

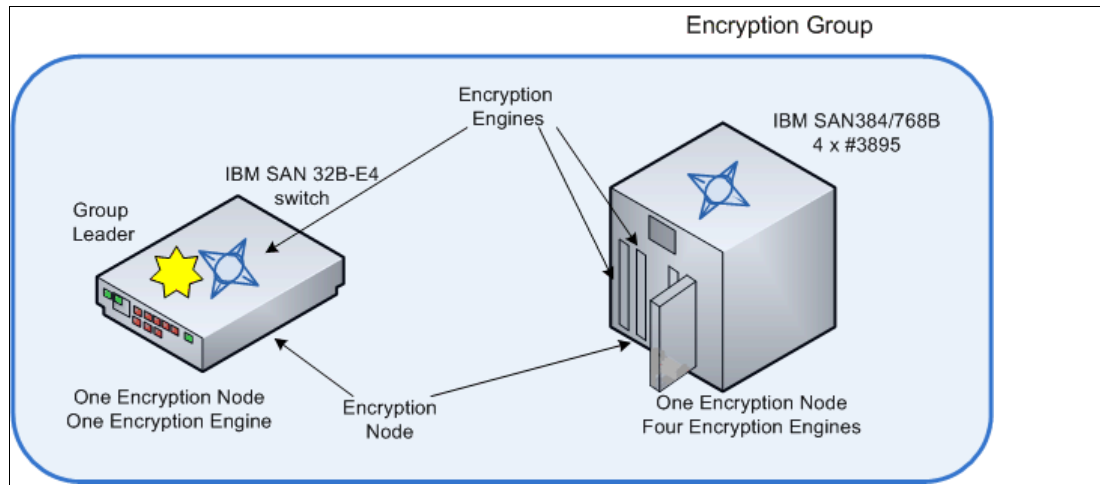


Figure 2-1 EG, encryption nodes, and EEs

2.3.5 Encryption certificates

We use these security certificates, which are generated both on encryption nodes and the key vault, for mutual authentication and access control. Each node generates a special type of security certificate (CPcert) during the initialization process. Each node in the group must export its CPcert to the group leader node, which provides the node access to the group and to the configuration settings of the group. Each key vault also has a unique security certificate for authorization, which must be imported to each encryption node to enable the encryption node to work with the key vault. We describe the creation of these certificates for TKLM in Chapter 3, "Initial setup for the IBM Tivoli Key Lifecycle Manager and the SAN32B-E4 Encryption Switch" on page 35.

2.3.6 Key vault

The *key vault* is an appliance, or a software solution, that establishes a trusted link with the encryption device for the secure exchange of DEKs. DEKs are encrypted with the link for transit between the encryption device and appliance. At the point of destination, the DEKs are re-encrypted, using the master key that is created and maintained by processes of the key vault, and then stored. Fabric-based encryption has the following types of key vaults:

- ▶ Opaque key vault. An opaque key vault is a storage location that provides untrusted key management functionality. Its contents might be visible to a third party. DEKs in an opaque key vault are stored encrypted in a master key to protect them. We do not use any opaque key vaults in this book.
- ▶ Trusted key vault. A trusted key vault is an extremely secure type of storage, which eliminates any possibility of a third party having access to the contents without authorization. We use IBM Tivoli Key Lifecycle Manager (TKLM) with Java Cryptography

Extension Key Store (JCEKS) as the trusted key vault. When we discuss key vaults in this book, we refer to trusted key vaults.

2.3.7 Recovery cards

Recovery cards are a set of smart cards that contain a backup master key. Each recovery card holds a portion of the master key. When smart cards are used for master key backup, you have the option to split the master key write over up to five cards. These cards can be kept and stored by up to five individuals, and all the cards are needed to restore the master key. Recovery cards can be stored in separate locations, making it extremely difficult to steal the master key.

The number of cards depends on how many cards you want to define. You define the number of cards by selecting the number of the quorum size. The actual number of registered authentication cards is always one more than the quorum size, so if you set the quorum size to 2, for example, you will need to register at least three cards in the subsequent steps. The maximum quorum size is 5 and, in that case, you need six cards.

You must have Admin or SecurityAdmin user privileges to activate, register, and configure smart cards. And to use the smart cards, you must have IBM Network Analyzer installed. To get the cards ready for use, you must have a card reader that is connected and installed on the management PC where the IBM Network Analyzer server is installed.

2.4 Terminology of the encryption process

In the following topics, we introduce part of the terminology that is used in the encryption process.

2.4.1 Crypto Target Container

The *Crypto Target Container* (CTC) is a configuration of virtual devices that is created for each target port that is hosted on an IBM SAN32B-E4 or IBM SAN768/384 Encryption Blade. The container holds the configuration information for a single target, including the associated hosts and LUN settings. A CTC interfaces between the EE, external storage devices (targets), and initiators (hosts) that can access the storage devices through the target ports. Virtual devices redirect the traffic between the host and target/LUN to EEs so that they can perform cryptographic operations. Although an EE can host more than one container for each target, do *not* use this approach.

The CTC has these components:

- ▶ Initiators PWWN and NWWN
- ▶ EE worldwide name (WWN)
- ▶ Target PWWN and NWWN
- ▶ LUNs

Figure 2-2 shows a diagram of the CTCs. We defined two CTCs:

- ▶ CTC 1 has target port 1, initiator port 1, and LUN set 1 (several LUNs)
- ▶ CTC 2 has target port 2, initiator ports 1 and 2, and LUN set 2 (several LUNs)

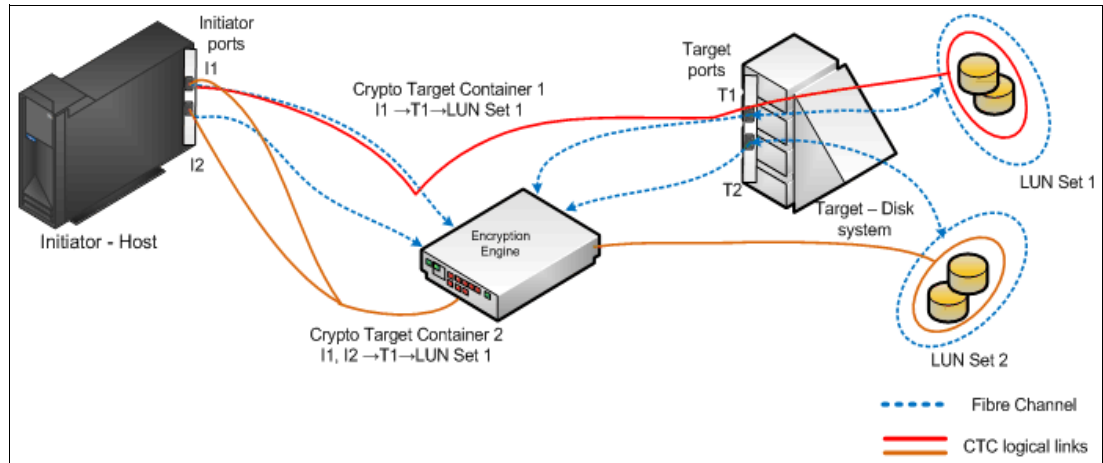


Figure 2-2 Diagram of CTCs

CTCs: One CTC is created for each target WWN, not for the LUN. The CTC can reside several LUNs from one target and multiple initiators.

2.4.2 Data encryption key

The *data encryption key* (DEK) is an encryption key that is generated by the EE. The DEK is used to encrypt cleartext that is received from a host before it is sent to a target LUN, and to decrypt that data when it is retrieved by the host. DEKs are created by the EE when CTCs are defined. Data encryption keys are stored in the key vault. Each LUN has its own DEK assigned. Its ID is compressed and written to the Master Boot Record (MBR) area of the LUN for recovery purposes. For example, if an EE has been replaced with a new EE for any reason (malfunction or any other reason), the new EE has no knowledge of the DEKs that have been used to encrypt the LUN. Immediately after the first request is made to the LUN, the EE reads the ID of the key from the MBR area of the LUN, requests the DEK from the key vault, and then continues the normal encryption process with the DEK.

Figure 2-3 shows the flow of actions with the DEK.

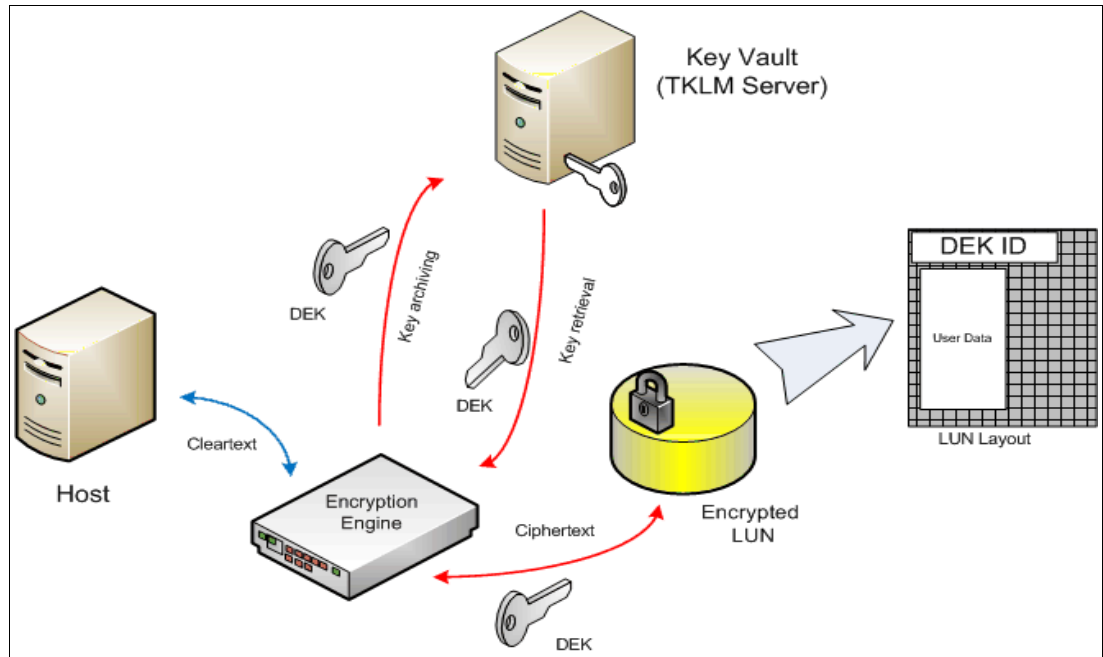


Figure 2-3 DEK in action

2.4.3 Data encryption key lifecycle management

Data encryption keys are generated by the EE. Data is encrypted and decrypted using the same DEK, so a DEK must be preserved at least long enough to decrypt the ciphertext that it created. The length of time that data is stored before it is retrieved can vary greatly, and certain data might be stored for years or decades before it is accessed. To be sure that the data remains accessible, DEKs might also need to be stored for years or decades. Key management systems provide lifecycle management for all DEKs that are created by the EE. Regardless of the length of the lifecycle, there are four stages in the life of a DEK, as shown in Figure 2-4.

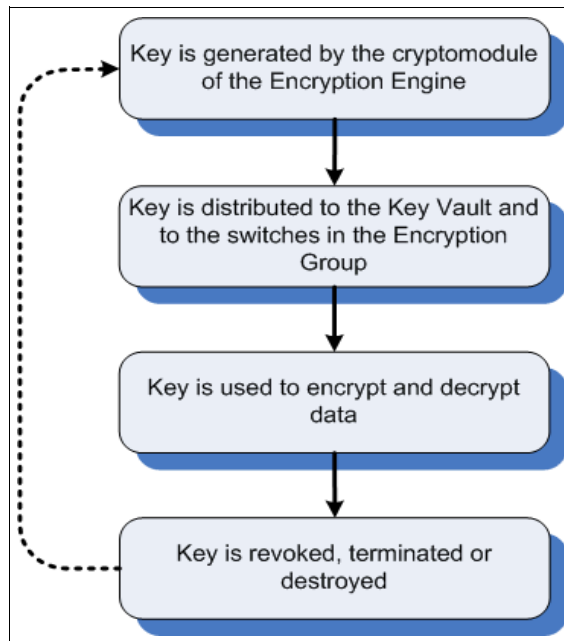


Figure 2-4 DEK lifecycle

A DEK is created by an EE, distributed, and then stored in a key vault. The key is used to encrypt and decrypt data at least once, and possibly many times. A DEK can be configured to expire in a certain time frame to avoid becoming compromised. Under those conditions, it must be used one more time to decrypt the data, and the resulting cleartext is encrypted with a new key (rekeyed).

2.4.4 First-time encryption

Each LUN that contains data to be encrypted must go through the first-time encryption process. In a first-time encryption operation, cleartext data is read from a LUN, encrypted with the current key, and written back to the same LUN at the same logical block address (LBA) location. This process effectively encrypts the LUN and is referred to as *in-place encryption*. First-time encryption can be performed under the following conditions:

- ▶ Off-line encryption: The hosts that are accessing the LUN are off-line or host I/O is halted while encryption is in process. Off-line encryption is the best way to perform first-time encryption of the LUN. The off-line encryption activity generates a number of I/Os to the disks. Therefore, it can increase response times, and applications might have performance degradation.
- ▶ Online encryption: The hosts that are accessing the LUN are online and host I/O is active during the encryption operation. This mode can be used when off-line mode is not possible. In this case, you can expect an increase in the response times of the LUN being encrypted, and you might even have denial of service for the LUN.

First-time encryption options are configured at the LUN level either during CTC configuration when the LUN is being added, or at a later time when an existing cleartext LUN needs to be encrypted. The following process describes first-time encryption:

1. Set the LUN policy to encrypt to enable encryption on the LUN. All other options that relate to encryption are enabled. A DEK is generated and associated with the LUN.
2. Enable first-time encryption by setting the **special** parameter. The existing data on the disk is encrypted using the configured DEK.

Important: At the first-time encryption of the LUN, you must specify if it has data on it or not. If you do not specify whether data exists on the LUN, the LUN is considered empty and any data that resides on the LUN will become unusable. You need to use the `-enable_encexistingdata` parameter in this case to prevent data corruption.

3. Optional: Set the options for auto rekeying, and specify the interval at which the key expires and automatic rekeying will take place (specify the time period in days).

2.4.5 Rekeying operation

Rekeying refers to decrypting data with the current DEK, and encrypting it with a new DEK. Use rekeying when the security of the current key is compromised, or when a DEK is configured to expire in a specific time frame. The rekeying operation can be used to encrypt existing data that is currently stored as cleartext. In that case, no DEK exists, and the data does not have to be decrypted before it is encrypted using the new DEK. In a rekeying operation, encrypted data on a LUN is decrypted with the current key, re-encrypted with a new key, and written back to the same LUN at the same logical block address (LBA) location. Rekeying operations can be performed under the following conditions:

- ▶ Offline rekeying: The hosts accessing the LUN are offline, or host I/O is halted.
- ▶ Online rekeying: The hosts accessing the LUN are online, and host I/O is active.
- ▶ Automatic rekeying: Rekeying options are configured at the LUN level either during initial LUN configuration in the CTC, or at a later time when the modifications are applied to the LUN. Automatic rekeying is done with the command that specifies the interval at which the key expires and automatic rekeying needs to take place (the time period in days). Enabling automatic rekeying is valid only if the LUN policy is set to encrypt.
- ▶ Manual rekeying: You can initiate a rekeying session manually at your own convenience. All EEs in a certain HA cluster, DEK cluster, or EG must be online for this operation to succeed. The manual rekeying feature is useful when the key is compromised and you want to re-encrypt existing data on the LUN before taking action on the compromised key.

2.4.6 First-time encryption and rekey operation details

First-time encryption and rekey operations are similar to each other and can be treated as one in terms of a detailed explanation. A maximum of 10 concurrent rekey sessions is supported per EG, with a maximum of 10 concurrent rekey/encryption sessions per target container and 10 concurrent sessions per physical initiator. If your configuration has two containers that are accessed by the same physical initiator, you cannot have more than 10 concurrent rekey or encryption sessions. This total includes both rekey (auto and manual) and first-time encryption sessions.

When scheduled rekey sessions or first-time encryption sessions exceed the maximum allowable limit, these sessions will be pending and a “temporarily out of resources” message is logged. Whenever an active rekey or first-time encryption session completes successfully, the next pending session is scheduled.

The system checks once every 15 minutes to determine if there are any pending rekey or first-time encryption sessions. If resources are available, the next session in the queue is processed. There might be up to an hour lag before the next session in the queue is processed. We advise that you do not schedule more than 10 rekey or first-time encryption sessions.

Figure 2-5 shows the process of first-time encryption or the rekey operation.

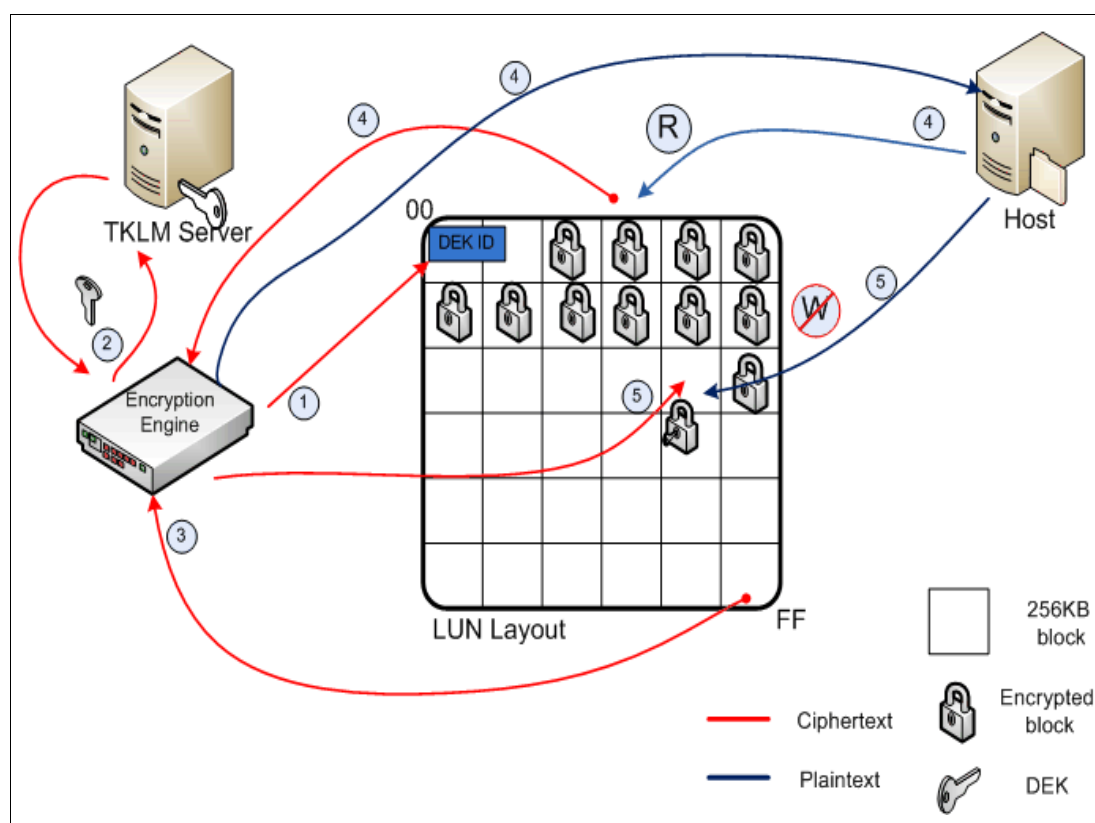


Figure 2-5 First-time encryption/rekey operation process

The following numbers correspond to the numbers that are shown in Figure 2-5:

1. The DEK is generated in the EE and the process of first-time encryption or rekeying is started. The DEK's ID is compressed and placed in the MBR area of the LUN.
2. The DEK is placed to the key vault (IBM TKLM Server).
3. The last 256 KB block of the LUN is read into the memory of the EE for data consistency reasons. The first-time encryption or rekey process continues block-by-block.
4. For example, if the host issues the read request to the already encrypted block, the request goes to the EE, which decrypts the data for that block and sends it to the host.
5. If host tries to write data to the block that is being encrypted, the host gets an "Error" reply and has to try again. The host will have no access to the block until the encryption ends.

Both operations use the AES256-XTS algorithm, which does not change the size of the data.

Performance impact of the operations

These operations use 256 KB blocks to access the data on the LUN. Because these operations are sequential processes, they can affect random reads and writes. The total performance of the operations depends on the abilities of the disk system, its utilization, and the number of the drives that make up the LUN. However, the lab tests show about 1 TB per 24 hours of rekeying operations. So, you can expect this type of performance on the disk systems with average utilization.

Tip: Use this formula to calculate the planned performance effect on the LUN:

1 TB = 1024 GB

$1024 \text{ GB} / 24 \text{ hours} / 3600 \text{ sec} / 256 \text{ KB} = 48.5 \text{ I/O per second (IOPS)}$

So, you will have approximately 50 IOPS per LUN of additional 256 KB, 50% read, 100% sequential workload. Note that your results might vary.

2.4.7 Frame redirection zone

The *frame redirection zone* or *redirection zone* is a logical instance inside the EE. It is created automatically after encryption starts, that is, after the LUN in the CTC has applied the encrypt policy. Then, a virtual target and virtual initiator are created and zoned with each other. With redirection zones, the name server sends a registered state change notification (RSCN) to both the host and target. When the host and target query the name server, the WWN of the physical device stays the same, but the port ID (PID) is replaced with the virtual initiator or target. Therefore, the zoning remains the same, and the data is redirected and encrypted by the redirection zone, as shown in Figure 2-7.

We suggest that you zone the host and target together before configuring them for encryption. Configuring a host/target pair for encryption normally creates a redirection zone to redirect the host-target traffic through the EE. But, redirection zones can only be created if the host and target are already zoned. If the host and target are not already zoned, you can still configure them for encryption. But afterward, you will need to zone the host and target together and then commit to create the redirection zones as a separate step.

2.4.8 Key encryption key

The *key encryption key* (KEK) is a key that is used to encrypt and decrypt DEKs within encryption devices so that the DEKs are transmitted in a secure manner outside of the EEs, and stored persistently inside key vaults. Because TKLM servers and encryption nodes are connected through the company's regular LAN, each DEK that is transmitted to the key vault can be captured and used for data access. To prevent such an undesirable situation, every key, which is to be transmitted to the key vault, is encrypted with the special key, which is known by the encryption node and the key vault (Figure 2-6).

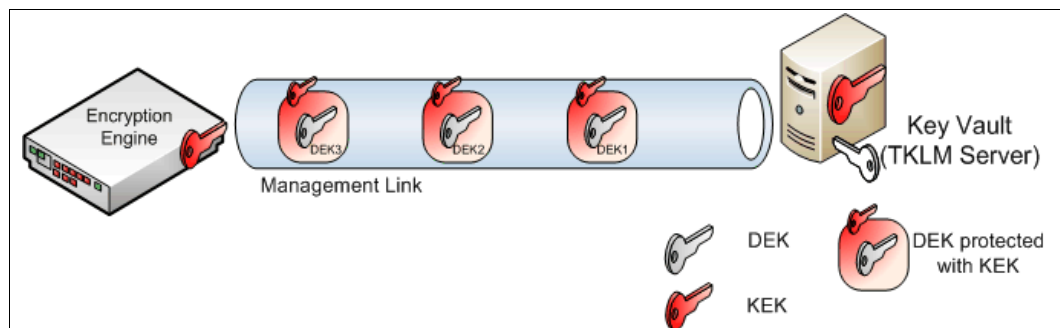


Figure 2-6 DEKs are protected with the KEK while transferred to the key vault

2.4.9 Master key

The *master key* is a key encryption key that is used to encrypt and decrypt DEKs when storing DEKs in key vaults. One master key exists per EG. Therefore, all node EEs within an EG use the same master key to encrypt and decrypt the DEKs. The master key status indicates whether a master key is used and whether it has been backed up. Encryption is not allowed until the master key has been backed up. Only the active master key can be backed up, and multiple backups are recommended. You can back up or restore the master key to the key vault, a file, or a recovery card set. A recovery card set is set of smart cards. Each recovery card holds a portion of the master key. The cards must be gathered and read together from a card reader that is attached to a PC running the Management application to restore the master key. Master keys belong to the group and are managed from Group Properties.

Active master key

The *active master key* is used to encrypt newly created data encryption keys (DEKs) prior to sending them to a key vault to be stored. You can restore the active master key under the following conditions:

- ▶ The active master key has been lost, which happens if all EEs in the group have been “zeroized” or replaced with new hardware at the same time.
- ▶ You want multiple EGs to share the same active master key. Groups must share the same master key if the groups share the same key vault and if disks are going to be exchanged regularly between the groups.

Alternate master key

The *alternate master key* is used to decrypt data encryption keys that were not encrypted with the active master key. Restore the alternate master key for the following reasons:

- ▶ To read old data that was created when the group used a separate active master key
- ▶ To read a disk from a separate EG that uses a separate active master key

Master key actions

Master keys have these actions:

- ▶ *Backup master key* is enabled any time that a master key exists. You can back up the master key to a file, key vault, or smart card. You can back up the master key multiple times to any of these media in case you forget the passphrase that you originally used to back up the master key, or if multiple administrators each need a passphrase for recovery.
- ▶ *Restore master key* is enabled when no master key exists or the previous master key has been backed up.
- ▶ *Create new master key* is enabled when no master key exists or the previous master key has been backed up.

2.4.10 Virtual targets and virtual initiators

Any given physical target port that is hosted on one SAN32B-E4 Encryption Switch or Encryption Blade is a *virtual target* (VT). If the target LUN is accessible from multiple target ports, each target port is hosted on a separate SAN32B-E4 Encryption Switch or Encryption Blade. A one-to-one mapping exists between the virtual target and physical target to the fabric whose LUNs are being enabled for cryptographic operations.

For each physical host that is configured to access a specific physical target LUN, a *virtual initiator* (VI) is generated on the SAN32B-E4 Encryption Switch or Encryption Blade that hosts the target port. If a physical host has access to multiple targets that are hosted on

separate Encryption Switches or Blades, you must configure one virtual initiator on each SAN32B-E4 Encryption Switch or Encryption Blade that is hosting one of the targets. The mapping between physical host and virtual initiator in a fabric is one-to- n , where n is the number of Encryption Switches or Blades that host targets. Figure 2-7 shows virtual targets and virtual initiators.

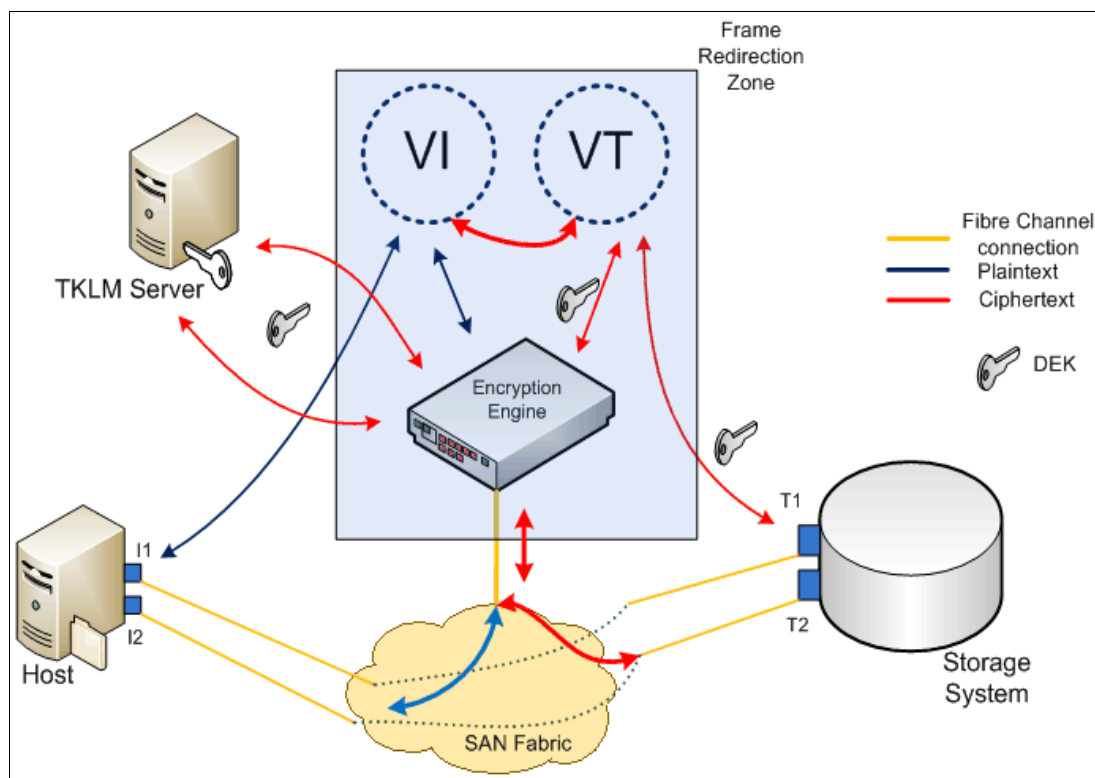


Figure 2-7 Virtual targets and virtual initiators

After the VT and VI are created, all traffic from the associated WWNs goes through the EE. The EE receives, stores, encrypts (decrypts), and then forwards the frame to the initiator. Figure 2-7 shows this process.

VT, VI, CTC, and the redirection zone create a process that is similar to the LUN masking process. You can decide which LUN is encrypted and which LUN is not while they are in one CTC (which LUN is redirected through the EE and which LUN is not). For example, if a server boots from the SAN and has LUNs with user data as well, you might choose not to encrypt the boot LUNs and have only the LUNs with data encrypted. However, all LUNs will be in the same CTC, but only selected LUNs will be encrypted.

Important: When configuring a LUN with multiple paths, you face a considerable risk of ending up with potentially catastrophic scenarios if multiple policies exist for each path of the LUN. Or, you might have a situation where one path ends up being exposed through the SAN32B-E4 Encryption Switch and another path has direct access to the device from a host outside the secured realm of the encryption platform. *The failure to follow the correct configuration procedures for multiple path LUNs results in data corruption.*

2.5 Cluster technology

In the following topics, we describe the cluster technology approaches.

2.5.1 High availability cluster configurations

A *high availability cluster* (HA or HA cluster) is a special mode of operation of two EEs that are aware of the same DEK and that have access to the same key vault. A *cluster* consists of two EEs in separate encryption nodes that are configured to host the same crypto targets and to provide active/standby failover and failback capabilities in a single fabric. Failover is automatic (not configurable) and occurs automatically, by default; however, it is configurable with a manual failback option. To be able to configure a manual failback option, two 1 Gigabit Ethernet (GbE) ports are connected to the private LAN to exchange status information and keep alive frames.

The HA cluster has these rules and limitations:

- ▶ The EEs that are part of an HA cluster must belong to the same EG and be part of the same fabric.
- ▶ An HA cluster cannot span fabrics, and it cannot provide failover/failback capability within a fabric that is transparent to host multipath I/O (MPIO) software.
- ▶ All HA cluster configuration and related operations must be performed on the group leader.
- ▶ Cluster links must be configured before creating an HA cluster.
- ▶ Configuration changes must be committed before they take effect. Any operation that is related to an HA cluster that is performed without a commit operation will not survive across switch reboots, power cycles, CP failover, or HA reboots.
- ▶ We advise that you complete the HA cluster configuration before you configure storage devices for encryption.
- ▶ It is mandatory that the two EEs in the HA cluster belong to two separate nodes for true redundancy. This rule is always the case for IBM SAN32B-E4 switches, but it is not true if two IBM SAN768/384 Encryption Blades in the same IBM SAN768/384 chassis are configured in the same HA cluster.
- ▶ In Fabric OS V6.3.0 and later releases, HA cluster creation is blocked when EEs belonging to IBM SAN768/384 Encryption Blades in the same IBM SAN768/384 chassis are specified.

HA active-standby cluster

This type of setup has only one active EE and one active CTC. As shown in Figure 2-8 on page 28 in our cluster setup, EE2 has fully replicated the configuration of EE1 and does not route traffic through itself (EE2). All traffic goes through EE1. In the case of a failover, EE2 starts to route traffic through itself.

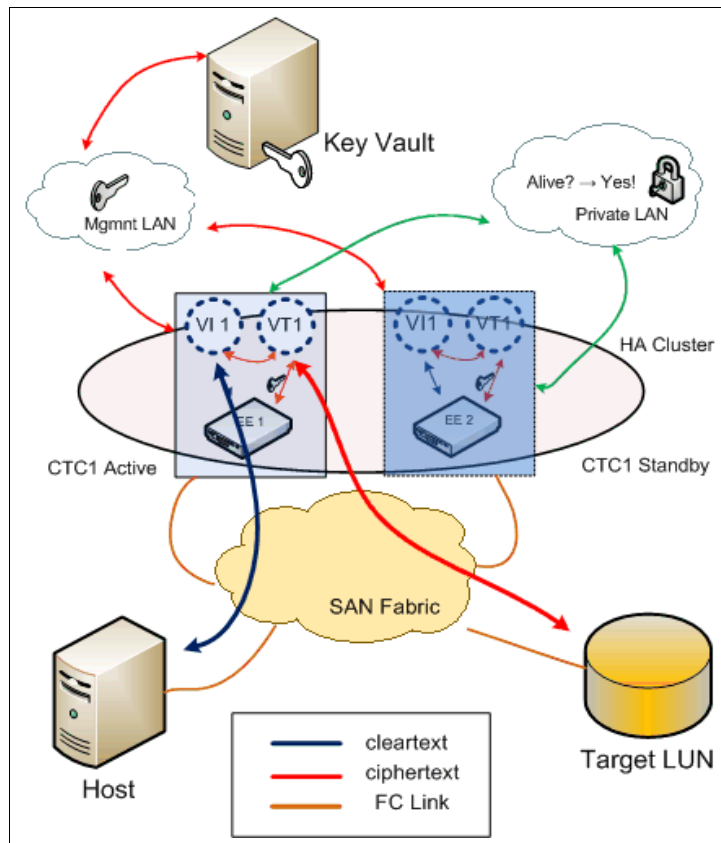


Figure 2-8 HA active-standby cluster

HA active-active cluster

This type of cluster setup is intended to use both EEs to provide access to both active CTCs. Figure 2-9 shows the cluster setup.

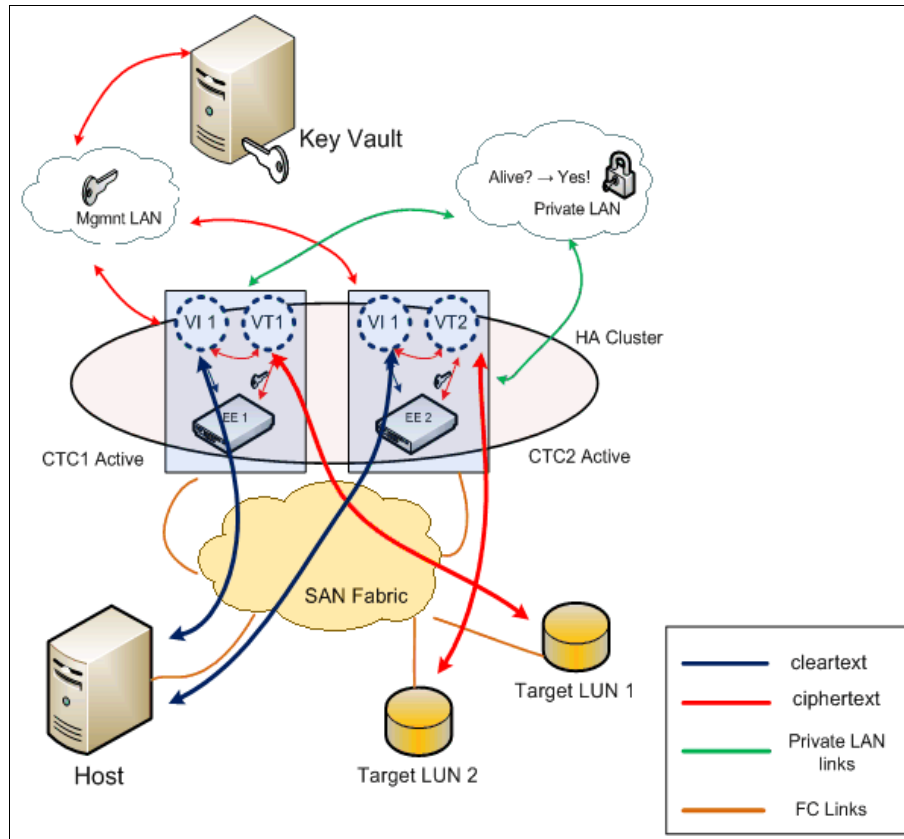


Figure 2-9 HA active-active cluster

The following conditions exist in our setup:

- ▶ EE1 hosts CTC1 and routes traffic to the LUN 1.
 - ▶ EE2 hosts CTC2 and routes traffic to the LUN 2.
 - ▶ Both engines are active and aware of each other's configuration.
 - ▶ Both engines have copies of the DEKs from each engine.
 - ▶ In the case of a failover, the CTC from the failed engine will be moved to another engine.
- For more details, see Chapter 6, "Maintenance and troubleshooting" on page 209.

2.5.2 Failover and failback of the HA cluster

In the following sections, we discuss failover and failback.

Failover

In the context of this implementation of encryption, *failover* refers to the automatic transfer of devices that are hosted by one EE to another SAN32B-E4 Encryption Switch within a high availability cluster (HA cluster). It is also true for the traffic routing through the EEs. See Figure 2-10 on page 30 and Figure 2-11 on page 31 for examples.

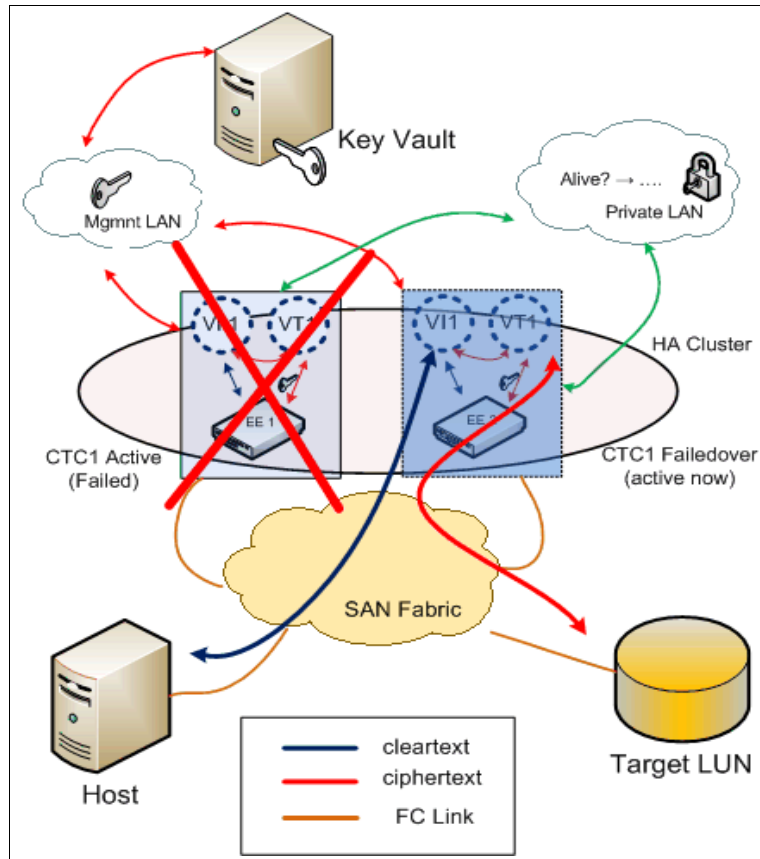


Figure 2-10 HA active-standby cluster failover

If the failover event occurs in the active-standby implementation of the HA cluster, the surviving engine activates the CTC of the failed engine, changes the Fibre Channel port IDs (PIDs) of the VI and VT, and begins to maintain the traffic. Figure 2-10 illustrates the process.

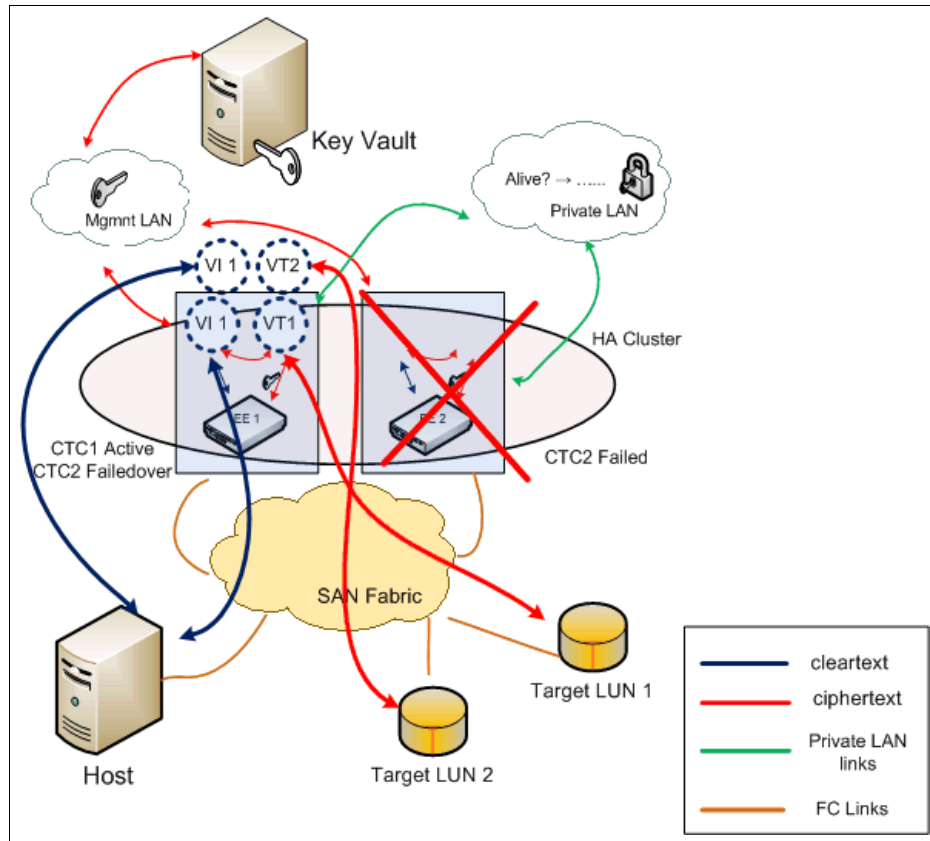


Figure 2-11 HA active-active cluster failover

If the failover event occurs in the active-active implementation of the HA cluster, the CTC from the failed engine will be activated on the surviving engine. Because both engines have the same set of DEKs, the CTC gets activated and the surviving engine starts to route traffic through itself. On Figure 2-11, the access to LUN 2 will be restored shortly. PIDs are also changed.

Important: Changing the PID might change the volume ID, which is important for IBM AIX and HP-UX operating systems. Volumes might become inaccessible in this case. To manage this situation, rediscover the volumes. Also, special actions might be required.

Failback

In this implementation of encryption, *failback* refers to the behavior after a failed SAN32B-E4 Encryption Switch recovers. Devices that were transferred to another switch by failover processing can automatically be transferred back, or they can be manually switched back. This choice is determined as a configuration option.

The failback policy can be set to automatic or manual failback mode. The automatic failback mode provides a policy where failback occurs automatically within an HA cluster when a SAN32B-E4 Encryption Switch or Encryption Blade that failed earlier has been restored or replaced. Automatic failback mode is the default mode.

The manual failback mode provides a policy where failback must be initiated manually when a SAN32B-E4 Encryption Switch or Encryption Blade that failed earlier has been restored or replaced.

To provide failover/failback operations, both engines use private network connections.

Failover/failback policy configuration

You can set the failover/failback policy parameters, as outlined in Table 2-1, for the entire EG on the group leader. Policies are automatically propagated to all member nodes in the EG.

Table 2-1 Failover/failback policy configuration

Policy name	Description
Failover policy	<p>Sets the failback mode. Valid values for failback mode are:</p> <ul style="list-style-type: none"> ► Auto: Enables automatic failback mode. Failback occurs automatically within an HA cluster when a SAN32B-E4 Encryption Switch or Encryption Blade that failed earlier has been restored or replaced. Automatic failback mode is the default. ► Manual: Enables manual failback mode. In this mode, failback must be initiated manually when a SAN32B-E4 Encryption Switch or Encryption Blade that failed earlier has been restored or replaced.
Heartbeat misses	<p>Sets the number of heartbeat misses that are allowed in a node that is part of an EG before the node is declared unreachable and the standby takes over. The default value is 3. The range is 1 - 15 in integer increments only.</p>
Heartbeat time-out	<p>Sets the time-out value for the heartbeat in seconds. The default value is 2 seconds. Valid values are integers in the range between 1 - 30 seconds.</p> <p>The relationship between -hbmisses and -hbtimeout determines the total amount of time that is allowed before a node is declared unreachable. If a switch does not sense a heartbeat within the heartbeat time-out value, it counts as a heartbeat miss. The default values result in a total time of 6 seconds (time-out value of 2 seconds times three misses). We suggest a total time of 6 - 10 seconds. A smaller value might cause a node to be declared unreachable prematurely, and a larger value might result in inefficiency.</p>

We describe the configuration of the HA cluster in Chapter 4, “Implementation scenarios and recommendations for managing the SAN32B-E4 Encryption Switch” on page 89.

2.5.3 Data encryption key cluster

The *data encryption key cluster* (DEK cluster) is a cluster of EEs that can host all paths to a LUN and share the same data encryption key (DEK) set. The EEs can be in the same or separate fabrics. DEK clusters enable host multipath I/O (MPIO) failover. See Figure 2-12 on page 33 for the layout of the DEK cluster.

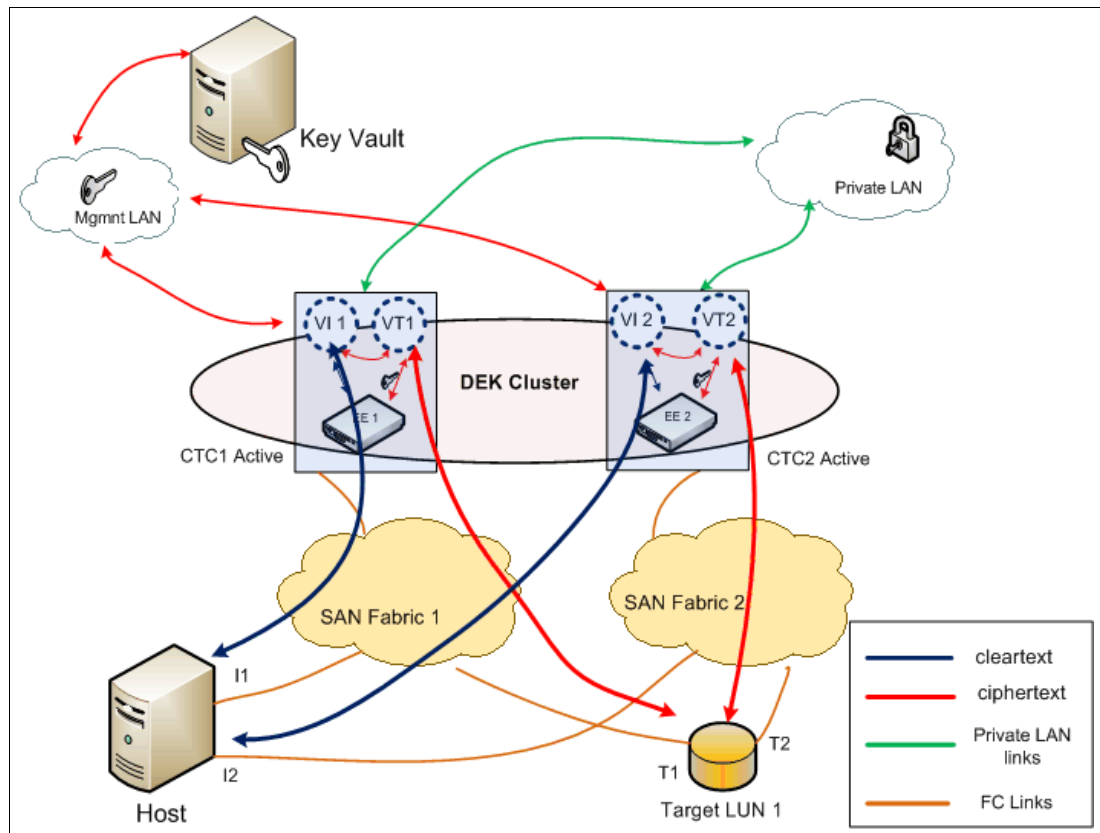


Figure 2-12 DEK cluster

Unlike an HA cluster, the DEK cluster does not provide failover/failback functionality of the EEs in terms of the resource relocation. It hosts separate CTCs related to the separate targets. You use the DEK cluster to provide MPIO functionality for the encrypted LUNs.

Figure 2-12 shows the target LUN accessed by the two paths through Fabric 1 and Fabric 2. Each fabric hosts separate CTCs, which are created for the separate target ports. This setup is typical for MPIO access to the LUN. Dual-fabric setup and MPIO protect the host if it loses one path to the LUN. We follow the same concept as in the MPIO solution: if one node of the cluster fails, we have the other node to access the encrypted LUN through the other fabric. Cluster nodes are only aware of the CTC's configuration, because it is propagated throughout the EG by the group leader. Engines in the cluster also share the same key vault.

Figure 2-13 on page 34 illustrates an assumed DEK cluster failover. If you compare it to Figure 2-12, the difference is the loss of EE2 and all connections that are associated with it. Practically, the host has lost one path to the LUN, and MPIO has worked. The overall access to the LUN remains, and data is still accessible. There was not any resource relocation from one node to another. The DEK cluster simply disconnects a failed path, and then path failure can be detected more quickly than with HA cluster. It is a good practice to have DEK cluster for MPIO functionality and HA cluster for high availability.

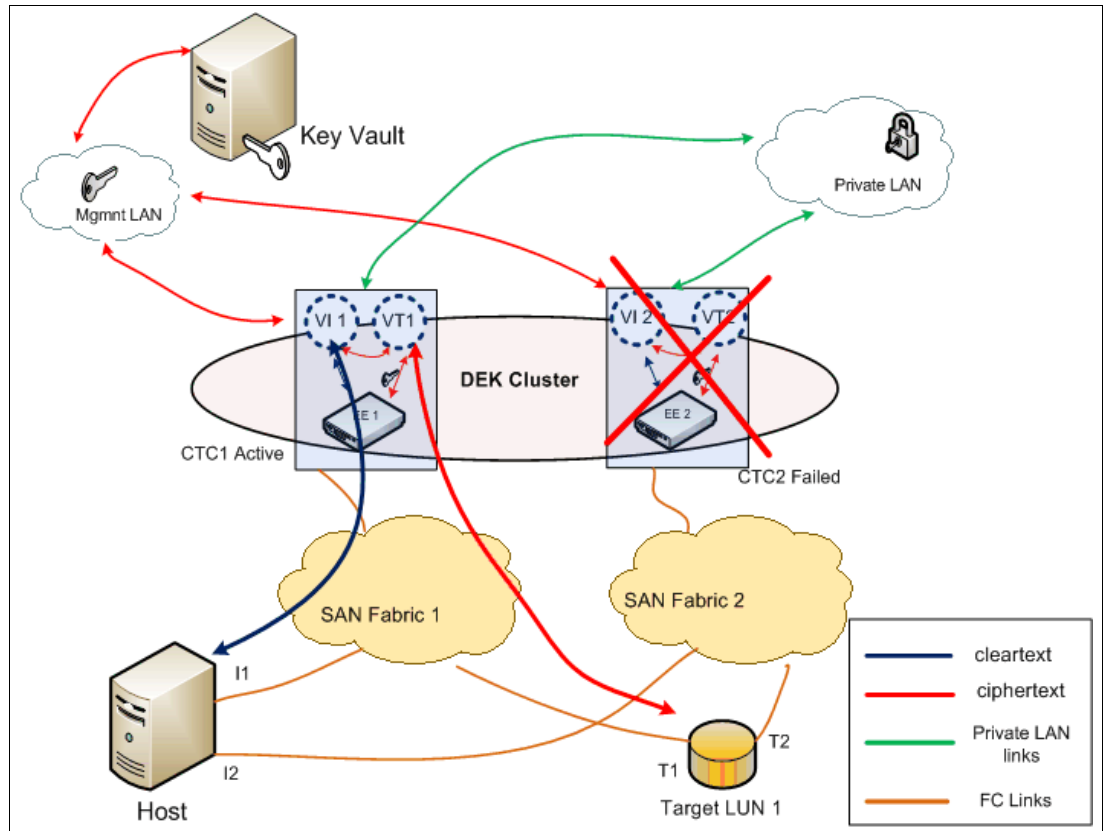


Figure 2-13 DEK cluster failover



Initial setup for the IBM Tivoli Key Lifecycle Manager and the SAN32B-E4 Encryption Switch

This chapter provides you with more information about IBM Tivoli Key Lifecycle Manager (TKLM) and the need for a centralized key management tool. We explain how TKLM and the IBM System Storage SAN32B-E4 Encryption Switch work together and how to perform the initial setup between them.

3.1 The need for a key management tool

Encrypted data depends on the security of, and the accessibility to, its encryption key or keys. The disclosure of an encryption key to an unauthorized agent (individual person or system component) creates a security exposure. The unauthorized agent will also have access to the encrypted data that was generated with the associated encryption key.

In addition, if all copies of the encryption key or keys are lost (whether intentionally or accidentally), no feasible way exists to decrypt the associated encrypted data (ciphertext). And, the data contained in this ciphertext is said to have been cryptographically erased, even if it still exists. If the only copies of certain data that exist are cryptographically erased, access to that data has been permanently lost for all practical purposes.

Therefore, the security and accessibility characteristics of encrypted data have considerations that do not exist with storage devices that do not contain encrypted data. Encryption key material must be kept secure from disclosure, or use by any agent that does not have authority to it. At the same time, it must be accessible to any agent that has both the authority and the requirement to gain access.

Two considerations are important in this context:

- **Key security**

To preserve the security of encryption keys, the implementation must ensure that no one individual (system or person) has access to the information that is required to determine the encryption key.

- **Key availability**

To preserve access to encryption keys, you can provide redundancy by having multiple independent key servers that have redundant communication paths to the encrypting devices. This approach ensures that the backup of each key server's data is maintained. The failure of any one key server or any one network will not prevent devices from obtaining access to the encryption keys that are needed to provide access to the encrypted data.

The sensitivity of possessing and maintaining encryption keys, and the complexity of managing the number of encryption keys in a typical environment, results in a client requirement for a key server. A key server is integrated with encrypting products to resolve most of the security and usability issues that are associated with key management for encrypted devices. However, you must still be sufficiently aware of how these products interact in order to provide the appropriate management of your data center environment.

Master key: Be aware that even with one or more key servers, generally at least one encryption key, which is normally called the master key (MK), must be maintained and backed up manually. For example, the MK manages access to all other encryption keys; it is a key that encrypts the data that is used by the key server to exchange keys. This copy can be electronically exported in multiple ways, for example, to a file and then burned on a CD. This exported MK also must be secured in a safe place, such as a safe deposit box in a bank.

Preferred practice: With TKLM, we suggest that you use the TKLM backup and restore functions via the Tivoli Integrated Portal. Tivoli Integrated Portal is the web-based graphical user interface (GUI) of TKLM. This mechanism will back up and restore all critical TKLM files and data. A backup file includes the keystore, configuration file, and the current database information of the specific TKLM server.

3.1.1 Why IBM Tivoli Key Lifecycle Manager

In your enterprise, a large number of symmetric keys, asymmetric keys, and certificates can exist. All of these keys and certificates have to be managed.

Tivoli Key Lifecycle Manager (TKLM) provides you with a simplified key management solution that is easy to install, deploy, and manage. TKLM allows you to create, back up, and manage the keys and certificates that your enterprise uses. Through its graphical and command-line interfaces, you can manage symmetric keys, asymmetric keys, and certificates efficiently. TKLM is an application that performs key management tasks for the following IBM encryption-enabled hardware:

- ▶ Disk systems:
 - IBM System Storage DS5000 series
 - IBM System Storage DS8000 series
- ▶ Tape drives:
 - IBM System Storage TS1140 Model E07 Tape Drive
 - IBM System Storage TS1130 Model E06 and Model EU6 tape drives
 - IBM System Storage TS1120 Model E05 Tape Drive (the tape drives that were manufactured after October 2006 and contain the label “Enc” on the rear of the drive canister)
 - IBM System Storage Linear Tape-Open (LTO) Ultrium Generation 4 and 5 tape drive
- ▶ Encryption Switch:
 - SAN32B-E4 (2498-E32)
 - FC Encryption Blades

TKLM provides, protects, stores, and maintains encryption keys that are used to encrypt information being written to, and decrypt information being read from, any of these products. Tivoli Key Lifecycle Manager V2.0 operates on the following supported operating systems:

- ▶ AIX 5.3, AIX 6.1, and AIX 7.1: 64-bit
- ▶ Red Hat Enterprise Linux (RHEL) 4.0 AS/ES x86-32
- ▶ Red Hat Enterprise Linux (RHEL) 5.0 Advanced Platform x86-32 and x86-64
- ▶ SUSE Linux Enterprise Server (SLES) 10.0 x86-32 and x86-64
- ▶ SUSE Linux Enterprise Server (SLES) 11.0 x86-32 and x86-64
- ▶ SUSE Linux Enterprise Server (SLES) 9.0 x86-32
- ▶ Solaris 9 and 10 SPARC: 64-bit
- ▶ Microsoft Windows Server 2003 R2 Enterprise Edition x86-32 and x86-64
- ▶ Microsoft Windows Server 2003 R2 Standard Edition x86-32 and x86-64
- ▶ Microsoft Windows Server 2008 Enterprise Edition x86-32 and x86-64
- ▶ Microsoft Windows Server 2008 R2 Enterprise Edition x86-64
- ▶ Microsoft Windows Server 2008 R2 Standard Edition x86-64
- ▶ Microsoft Windows Server 2008 Standard Edition x86-32 and x86-64

Go to this website for the latest releases and updates:

<http://www.ibm.com/support/docview.wss?uid=swg27020567>

Table 3-1 lists a summary of the hardware requirements for Windows server systems that are required to run the distributed Tivoli Key Lifecycle Manager. For more details and other platforms, see the *IBM Tivoli Key Lifecycle Manager Version 2 Release 0 Installation and Configuration Guide*, SC27-2741-00, which can be downloaded from this website:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tk1m.doc_2.0/pdf/tk20_installguide.pdf1m.doc/welcome.htm

Table 3-1 TKLM hardware requirements for a Microsoft Windows Server system

System components	Minimum values*	Typical values**
System memory (RAM)	4 GB	4 GB
Processor speed	2.66 GHz single processor	3.0 GHz dual processors
Required disk space	5.2 GB	11.4 GB
<p>All file systems must be writable.</p> <p>* Minimum values: These values enable a basic use of Tivoli Key Lifecycle Manager.</p> <p>** Typical values: You might need to use larger values that are appropriate for your production environment. The most critical requirements are to provide adequate system memory, and free disk and swap space. Processor speed is less important. Installing into mapped network drives is not supported. If the installation locations of more than one system component fall on the same Windows drive, the cumulative space that is required to contain all those components must be available in that drive.</p>		

TKLM for IBM z/OS® was replaced by the software IBM Security Key Lifecycle Manager for z/OS, which was made available on 15 April 2011. Security Key Lifecycle Manager is the latest IBM key manager for z/OS. This new version removes the dependency on the System Shell Runtime Environment (SSRE) and a customized IBM DB2® database. However, IBM currently has no support for using Security Key Lifecycle Manager together with the SAN32B-E4 Encryption Switch.

Support is also available for virtualized server environments, especially for AIX, as shown in Table 3-2, and for Windows, as shown in Table 3-3.

Table 3-2 Supported virtualized server environments for TKLM under AIX

Operating system	Product	Applicable OS	Notes
AIX	IBM PowerVM® Hypervisor (logical partition (LPAR), dynamic LPAR (DPAR), and micro-partition): any supported version	AIX Version 6.1 (IBM POWER® System), AIX Version 5.3 (POWER System), and AIX Version 7.1 (POWER System)	Support is available for Power5, IBM POWER6®, and IBM POWER7®

Table 3-3 Supported virtualized server environments for TKLM under Windows

Operation system	Product	Applicable OS
Windows	Red Hat Enterprise Virtualization Hypervisor (RHEV-H)/KVM 5.4	All applicable Windows OS
	VMware ESX 4.0	All applicable Windows OS

Important: The IBM System Storage SAN32B-E4 Encryption Switch (2498-E32) and IBM FC Encryption Blades must be used with TKLM V2.0 or later.

Preferred practice: For the implementations and examples in this book, we used TKLM Version 2.0.0.2 and the latest fix pack, which is Fix Pack 2 of TKLM. Our TKLM servers were installed under VMware clients running Microsoft Windows Server 2008 R2 Enterprise Edition. Running TKLM servers on VMware clients is advantageous, because it is easy to add an additional server or go back to a clean starting installation that you saved initially as a VM image. But, in terms of high availability, it is more likely that you will install each TKLM on its own physical hardware platform, for example, an INTEL-based server and a supported Windows version.

TKLM is designed to be a shared resource that is deployed in several locations within an enterprise. It is capable of serving numerous IBM encryption-enabled hardware devices, regardless of where those devices reside. The communication and key exchange between TKLM and the devices is typically done via standard IP communication protocol (out-of-band-managed keys). Nevertheless, certain devices support key exchange within the storage communication protocol, for example, native attached tape drives to System z via the IBM FICON® protocol. This method of exchanging keys is called the *in-band* exchange of keys. This specific architecture of key exchange is not part of the examples that are covered in this book.

TKLM provides the following functions:

- ▶ Key serving with lifecycle management using a GUI, such as Tivoli Integrated Portal and a command-line interface (CLI)
- ▶ Support for the new IBM SAN Encryption Switch products SAN32B-E4 (2498-E32) and IBM FC Encryption Blades
- ▶ Support for encryption-enabled IBM System Storage TS1100 family tape drives (3592 tape drives)
- ▶ Support for IBM Systems Storage Linear Tape-Open (LTO) Ultrium Generation 4 and 5 tape drives
- ▶ Support for the IBM System Storage DS8000/DS5000 disk series
- ▶ Backup and recovery to protect your keys and certificates
- ▶ Notification of the expiration of certificates
- ▶ Audit records to allow you to track the encryption of your data
- ▶ Automatic rollover of key groups and certificates (This capability applies to 3592 and LTO drives; it does not apply to DS8000/DS5000 series.)
- ▶ Key lifecycle management function that allows a user to define when to use a new key group with LTO tape drives or new certificates with TS1100 tape drives

3.2 Tivoli Key Lifecycle Manager components and resources

TKLM does not perform any cryptographic operations, such as generating encryption keys, and it does not provide storage for keys and certificates. The major purpose for TKLM is to serve and manage the keys of an enterprise.

In addition to the key-serving function, TKLM also performs the following functions, which can be used for IBM encryption-enabled devices:

- ▶ Lifecycle functions:
 - Notification of certificate expiration through Tivoli Integrated Portal (TIP)
 - Automated rotation of certificates
 - Automated rotation of groups of keys
 - ▶ Usability features:
 - Web-based GUI
 - Initial configuration wizards
 - Migration wizards
 - ▶ Integrated backup and restore of TKLM files
- Only one click in the Tivoli Integrated Portal is required to create and restore a single backup, which is packaged as a .jar file.

To perform all these tasks, TKLM relies on several more external components. The TKLM solution includes the TKLM server, an IBM embedded WebSphere® Application Server, and a database server (IBM DB2). The solution also incorporates the Tivoli Integrated Portal installation manager, which provides an easy to use installation for Windows, Linux, AIX, and Solaris operation systems.

In Tivoli Key Lifecycle Manager, the drive table, key group, and metadata are all kept in DB2 tables. The TKLM DB2 tables enable the user to search and query that information more easily. Note that the keystore, configuration file, audit log, and debug log are still flat files.

TKLM relies on the following resources:

- ▶ Configuration file
- TKLM has an editable configuration file with additional configuration parameters that is not offered in the GUI. The file can be text-edited, but the preferred method is modifying the file through the TKLM CLI.
- ▶ Java security keystore
- The keystore is defined as part of the Java Cryptography Extension (JCE) and an element of the Java Security components, which are, in turn, part of the Java Runtime Environment. A *keystore* holds the certificates and keys (or pointers to the certificates and keys) that are used by TKLM to perform cryptographic operations. A keystore can be either hardware-based or software-based.
- The TKLM product family supports several types of Java keystores, offering a variety of operational characteristics to meet your needs. TKLM on open systems supports the JCEKS keystore (JCEKS). This software-based keystore supports both CLEAR key symmetric keys, and CLEAR key asymmetric keys. Table 3-4 on page 41 shows you a summary of the current supported keystore types for TKLM and Table 3-5 on page 41 lists the supported key sizes for each keystore type that TKLM supports.

Table 3-4 Summary of supported keystore types

Keystore	Operating system	TS1100 and DS8000 (store keypairs and certificates)	LTO and BRCD_ENCRYPTOR (store symmetric keys)	DS5000 and ONESECURE (store symmetric keys)	TS1100, DS8000, and LTO
JCEKS	All	✓	✓	✓	✓

Table 3-5 Supported key sizes and keystore types

Keystore type	Import PKCS#12 file	Export PKCS#12 file	Key generation size in bits
JCEKS	✓	✓	2048

TKLM can serve either 2048-bit or 1024-bit keys to devices. You can continue to use older keys that were generated as 1024-bit keys.

► Cryptographic services

TKLM uses the IBM Java Security components for its cryptographic capabilities. TKLM does not provide cryptographic capabilities. Therefore, it does not require and is not allowed to obtain FIPS 140-2 certification. However, TKLM takes advantage of the cryptographic capabilities of the IBM Java Virtual Machine (JVM) in the IBM Java Cryptographic Extension (JCE) component. TKLM allows the selection and use of the IBMJCEFIPS cryptographic provider, which has a FIPS 140-2 level 1 certification.

Ready to use: After the successful installation of all components and resources, you can start working with TKLM immediately.

For more details about Tivoli Key Lifecycle Manager, installation planning, installation, and configuration, see the *IBM Tivoli Key Lifecycle Manager Version 2 Release 0 Installation and Configuration Guide*, SC27-2741-00, which can be downloaded from this website:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tk1m.doc_2.0/pdf/tk20_installguide.pdf

Also, the IBM Tivoli Key Lifecycle Manager Information Center contains helpful information describing the TKLM product and features. It also will help you with basic, as well as advanced, tasks and provides documentation in PDF files:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp>

3.3 Initial setup of Tivoli Key Lifecycle Manager and the Encryption Switch

In this topic, we describe the initial setup of TKLM and the Encryption Switch.

3.3.1 Basic installation of Tivoli Key Lifecycle Manager

TKLM can run on various operating systems (see the list on page 37). Depending on the operating systems, there are many installation scenarios available. It is beyond the scope of

this book to cover all those scenarios, and we do not cover the software installation of TKLM here. For more details about the Tivoli Key Lifecycle Manager, installation planning, installation, and configuration, see the *IBM Tivoli Key Lifecycle Manager Version 2 Release 0 Installation and Configuration Guide*, SC27-2741-00.

In our scenario, we installed the TKLM servers under VMware clients running Microsoft Windows Server 2008 R2 Enterprise Edition.

When the installation of the TKLM software is complete, you can continue to set up the Encryption Switch and the TKLM servers to communicate to handle the keys for the IBM Encryption Switches. *Do not continue* with the setup and initial encryption enablement on the Encryption Switches in Chapter 4, “Implementation scenarios and recommendations for managing the SAN32B-E4 Encryption Switch” on page 89 until TKLM and the IBM Encryption Switches can communicate successfully with each other.

Certain one-time steps must be performed during the configuration of the TKLM servers, and there are steps that must be performed every time that you configure a new client, or when a new client certificate is generated. A new client, for example, might be a newly added SAN32B-E4 Encryption Switch.

Important: Make sure that the time setting on the TKLM server is the same as the time setting on the IBM Encryption Switch or on the chassis of the Encryption Blade. A variation between the two time settings can cause the Transport Layer Security (TLS) connectivity to fail. The switch will fail with the state: Failed authentication.

Preferred practice: Use a Network Time Protocol (NTP) server to keep the date and time synchronized across the entire enterprise and all components.

3.3.2 Multiple Tivoli Key Lifecycle Managers for redundancy

For reasons of redundancy and availability, we highly advise that you run more than one TKLM instance. At least a minimum of two independent, but duplicate TKLM servers must be in place.

In order to run duplicate Tivoli Key Lifecycle Managers for redundancy and availability, you must keep the two (or more) Tivoli Key Lifecycle Managers and their content synchronized. The keys, certificates, TKLM database, and configuration files must be the same on all Tivoli Key Lifecycle Managers.

In addition, remember that TKLM is not cluster-aware and not enabled for automatic failover. Therefore, Tivoli Key Lifecycle Managers cannot share keystores or configuration files itself, and so, multiple Tivoli Key Lifecycle Managers cannot be configured for performance or automatic failover. However, performance is not a major concern. The IP traffic to TKLM is generally not extremely high volume. The IP traffic is used mainly to provide the data keys and for web browser access to the TKLM server (which, after TKLM is configured, will be infrequent).

Therefore, a primary/secondary TKLM server configuration is not a failover or clustered server from a TKLM point of view.

The redundancy typically is managed by setting up multiple independent TKLM server destinations (their IP addresses) at the separate storage subsystems, for example, in a DS8000 System Storage Disk subsystem or an IBM TS3500 Enterprise Tape-Library, both of

which are out-of-band-managed devices. Those IBM storage systems can handle typically up to four separate IP addresses to four separate TKLM servers.

In those cases, synchronization between the TKLM servers was typically achieved by backing up one (the first) TKLM server's repository after the initial setup or after making changes and restoring this backup configuration on the other TKLM servers. This approach assumes that all Tivoli Key Lifecycle Managers are on the same OS platform. In certain publications and other IBM Redbooks publications, you might find this method of synchronizing separate TKLM servers as a preferred practice.

Important: Do not use the method of TKLM backup/restore for synchronization between multiple TKLM servers when IBM Encryption Switches will be managed and served. We will explain later in this section why not to use this method, and we suggest a method. Refer also to “Master key backup” on page 47, 3.3.6, “Setting up the SAN32B-E4 Encryption Switch and TKLM using the switch CLI” on page 50, and “Step 17: Importing the TKLM server certificates on the GL node” on page 67).

Another general method of synchronization between TKLMs is through the export/import functions of TKLM. You must use this function regardless if multiple TKLM servers are on dissimilar platforms, because synchronization via backup/restore only works between the same operating system platforms. For any other action in this case, such as making configuration changes, perform the action manually on each TKLM server.

Nevertheless, plan to perform this standard TKLM backup process on every TKLM server for a general backup, when the following events take place. The following events change the keystore or configuration files within TKLM:

- ▶ After initial configuration
- ▶ After adding keys or devices
- ▶ After changing the key or certificate replacement intervals
- ▶ After making a certificate authority (CA) request
- ▶ After upgrading TKLM or its middleware, such as DB2

TKLM shows an alert when you make any change that requires making a new backup. On Windows systems and other systems, such as Linux or AIX, to use backup/restore, the computers must have the required memory, speed, and available disk space to meet this job requirement.

New communication path with the TKLM server and SAN32B-E4 Encryption Switch

TKLM support of the SAN32B-E4 Encryption Switch has a significant change in the communication path to and from TKLM. A native disk-encryption or tape-encryption solution uses and queries keys and certificates that previously were entered into the TKLM server by the TKLM administrator.

The SAN32B-E4 Encryption Switch behaves differently. Whenever a new LUN (volume from the disk storage subsystem) is added to a Crypto Target Container (CTC) within a SAN32B-E4 Encryption Switch, the encryption engine (EE) in the Encryption Switch generates a new data key (DEK) and a new key ID (key label). The key label is compressed and stored within the Master Boot Record (MBR) of the LUN, and the new DEK will be sent encrypted by using the MK to the TKLM server to be stored there automatically “in-flight”. In this case, the MK acts as a key encryption key (KEK). Within this automatic key exchange, additional read-verification is performed between the EE and TKLM before the new DEK is actually used for data encryption operations (see Figure 3-1 on page 45).

DEKs: The DEKs are generated by the SAN32B-E4 Encryption Switch (or EE). A DEK is generated per LUN from the EE. These DEKs are sent encrypted using the EE's MK from the SAN32B-E4 Encryption Switch over the IP network to the TKLM servers to be stored there automatically in-flight.

Data is encrypted and decrypted using the same DEK, so a DEK must be preserved at least long enough to decrypt the ciphertext that it created. The length of time that data is stored before it is retrieved can vary greatly, and certain data might be stored for years or decades before it is accessed. To be sure that the data remains accessible, DEKs might also need to be stored for years or decades. TKLM provides lifecycle management for all DEKs that are created by the SAN32B-E4 Encryption Switch (or EE). There are also rekey functions available if you need to change the DEK later because of legal requirements or for other security reasons.

Important: A SAN32B-E4 Encryption Switch solution will send and store its keys automatically to the TKLM servers. This solution differs greatly from the existing methods with native tape and disk encryption where the keys and certificates are stored or created by the administrator within the TKLM server.

The concept of the SAN32B-E4 Encryption Switch interacting with multiple TKLM servers differs from the method that is used by disk or tape encryption solutions. Those solutions can have multiple (typically, up to four) TKLM server IP addresses set up in their storage subsystem configurations. The storage subsystems decide during runtime with which TKLM server they actually communicate if a key exchange is needed. The disk or tape encryption solutions assume that every TKLM server is identical in configuration and content.

The SAN32B-E4 Encryption Switch can handle two TKLM server instances, and these server instances are named *key vaults*. They are used as the primary and secondary key vaults. Whenever a new LUN is added to a CTC as a candidate for encryption and a new DEK is generated, the new DEK is sent simultaneously to both TKLM servers (the primary and the secondary). Figure 3-1 on page 45 shows the steps for the exchange of a new DEK creation in the case of a disk subsystem encryption.

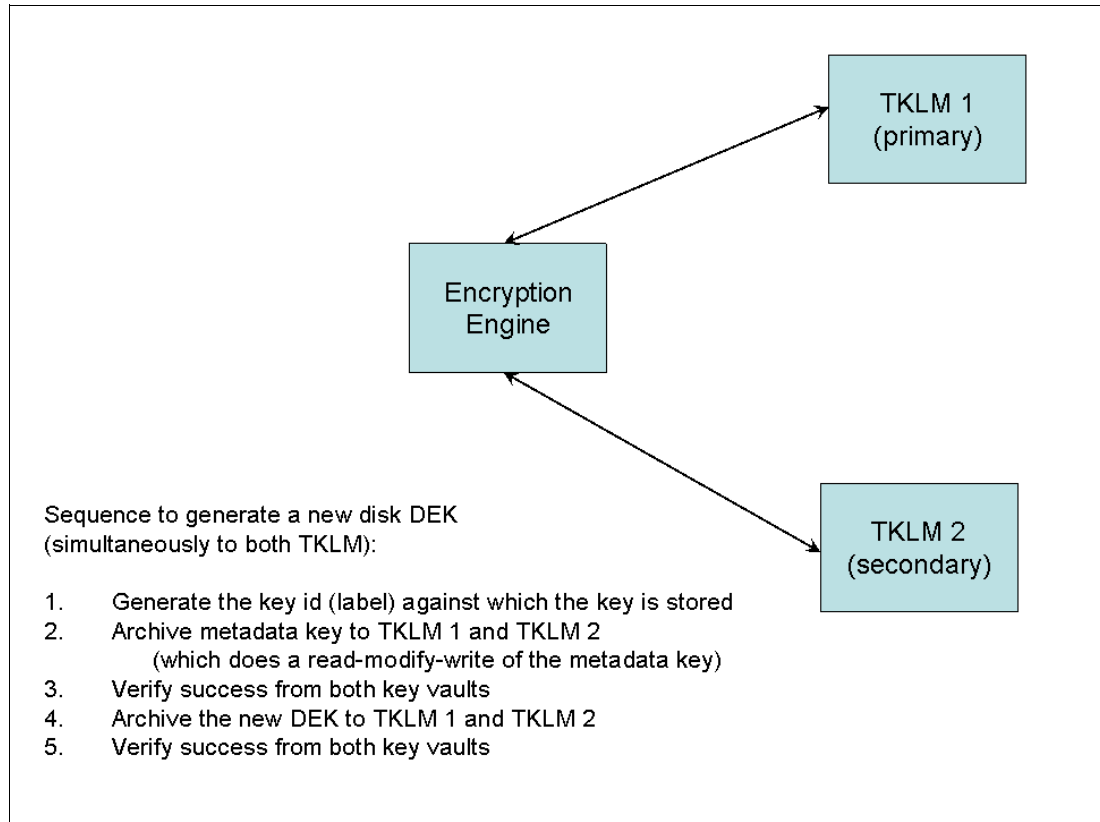


Figure 3-1 Sequence of TKLM communication to generate a new disk DEK

If you use multiple EEs, for example, as an HA cluster or DEK cluster, the EE that owns the CTC when you provisioned the corresponding LUN will perform both TKLM server updates. That is, the EE does not act as the group leader (GL).

Note: The GL's only role is to ensure that every member node is ready and available within the encryption group (EG). When you make any configuration changes, the GL will ensure that the configuration change is distributed to the group. The GL role can move around to other member nodes, for example, after reboots of EEs. This role has no other function; it only ensures the consistency of the configuration.

Tip: If you make configuration changes via the CLI, you always must make them on the GL. The GUI handles this requirement automatically, but if you use the CLI and you are on the wrong EE (not the GL), the CLI command response will let you know.

Important: If, in a dual-TKLM server setup, one TKLM server is down, do not perform LUN provisioning, because the unavailable TKLM server will miss any newly distributed DEKs that are sent from the EEs. It is possible to continue, but you need to export those newly generated DEKs at a later time from the running TKLM and import them to the TKLM that was down by using the `wsadmin` tool.

Before the TKLM servers can talk to the SAN32B-E4 Encryption Switch and vice versa, both parties need to prove to each other that they are the correct system to which to talk. Also, they need to prove that they are not a third party in the middle that is trying to steal keys while they are being exchanged, which is known as a “*man in the middle attack*”. Therefore, the TKLM

servers and the IBM Encryption Switches need to prove their identities. The identities are verified through an exchange of self-signed certificates between systems: TKLM Server to Encryption Switch, and Encryption Switch to TKLM. The self-signed certificates are the KAC and certificate authority (CA) certificates. In 3.3.6, “Setting up the SAN32B-E4 Encryption Switch and TKLM using the switch CLI” on page 50, we describe this process step-by-step.

Summary: For our implementation, we configured both TKLM servers in parallel. Therefore, we did not need (at the end of the TKLM/Encryption Switch setup) to make a backup/restore to get the first TKLM server to be identical to the second server after the initial configuration.

3.3.3 Installation of the Encryption Switch

The SAN32B-E4 Encryption Switch is a stand-alone SAN Encryption Switch that can be connected to a fabric using inter-switch link (ISL) links and trunks. You can configure this switch as a remote stand-alone Fibre Channel switch with full encryption for Data-at-Rest. The SAN768B and SAN384B Encryption Blades are managed as any other blade within the director. In our example, we set up a SAN32B-E4.

When connecting a SAN32B-E4 within a larger fabric, only use the switch ports on the SAN32B-E4 for ISL trunk links to provide the maximum bandwidth for encryption. For our simple installation of the SAN32B-E4 Encryption Switch, we assume that all Encryption Switches that will be configured are physically installed and powered on. We also assume that the Management Interface IP address is set and that we log on to the CLI.

Tip: If available, use the optional management software, Data Center Fabric Manager (DCFM). This GUI might be easier to handle in certain cases and more user-error tolerant. However, for general understanding of how the setup process works, use the CLI, which provides more information as you perform the setup process.

Also, we assume that if more than one Encryption Switch is used, for example, to form a DEK cluster or an HA solution, that both 1 Gbps IP links are set. Specifically, we assume that the 1 Gbps IP links, between all the switches forming an EG, are physically installed, connected, and that the virtual IP addresses are set. Each EE has a single virtual IP address configured on its 2 Gb Ethernet ports for this cluster interconnect, especially for the HA cluster failover/failback detection. In a DEK cluster solution, these links are utilized for rekey state synchronization across cluster member EEs and for DEK distribution across cluster member EEs.

Setting this virtual IP address requires a switch reboot. Even it is technically only needed when more than one EE will be present, the preferred practice is to always set this virtual IP address to prevent future switch reboots when a second EE is added. Example 3-1 on page 47 shows you how to set up this virtual IP address. Do not forget to reboot the switch after setting this address.

Tip: For more general information about the installation planning and installation of the SAN32B-E4 Encryption Switch, see *Implementing the IBM System Storage SAN32B-E4 Encryption Switch*, SG24-7922, or the *Fabric OS Encryption Administrator's Guide, Supporting Tivoli Key Lifecycle Manager (TKLM) Environments and Supporting Fabric OS v7.0.0*, 53-1002162-02.

Example 3-1 CLI command to set the EE's virtual IP address

```
SAN32B-E4-1:root> ipaddrset -eth0 --add 192.168.0.120/24  
IP address is being changed...Done.
```

Fabric operating system (FOS): The IBM System Storage SAN32B-E4 Encryption Switch (2498-E32) and Encryption Blades that we used in our implementation were at FOS level v7.0.0a.

3.3.4 Master key management

In the following topics, we describe master key (MK) management.

Master key generation

An MK must be generated by the GL encryption engine (EE). This MK can be generated once by the GL and then propagated to the other members of an EG.

Master key backup

It is essential to back up the MK immediately after it is generated. You can back up the MK to any of the following locations or devices:

- ▶ A plain file as an encrypted key, which later can be stored on any type of media, such as USB, CD, and so on.
- ▶ The key management system (Tivoli Key Lifecycle Manager) as an encrypted key record within the keystore.
- ▶ A set of recovery smart cards. This option is only available if the switch is managed by the DFCM, and if a card reader is available for attachment to the management workstation.

The use of smart cards provides the highest level of security. When smart cards are used, the key is split and written on up to 10 cards. Each card can be kept and stored by a separate individual. A quorum of key holders is necessary to restore the key. If five key holders exist and the quorum is set to three, any three of the five key holders are needed to restore the key.

Preferred practice: When running two TKLM servers, this MK can be exported to those TKLM instances and backed up with the TKLM backup/restore function. Ensure that you store a copy of the TKLM backup somewhere else, for example, in a safe deposit box in a bank.

Important: You must register both the primary and secondary key vaults before exporting the MK or encrypting LUNs. If the secondary key vault is registered after encryption has been performed for part of the LUNs, the key database of the primary TKLM server must be backed up and restored on the secondary TKLM server before registering the secondary TKLM server on the switch.

3.3.5 Considerations before the first TKLM and SAN32B-E4 Encryption Switch setup

You must check the date and time on the switch and compare it with the time on your TKLM servers. During the writing of this book, we received various errors if the time of the TKLM servers and the time of the EEs were not synchronized. Also, be aware of the local time zones that you might need to configure in each component.

Important: It is best to use an NTP server to keep date and time in synchronization across the entire enterprise (especially the encryption nodes and the TKLM servers). If you do not have an NTP server in your company, you might consider using an externally provided NTP server from the Internet. For example, go to <http://www.pool.ntp.org>. Also, refer to 4.2.2, “Synchronizing switch times” on page 96 to see commands to configure an NTP server within the SAN32B-E4 Encryption Switch. For details about how to set up the NTP server on your TKLM server, check the relevant documentation of the operating system that you use on the TKLM servers.

The SAN32B-E4 Encryption Switch or IBM Encryption Blade can be managed by either the CLI or the GUI using DCFM. We show the setup of the Encryption Switches using the CLI. You must perform a series of steps to initialize the SAN32B-E4 Encryption Switch or IBM Encryption Blade that is expected to perform encryption as a GL within the fabric. You perform all configuration on the switch using the **cryptocfg** CLI command.

User IDs: To configure the switch, you have to log on to it as the Admin user or the SecurityAdmin user. Both user IDs have the necessary permissions to use the setup commands. Certain switch commands require the Root user. We point out when the Root user is needed.

To get help about the command syntax, use **cryptocfg --help**, as shown in Example 3-2.

Example 3-2 The cryptocfg --help command

```
SAN32B-E4-1:admin> cryptocfg --help
Usage: cryptocfg
--help -nodecfg:
    Display the synopsis of node parameter configuration.
--help -groupcfg:
    Display the synopsis of group parameter configuration.
--help -hacluster:
    Display the synopsis of hacluster parameter configuration.
--help -devicecfg:
    Display the synopsis of device container parameter configuration.
--help -transcfg:
    Display the synopsis of transaction management.
--help -decommission:
    Display the synopsis of decommissioning a lun.
```

To get more help about other parameters, for example, issue **cryptocfg --help --nodecfg**, as shown in Example 3-3.

Example 3-3 The cryptocfg --help -nodecfg command

```
SAN32B-E4-1:admin> cryptocfg --help -nodecfg
Usage: cryptocfg
--help -nodecfg:
    Display the synopsis of node parameter configuration.
--initnode:
    Initialize the node for configuration of encryption options.
--initEE [<slotnumber>]:
    Initialize the specified encryption engine.
--regEE [<slotnumber>]:
    Register a previously initialized encryption blade.
```

```

--setEE [<slotnumber>] -routing <shared | partitioned>:
    Set encryption routing policy.
--reg -membnode <member node WWN> <member node certfile> <IP addr>:
    Register a member node with the system.
--reg -groupleader <group leader WWN> <group leader certfile> <IP addr>:
    Register a group leader node with the system.
--dereg -membnode <member node WWN>:
    Deregister a member node with the system.
--reg -systemcard [<slotnumber>] <cert label> <certfile>:
    Register a system card with the system.
--dereg -systemcard [<slotnumber>]:
    Deregister a system card with the system.
--dhchallenge <vault IP addr>:
    Generates the Diffie-Hellman challenge passed from the
    EE to the specified NetApp LKM.
--dhresponse <vault IP addr>:
    Accepts the LKM Diffie-Hellman response from the NetApp
    LKM and generates the Link Key on the specified EE.
--disableEE [<slotnumber>]:
    This command reboots the Mace switch or power cycle the Lance blade in
case
of chassis system.
--enableEE [<slotnumber>]:
    Enables the system to perform encryption.
--zeroizeEE [<slotnumber>]:
    Zeroize all critical security parameter.
--import -scp <local name> <host IP> <host username> <host path>:
    Import a file from an external host via scp.
--import -usb <dest filename> <source filename>:
    Import a file from a mounted USB storage device.
--export -scp [-dhchallenge <vault IP addr> | -currentMK | -KACcert | -KACcsr |
-CPcert] <host IP> <host username> <host path>:
    Export a specified file to an external host via scp.
--export -usb [-dhchallenge <vault IP addr> | -currentMK | -KACcert | -KACcsr |
-CPcert] <dest filename>:
    Export a specified file to a mounted USB storage device.
--delete -file <file name>:
    Delete a file previously imported to the switch.
--show -nodecerts:
    Display all authorization lists certificates for Cluster Members,
    Key Vaults, CP certificate and local EE certificates.
--show -file -all:
    Display the files that are imported or to be exported.
--show -localEE:
    Display status of EEs on the local node.
--rebalance [slot_num]:
    Rebalance containers and initiators per EE

```

3.3.6 Setting up the SAN32B-E4 Encryption Switch and TKLM using the switch CLI

In our example, we set up one IBM SAN32B-E4 Encryption Switch and two TKLM servers. The focus in this part of the setup is to register all the components to each other. Typically, this registration is done by exchanging certificates. Within the following steps, various certificates are exchanged and registered in the various components:

- ▶ *CP or CPCert certificates*: These CP certificates register all other encryption member nodes to the encryption GL node. This process only affects the SAN32B-E4 Encryption Switch setup.
- ▶ *TKLM self-signed server certificates*: These TKLM self-signed server certificates register each TKLM server to the EE's GL. The EE GL also registers these TKLM self-signed server certificates in all the encryption member nodes.
- ▶ *KAC certificates*: These certificates are provided by each encryption node (the members and the GL).

In Figure 3-2, we show where these types of certificates are generated and registered.

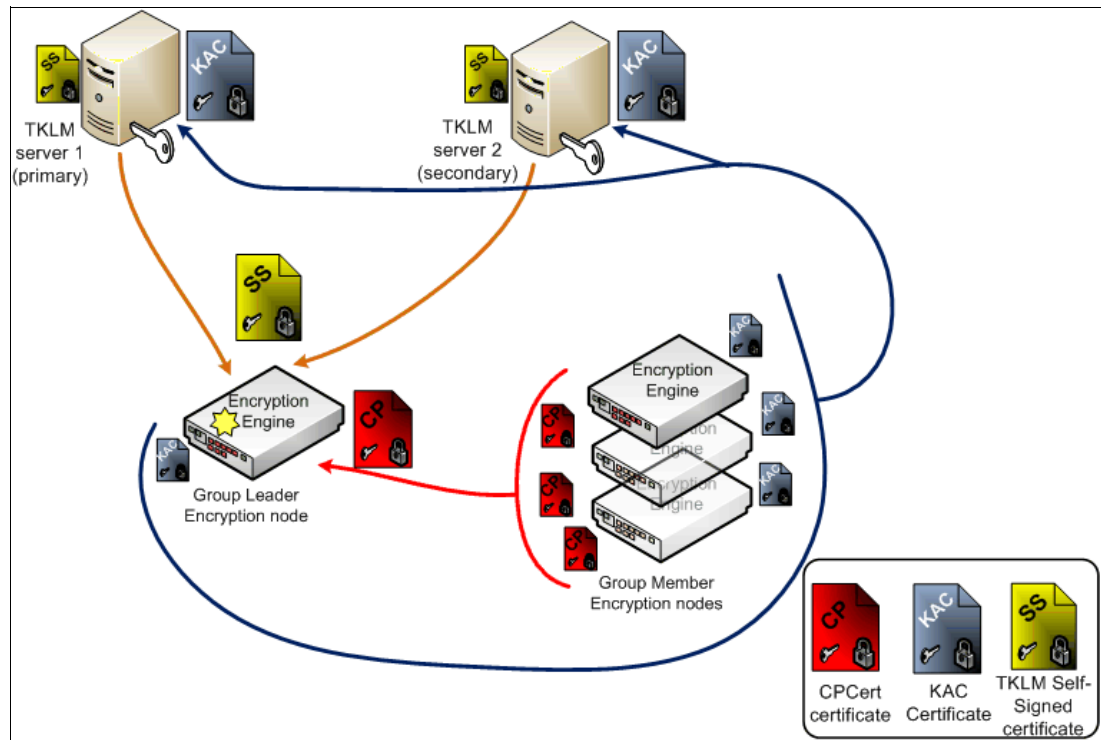


Figure 3-2 Generation and registration flow of the various certificates

Slot number: You need to enter a slot number in certain switch CLI commands when you use an Encryption Blade instead of a switch.

Multiple TKLM servers: If you plan to use more than one Encryption Switch, for example, as an HA cluster or DEK cluster solution, connect all Encryption Switches that will be on the same dedicated

1 Gbps LAN for Encryption Switch intercommunication and perform the IP setup. In these scenarios, you do not have to perform all steps on the other TKLMs; in each of the following steps, we explain what needs to be done if there are multiple TKLM servers. From a SAN32B-E4 Encryption Switch, certain commands only have to be executed via the EE's GL, and the GL will synchronize these settings across all other EEs.

Step 1: TKLM initial configuration: Creating the master keystore

During the first logon at the TKLM server via the Tivoli Integrated Portal, you are prompted to create the master keystore. Connect to the TKLM web GUI via your preferred browser using the user `tkladmin` and the password that you created during the installation, as shown in Figure 3-3. After this first logon, the system prompts you to create the default master keystore, which you can see in Figure 3-4.

Tip: With a TKLM installation that is based on accepting the defaults, the URL to the Tivoli Integrated Portal is `https://<your TKLM IP address>:16316/ibm/console/login.jsp`.

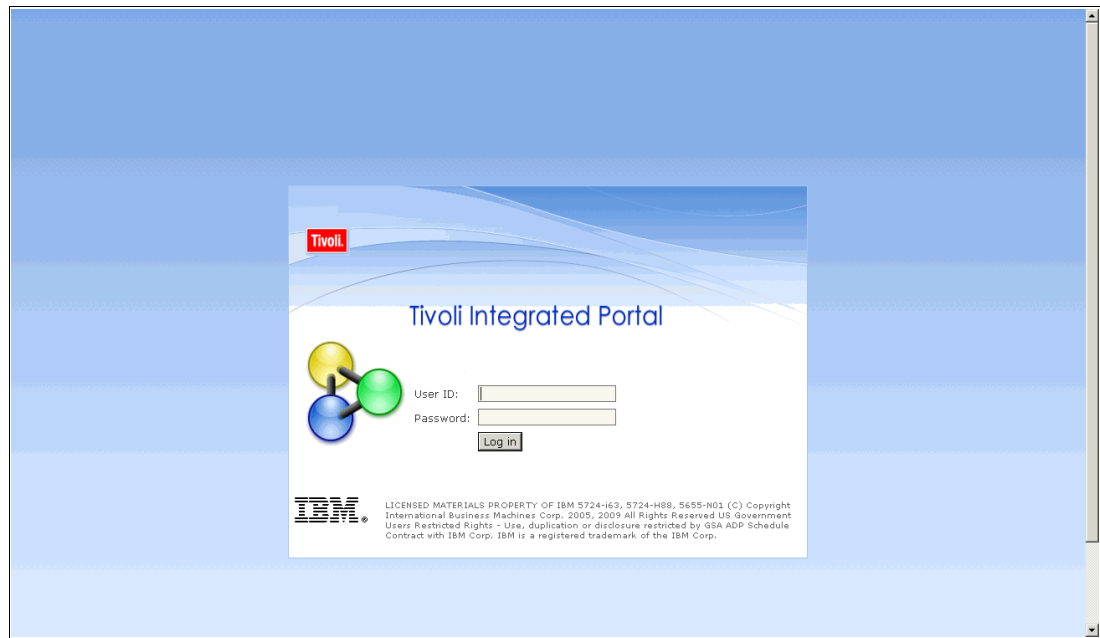


Figure 3-3 TKLM logon window

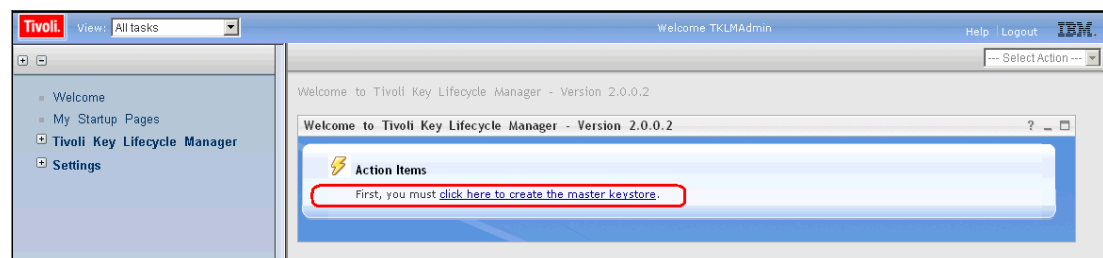


Figure 3-4 Create the TKLM master keystore during the first logon to TKLM

You must define a master password for the master keystore. You can use the same password that you used for the tkmadmin user for the TKLM logon. Use the defaults for the rest of the options in this window, as shown in Figure 3-5. Confirm that **JCEKS** is selected as the type of keystore.

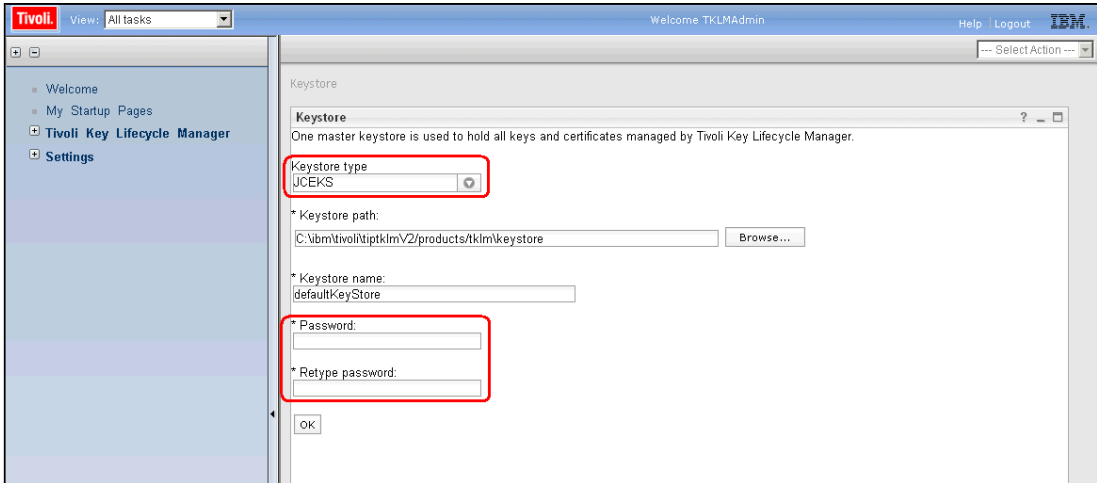


Figure 3-5 Specify the master password for the master keystore

You see the window as shown in Figure 3-6 after you have successfully accomplished this task.

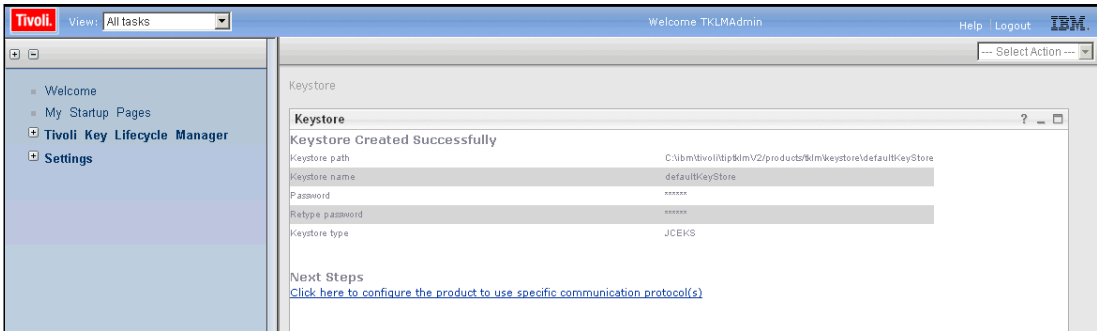


Figure 3-6 Master keystore created successfully after first logon to TKLM

Secondary TKLM server: If you have a secondary TKLM server, you must perform this task on the secondary TKLM server in the same way.

Important: Do not confuse the terms “master keystore” and “master key”.

Step 2: TKLM initial configuration: Creating the device group
In this next step, we check if the device group already exists. It typically exists if you are using TKLM v2.0.0.2. If it does not exist, you create a new device group by using the **create** command in this TKLM window. This device group is named BRCD_ENCRYPTOR, and it belongs to the device family LT0. IBM Encryption Switch nodes are associated with this device group.

You can browse to this window by selecting **Tivoli Lifecycle Manager → Advanced Configuration → Device Group**, as shown Figure 3-7 on page 53.

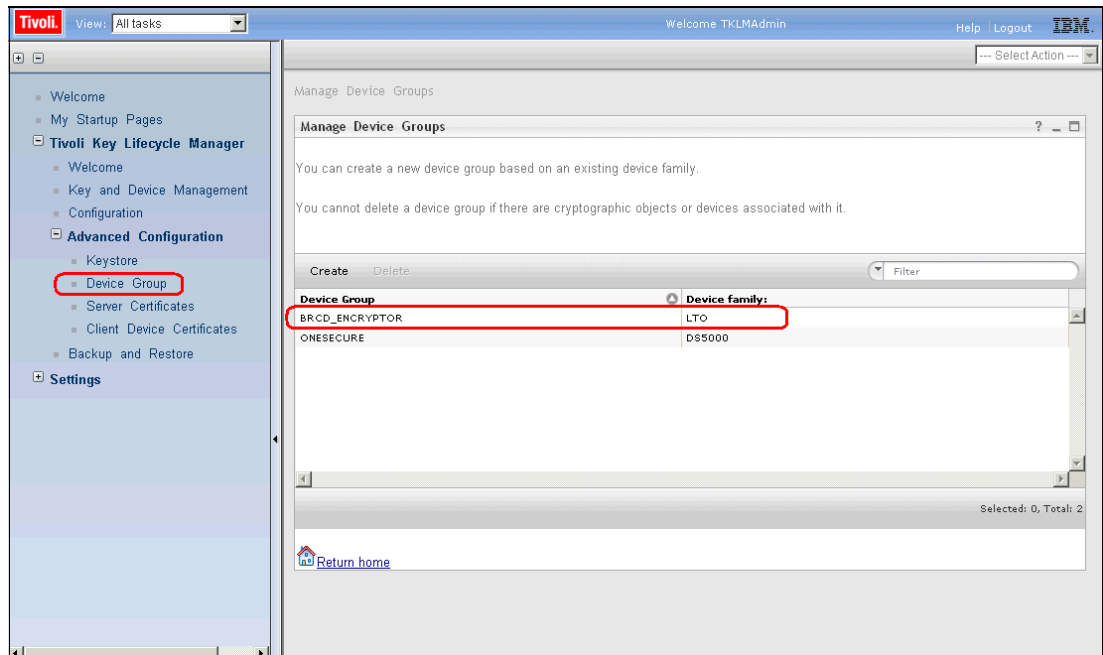


Figure 3-7 Manage Device Groups window of TKLM

In later steps, we continue the setup of this specific device group by adding devices (EEs) to that group.

Secondary Server: If you use more than one TKLM server, you must perform this task on the secondary TKLM server in the same way.

Step 3: TKLM initial configuration: Creating a self-signed server certificate

Whenever the TKLM servers and the EEs exchange keys, they must verify that the receiver of the keys is the correct receiver and can be trusted. Ensure that a third party is not in the middle trying to steal the keys while they are being exchanged. Therefore, in this step, we generate a self-signed server certificate of the TKLM, which in the next step will be exported. And, in a later step, the self-signed server certificate is imported and activated into the EEs.

You use the window that is shown in Figure 3-8 on page 54 to create a self-signed server certificate of TKLM.

Use label and descriptive parameters that are appropriate for your installation. You must document and remember these labels to use in a later step.

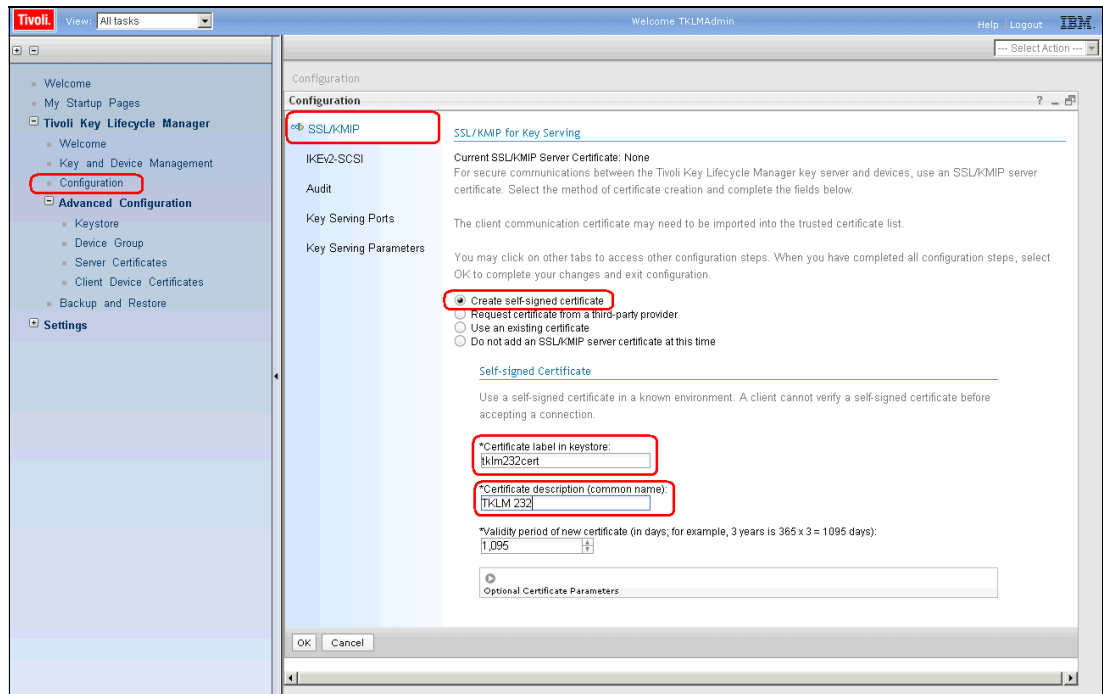


Figure 3-8 TKLM window to create a self-signed server certificate

A separate window confirms the certification creation, as shown in Figure 3-9.

TKLM server restart: A TKLM server restart is required after creating this self-signed server certificate. The commands for the restart of the TKLM server vary according to the type of operating system.

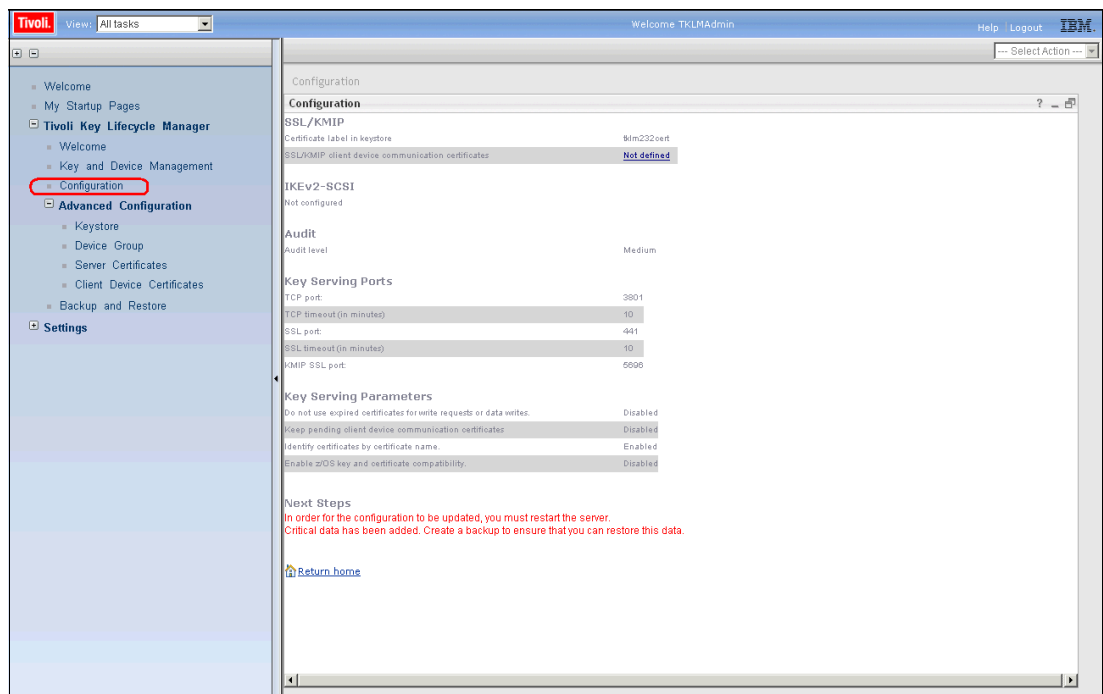


Figure 3-9 TKLM window to confirm the creation of a self-signed server certificate

In our setup, the TKLM server was running on a Windows environment. We used the following commands and sequence for the TKLM server restart. We executed the batch files stopServer.bat and startServer.bat in the TKLM server bin directory, C:\ibm\tivoli\tiptklmV2\bin\, as shown in Example 3-4.

Example 3-4 Command sequence to restart the TKLM server via the DOS command line

C:\ibm\tivoli\tiptklmV2\bin\stopServer.bat server1

ADMU0116I: Tool information is being logged in file
C:\ibm\tivoli\tiptklmV2\profiles\TIPProfile\logs\server1\stopServer.log

ADMU7702I: Because server1 is registered to run as a Windows Service, the request to stop this server will be completed by stopping the associated Windows Service.

ADMU0116I: Tool information is being logged in file
C:\ibm\tivoli\tiptklmV2\profiles\TIPProfile\logs\server1\stopServer.log

ADMU0128I: Starting tool with the TIPProfile profile

ADMU3100I: Reading configuration for server: server1

ADMU3201I: Server stop request issued. Waiting for stop status.

ADMU4000I: Server server1 stop completed.

C:\ibm\tivoli\tiptklmV2\bin\startServer.bat server1

ADMU0116I: Tool information is being logged in file
C:\ibm\tivoli\tiptklmV2\profiles\TIPProfile\logs\server1\startServer.log

ADMU7701I: Because server1 is registered to run as a Windows Service, the request to start this server will be completed by starting the associated Windows Service.

ADMU0116I: Tool information is being logged in file
C:\ibm\tivoli\tiptklmV2\profiles\TIPProfile\logs\server1\startServer.log

ADMU0128I: Starting tool with the TIPProfile profile

ADMU3100I: Reading configuration for server: server1

ADMU3200I: Server launched. Waiting for initialization status.

ADMU3000I: Server server1 open for e-business; process id is 2152

After the TKLM server restart, log on again to TKLM and verify by selecting **Tivoli Key Lifecycle Manager** → **Advanced Configuration** → **Server certificates** that the self-signed certificate that you have created is valid and active. The green square in the Status column indicates this condition, as shown in Figure 3-10 on page 56.

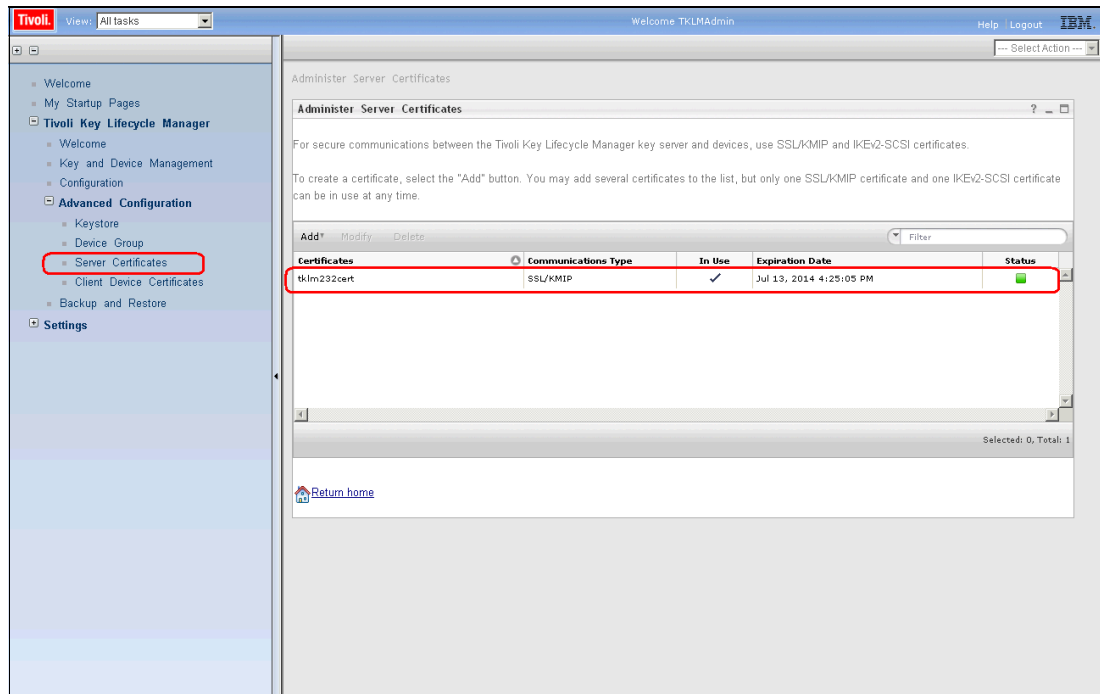


Figure 3-10 TKLM window to verify the self-signed server certificate is active

Secondary server: If you use more than one TKLM server, you must perform this task on the secondary TKLM server in the same way.

Step 4: TKLM initial configuration: Exporting the self-signed server certificate

In this step, we export the self-signed server certificate that we created in “Step 3: TKLM initial configuration: Creating a self-signed server certificate” on page 53 from the TKLM keystore to a file. This step is necessary, because in a later step, we import this generated file to the EEs.

You cannot perform this step by using the TKLM server GUI. A command-line tool, **wsadmin**, is available to perform this operation. WebSphere Application Server provides this interface. You must specify the Jython syntax.

The path to this **wsadmin** utility varies on the type of operating system on the TKLM server.

For a LINUX/AIX-based TKLM server platform, navigate to `/opt/IBM/tivoli/tiptklmV2/bin/` in a terminal window.

For a Windows-based TKLM server platform, navigate to `C:\ibm\tivoli\tpklmV2\bin\` in a CMD window.

Now, connect first to the **wsadmin** utility using the command that is shown in Example 3-5 on page 57.

Example 3-5 Log on to the wsadmin tool on a Windows-based TKLM server in a CMD window

```
c:\ibm\tivoli\tipctlmv2\bin\wsadmin.bat -username TKLMAdmin -password pass -lang
jython
WASX7209I: Connected to process "server1" on node TIPNode using SOAP connector;
The type of process is: UnManagedProcess*sys-package-mgr*: processing modified
jar, 'C:\ibm\tivoli\tipctlmv2\plugins\com.ibm.ws.runtime_6.1.0.jar'
WASX7031I: For help, enter: "print Help.help()"

wsadmin>
```

After the successful start of the **wsadmin** session, the tool is ready to receive your commands on its CLI.

Before we export the self-signed server certificate, we first want to display it, as shown in Example 3-6. This step is necessary, because we need to obtain certain parameters for the next task. You must obtain the Universally Unique Identifier (UUID) prior to exporting it. The command that is shown in Example 3-6 is the same command for LINUX, AIX, and Windows.

Example 3-6 The wsadmin tool output showing the TKLM self-signed server certificate details

```
wsadmin>print AdminTask.tklmCertList('[]')
CTGKM0001I Command succeeded.
uuid = CERTIFICATE-72a98eeb-9c77-4833-91de-24c66100a90d
alias = tklm232cert
key store name = defaultKeyStore
key state = ACTIVE
issuer name = CN=TKLM 232
subject name = CN=TKLM 232
creation date = 7/14/11 4:25:07 PM PDT
expiration date = 7/13/14 4:25:05 PM PDT
serial number = 25520783099248
```

Important: This listing contains all certificates that are stored in the TKLM database. Use the server certificate's UUID to export the certificate from the database to the file system. You can recognize the UUID from the alias-name that you specified when you created this certificate within TKLM (in Example 3-6, it is tklm232cert).

After we have successfully identified our self-signed server certificate, we continue to export this TKLM self-signed server certificate to a file. To accomplish this task, we use the **wsadmin** command, **print AdminTask.tklmCertExport**, as shown in Example 3-7.

For the **-uuid** parameter, we used the ID from Example 3-6. And for the **-filename** parameter, we specify any name, which will become the exported certificate's name in the file system. For example, we used **tklm-solc-win2008-232.der** as the file name, because it is from a Microsoft Windows Server 2008 server and its IP address ended in x.x.x.232.

Example 3-7 Exporting the TKLM self-signed server certificate to a file using the wsadmin tool

```
wsadmin>print AdminTask.tklmCertExport('[-uuid CERTIFICATE-72a98eeb-9c77-4833-91
de-24c66100a90d -fileName tklm-solc-win2008-232.der -format DER]')
CTGKM0001I Command succeeded.
C:\ibm\tivoli\tipctlmv2\products\tklm\tklm-solc-win2008-232.der
```

After you finish and the command completes successfully, you can exit the **wsadmin** tool by using **quit** or **exit**. You return to the Windows CMD box prompt.

Verify with a simple DOS command (**dir**) that the exported self-signed server certificate is in the Windows file system, as shown in Example 3-8.

Example 3-8 Showing the exported TKLM self-signed server certificate in the file system

```
C:\dir C:\ibm\tivoli\tiertklmV2\products\tklm\*.der
```

```
Volume in drive C has no label.
```

```
Volume Serial Number is 1825-978F
```

```
Directory of C:\ibm\tivoli\tiertklmV2\products\tklm
```

```
07/14/2011  04:58 PM
```

```
715  tklm-solc-win2008-232.der
```

Secondary server: If you use more than one TKLM server, you must perform this task d on the secondary TKLM server in the same way. Be aware that you will get a separate exported self-signed server certificate from the other TKLM server. Give this certificate a separate name. In our setup, the secondary TKLM server had the IP address x.x.x.231, and we stored this certificate as tklm-solc-win2008-231.der.

In later steps, we will import these two certificates to the GL node.

Step 5: SAN32B-E4 Encryption Switch: Cleaning up old config files

Important: This step is not necessary if you have received a new, unused SAN32B-E4. If you have any doubt about issuing this command and its consequences, contact your local support team.

This step ensures that you start with a clean SAN32B-E4 Encryption Switch configuration. The commands will remove various old configuration files within the EE. You must run all the commands that are shown in Example 3-9 using the root user ID. This step is only necessary if a prior configuration needs to be removed. However, you can run these commands on your new setup, as well, to ensure a clean starting point.

Example 3-9 These commands clean up old config files in the Encryption Switch

```
SAN32B-E4-1:root> rm -r /etc/fabos/mace/*
SAN32B-E4-1:root> rm /etc/fabos/certs/sw0/*
SAN32B-E4-1:root> rm /etc/fabos/certs/mace/*
SAN32B-E4-1:root> rm -r /mnt/etc/fabos/mace/*
SAN32B-E4-1:root> rm /mnt/etc/fabos/certs/sw0/*
SAN32B-E4-1:root> rm /mnt/etc/fabos/certs/mace/*
SAN32B-E4-1:root> diagdisablepost
SAN32B-E4-1:root> reboot
```

We issued the command sequence in Example 3-9 on an IBM System Storage SAN32B-E4 Encryption Switch (2498-E32).

Step 6: SAN32B-E4 Encryption Switch: Zeroizing the EE

In this task, we “zeroize” the EE. Perform this task on the GL *and* the member nodes.

An encryption node is either an Encryption Switch or a DCX/DCX-4S with one or more Encryption Blades. An Encryption Switch is both a single node and a single EE. A DCX is a single node, but it can have multiple EEs. Example 3-10 shows how to use this command.

Example 3-10 Command to zeroize the EE

```
SAN32B-E4-1:admin> cryptocfg --zeroizeEE
This action will zeroize all critical security parameters.
Any decommissioned keyids in the EG will also be deleted.
Following the zeroizing of this EE, the switch/blade will be rebooted.
ARE YOU SURE (yes, y, no, n): [no] y
Operation succeeded.

Rebooting...
```

Step 7: SAN32B-E4 Encryption Switch: Initializing the encryption node

The command that is shown in Example 3-11 initializes the encryption node. Perform this task, as well, on the GL *and* the member nodes.

This **initnode** command creates the Node CP certificate and the KAC certificate on the encryption nodes.

Important: Later, we export this KAC certificate to the TKLM servers.

Example 3-11 Command to initialize the encryption node

```
SAN32B-E4-1:admin> cryptocfg --initnode
This will overwrite all identification and authentication data
ARE YOU SURE (yes, y, no, n): [no] y
Operation succeeded
```

Step 8: SAN32B-E4 Encryption Switch: Creating the encryption group

The EG is created on the node that is designated as the group leader (GL). A GL is a special node within an EG that acts as a group manager and cluster manager. The GL manages and distributes all group-wide and cluster-wide configurations to all members of the group or cluster. The GL flag can move around, for example, to another node because of a failure of the previous GL or a reboot. Example 3-12 shows the command to create an EG.

Example 3-12 Command to create the EG

```
SAN32B-E4-1:admin> cryptocfg --create -encgroup support_TKLM

Encryption group create status: Operation Succeeded.
```

In Example 3-12, we used the default name, support_TKLM, for the EG.

Step 9: SAN32B-E4 Encryption Switch: Importing the node certificates

In this step, we import the node CP certificates from the other encryption member nodes if there are any other encryption member nodes. *Perform this task only on the GL.*

The CP certificate is created during node initialization (**cryptocfg --initnode**). This certificate is exchanged with the GL node, and it is used for authenticating the member nodes to the encryption GL. The location of the CP certificate file is /etc/fabos/certs/sw0/my_cp_cert.pem. This name is generated by the system, and it is not a name that the user can assign. The GL node can import this file directly from the member nodes, bypassing the need to use an intermediate Secure Shell (SSH) server (Example 3-13).

User access: This procedure requires the root user access.

Example 3-13 Command to import the CP certificate of the member node

```
SAN32B-E4-1:root> cryptocfg --import -scp BES2_CPcert.pem 10.18.235.56 root
/etc/fabos/certs/sw0/my_cp_cert.pem
Available Space:24576
Make sure your file size is not greater than 24576.
The switch will be unstable or the operation will fail if you exceed this limit.
Do you want to proceed?
ARE YOU SURE (yes, y, no, n): [no] y
root@10.18.235.56's password:
Operation succeeded.
```

In Example 3-13 on page 60, we imported the `my_cp_cert.pem` file (the CP certificate) from a second SAN32B-E4 Encryption Switch to our GL node.

Step 10: SAN32B-E4 Encryption Switch: Registering member nodes on the GL node

After the CP certificates are imported from the member nodes to the GL as described in “Step 9: SAN32B-E4 Encryption Switch: Importing the node certificates” on page 59, the member nodes will then be registered within the GL. The worldwide node name (WWNN) of *each member node* is required to complete this step. Use the `wwn` command on the member switch or node CLI to obtain this information, as shown in Example 3-14.

Example 3-14 Command to obtain the node wwnn

```
SAN32B-E4-2:root> wwn
10:00:00:05:1e:54:16:53
```

You will need the WWNN of the member node for the next task. To register the member node on the GL node, you use the command that is shown in Example 3-15 on the GL CLI.

Example 3-15 Command to register member nodes on the GL node

```
SAN32B-E4-1:root> cryptocfg --reg -membnode 10:00:00:05:1e:54:16:53
BES2_CPcert.pem 10.18.235.56
Operation succeeded.
```

User access: This procedure requires the root user access.

Step 11: SAN32B-E4 Encryption Switch: Setting the key vault type

In this task, we register the type of key vault that we are going to use within the EG. In this particular case, we use the IBM Tivoli Key Lifecycle Manager.

The command that is shown in Example 3-16 is run on the GL node.

Example 3-16 Command to set the key vault type within the GL node

```
SAN32B-E4-1:root> cryptocfg --set -keyvault TKLM
Set key vault status: Operation Succeeded.
```

User access: This procedure and the following procedure will also work when using the admin user access, but in our setup, we continued using the root user access at this point.

Step 12: Initializing, registering, and enabling the encryption engine

Finally, we initialize, register, and enable the EE. We issue the command sequence that is shown in Example 3-17 on page 61 from the GL node.

Example 3-17 Command sequence to initialize, register, and enable the EE

```
SAN32B-E4-1:root> cryptocfg --initEE
This will overwrite previously generated identification and authentication data
ARE YOU SURE (yes, y, no, n): [no] y
Operation succeeded.

SAN32B-E4-1:root> cryptocfg --regEE
Operation succeeded.

SAN32B-E4-1:root> cryptocfg --enableEE
Operation succeeded.
```

Important: If running these commands on an Encryption Blade within a DCX/DCX-4S director configuration, ensure that at the end of each command, you add the slot number of the corresponding Encryption Blade.

Example 3-18 shows the same command sequence for an Encryption Blade within a DCX configuration. In this case, the Encryption Blade was located in slot number 2.

Example 3-18 Command sequence to initialize, register, and enable the EE on an Encryption Blade

```
cryptocfg --initEE 2
cryptocfg --regEE 2
cryptocfg --enableEE 2
```

Step 13: SAN32B-E4 Encryption Switch: Obtaining the discrete device serial number

Beginning with FOS v7.0.0, each node needs to be registered to the TKLM servers with its discrete device serial number rather than grouped under a generic serial number. This device serial number is based on the output from the command **cryptocfg --reg -KAClogin**, as shown in Example 3-19. This task needs two actions to be accomplished:

1. Get the discrete device serial number of *all* encryption nodes via the switch CLI.
2. Register each node's discrete device serial number within *all* TKLM servers.

In Step 13: SAN32B-E4 Encryption Switch: Obtaining the discrete device serial number, we show how to query the discrete device serial number of an encryption node. In Step 14: TKLM: Adding each node's discrete device serial number, we show how to add this information to the TKLM servers.

Example 3-19 Command to query the discrete device serial number of an encryption node

```
SAN32B-E4-1:root> cryptocfg --reg -KAClogin
Please register device on the TKLM Server with Serial B10_00_00_05_1e_54_17_10
under device group BRCD_ENCRYPTOR
```

In the next step, we will add the device serial number that is shown in Example 3-19 to the TKLM server.

Tip: If you use more than one encryption node, ensure that you collect the discrete device serial number of each node. Each node's discrete device serial number must be queried and registered to all TKLM servers.

Step 14: TKLM: Adding each node's discrete device serial number

On each TKLM server, we register the discrete device serial number of each encryption node. In the previous step, we explained how to query the discrete device serial number of each encryption node.

Go to the TKLM server. If you have more than one TKLM server, start at the first server and repeat this activity on the second TKLM server.

Select **Key and Device Management**. Select the **BRCD_Encryptor** device group. Click **Go**, as shown in Figure 3-11.

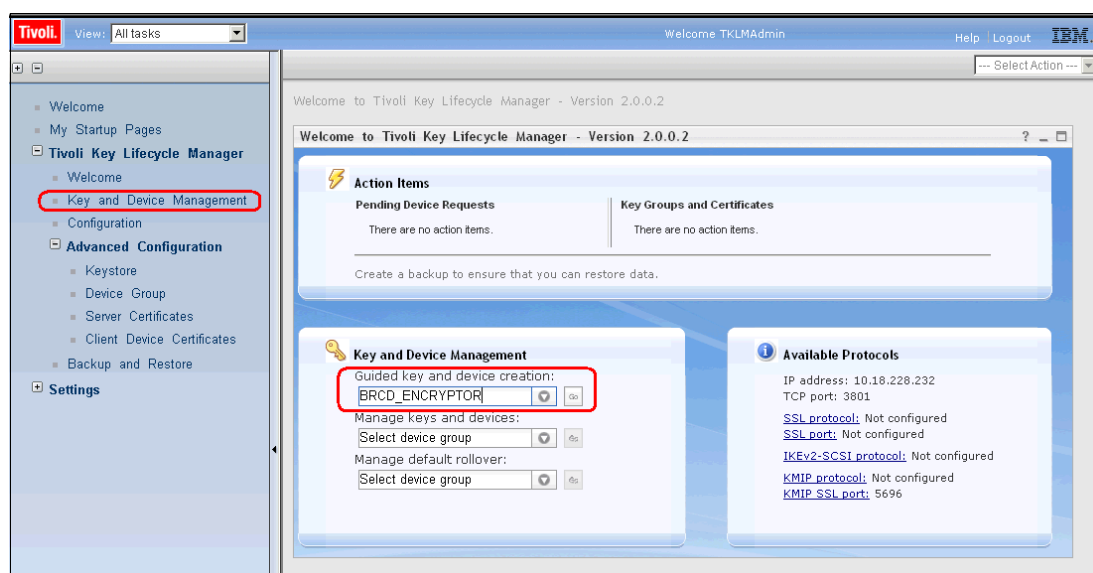


Figure 3-11 TKLM Key and Device Management window

In the window that opens, as shown in Figure 3-12, select **Only accept manually added devices for communication** and click **Add**.

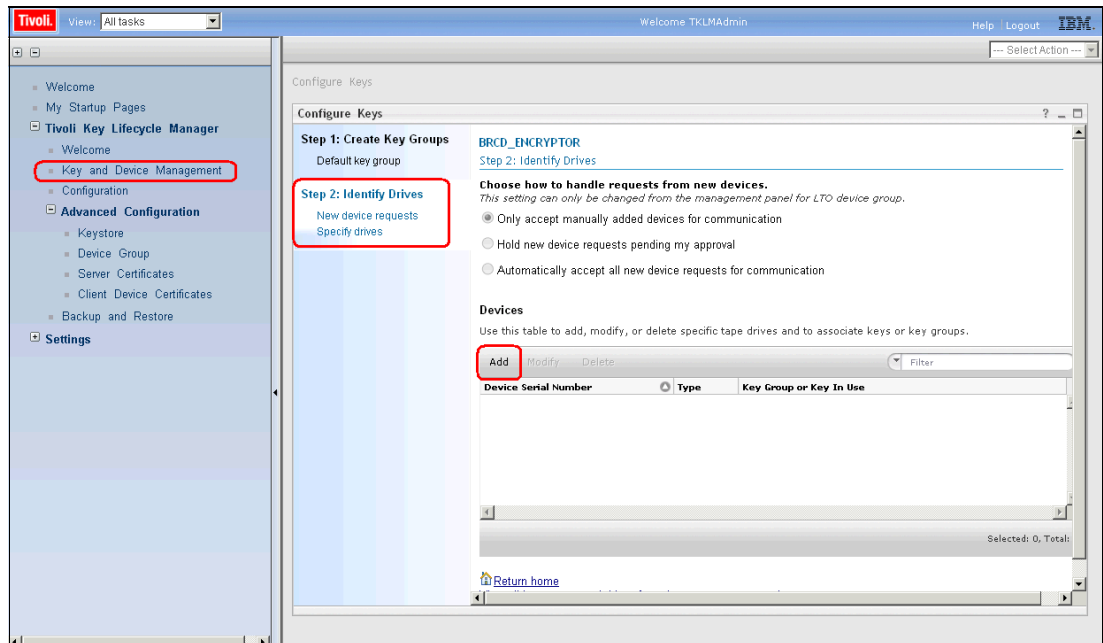


Figure 3-12 TKLM Key and Device Management window to add new devices

In the window that is shown in Figure 3-13, you can enter the encryption node discrete device serial number that you want to add to TKLM. For the Key group or key in use, keep the Default selection, and click **Add Tape Drive**.

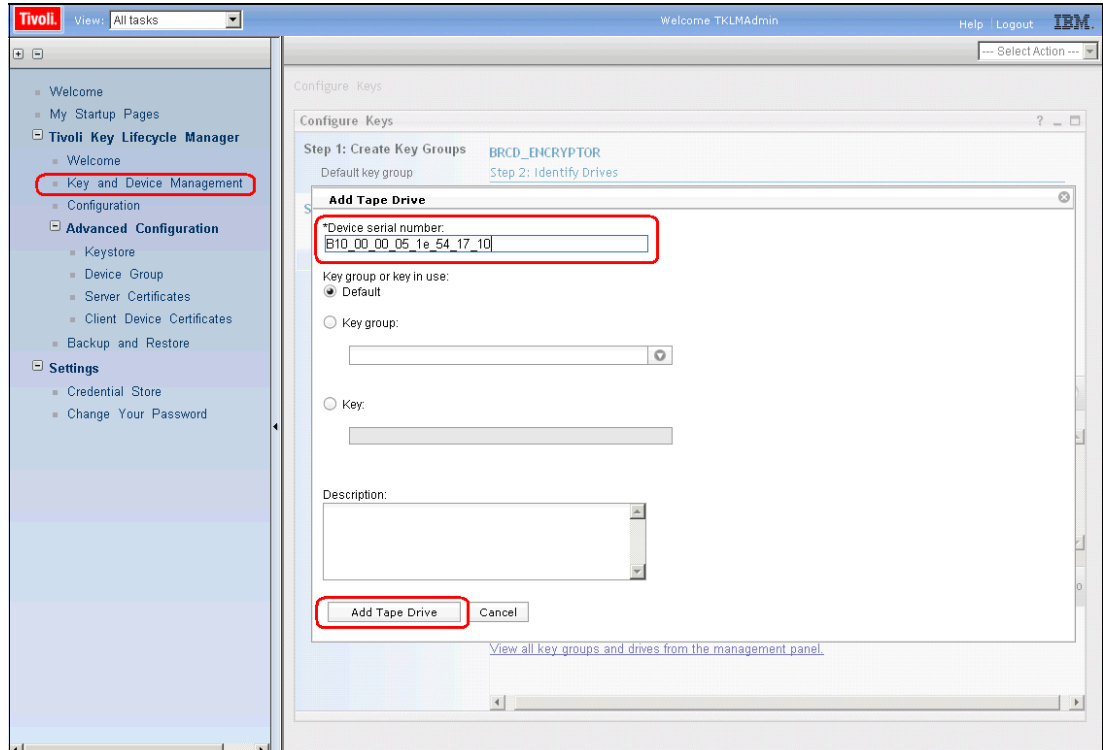


Figure 3-13 TKLM Key and Device Management: Adding the new device window

You can confirm the successful creation of the newly added device serial number, as shown in Figure 3-14 on page 64.

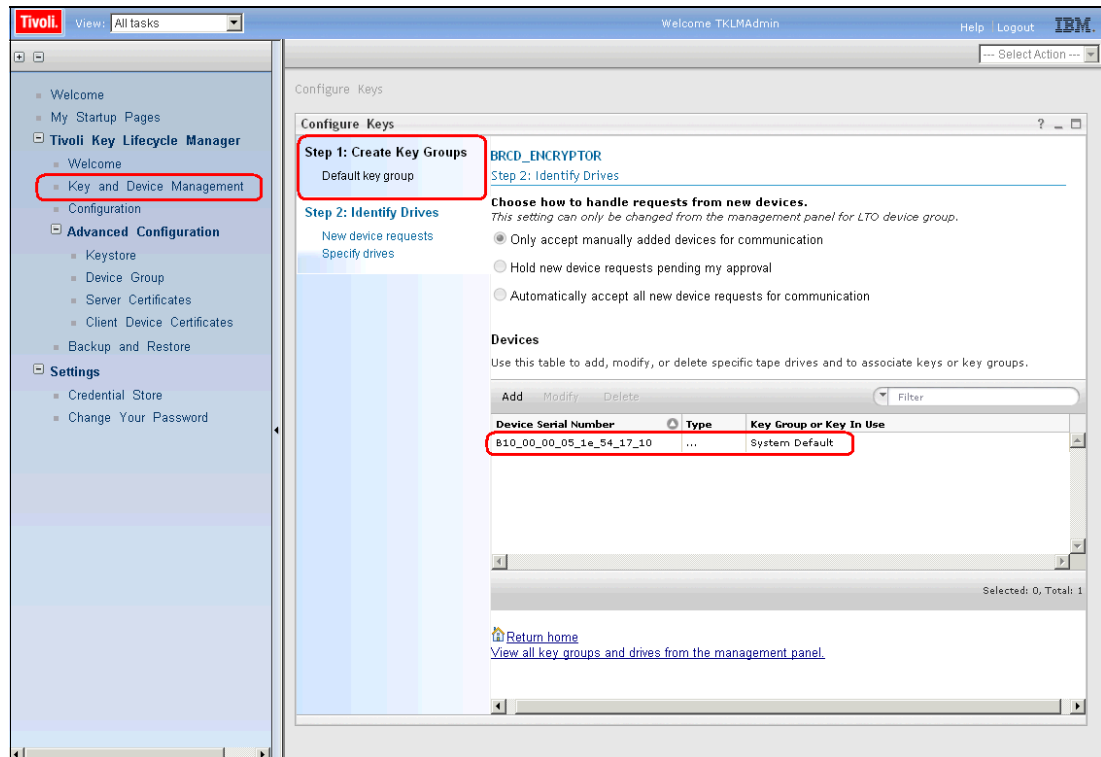


Figure 3-14 TKLM Key and Device Management window confirming that the new device was added

Step 15: Transferring the EE nodes' KAC certificates to the TKLM servers

In the previous steps, we created the KAC certificates on each encryption node (refer to Step 7: SAN32B-E4 Encryption Switch: Initializing the encryption node). In this step, the KAC certificates from the GL and the member nodes are transferred to the TKLM servers prior to importing them using the TKLM web interface.

The transfer method varies depending on the TKLM server operating system. The transfer method can be a file transfer protocol (**ftp**), for example, File Transfer Protocol (FTP), Secure Copy Protocol (SCP), WinSCP, or another protocol.

In our setup, we used a Windows-based TKLM server setup. Windows does not come with a built-in SCP server module. We installed a third-party software SCP server program on each TKLM server to upload the KAC certificates from the encryption nodes to the TKLM servers.

Bitvize: We used the 30-day trial version of the SCP Microsoft Windows server program Bitvize (<http://bitvize.com>).

Example 3-20 on page 65 shows how to use the **cryptocfg** command in this step. We used `supt_bes1_KACCert.pem` for the target name on the Windows TKLM server, because this name was the KAC certificate name of our SAN32B-E4 Encryption Switch that we called BES1.

Example 3-20 Command to transfer the KAC certificates from the encryption nodes to TKLM servers

```
SAN32B-E4-1:root> cryptocfg --export -scp -KACcert 10.18.228.232 Administrator  
supt_bes1_KACCert.pem
```

Administrator@10.18.228.232's password:
Operation succeeded.

Step 16: Importing the KAC certificates to the TKLM servers

The KAC certificate files, which were transferred from the encryption GL and then the member nodes, are imported into the TKLM servers. Click **Client Device Certificates**, as shown in Figure 3-15, and select **Import**, as shown in Figure 3-16 on page 66, selecting the certificate type **SSL/KMIP Certificate**.

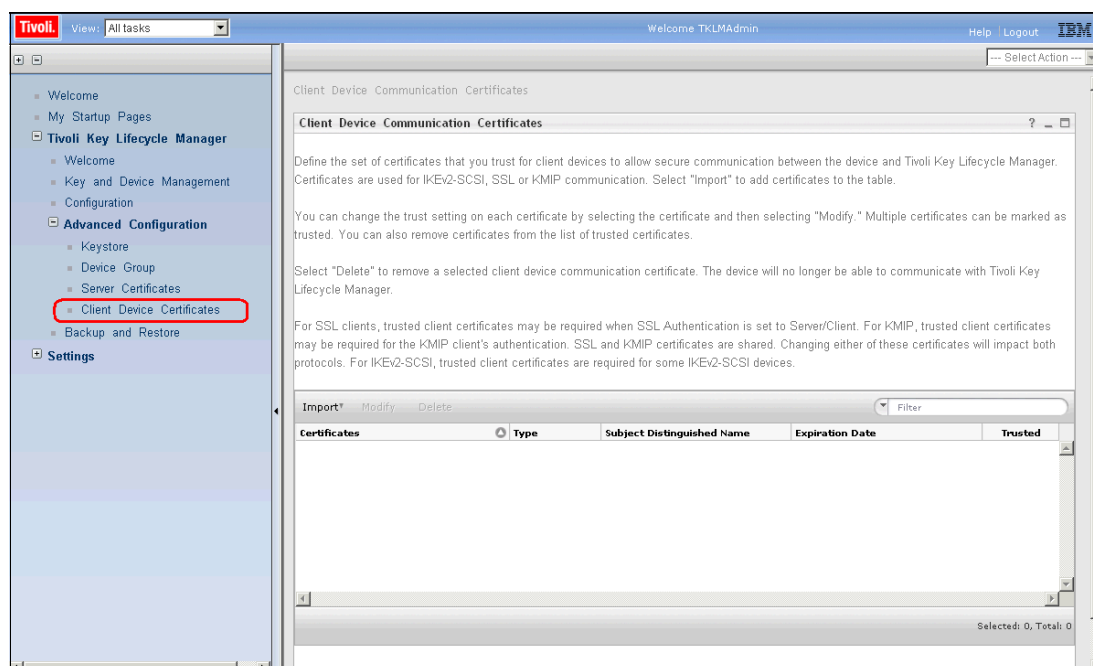


Figure 3-15 TKLM window to manage client device communication certificates

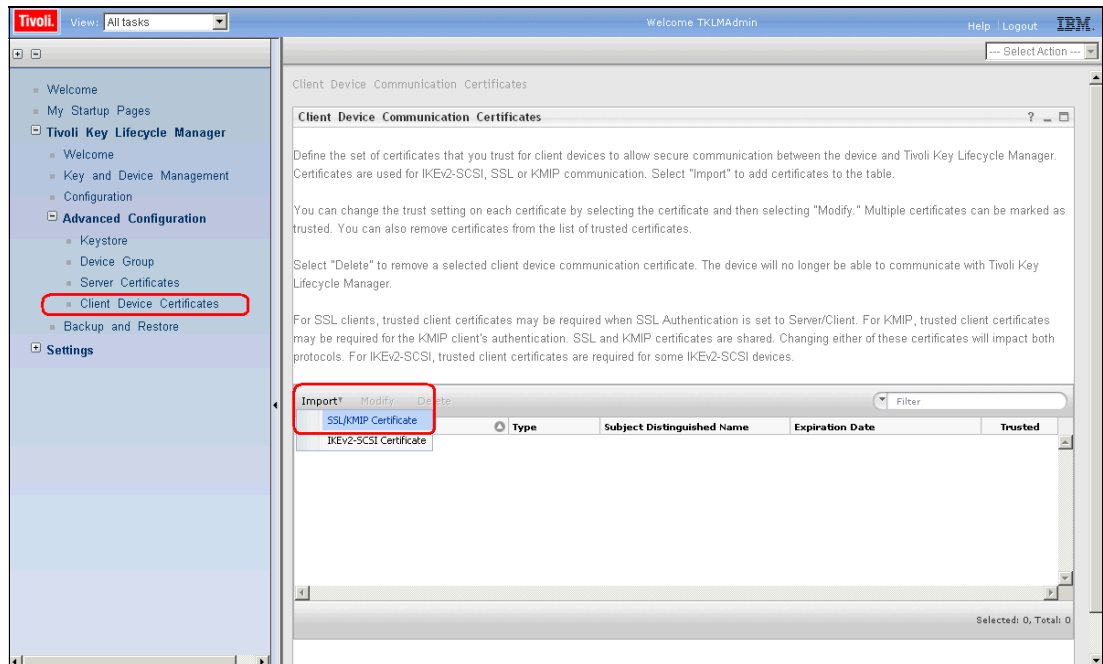


Figure 3-16 TKLM window to import an SSL/KMIP client certificate

In the next window, enter more information. First, enter a user-defined certificate name of your choice. Next, browse to the Windows server directory to which the KAC certificate was transferred earlier (via the SCP operations of the `cryptocfg` command that was explained in Step 15: Transferring the EE nodes' KAC certificates to the TKLM servers). Select the appropriate certificate. Select **Allow the server to trust this certificate and communicate with the associated client device** to trust this certificate (Figure 3-17).

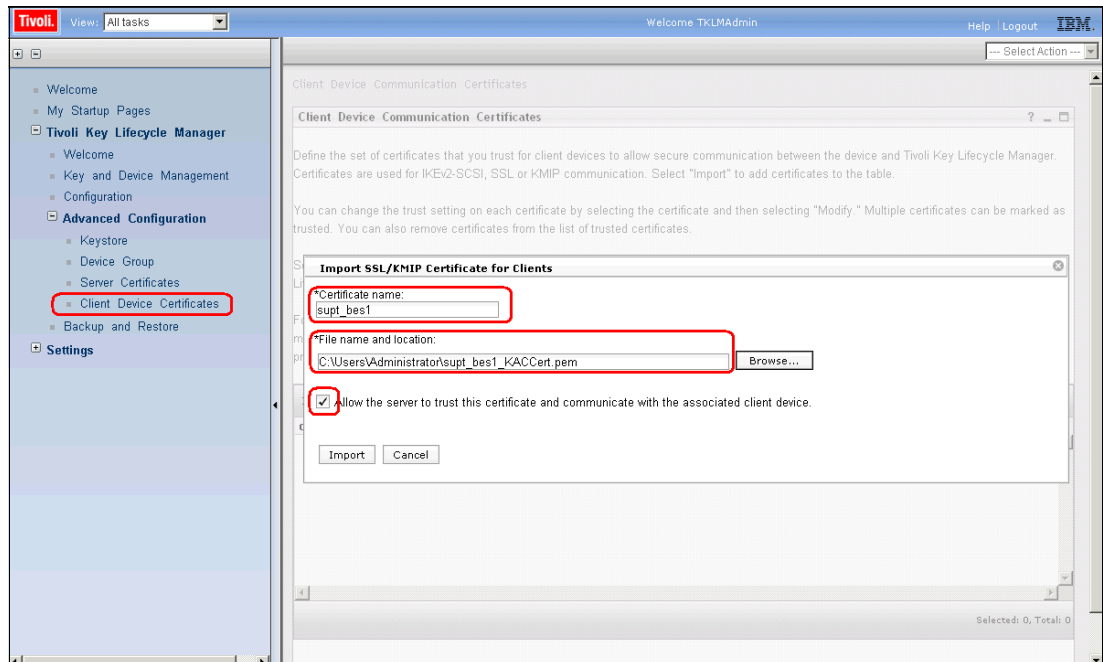


Figure 3-17 TKLM Import SSL/KMIP certificate for clients

Verify the successful KAC import to TKLM, as shown in Figure 3-18. Use the menu **Tivoli Key Lifecycle Manager** → **Advanced Configuration** → **Client Device Certificates**. The green status box indicates that the certificate is valid.

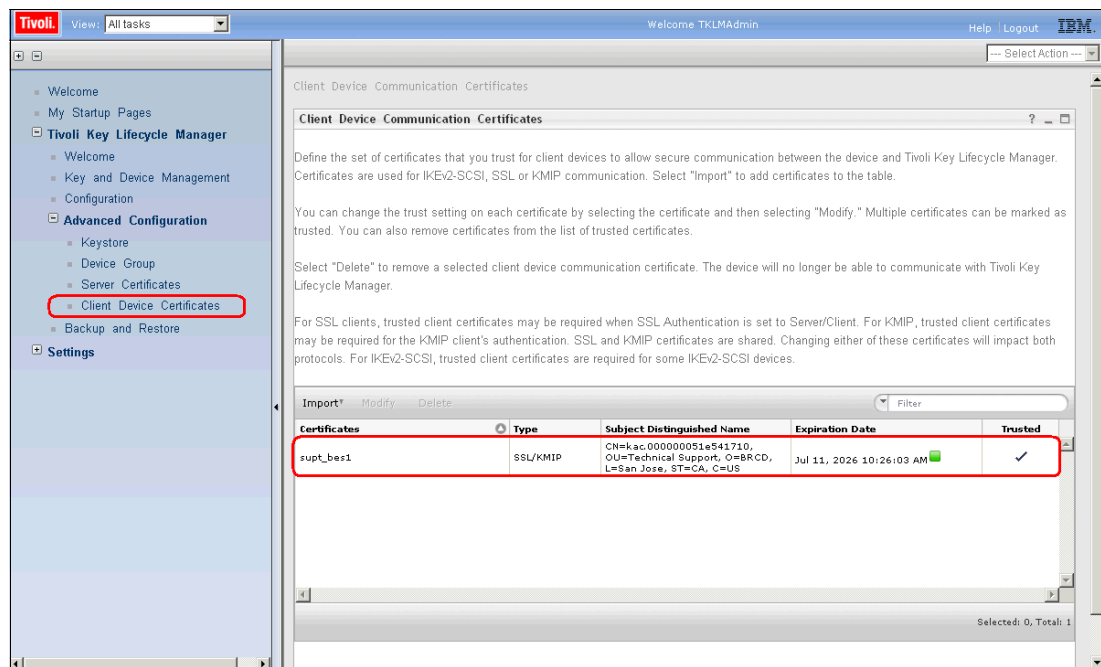


Figure 3-18 TKLM acknowledgement of KAC certification import process success

Step 17: Importing the TKLM server certificates on the GL node

In an earlier step, we exported the TKLM server certificates from the TKLM server database to the TKLM server file system (refer to Step 4: TKLM initial configuration: Exporting the self-signed server certificate). This file is now imported by the GL node and used to register the TKLM as the key vault (the primary and the secondary key vaults in our scenario, because we set up two TKLM servers).

Also, this FOS command requires a file transfer between TKLM and the encryption GL node. The transfer method varies depending on your TKLM server's operating system. The transfer method can include any file transfer protocol, for example, FTP, SCP, WinSCP, or another protocol.

We used a Windows-based TKLM server setup. Because Windows does not include a built-in SCP server module, we installed a third-party software SCP server program on each TKLM server to accomplish this transfer and also to download the TKLM server certificates to the encryption node (GL).

Example 3-21 shows the command for this operation.

Example 3-21 Transfer TKLM server certificates to the GL node

```
SAN32B-E4-1:root> cryptocfg --import -scp solc-tklm231.der 10.18.228.231
Administrator tklm-solc-win2008-231.der
```

```
Available Space:24576
Make sure your file size is not greater than 24576.
The switch will be unstable or the operation will fail if you exceed this limit.
Do you want to proceed?
ARE YOU SURE (yes, y, no, n): [no] y
```

```
Administrator@10.18.228.232's password:  
Operation succeeded.
```

```
SAN32B-E4-1:root> cryptocfg --import -scp solc-tklm232.der 10.18.228.232  
Administrator tklm-solc-win2008-232.der  
Available Space:20480  
Make sure your file size is not greater than 20480.  
The switch will be unstable or the operation will fail if you exceed this limit.  
Do you want to proceed?  
ARE YOU SURE (yes, y, no, n): [no] y  
Administrator@10.18.228.232's password:  
Operation succeeded.
```

In Example 3-21 on page 67, we used `solc-tklm23x.der` as the target name to store the certificate under FOS. Earlier, we specified the `tklm-solc-win2008-23x.der` name on the TKLM Windows server file space when we generated and exported the certificate via the `wsadmin` command. You can see, in Example 3-21 on page 67, that we issued the command twice: one time to the primary TKLM server and again to the secondary TKLM server to get from each TKLM server the correct self-signed server certificate.

Important: Each TKLM server must be registered at the GL node.

Step 18: Registering the TKLM server certificates on the GL node

In this step, we continue to register the previously imported TKLM server certificates as the primary and the secondary key vaults. Refer to Step 17: Importing the TKLM server certificates on the GL node.

Example 3-22 shows how to register the primary and secondary keystores to the encryption GL node. You can specify a name for each key vault. We used TKLM231 for the primary key vault, and we used TKLM232 for the secondary key vault. In addition, specify each TKLM server IP address, as well as the imported server certificate. You must specify which TKLM server acts as the primary TKLM server and which TKLM server acts as the secondary TKLM server.

Example 3-22 Command to register the primary and secondary key vaults to the GL node

```
SAN32B-E4-1:root> cryptocfg --reg -keyvault TKLM231 solc-tklm231.der 10.18.228.231  
primary  
Register key vault status: Operation Succeeded.  
  
SAN32B-E4-1:root> cryptocfg --reg -keyvault TKLM232 solc-tklm232.der 10.18.228.232  
secondary  
Register key vault status: Operation Succeeded.
```

Step 19: Configuration Checkpoint: Verifying the TKLM connectivity

At this point, the communication between the encryption nodes and the TKLM servers is established. You can use the `cryptocfg --show -groupc` command to verify the readiness for key operations. Example 3-23 on page 69 shows the partial output of this FOS command and where to look to confirm successful TKLM communication.

Example 3-23 Command to verify TKLM connectivity

SAN32B-E4-1:root> cryptocfg --show -groupc

```
Encryption Group Name:  support_TKLM
  Failback mode:        Auto
  Replication mode:     Disabled
  Heartbeat misses:     3
  Heartbeat timeout:    2
  Key Vault Type:       TKLM
  System Card:          Disabled
Primary Key Vault:
  IP address:           10.18.228.231
  certificate ID:        TKLM
  certificate label:     TKLM231
  State:                Connected
  Type:                 TKLM
Secondary Key Vault:
  IP address:           10.18.228.232
  certificate ID:        TKLM
  certificate label:     TKLM232
  State:                Connected
  Type:                 TKLM
Additional Primary Key Vault Information::
  Key Vault/CA certificate Validity:  Yes
  Port for Key Vault Connection:      5696
  Time of Day on Key Server:          N/A
  Server SDK Version:                 TKLM 2.0.0.2 KMIP 1.0 BUILD 201
Additional Secondary Key Vault Information:
  Key Vault/CA certificate Validity:  Yes
  Port for Key Vault Connection:      5696
  Time of Day on Key Server:          N/A
  Server SDK Version:                 TKLM 2.0.0.2 KMIP 1.0 BUILD 201
Encryption Node (Key Vault Client) Information:
  Node KAC certificate Validity:      Yes
  Time of Day on the Switch:          2011-07-16 12:39:56
  Client SDK Version:                 N/A
  Client Username:                    N/A
  Client Usergroup:                  N/A
  Connection Timeout:                 10 seconds
  Response Timeout:                   10 seconds
  Connection Idle Timeout:             N/A
Key Vault configuration and connectivity checks successful, ready for key
operations.
Authentication Quorum Size:           0
Authentication Cards not configured
NODE LIST
Total Number of defined nodes: 1
Group Leader Node Name:               10:00:00:05:1e:54:17:10
Encryption Group state:                CLUSTER_STATE_CONVERGED
Crypto Device Config state:            In Sync
Encryption Group Config state:         In Sync
  Node Name      IP address      Role
10:00:00:05:1e:54:17:10  10.18.235.54  GroupLeader  (current node)
  EE Slot:      0
```

SP state:

Operational; Need Valid KEK

You might notice, at the end of the output in Example 3-23 on page 69, a message stating “SP state: Operational; Need Valid KEK”. We get this message because we have not created the encryption GL MK yet. This MK will be used, after it is created, for a safe key exchange of the individual data encryption keys (DEKs) from the EEs to the TKLM servers. In Step 20: Generating and exporting the master key from the GL node, we generate and export this MK from the GL node to the TKLM servers.

Step 20: Generating and exporting the master key from the GL node

This last step generates (see Example 3-24) and exports (see Example 3-25) the MK from the encryption GL to the primary and secondary key vaults (TKLM servers).

Example 3-24 Command to generate the MK from the GL node

```
SAN32B-E4-1:root> cryptocfg --genmasterkey
```

Master Key generated. The Master Key must be exported before further operations are performed.

Example 3-25 Command to export the MK from the GL node to the TKLM servers

```
SAN32B-E4-1:root> cryptocfg --exportmasterkey
```

Enter passphrase:

password

Confirm passphrase:

password

Master Key exported.

Master Key ID: e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5

Exported Key ID: e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5

When you export the MK to the key vaults, you must specify a pass phrase for the MK. You also need to store the information in a safe place for recovery purposes.

You can use the **cryptocfg --show -groupc** command to look at the end of the output for the SP state showing **Online** (see Example 3-26). This state indicates that the MK was created and exported successfully.

Example 3-26 Command to confirm that the MK was accepted by the keystore

```
Encryption Group Config state: In Sync
```

Node Name	IP address	Role
10:00:00:05:1e:54:17:10	10.18.235.54	GroupLeader (current node)
EE Slot:	0	
SP state:	Online	

Alternatives: You also can export this MK to a file or a supported smart card using DCFM.

Preferred practice: Use supported smart cards and DCFM to keep at least two copies of the MK in a safe place.

Nevertheless, after the MK is exported to the TKLM servers, it is also backed up with the TKLM server backup/restore functions.

On the TKLM server, you can display the certificates and also list the keys by using the **wsadmin** command. Example 3-27 shows the command to display all keys belonging to the BRCD_ENCRYPTOR device group. Within this list, you can identify via the alias entry the previously generated and exported MK from the EE's GL. You can see its state and the type of underlying key algorithm (in this case, it is Advanced Encryption Standard symmetric key).

Example 3-27 Command to display all keys within the BRCD_ENCRYPTOR device group

```
wsadmin>print AdminTask.tklmKeyList('[-usage BRCD_ENCRYPTOR]')
CTGKM0001I Command succeeded.
```

```
uuid = KEY-b3db8115-7124-494b-bec7-e759dba6c92a
alias = b10_00_00_05_1e_54_17_10_00_00_00_00_00_00_00
key algorithm = AES
key store name = defaultKeyStore
key state = ACTIVE
creation date = 7/15/11 12:38:23 PM PDT
expiration date = null
```

```
uuid = KEY-5a5c04b8-7b04-454a-8e6c-d5c539f67337
alias = be5_ec_ce_12_c8_33_a2_04_a7_17_ab_73_2c_83_9f_a4
key algorithm = AES
key store name = defaultKeyStore
key state = ACTIVE
creation date = 7/15/11 12:45:06 PM PDT
expiration date = null
```

```
uuid = KEY-88090ebe-c285-4a86-aa28-15105351c21b
alias = be5_ec_ce_12_c8_33_a2_04_a7_17_ab_73_2c_83_9f_a5
key algorithm = AES
key store name = defaultKeyStore
key state = ACTIVE
creation date = 7/15/11 12:45:06 PM PDT
expiration date = null
```

wsadmin: The TKLM CLI provides additional commands to enhance the functions in the TKLM GUI. The commands use the **wsadmin** interface that is provided by WebSphere Application Server. You can invoke **wsadmin** commands in batch or interactive mode, in a script, or from a command prompt using the **wsadmin -c** command. Use the IBM Tivoli Key Lifecycle Manager Information Center to see the complete command reference information for the **wsadmin** command-line tool of the TKLM server. Use this link for direct access to this reference:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tklm.doc_2.0/ref/ref_ic_cli.html

3.3.7 Setting up the SAN32B-E4 Encryption Switch and TKLM using the DCFM GUI

In this topic, we show the generic steps of the initial setup of the SAN32B-E4 Encryption Switch to the TKLM servers using the DCFM GUI. You can execute all the previously discussed individual Encryption Switch CLI commands via the DCFM GUI (refer to 3.3.6, "Setting up the SAN32B-E4 Encryption Switch and TKLM using the switch CLI" on page 50). Certain individual commands via the CLI are combined in one task or a wizard in the GUI.

DCFM: The DCFM is an additional software component. It is optional and priced separately. It is not provided with the delivery of the SAN32B-E4 Encryption Switch.

Using the DCFM GUI is advantageous in that it is more tolerant of human errors, especially typographical errors. Nevertheless, using the CLI has an advantage over the DCFM GUI, because it is easier to understand each step. Also, following the sequence of steps in command mode is more logical and intuitive and helps you to understand each step's part of the overall context.

Important: The sequence of steps provided in this chapter do not include the activities that you must perform at the TKLM servers. The order of steps in this chapter does not mean it is the order of the overall execution to set up TKLM communication with the Encryption Switch. It only shows which tasks of the Encryption Switch setup via CLI from 3.3.6, “Setting up the SAN32B-E4 Encryption Switch and TKLM using the switch CLI” on page 50 can be handled via the DCFM GUI.

Figure 3-19 shows the main window of the DCFM GUI.

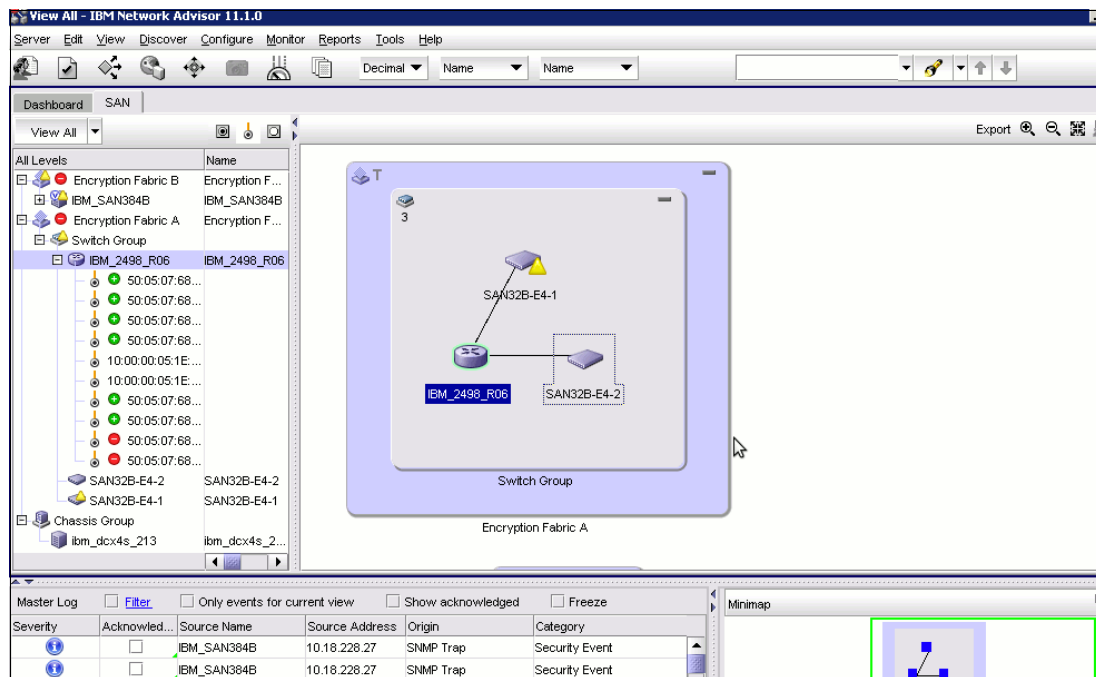


Figure 3-19 DCFM main window

From this window, you can navigate to the DCFM Encryption Center (see Figure 3-20 on page 73 and Figure 3-21 on page 73) where you can select and use all the specific functions that relate to encryption.

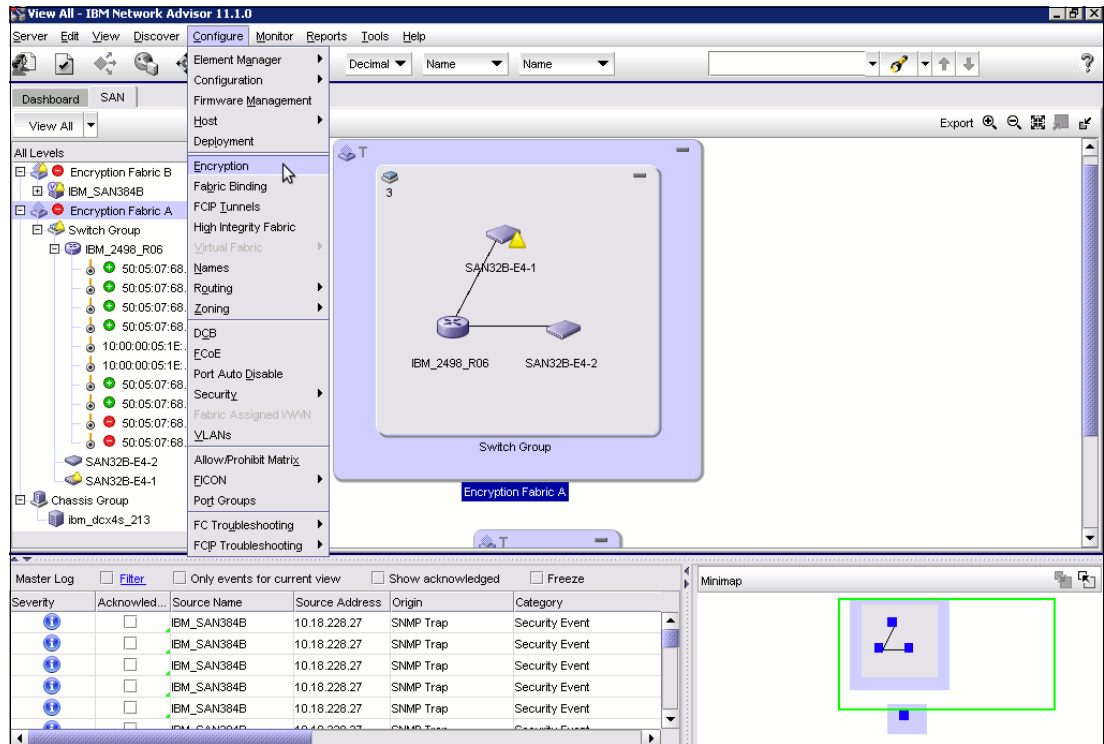


Figure 3-20 DCFM: How to start the Encryption Center

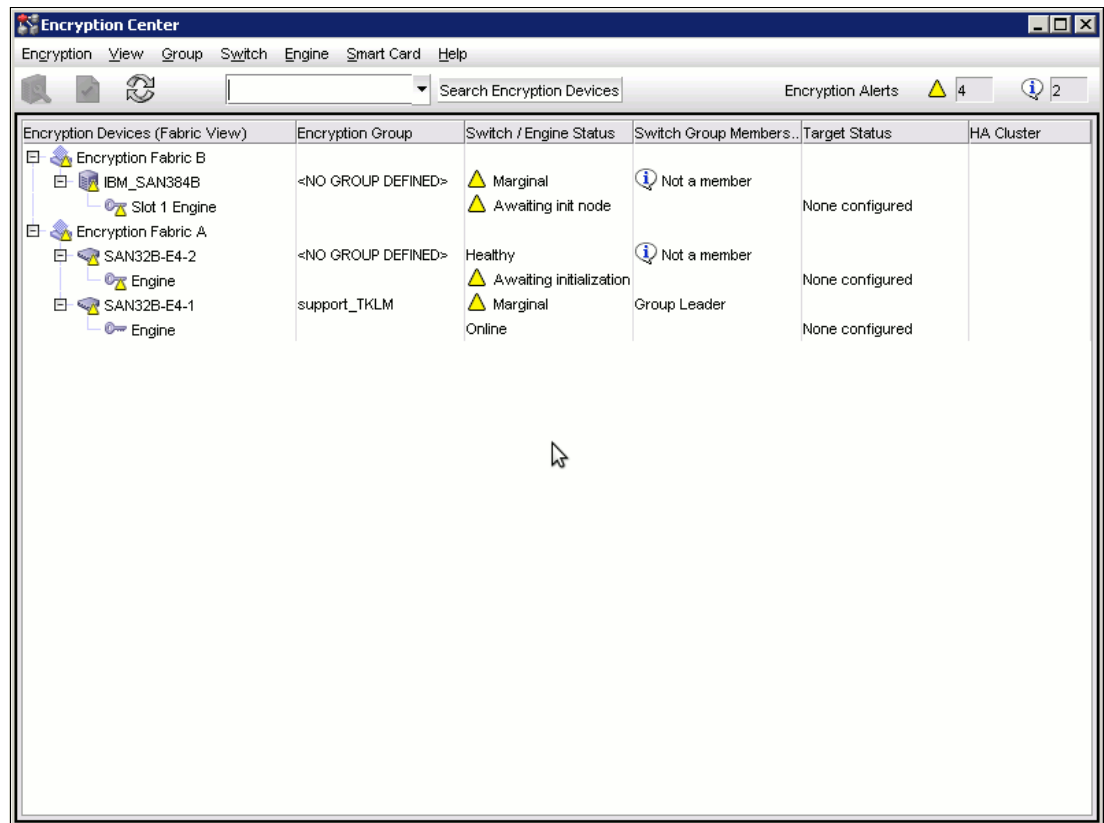


Figure 3-21 DCFM Encryption Center window

SAN32B-E4 Encryption Switch: Zeroizing the Encryption Engine via DCFM

This step relates to “Step 6: SAN32B-E4 Encryption Switch: Zeroizing the EE” on page 58 using the switch CLI interface. You need to perform this task on all members of the EG: the GL *and* the member nodes.

The following two DCFM windows show how to “zeroize” the EE (Figure 3-22 and Figure 3-23 on page 75) via the DCFM GUI.

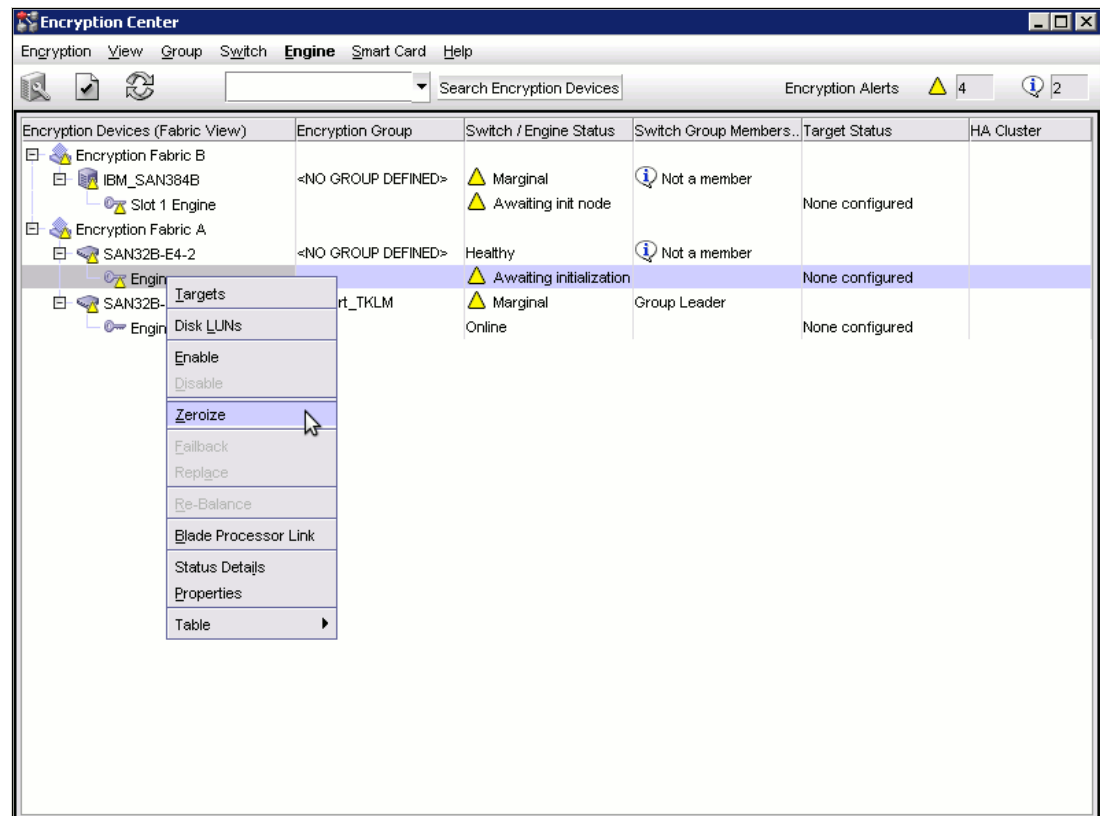


Figure 3-22 How to zeroize the EE from the DCFM (step 1 of 2)

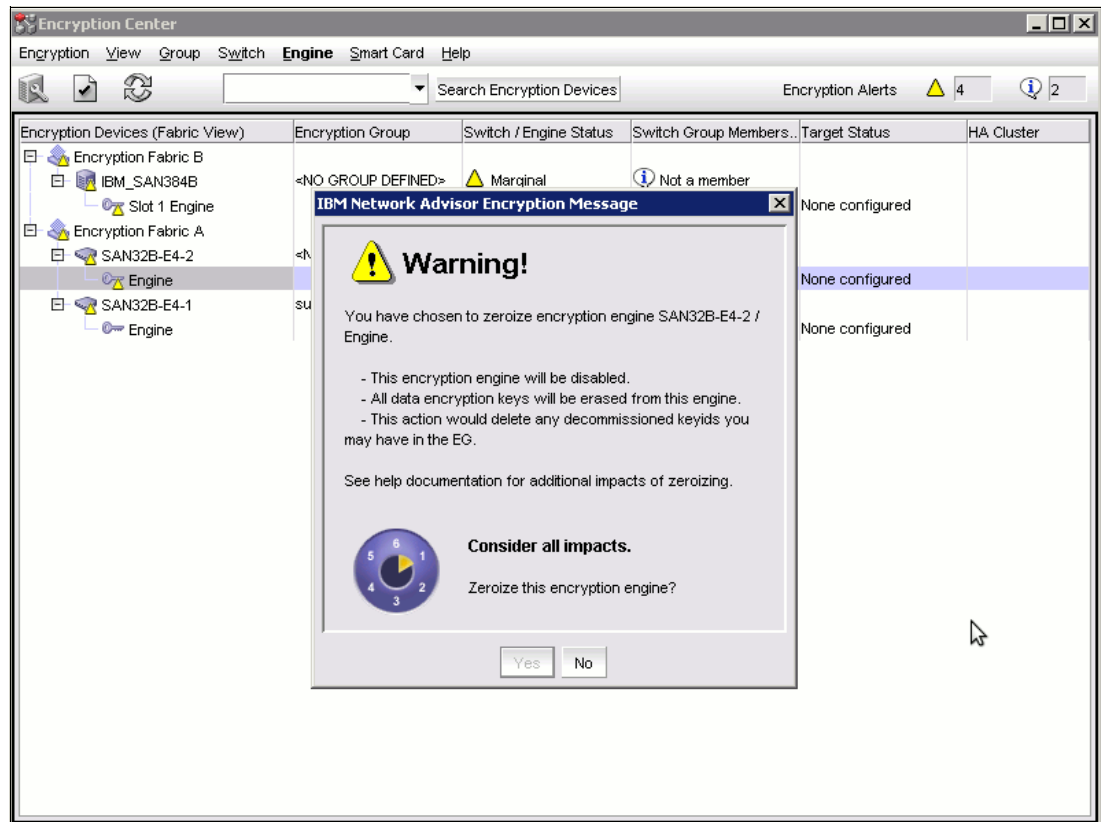


Figure 3-23 How to zeroize the EE from DCFM (step 2 of 2)

SAN32B-E4 Encryption Switch: Initializing the encryption node via DCFM

The following action relates to “Step 7: SAN32B-E4 Encryption Switch: Initializing the encryption node” on page 59 and will initialize the encryption node. Perform this task on the GL *and* the member nodes. Remember, this step will also internally create the Node CP certificate and the KAC certificate on the encryption nodes. Figure 3-24 on page 76 and Figure 3-25 on page 77 show how to invoke this task.

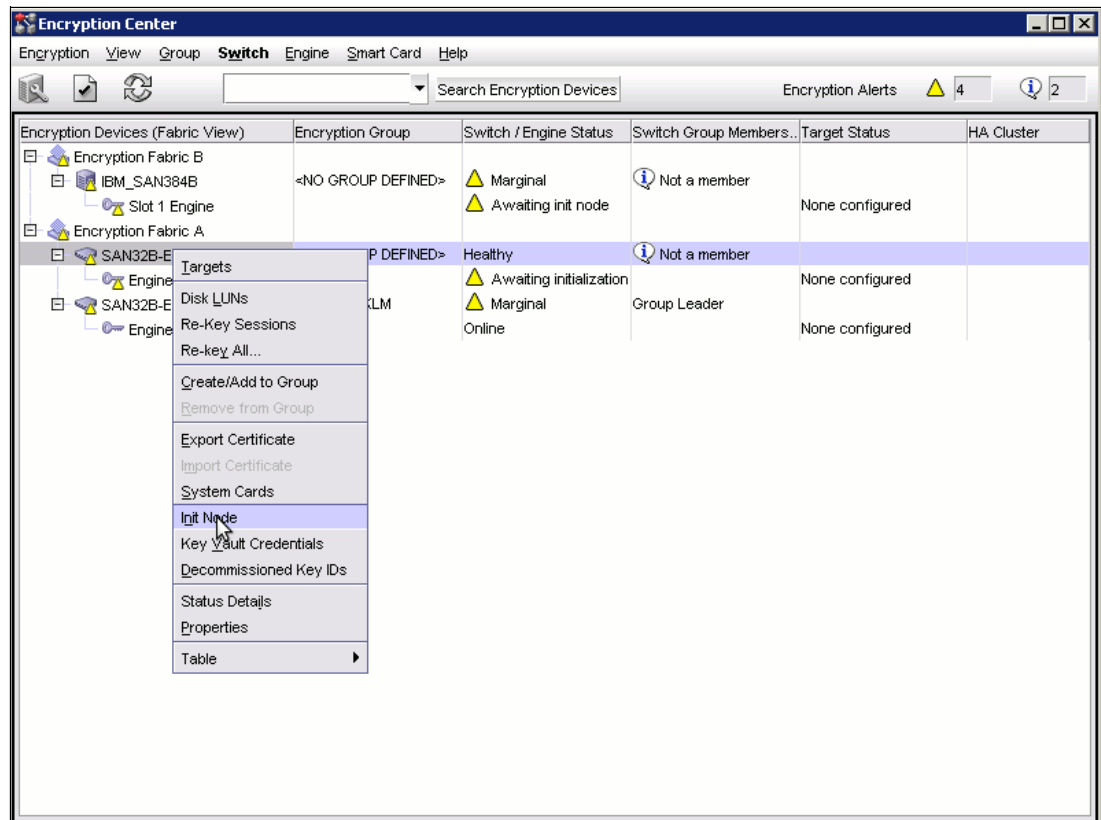


Figure 3-24 How to initnode the EE from DCFM (step 1 of 2)

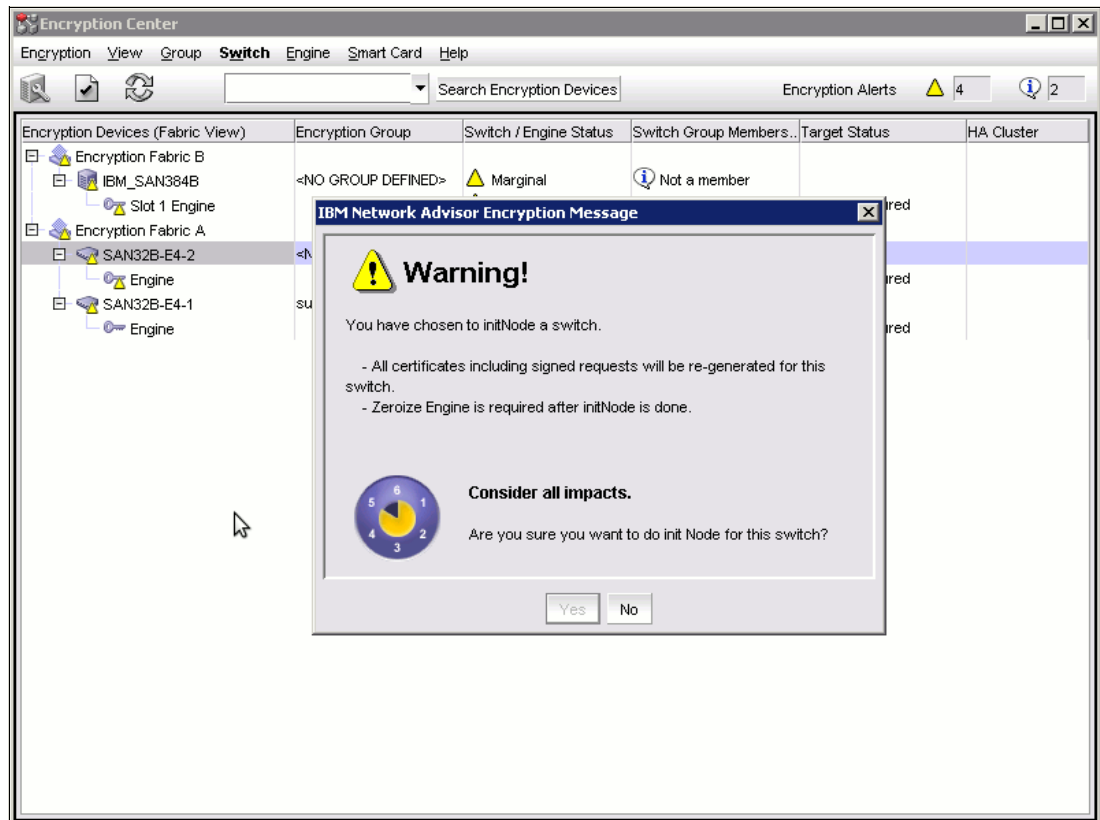


Figure 3-25 How to initnode the EE from DCFM (step 2 of 2)

SAN32B-E4 Encryption Switch: Creating the encryption group via DCFM

The EG is created on the node that is designated as the group leader (GL). A GL is a special node within an EG that acts as a group and cluster manager. The GL manages and distributes all group-wide and cluster-wide configurations to all members of the group or cluster. This step mainly relates to the CLI command on “Step 8: SAN32B-E4 Encryption Switch: Creating the encryption group” on page 59, but this step includes other previous steps in one window that is presented to the user (wizard) to create a new EG:

- ▶ Create EG (“Step 8: SAN32B-E4 Encryption Switch: Creating the encryption group” on page 59)
- ▶ Set key vault type (“Step 11: SAN32B-E4 Encryption Switch: Setting the key vault type” on page 60)
- ▶ Get the Encryption Switch discrete device serial number (“Step 13: SAN32B-E4 Encryption Switch: Obtaining the discrete device serial number” on page 61)
- ▶ Export and save the KAC certificate of the Encryption Switch (“Step 15: Transferring the EE nodes’ KAC certificates to the TKLM servers” on page 64)
- ▶ Init, register, and enable EE (“Step 12: Initializing, registering, and enabling the encryption engine” on page 61)
- ▶ Import and register the TKLM self-signed server certificates (“Step 17: Importing the TKLM server certificates on the GL node” on page 67 and “Step 18: Registering the TKLM server certificates on the GL node” on page 68)

Figure 3-26 on page 78 through to Figure 3-32 on page 81 show the progress through the wizard.

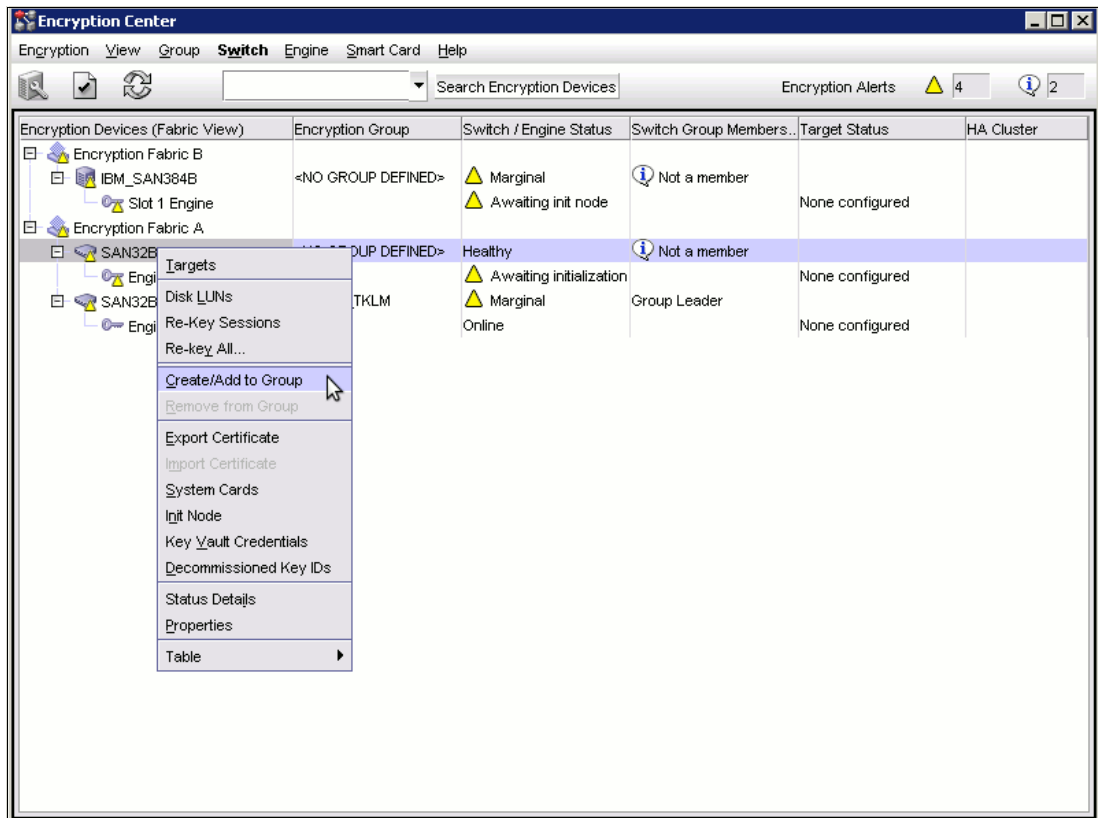


Figure 3-26 DCFM Create a new Encryption Group selection window

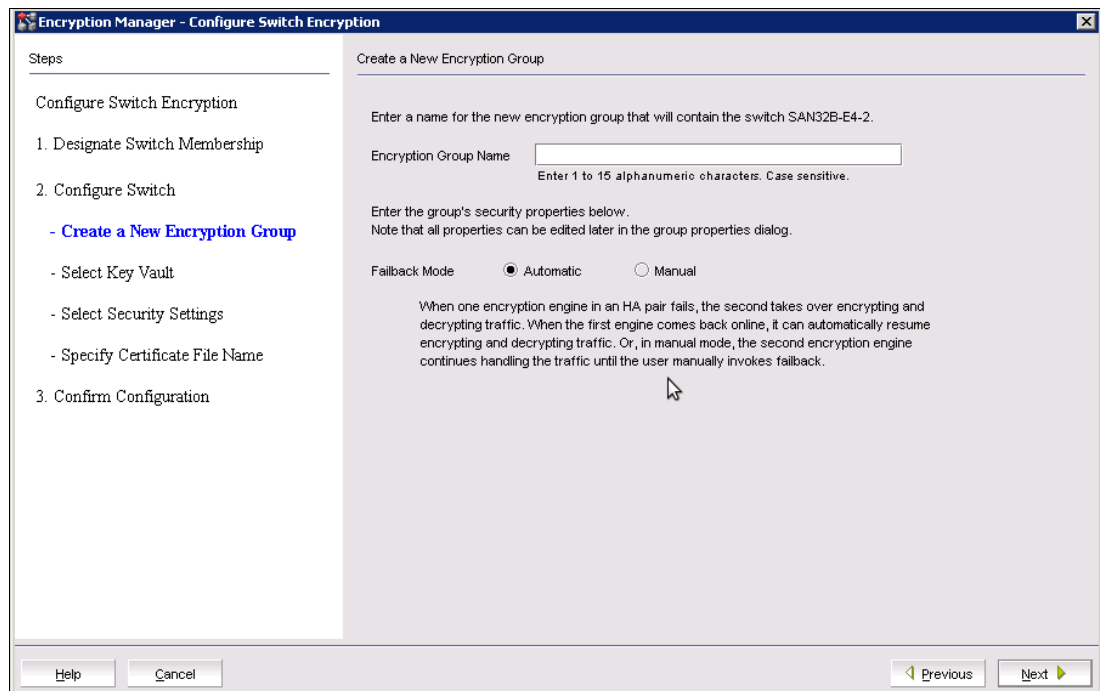


Figure 3-27 DCFM Create a New Encryption Group: Enter the Encryption Group Name

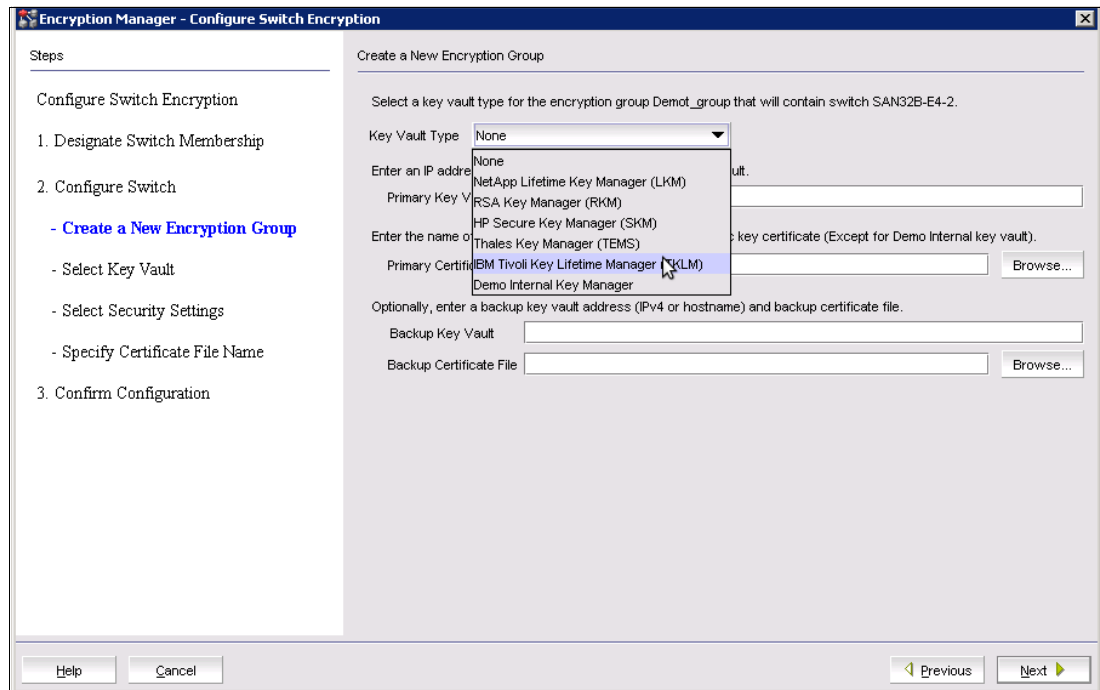


Figure 3-28 DCFM Create a New Encryption Group: Select the Key Vault Type

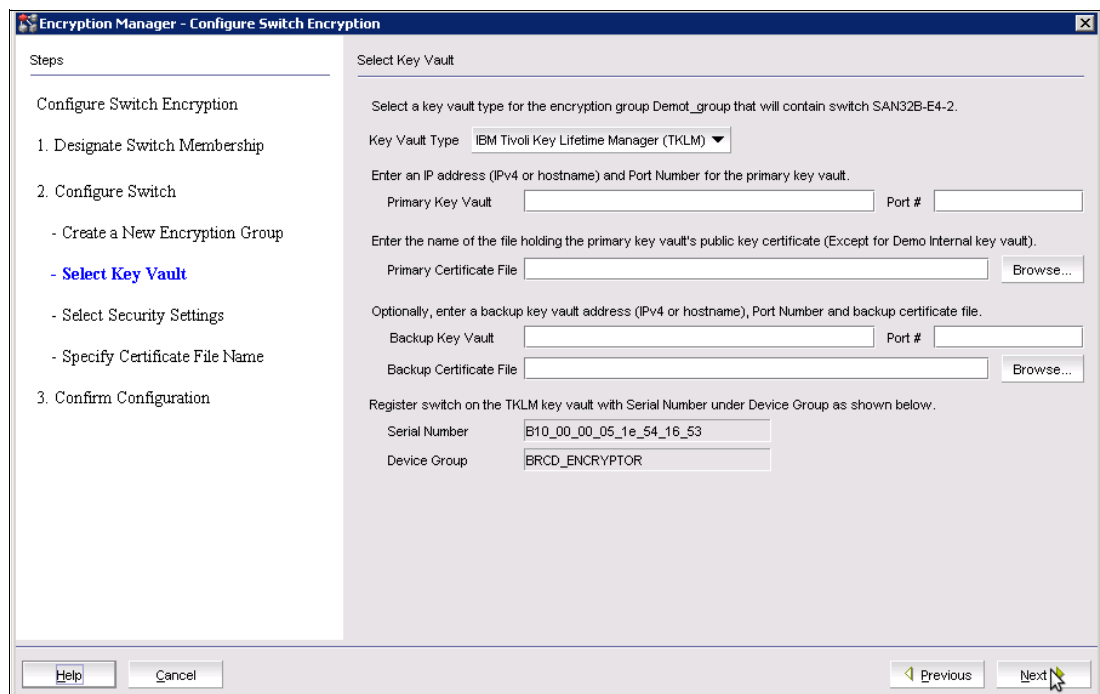


Figure 3-29 DCFM Create a New Encryption Group: Select key vault and get switch serial number

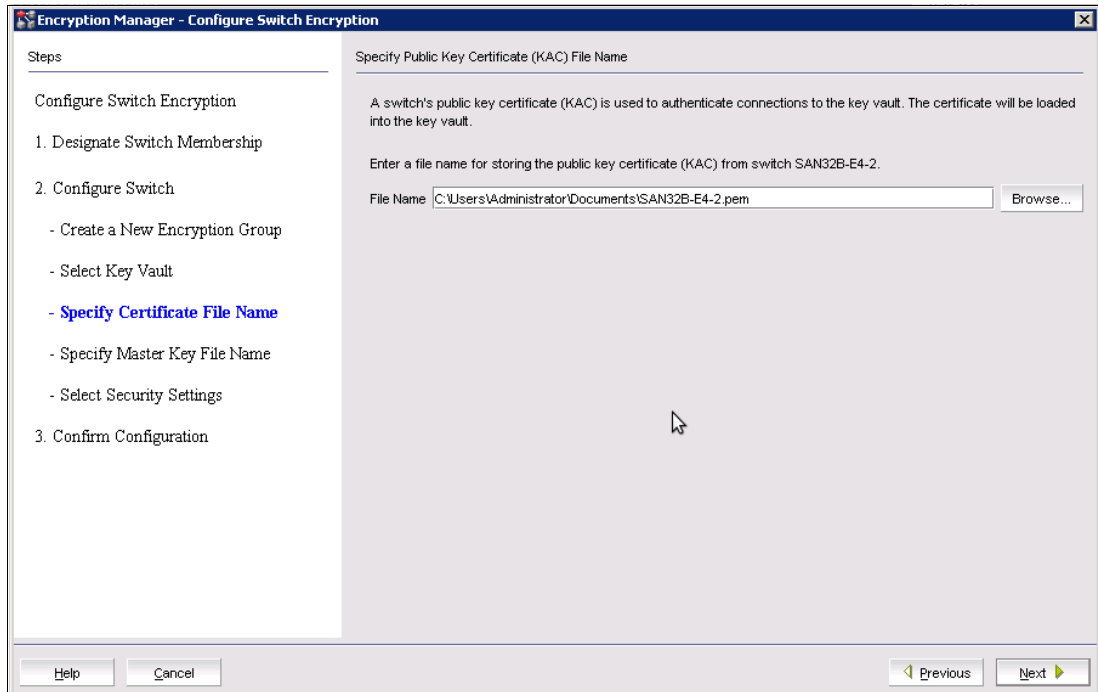


Figure 3-30 DCFM Create a New Encryption Group: Specify name to store switch KAC certificate

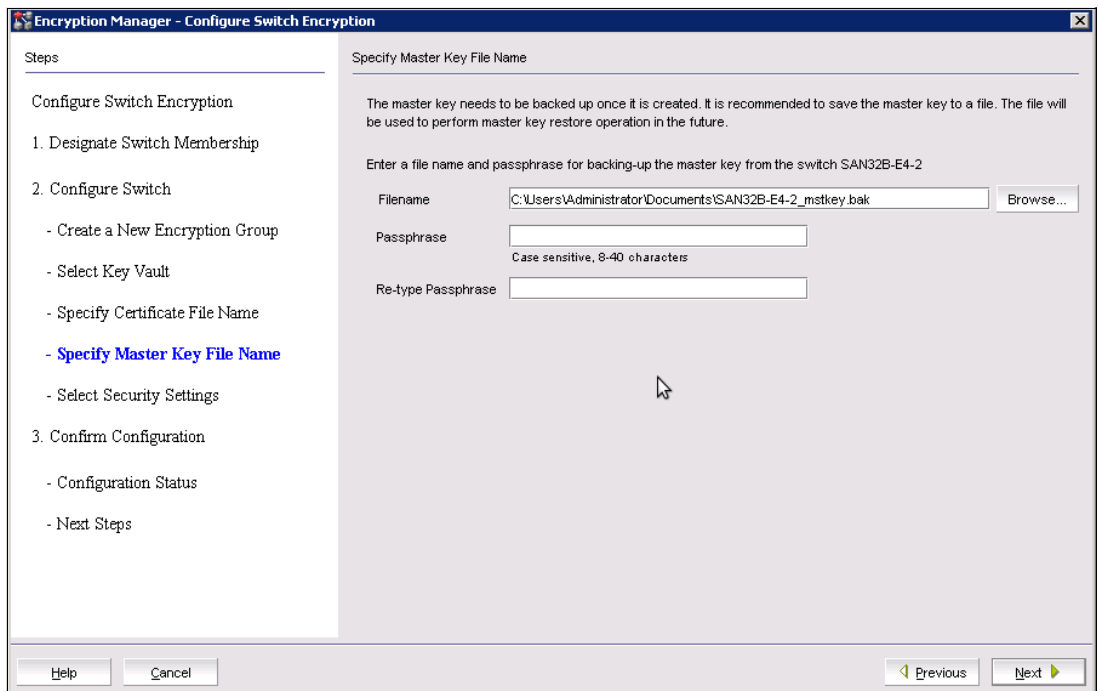


Figure 3-31 DCFM Create a New Encryption Group: Specify a name to back up the switch MK

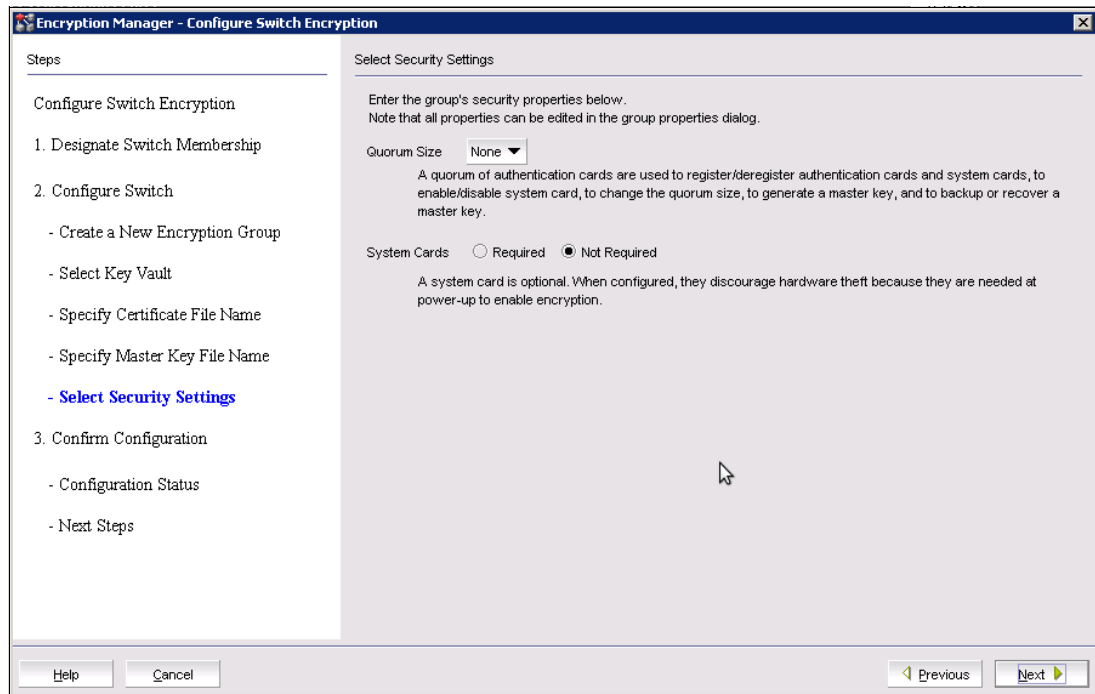


Figure 3-32 DCFM Create a New Encryption Group: Select Security Settings

SAN32B-E4 Encryption Switch: Registering member nodes via DCFM

In this step, we register the other encryption nodes and import the node CP certificates from the other encryption member nodes, if there are any more encryption nodes. Perform this task *only* on the GL. This task relates to “Step 9: SAN32B-E4 Encryption Switch: Importing the node certificates” on page 59 and “Step 10: SAN32B-E4 Encryption Switch: Registering member nodes on the GL node” on page 60. If using the DCFM wizard to add a switch to an EG, the DCFM wizard will guide you automatically through the sequence of steps.

Remember that the CP certificate is created during node initialization. This certificate is exchanged with the GL, and it used for authenticating the member nodes to the GL. The location of the CP certificate file is `/etc/fabos/certs/sw0/my_cp_cert.pem` within the switch FOS. This name is a fixed name that is generated by the system, and it is not a name that the user can assign or change. If you use the DCFM GUI, the GL node also imports this file directly from the member nodes, bypassing the need to use an intermediate SSH server.

Figure 3-33 on page 82 through Figure 3-35 on page 83 guide you through the sequence of adding an encryption member node to an existing EG. To invoke the wizard, you start via the DCFM Encryption Center view with a right mouse click on the switch that you want to add. This process is similar to the process that you used to create a new EG (as shown in Figure 3-26 on page 78).

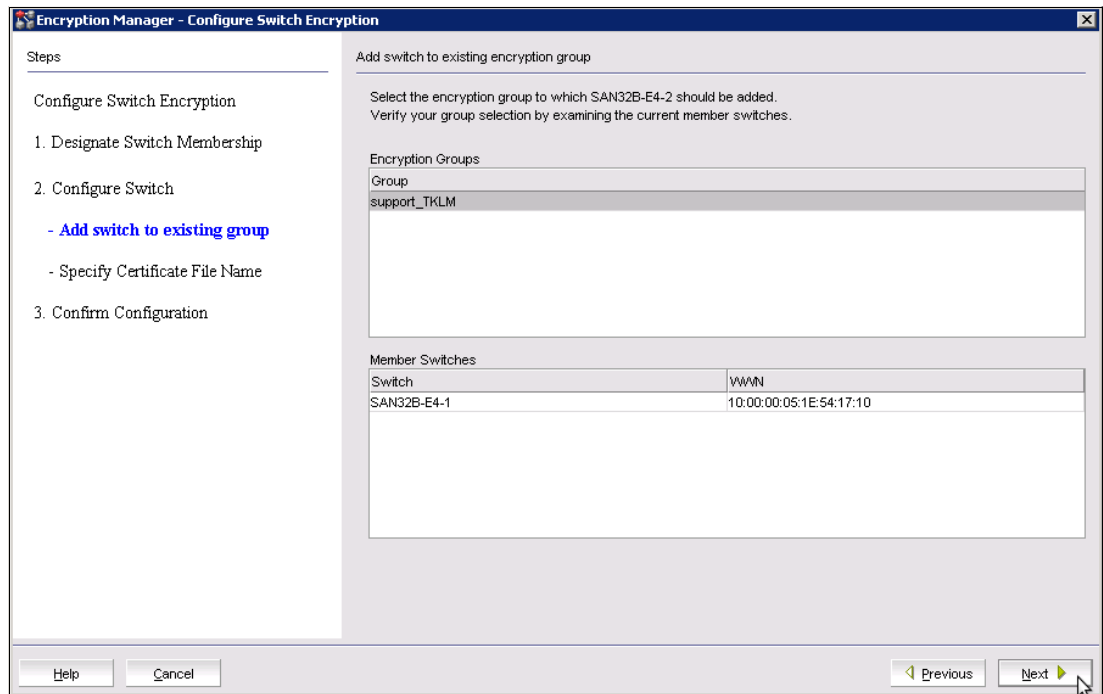


Figure 3-33 DCFM wizard to add an Encryption Switch to an existing encryption group

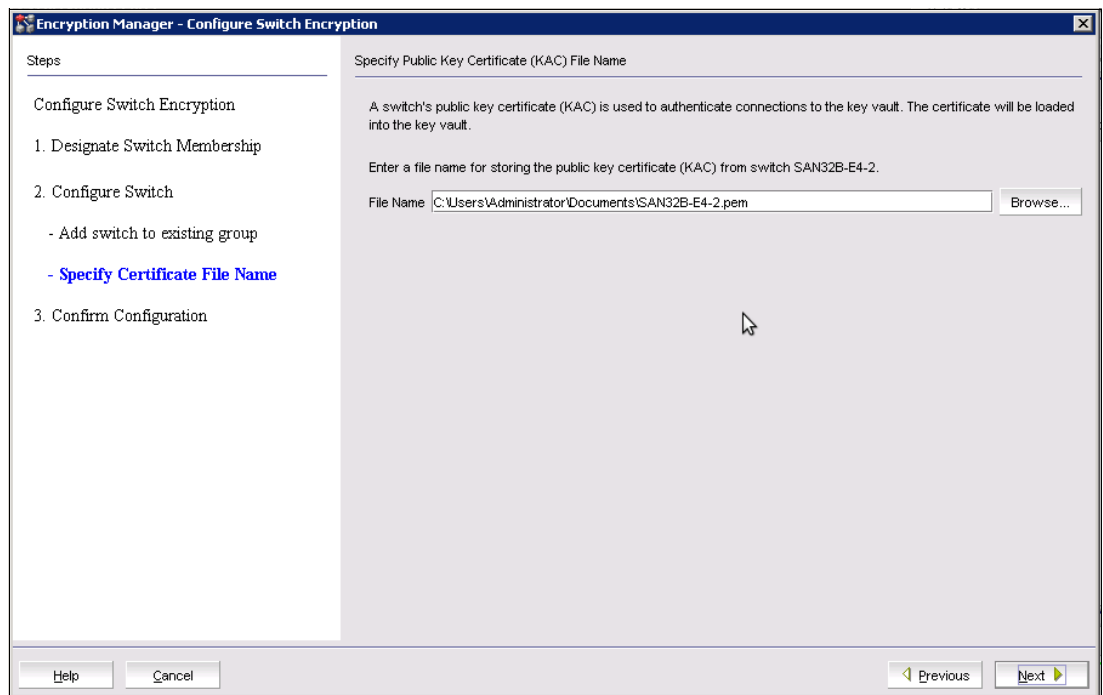


Figure 3-34 DCFM wizard to add an Encryption Switch to an existing encryption group

You get a final confirmation window, as shown in Figure 3-35 on page 83.

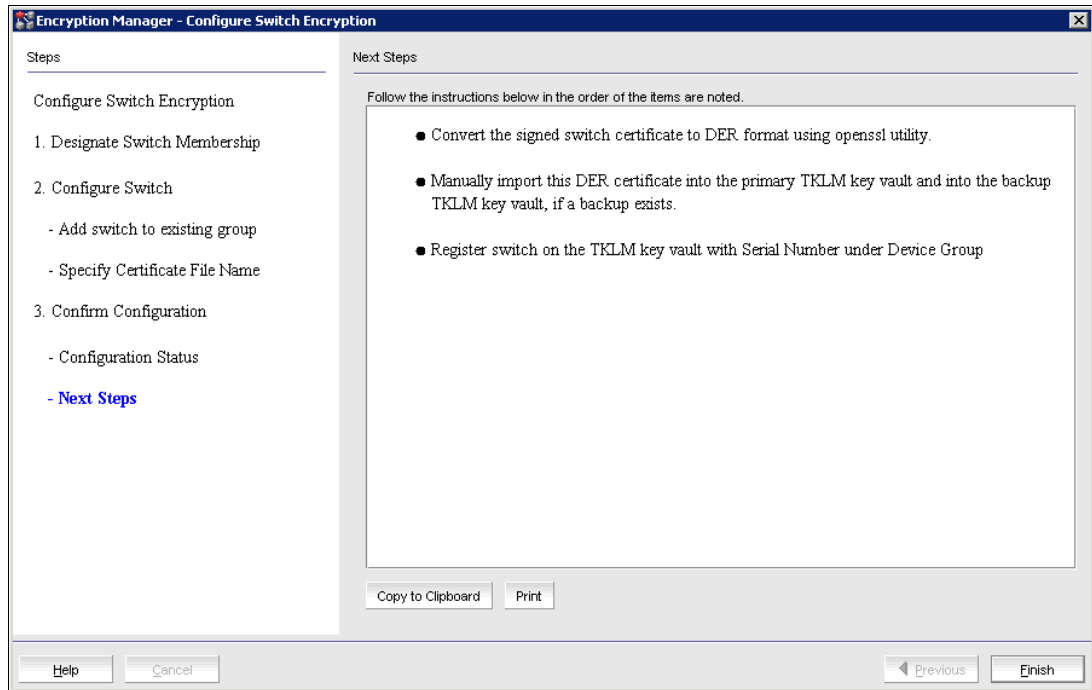


Figure 3-35 DCFM wizard window after adding an Encryption Switch to the existing encryption group

Important: Although the final DCFM wizard window, which is shown in Figure 3-35 on page 83, instructs you, after you add an encryption node, to convert the exported switch certificate (KAC) to DER format using the **openssl** utility, *ignore this instruction*. The TKLM server can import the provided certificate file immediately, as described in “Step 16: Importing the KAC certificates to the TKLM servers” on page 65.

Important: The final DCFM wizard window, which is shown in Figure 3-35 on page 83, also instructs you to register the newly added encryption member node to the TKLM servers with its serial number. Remember that this number is not the raw serial number; it is the discrete device serial number in a format as `x_y_z_a_b_c_d_e_f_g`.

Unfortunately, in this step, the DCFM GUI does not show it on the window, so at this point you must return to the new encryption member CLI and use the **cryptocfg --reg -KAClogin** command, as described in “Step 13: SAN32B-E4 Encryption Switch: Obtaining the discrete device serial number” on page 61, to get the discrete device serial number. With this unique discrete device serial number string, you need to register the new EG member on each TKLM server within the device group `BRCD_ENCRYPTOR`.

SAN32B-E4 Encryption Switch: Verifying TKLM connectivity via DCFM

You need to verify the successful communication between the encryption nodes and the TKLM servers via the DCFM GUI (Encryption Center window). Navigate from the selected Encryption Switch using a right mouse click to **Properties** to display the following window (Figure 3-36 on page 84). Look for the highlighted entries, which we have highlighted, about the primary and secondary key vault connection status. The status must be Connected.



SAN32B-E4 Encryption Switch: Generating the master key via DCFM

In this final step, we show how to generate the MK via the DCFM GUI. Again, start at the Encryption Center window. Use a right mouse click on the Encryption Switch (most likely the GL) to navigate to the **Status Details** view. Continue to a new window by using **Show**. Click the **Security** tab. Under the Master Key Actions list box, select **Create New Master Key** to generate an initial MK. This MK is automatically distributed to the TKLM servers during this process for archival.

Figure 3-37 on page 85 through Figure 3-40 on page 88 show this process.

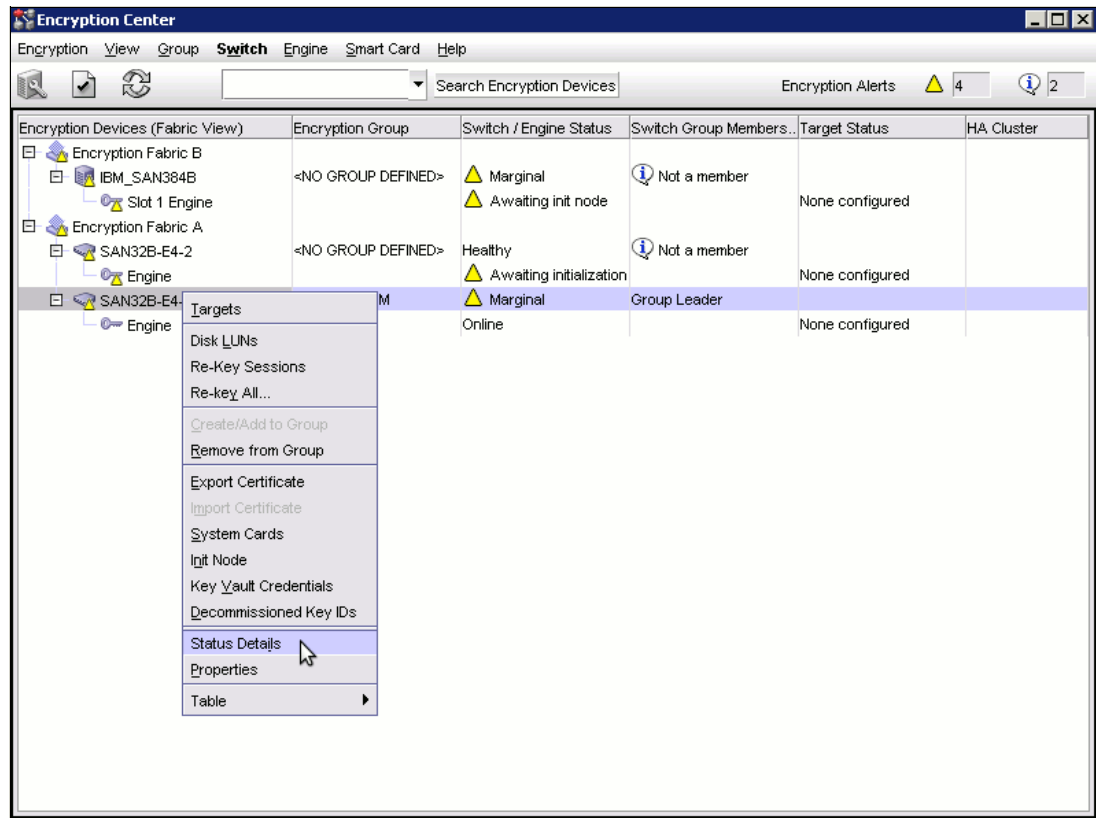


Figure 3-37 DCFM: Generating the encryption group's MK (step 1 of 3)

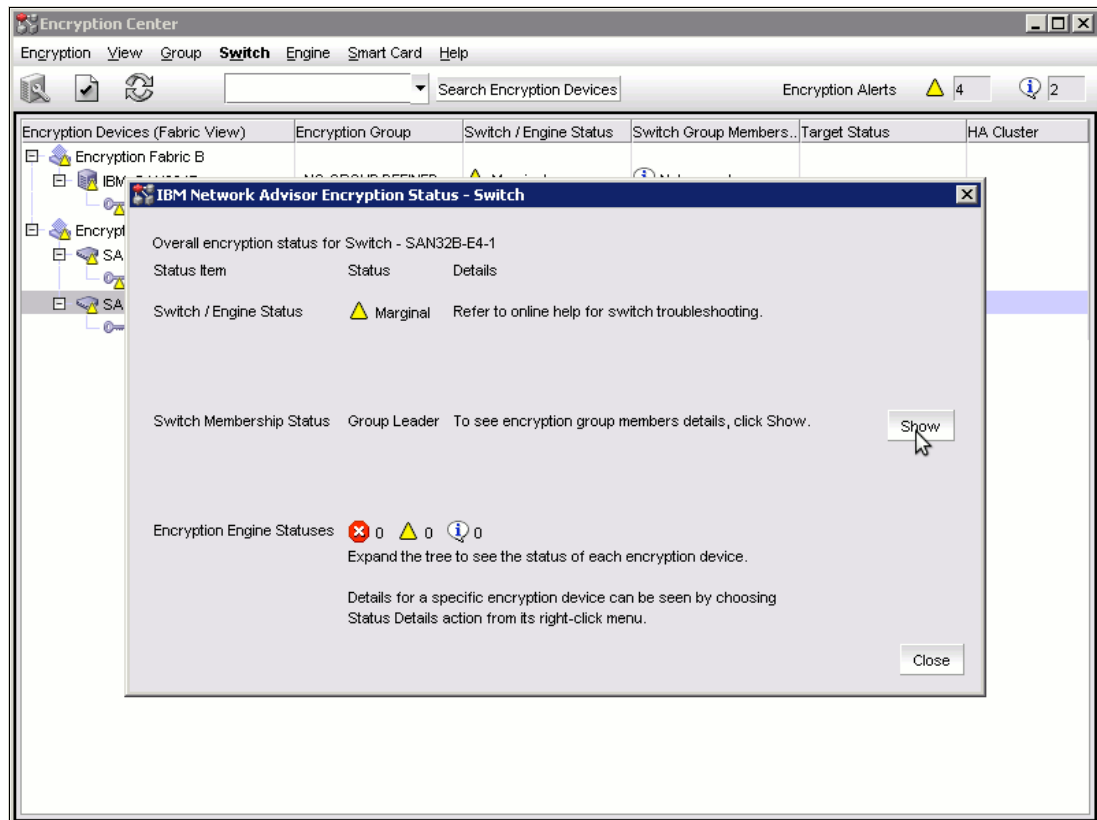


Figure 3-38 DCFM: Generating the encryption group's MK (step 2 of 3)

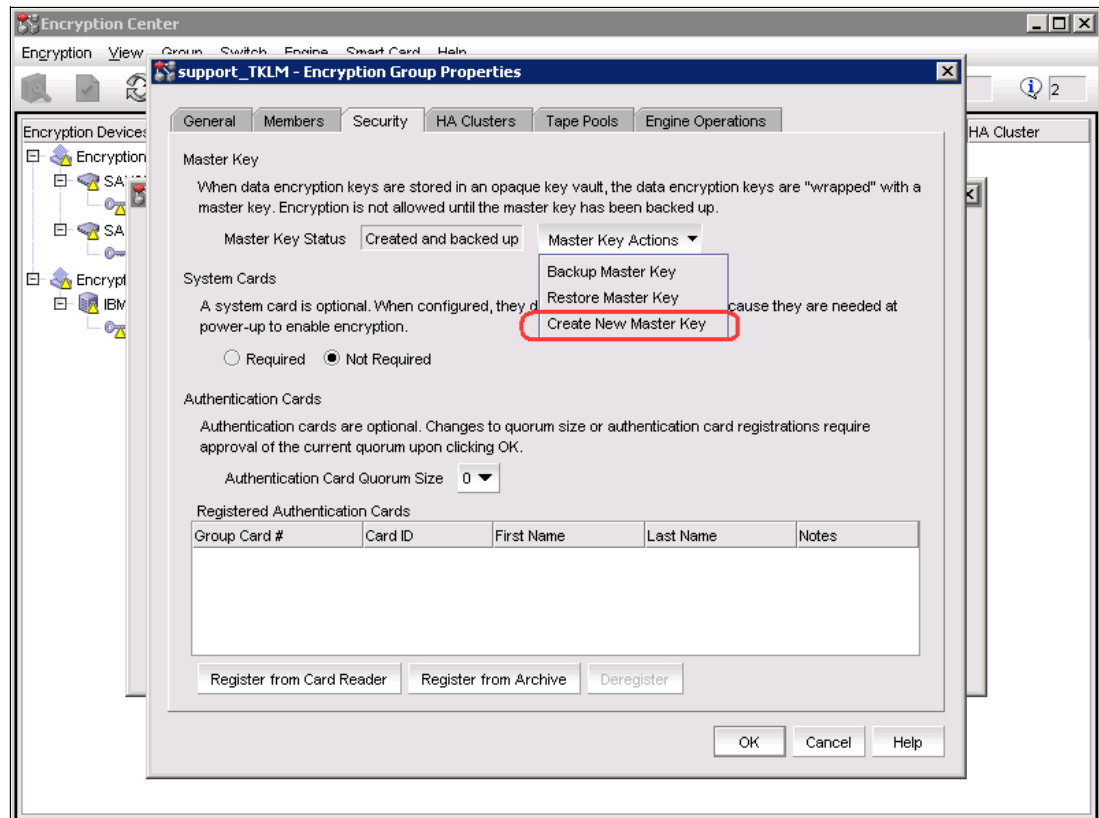


Figure 3-39 DCFM: Generating the encryption group's MK (step 3 of 3)

You also can use the same window to check the MK's status. If an MK exists, the window shows the message "Created and backed up" in the Master Key text box, as shown in Figure 3-40 on page 88.

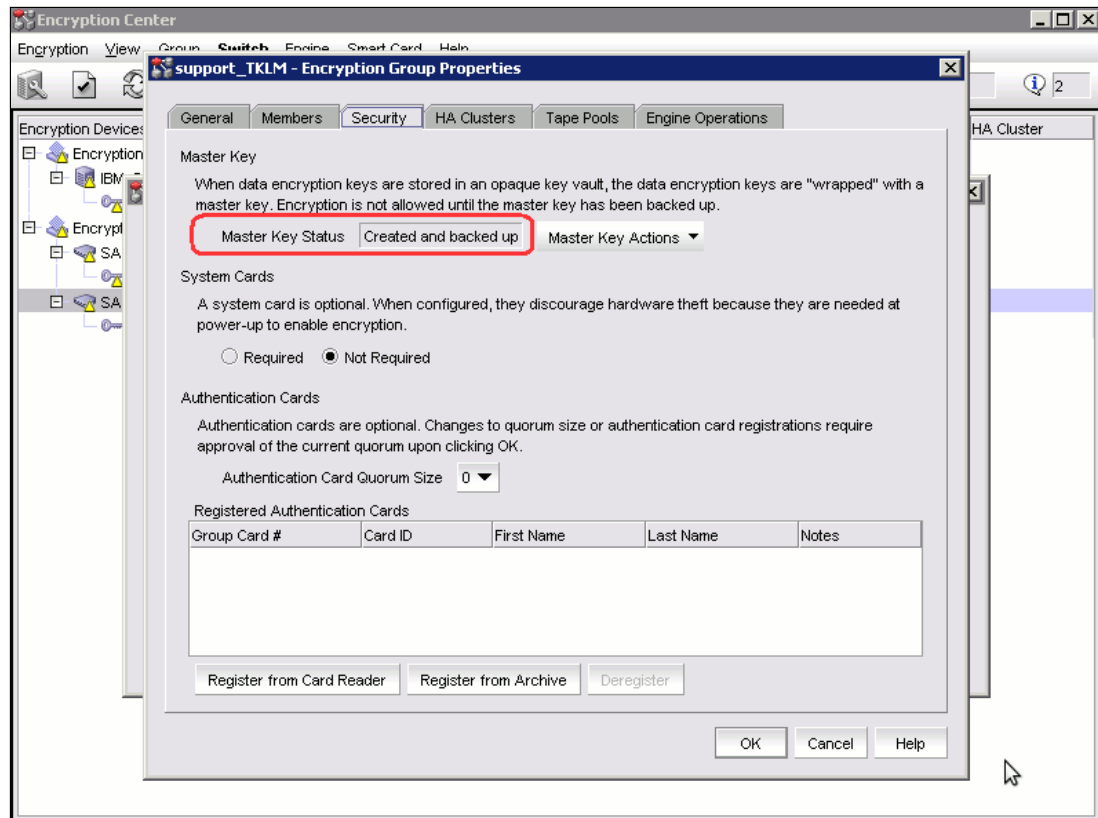



Figure 3-40 DCFM: Verifying the encryption group's MK

We have successfully set up and established communication between the IBM TKLM servers and the EEs. This task is mandatory and a key prerequisite for all upcoming processes.



Implementation scenarios and recommendations for managing the SAN32B-E4 Encryption Switch

In this chapter, we show the steps that we used to encrypt existing data on a host logical unit number (LUN). We initially perform this encryption on a single-fabric scenario. Installation on a single fabric is not a common requirement, but we chose it to show enabling encryption in its simplest form.

We then show the steps in a more typical, dual-fabric scenario, pointing out the additional requirements to be considered.

4.1 Configuring encryption on a single fabric

In this section, we show our existing configuration of a host to target LUN via a single fabric only. We will then show the steps that we took to encrypt this target LUN containing existing data. For this single fabric encryption process, we use command-line interface (CLI) commands.

4.1.1 Current configuration

In Figure 4-1, we see our current configuration of our host server accessing its cleartext data that resides on a LUN, which is provisioned from the Storwize V7000 over a single SAN fabric. We have two connections to each of the Storwize V7000 node canisters, which provides four paths to the target LUN0.

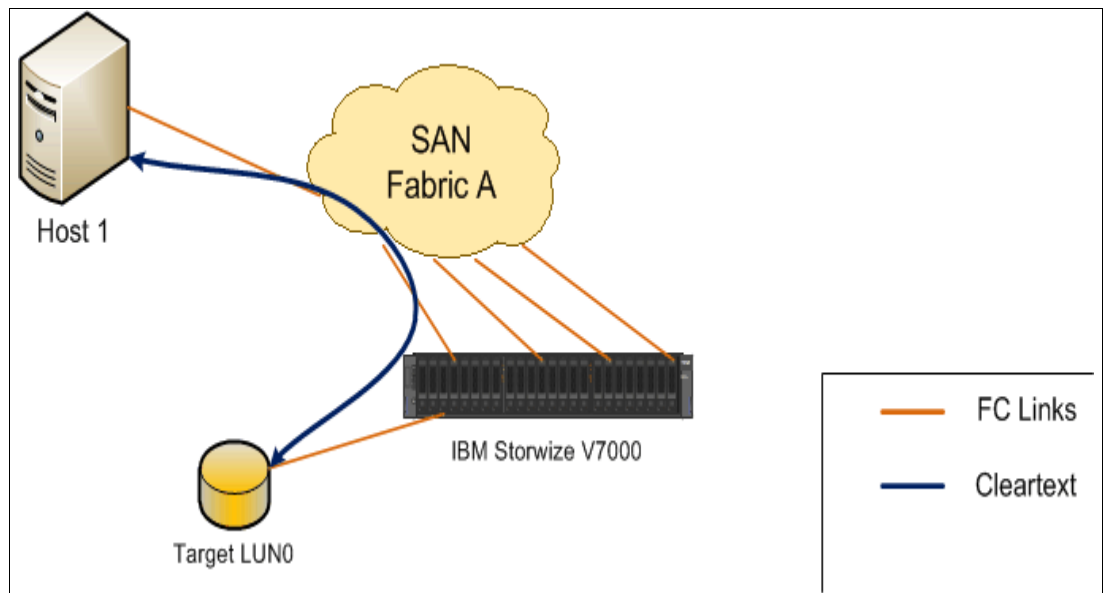


Figure 4-1 Single fabric environment

In Figure 4-6 on page 101, we confirm the definitions of the Storwize V7000 for our host (win2k8-1) and that it has access to a single LUN (win2k8_Lun0) via a single Fibre Channel (FC) port or host bus adapter (HBA).

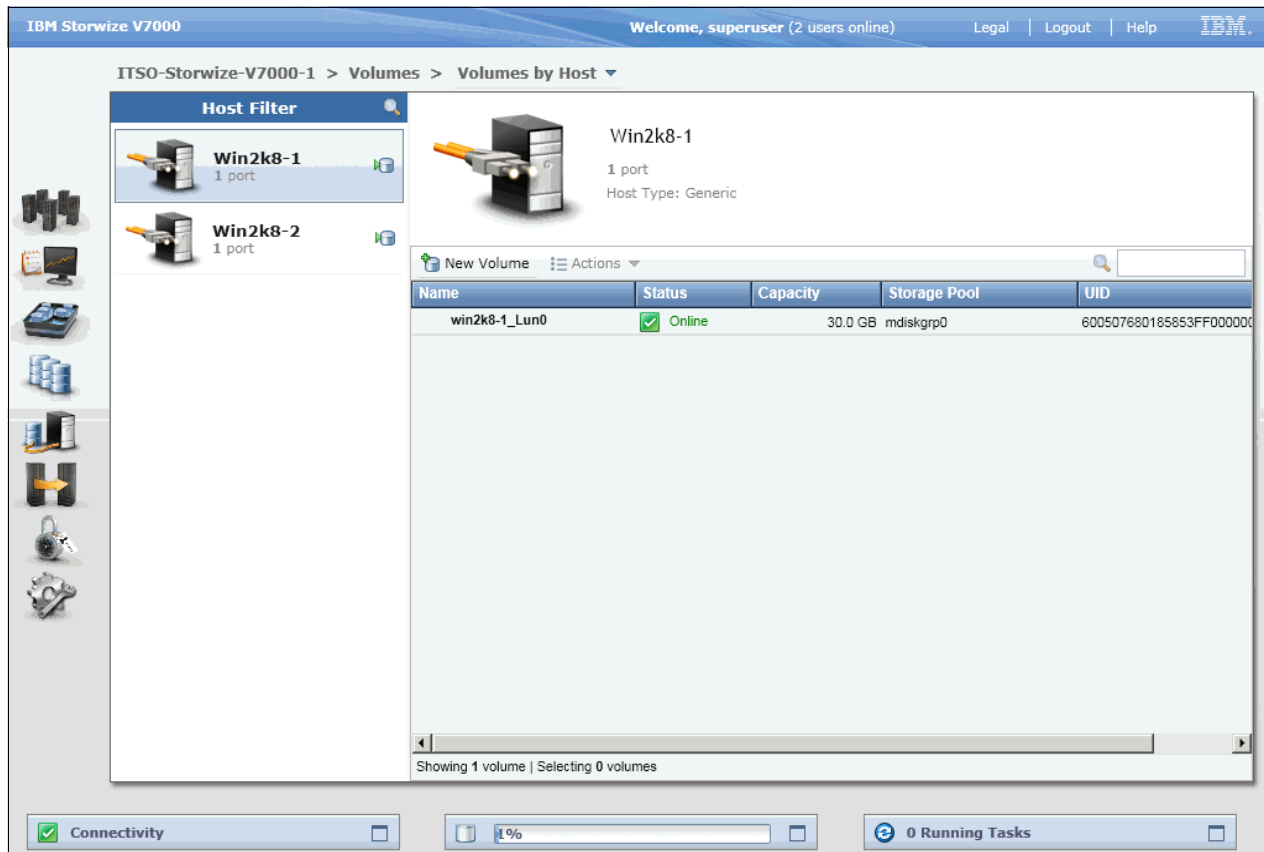


Figure 4-2 Viewing LUN assignment to our single path host

Our host and target Storwize V7000 ports attach to a single SAN switch. From the **switchshow** command output in Example 4-1, you can see that ports 4, 5, 6, and 7 have the Storwize V7000 ports attached (two ports from each node canister), port 9 has our host HBA port attached, and port 8 has another HBA connected (to use in later scenarios). Ports 0 and 1 form an inter-switch link (ISL) trunk to our SAN32B-E4 Encryption Switch that will be implemented in the coming procedure.

Example 4-1 Fabric A switchshow output

```
IBM_2498_R06:admin> switchshow
switchName:   IBM_2498_R06
switchType:   83.3
switchState:  Online
switchMode:   Native
switchRole:   Subordinate
switchDomain: 3
switchId:     fffc03
switchWwn:   10:00:00:05:1e:c3:be:29
zoning:       ON (ITS0_SG247977)
switchBeacon: OFF
FC Router:    OFF
FC Router BB Fabric ID: 1
Address Mode: 0
```

```
Index Port Address Media Speed State      Proto
=====
```

0	0	030000	id	N8	Online	FC	E-Port	(Trunk port, master is
Port 1)								
1	1	030100	id	N8	Online	FC	E-Port	10:00:00:05:1e:54:17:10
"SAN32B-E4-1" (downstream)(Trunk master)								
2	2	030200	id	N8	No_Light	FC		
3	3	030300	id	N8	No_Light	FC		
4	4	030400	id	N8	Online	FC	F-Port	50:05:07:68:02:30:a7:fe
5	5	030500	id	N8	Online	FC	F-Port	50:05:07:68:02:40:a7:fe
6	6	030600	id	N8	Online	FC	F-Port	50:05:07:68:02:30:a7:be
7	7	030700	id	N8	Online	FC	F-Port	50:05:07:68:02:40:a7:be
8	8	030800	id	N8	Online	FC	F-Port	10:00:00:05:1e:c7:6b:a2
9	9	030900	id	N8	Online	FC	F-Port	10:00:00:05:1e:c7:6b:8a
10	10	030a00	--	N8	No_Module	FC		
11	11	030b00	--	N8	No_Module	FC		
12	12	030c00	id	N8	No_Light	FC		
13	13	030d00	id	N8	No_Light	FC		
14	14	030e00	id	N8	No_Light	FC		
15	15	030f00	id	N8	No_Light	FC		

The Storwize V7000 is zoned to the host or hosts to provide the required connectivity. Because we have a single host HBA to four target ports on the Storwize V7000, the host sees a single disk LUN with four paths, as seen in the SDDPCM output in Example 4-2.

Example 4-2 IBM SDDPCM pathing to V7000 LUN

```
C:\Program Files\IBM\SDDDSM>datapath query device
```

```
Total Devices : 1
```

DEV#:	0	DEVICE NAME:	Disk1 Part0	TYPE:	2145	POLICY:	OPTIMIZED
SERIAL:	600507680185853FF000000000000000						
=====							
Path#	Adapter/Hard Disk			State	Mode	Select	Errors
0	Scsi	Port12	Bus0/Disk1 Part0	OPEN	NORMAL	1511	0
1	Scsi	Port12	Bus0/Disk1 Part0	OPEN	NORMAL	0	0
2	Scsi	Port12	Bus0/Disk1 Part0	OPEN	NORMAL	271	0
3	Scsi	Port12	Bus0/Disk1 Part0	OPEN	NORMAL	0	0

By issuing an SDDPCM command **datapath query wwpn**, as shown in Example 4-3, we display the HBA worldwide port names (WWPNs) on our host. We will need the WWPNs later in this procedure.

Example 4-3 Displaying host WWPNs

```
C:\Program Files\IBM\SDDDSM>datapath query wwpn
```

Adapter Name	PortWWN
Scsi Port11:	100000051EC76B89
Scsi Port12:	100000051EC76B8A

```
C:\Program Files\IBM\SDDDSM>
```

In Figure 4-3 on page 93, we display the host's Disk Management window where our Storwize V7000 LUN is seen as Disk 1 and has a label of V7000-1_LUN0.

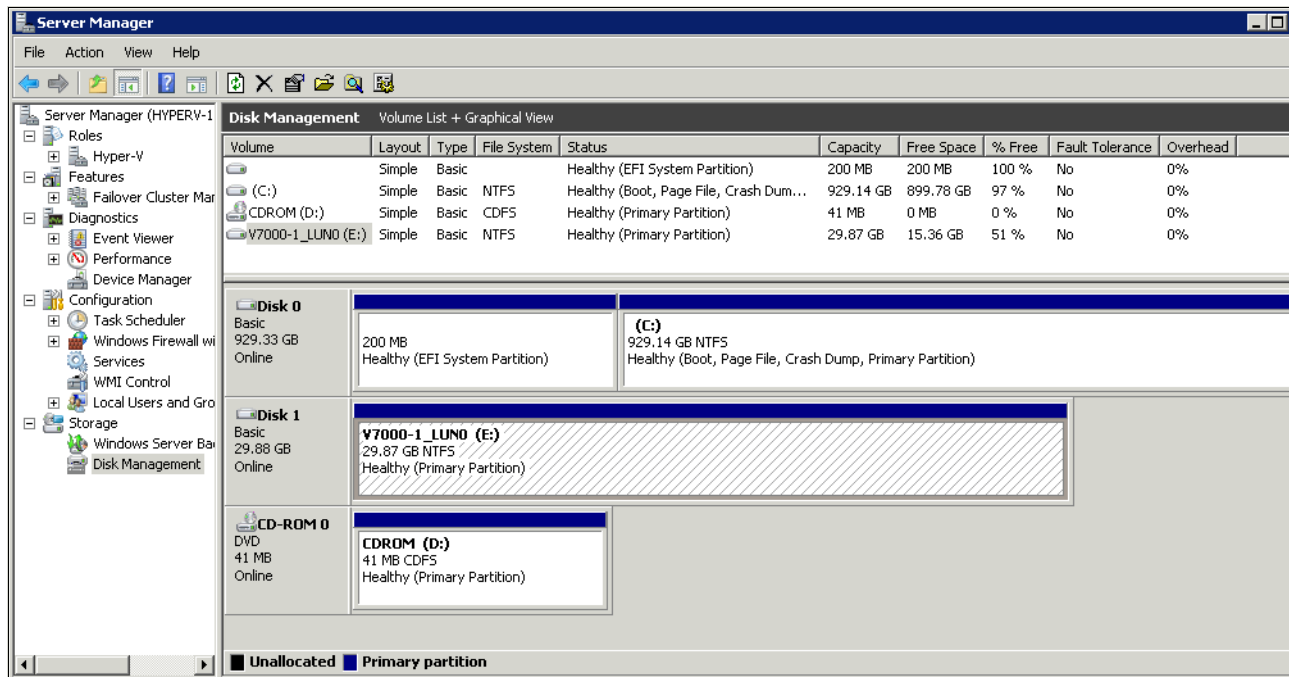


Figure 4-3 Disk Management view of the V7000 LUN

By opening Windows Explorer to view the contents of this disk, as shown in Figure 4-4, we see pre-existing data. In a typical environment, a LUN will be full of application-specific data. We have placed a text file on this LUN to show that the LUN data is readable now; however, you will not be able to read the LUN data after the successful encryption of the volume.

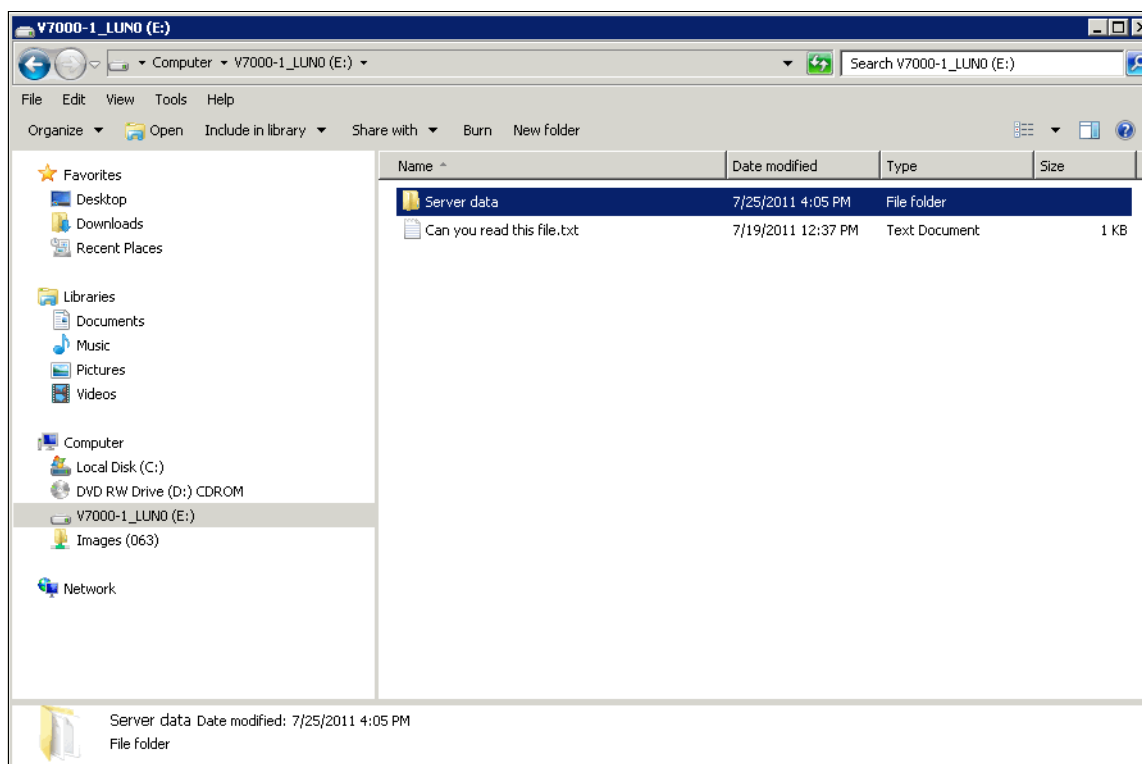


Figure 4-4 Viewing existing data on the V7000 LUN

In the following steps, we require both the worldwide node names (WWNNs) and WWPNs of the host initiator HBA, as well as for the Storwize V7000 target ports. In Example 4-4, we show the SAN switch **nscamshow** output, which provides a convenient way to collect this information for each port.

Example 4-4 Identifying WWPNs and WWNNs

```

SAN32B-E4-1:admin> nscamshow
nscam show for remote switches:
Switch entry for 3
state rev owner cap_available
known v700 0xffffc01 1
Device list: count 8
  Type Pid COS PortName NodeName
  N 030400; 3;50:05:07:68:02:30:a7:fe;50:05:07:68:02:00:a7:fe;
    FC4s: FCP
    Fabric Port Name: 20:04:00:05:1e:c3:be:29
    Permanent Port Name: 50:05:07:68:02:30:a7:fe
    Port Index: 4
    Share Area: No
    Device Shared in Other AD: No
    Redirect: No
    Partial: No
  N 030500; 3;50:05:07:68:02:40:a7:fe;50:05:07:68:02:00:a7:fe;
    FC4s: FCP
    Fabric Port Name: 20:05:00:05:1e:c3:be:29
    Permanent Port Name: 50:05:07:68:02:40:a7:fe
    Port Index: 5
    Share Area: No
    Device Shared in Other AD: No
    Redirect: No
    Partial: No
  N 030600; 3;50:05:07:68:02:30:a7:be;50:05:07:68:02:00:a7:be;
    FC4s: FCP
    PortSymb: [28] "IBM 2145 0000"
    Fabric Port Name: 20:06:00:05:1e:c3:be:29
    Permanent Port Name: 50:05:07:68:02:30:a7:be
    Port Index: 6
    Share Area: No
    Device Shared in Other AD: No
    Redirect: No
    Partial: No
  N 030700; 3;50:05:07:68:02:40:a7:be;50:05:07:68:02:00:a7:be;
    FC4s: FCP
    PortSymb: [28] "IBM 2145 0000"
    Fabric Port Name: 20:07:00:05:1e:c3:be:29
    Permanent Port Name: 50:05:07:68:02:40:a7:be
    Port Index: 7
    Share Area: No
    Device Shared in Other AD: No
    Redirect: No
    Partial: No
  N 030900; 3;10:00:00:05:1e:c7:6b:8a;20:00:00:05:1e:c7:6b:8a;
    FC4s: FCP
    PortSymb: [89] "Brocade-825 | 2.3.0.2 | HYPERV-1-046 | Windows Server 2008
R2 Datacenter | Service Pack 1"

```

```
Fabric Port Name: 20:09:00:05:1e:c3:be:29
Permanent Port Name: 10:00:00:05:1e:c7:6b:8a
Port Index: 9
Share Area: No
Device Shared in Other AD: No
Redirect: No
Partial: No
```

Notice the Pid heading in the device list output (Example 4-4 on page 94). This value is the port identifier. In our small switch environment, the middle byte of the 3-byte Pid correlates to the port number, which is marked in **bold** for each Pid.

We then can see the NodeName (WWNN) and the PortName (WWPN) that are associated to each port. A Storwize V7000 has two separate WWNNs (one WWNN for each node canister). In each WWPN, only the third last byte (in **bold**) changes compared to the WWNN. For the host HBA that attaches to port 9, we see that the first byte of the WWPN is changed compared to the WWNN.

4.2 General prerequisites for encryption

Requirements: It is a requirement of an encryption environment that the preferred practice of a single initiator to a single target and zoning by WWPN must be in place. All switches must have their default zones set to No Access.

4.2.1 Setting the default zone

To view the current default zone setting, we issue the command **defzone --show** on each switch to verify if this requirement is met. In Example 4-5, the output shows that the default zone for this switch is set for All Access; therefore, we must change this to No Access before continuing. To change the default zone, we ensure that we are working with the default Admin Domain, define No Access, and then save this new zoning configuration.

Performing this default zone change in a well-zoned environment is the same as activating a new zoneset, and this change typically does not cause any disruption. Although if you use a fabric without zoning, this change will stop all traffic through the switch. Implementing WWPN zoning must be done before changing the default zone and proceeding with implementing encryption.

Example 4-5 Resolving the default zone setting

```
SAN32B-E4-1:admin> defzone --show
Default Zone Access Mode
    committed - All Access
    transaction - No Transaction
SAN32B-E4-1:admin> ad --select AD0
SAN32B-E4-1:admin> defzone --noaccess
You are about to set the Default Zone access mode to No Access
Do you want to set the Default Zone access mode to No Access ? (yes, y, no, n):
[no] y
SAN32B-E4-1:admin> cfgsave
You are about to save the Defined zoning configuration. This
action will only save the changes on Defined configuration.
```

```
Any changes made on the Effective configuration will not
take effect until it is re-enabled. Until the Effective
configuration is re-enabled, merging new switches into the
fabric is not recommended and may cause unpredictable
results with the potential of mismatched Effective Zoning
configurations.
Do you want to save Defined zoning configuration only? (yes, y, no, n): [no] y
Updating flash ...
SAN32B-E4-1:admin>
```

4.2.2 Synchronizing switch times

After encryption keys are in use, it is critical that the synchronization of keys between encryption engines (EEs) and the Tivoli Key Lifecycle Manager (TKLM) key vaults is maintained, especially when performing functions, such as a rekey operation. To ensure that the time is kept in sync between the TKLM servers and the EEs, we must define a Network Time Protocol (NTP) server to each one of these devices. Although we do not show how to set an NTP server on the TKLM host operating system, it must be set to the same NTP server as the switches.

In Example 4-6, we show the commands that are issued on the SAN32B-E4 to set the NTP server and our time zone; we also verify each setting as we set it. We repeat these same commands on every switch in both our fabrics.

Example 4-6 Setting NTP servers on the switches

```
SAN32B-E4-1:admin> tsclockserver 199.4.29.166
Updating Clock Server configuration...done.
Updated with the NTP servers
SAN32B-E4-1:admin> tsclockserver
Active NTP Server          199.4.29.166
Configured NTP Server List 199.4.29.166
SAN32B-E4-1:admin> tstimezone America/Los_Angeles
SAN32B-E4-1:admin> tstimezone
America/Los_Angeles
SAN32B-E4-1:admin> date
Mon Aug  1 10:55:28 PDT 2011
SAN32B-E4-1:admin>
```

Tip: It might be simpler to use the **tstimezone --interactive** command to set your specific time zone.

By issuing the **date** command on the switch, we can display the current time according to the switch. We then compare this time to the other switches' time and the TKLM time to ensure that everything is synchronized.

4.2.3 Host requirements

The process of enabling encryption to a target LUN creates Virtual Initiator (VI) and Virtual Target (VT) WWPNs and WWNNs. Associated to these VIs and VTs are virtual port identifiers (PIDs). These VT PIDs will be presented to the host enabling the data paths. Because these virtual PIDs differ from the previous PIDs in the native fabric, the host OS must be able to handle this change.

For AIX, Dynamic Tracking (DYNTRK) needs to be enabled to avoid losing access to target LUNs during an encryption implementation. For further details, refer to this website:

https://www-304.ibm.com/support/docview.wss?uid=isg1520readmefb4520desr_lpp_bos

4.3 Defining the encryption configuration

Now that we are familiar with our existing environment and know how to find the required WWNN/WWPN information for our devices, we are ready to configure the encryption of our LUN.

In Figure 4-5, we have added the SAN32B-E4 and associated TKLM key vaults to our SAN environment, which we have already set up initially in Chapter 3, “Initial setup for the IBM Tivoli Key Lifecycle Manager and the SAN32B-E4 Encryption Switch” on page 35. We can proceed with the specific configuration of our target LUN for our host initiator.

With the SAN32B-E4 and TKLMs added to the SAN fabric, and no further configuration performed, the host to target LUN is still accessible as it always has been. The following steps to encrypt the target LUN will cause all traffic from the host to the LUN to be redirected via the SAN32B-E4. Therefore, you need to plan for enough ISL bandwidth to cater for the expected workload.

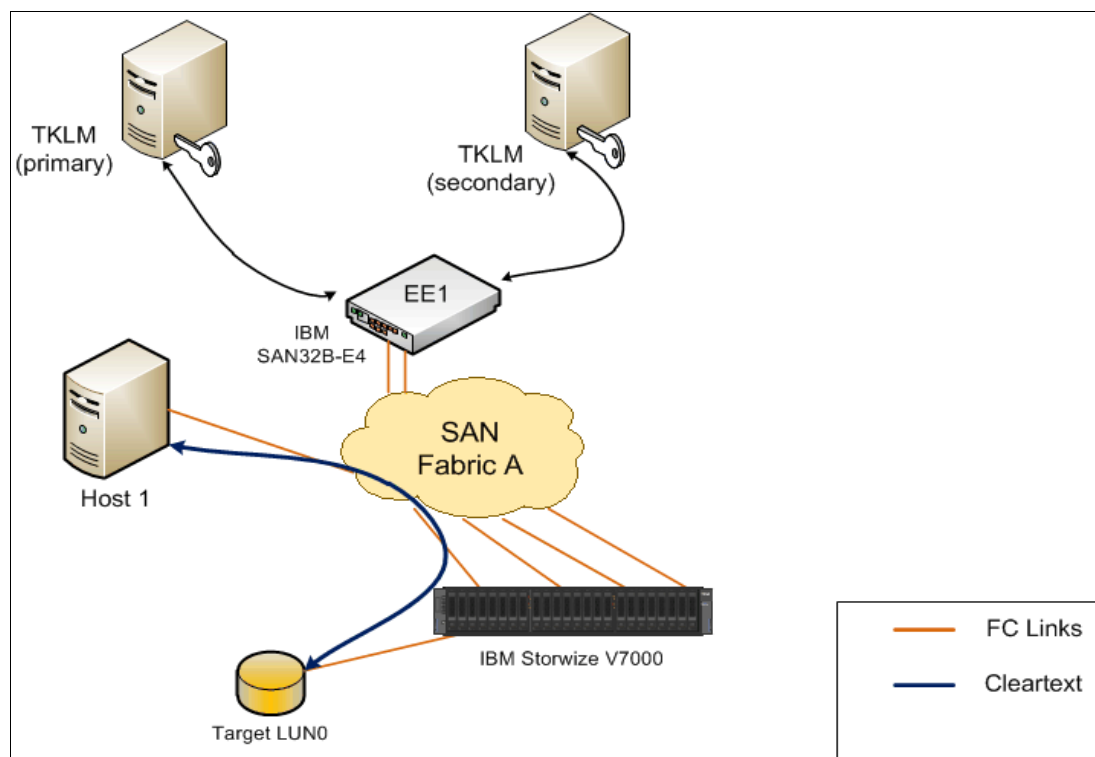


Figure 4-5 Adding EE and TKLM servers

First, we must create target port *containers*. Each port on the target Storwize V7000 must have a separately defined container. Because we have two ports from each node canister on the Storwize V7000, we must create four containers.

To create the container, one of the required parameters is the encryption node's WWN. The WWN is shown in the **switchshow** output previously displayed in Example 4-1 on page 91.

4.3.1 Creating containers

To create a container, we use the **cryptocfg** command with the **create** parameter. We specify that the container type is disk and we add a container name (of our choice). Then, we specify the EE Node WWN, the target WWPN, and the WWNN. We repeat this command for each target port on our Storwize V7000, as shown in Example 4-7.

Example 4-7 Creating containers for target ports

```
SAN32B-E4-1:admin> cryptocfg --create -container disk CTC_1
10:00:00:05:1e:54:17:10 50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
      Slot      Local/
      EE Node WWN      Number      Remote
10:00:00:05:1e:54:17:10      0      Local
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --create -container disk CTC_2
10:00:00:05:1e:54:17:10 50:05:07:68:02:40:a7:fe 50:05:07:68:02:00:a7:fe
      Slot      Local/
      EE Node WWN      Number      Remote
10:00:00:05:1e:54:17:10      0      Local
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --create -container disk CTC_3
10:00:00:05:1e:54:17:10 50:05:07:68:02:30:a7:be 50:05:07:68:02:00:a7:be
      Slot      Local/
      EE Node WWN      Number      Remote
10:00:00:05:1e:54:17:10      0      Local
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --create -container disk CTC_4
10:00:00:05:1e:54:17:10 50:05:07:68:02:40:a7:be 50:05:07:68:02:00:a7:be
      Slot      Local/
      EE Node WWN      Number      Remote
10:00:00:05:1e:54:17:10      0      Local
Operation succeeded.
SAN32B-E4-1:admin>
```

Node canisters: The Storwize V7000 has two node canisters operating as a cluster that function as the controllers for any disk that is managed by it. Each of these node canisters has a separate WWNN, and both node canisters must have defined containers.

4.3.2 Adding a host initiator to a container

To avoid disruption for the host to the target LUN access, we define and commit each host initiator port to Storwize V7000 port, one at a time. When we reach the point of discovering the target LUN number as known by the EE, we must commit the initiator to the target port Crypto Target Container (CTC) configuration for the discovery to work. But after this configuration is committed, the defined path is redirected via the EE, but without the target LUN defined, thus breaking the access. By completing the definitions for one path (CTC) at a time, we only disrupt one path to the target LUN until we complete the definition.

In Example 4-8 on page 99, we define our host HBA WWPN and WWNN, which were listed earlier in the **nsccamshow** output (Example 4-4 on page 94), to the CTC_1 container that was defined in the previous step.

Example 4-8 Adding the host WWPN and WWNN to a container

```
SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_1 10:00:00:05:1e:c7:6b:8a
20:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin>
```

To allow us to discover the target LUN ID that is required in the next step, we commit the configuration that we have defined so far. Because the CTC_1 definitions currently contain the host HBA details and the target port details, this container is able to discover LUNs presented from that target port to our initiator after it is committed. By doing this step, the path from the host to the target Storwize V7000 LUN, as defined by our CTC_1, will stop data access through this path, because it currently does not define the target LUN.

Default zone access: The error that we encountered during the commit, which is shown in Example 4-9, occurs if the default zone is set to All Access. This default zone access needs to be correct before you begin the CTC creation, or all CTC definitions that have been created so far will be lost.

Example 4-9 Commit failure

```
SAN32B-E4-1:admin> cryptocfg --commit
Operation failed: Default zone set to all access at one of nodes in EG
```

Refer to 4.2.1, “Setting the default zone” on page 95 to correctly set the default zone on all switches to resolve the error that is shown in Example 4-9. If your default zone setting is correct, the commit command will succeed, as shown in Example 4-10.

Example 4-10 CTC_1 successfully committed

```
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>
```

By viewing the **datapath query device** output on our host, as shown in Example 4-11, the closed state proves that we have lost a path due to the incomplete container definitions that are now active.

Example 4-11 Verifying LUN access on the host

```
C:\Program Files\IBM\SDDDSM>datapath query device
```

Total Devices : 1

```
DEV#: 0  DEVICE NAME: Disk1 Part0  TYPE: 2145          POLICY: OPTIMIZED
SERIAL: 600507680185853FF0000000000000000
=====
Path#          Adapter/Hard Disk          State  Mode      Select    Errors
  0      Scsi Port12 Bus0/Disk1 Part0    OPEN   NORMAL    2468      0
  1      Scsi Port12 Bus0/Disk1 Part0    OPEN   NORMAL      0      0
  2      Scsi Port12 Bus0/Disk1 Part0    OPEN   NORMAL   1163      0
  3      Scsi Port12 Bus0/Disk1 Part0    CLOSE   NORMAL    300      0
```

```
C:\Program Files\IBM\SDDDSM>
```

After the container configuration is committed to the fabric, by checking the **cfgshow** output on the switch, we see that a redirect zone has been defined, as shown in Example 4-12.

Redirect zones: Redirect zones show only in the defined zone configuration, not in the effective configuration. But if a redirect zone is defined, it *is* in use.

Example 4-12 Redirect zone shown in cfgshow output

```
zone: red_1109_support_TKLM500507680230a7be200800051e54170c
      10:00:00:05:1e:c7:6b:8a; 50:05:07:68:02:30:a7:be;
      20:08:00:05:1e:54:17:0c; 20:04:00:05:1e:54:17:0c
```

4.3.3 Adding a target LUN to a container

To discover our target LUN, we issue the **discoverLUN** command for the specific container. In this case, we discover the LUNs within CTC_1. In Example 4-13, we see the discovery output, showing the LUN information for specific hosts.

Example 4-13 Discovering a LUN within a container

```
SAN32B-E4-1:admin> cryptocfg --discoverLUN CTC_1
Container name:      CTC_1
Number of LUN(s):    1
Host:                10:00:00:05:1e:c7:6b:8a
LUN number:       0x0
LUN serial number:   600507680185853FF0000000000000000
LUN connectivity state: Connected
Key ID state:        Key ID not available

Operation succeeded.
SAN32B-E4-1:admin>
```

We must use the LUN Serial number to compare this discovered LUN to the LUN, which is displayed as the Volume Unique Identifier (UID) in Figure 4-6 on page 101, that is presented to our host. We have displayed the UID on the Storwize V7000 Host Mappings window, which is shown in Figure 4-6 on page 101. The Small Computer System Interface ID (SCSI ID) that is displayed on the Storwize V7000 window (Figure 4-6 on page 101) is the decimal value of the LUN number in the **discoverLUN** output (Example 4-13).

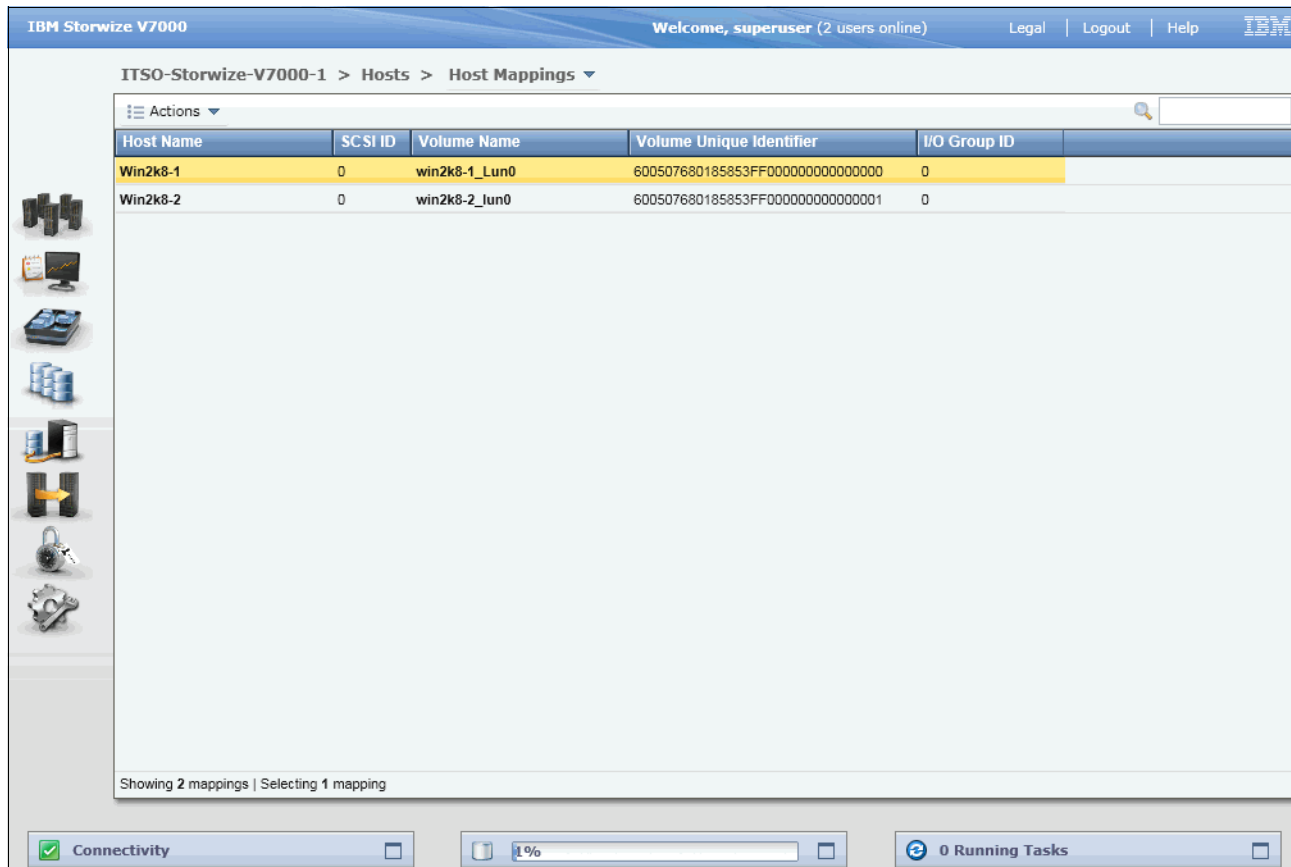


Figure 4-6 Displaying LUN details to each host

After we have confirmed that this LUN serial number, as discovered on the switch, matches the volume UID presented to our host on the Storwize V7000, we need to document the LUN number displayed in the switch output. This LUN number is required for the next step. For our LUN, this number is 0x0.

We add the LUN to the container for our host. As shown in Example 4-14, using the **add LUN** parameters, we specify the container, then the LUN number that we obtained in the previous step, and then our host initiator HBA WWPN and WWNN.

Example 4-14 Adding the target LUN to our CTC_1 container

```
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_1 0x0 10:00:00:05:1e:c7:6b:8a
20:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>
```

After committing this configuration, the path is available again for host to target LUN data access. Prove that the path is available for this access by looking at the **datapath query device** output on the host, as shown in Example 4-15 on page 102. After verifying that the path is again available, we can safely move to the second path/container definitions.

Example 4-15 Datapath query output

```
C:\Program Files\IBM\SDDDSM>datapath query device
```

```
Total Devices : 1
```

```
DEV#: 0  DEVICE NAME: Disk1 Part0  TYPE: 2145          POLICY: OPTIMIZED
SERIAL: 600507680185853FF0000000000000000
```

```
=====
Path#           Adapter/Hard Disk           State  Mode      Select  Errors
  0      Scsi Port12 Bus0/Disk1 Part0      OPEN   NORMAL    2468      0
  1      Scsi Port12 Bus0/Disk1 Part0      OPEN   NORMAL      0      0
  2      Scsi Port12 Bus0/Disk1 Part0      OPEN   NORMAL    1163      0
  3      Scsi Port12 Bus0/Disk1 Part0      OPEN   NORMAL     374      0
```

```
C:\Program Files\IBM\SDDDSM>
```

We repeat the steps that were performed for CTC_1. That is, we add the initiator, commit the configuration, discover the target LUN, define the target LUN, and commit the completed container configuration to return the path to operation for our CTC_2. Example 4-16 shows completing the CTC_2 definitions.

Example 4-16 Completing the CTC_2 definitions

```
SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_2 10:00:00:05:1e:c7:6b:8a
20:00:00:05:1e:c7:6b:8a
```

```
Operation succeeded.
```

```
SAN32B-E4-1:admin> cryptocfg --commit
```

```
Operation succeeded.
```

```
SAN32B-E4-1:admin> cryptocfg --discoverLUN CTC_2
```

```
Container name:      CTC_2
```

```
Number of LUN(s):      1
```

```
Host:                10:00:00:05:1e:c7:6b:8a
```

```
LUN number:          0x0
```

```
LUN serial number:    600507680185853FF0000000000000000
```

```
LUN connectivity state: Connected
```

```
Key ID state:        Key ID not available
```

```
Operation succeeded.
```

```
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_2 0x0 10:00:00:05:1e:c7:6b:8a
20:00:00:05:1e:c7:6b:8a
```

```
Operation succeeded.
```

```
SAN32B-E4-1:admin> cryptocfg --commit
```

```
Operation succeeded.
```

```
SAN32B-E4-1:admin>
```

We verify on the host, by using the **datapath query device** command, that all paths are again open and in use. Then, after the verification, we continue completing our CTC_3 definitions, as shown in Example 4-17 on page 103.

Example 4-17 Completing the CTC_3 definitions

```
SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_3 10:00:00:05:1e:c7:6b:8a
20:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>

SAN32B-E4-1:admin> cryptocfg --discoverLUN CTC_3
Container name:      CTC_3
Number of LUN(s):    1
Host:                10:00:00:05:1e:c7:6b:8a
LUN number:          0x0
LUN serial number:    600507680185853FF0000000000000000
LUN connectivity state: Connected
Key ID state:         Key ID not available

Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_3 0x0 10:00:00:05:1e:c7:6b:8a
20:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>
```

After committing the CTC_3 definitions, we verify the path operation by using the **datapath query device** command on the host. Then, we proceed with our last container, CTC_4, as shown in Example 4-18.

Example 4-18 Completing the CTC_4 configuration

```
SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_4 10:00:00:05:1e:c7:6b:8a
20:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin>
SAN32B-E4-1:admin> cryptocfg --discoverLUN CTC_4
Container name:      CTC_4
Number of LUN(s):    1
Host:                10:00:00:05:1e:c7:6b:8a
LUN number:          0x0
LUN serial number:    600507680185853FF0000000000000000
LUN connectivity state: Connected
Key ID state:         Key ID not available

Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_4 0x0 10:00:00:05:1e:c7:6b:8a
20:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>
```

4.3.4 Displaying a container

With all of our containers defined, we display them to verify the configuration and the container status. We show the CTC_1 configuration in Example 4-19. We see the target Storwize V7000 WWPN and WWNN as defined. A VT has been allocated by the EE. This container has a total of one defined host, and the EE has allocated a VI for the host HBA WWPN/WWNN that we defined. This host has one LUN defined to it.

Example 4-19 Displaying the container configuration

```
SAN32B-E4-1:admin> cryptocfg --show -container CTC_1 -cfg
Container name:      CTC_1
Type:                disk
EE node:             10:00:00:05:1e:54:17:10
EE slot:             0
Target:              50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
VT:                  20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
Number of host(s):   1
Configuration status: committed
Host:                10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
VI:                  20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
Number of LUN(s):    1
Operation succeeded.
SAN32B-E4-1:admin>
```

By looking at the status of a container (Example 4-20), we can display more information about the LUN. We see the VT and VI WWPNs, the target PID, and the associated Virtual PID (VT PID).

More importantly, we see the encryption state of this LUN. In Example 4-20, the encryption mode is `cleartext`, which means that this LUN has not been encrypted yet. Therefore, no assigned encryption key ID exists, which is expected for a `cleartext` LUN.

Example 4-20 Displaying the container status

```
SAN32B-E4-1:admin> cryptocfg --show -container CTC_1 -stat

Container name:      CTC_1
Type:                disk
EE node:             10:00:00:05:1e:54:17:10
EE slot:             0
EE hosting container: current
Target:              50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:          030400
VT:                  20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
VT PID:              012001
Number of host(s):   1
Number of rekey session(s): 0
Host:                10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Host PID:            030900
VI:                  20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:              012401
Number of LUN(s):    1
LUN number:          0x0
LUN type:             disk
LUN serial number:    600507680185853FF0000000000000000
```

```
Encryption mode:      cleartext
Encryption format:    native
Encrypt existing data: disabled
Rekey:                disabled
Internal EE LUN state: Clear text
Encryption algorithm: None
Key ID state:         Key ID not Applicable
New LUN:              No
Operation succeeded.
SAN32B-E4-1:admin>
```

4.3.5 Enabling encryption

We are ready to encrypt the data on the target LUN, but to do so, we must ensure that *all* paths to the LUN are enabled for encryption at the same time. In a dual-fabric configuration, we also must ensure that *all* paths to the LUN are enabled for encryption at the same time. All paths must be defined to enable encryption across all fabrics. The **commit** parameter enables encryption across the fabrics.

Important: If a path/container is not modified to enable encryption to the LUN at the same time as the other paths/container, data corruption occurs.

Using the **modify** parameter, we change each CTC container to enable the encryption of existing data on LUN 0x0 for our specific host initiator WWPN. In our case, we define all four CTCs in the same way. After we are sure that all paths to this target LUN have been modified, we issue the **commit** to commence the encryption, as shown in Example 4-21.

Example 4-21 Enabling encryption on all paths

```
SAN32B-E4-1:admin> cryptocfg --modify -LUN CTC_1 0x0 10:00:00:05:1e:c7:6b:8a
--encrypt --enable_encexistingdata
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --modify -LUN CTC_2 0x0 10:00:00:05:1e:c7:6b:8a
--encrypt --enable_encexistingdata
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --modify -LUN CTC_3 0x0 10:00:00:05:1e:c7:6b:8a
--encrypt --enable_encexistingdata
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --modify -LUN CTC_4 0x0 10:00:00:05:1e:c7:6b:8a
--encrypt --enable_encexistingdata
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>
```

With the CTCs redirecting data flow through the EE and our LUN set for encryption, the SAN32B-E4 will actively modify all data on the LUN by reading in the data, encrypting it, and writing it back to the LUN. This encryption process takes several hours. The length of time depends on the size of the LUN and the workload to it. During this time, any host read access to data on the LUN is handled by the EE. The not yet encrypted data is read. The already encrypted data is read and decrypted. Then, the data is passed to the host. Any write activity from the host is encrypted and written to the LUN. Figure 4-7 on page 106 shows this redirected data flow.

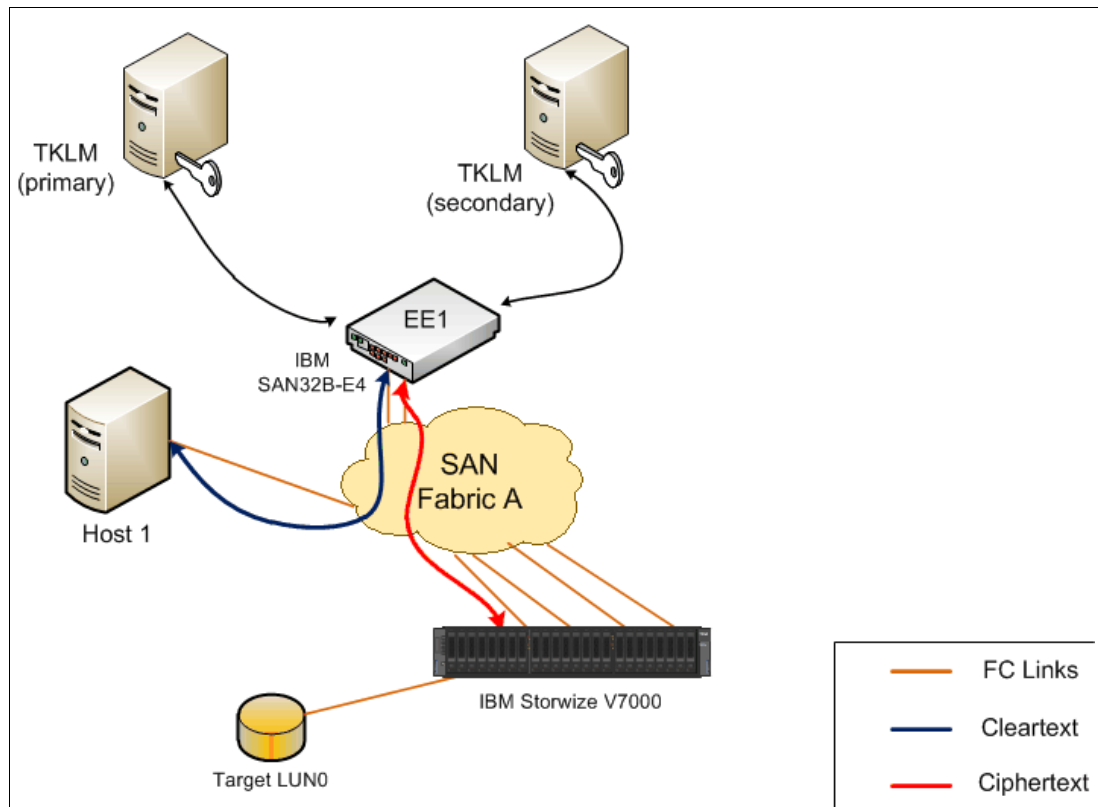


Figure 4-7 Final configuration showing redirected data flow

If we display the LUN status, we see that the encryption mode has changed to encrypt and that the internal EE LUN state is First time re-key. Note that a key ID is associated to the LUN. We can see a percentage of completion for this first time rekey operation. Example 4-22 shows that the rekey operation is 6% complete.

Example 4-22 Displaying the encryption progress

```
SAN32B-E4-1:admin> cryptocfg --show -container CTC_1 -stat
```

```
Container name:      CTC_1
Type:               disk
EE node:            10:00:00:05:1e:54:17:10
EE slot:            0
EE hosting container: current
Target:             50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:         030400
VT:                20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
VT PID:             012001
Number of host(s):  1
Number of rekey session(s): 1
Host:               10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Host PID:           030900
VI:                 20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:             012401
Number of LUN(s):   1
LUN number:         0x0
LUN type:           disk
LUN serial number:  600507680185853FF0000000000000000
```



```

Encryption mode:      encrypt
Encryption format:    native
Encrypt existing data: enabled
Rekey:                disabled
Internal EE LUN state: First time re-key
Encryption algorithm: AES256-XTS
Key ID state:         Re-key
New LUN:              No
Key ID:               47:62:48:66:9d:f8:4a:76:91:28:6c:f1:9a:a4:cb:dd
Key creation time:    Tue Jul 26 17:13:46 2011
Rekey session number: 0
Percentage complete:  6
Rekey state:          Write Phase
Rekey role:           Primary/Active
Block size:           512
Number of blocks:     62914560
Current LBA:          3862369
Operation succeeded.
SAN32B-E4-1:admin>

```

By using the **cryptocfg --show -rekey -all** command, we display all of our CTCs at one time. We verify that we have all four CTCs actively encrypting our target LUN, as shown in Example 4-23.

Example 4-23 Displaying all four containers' rekey states

```

SAN32B-E4-1:admin> cryptocfg --show -rekey -all
Number of rekey session(s):      4

Container name:      CTC_1
EE node:             10:00:00:05:1e:54:17:10
EE slot:             0
Target:              50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:          030400
VT:                  20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
VT PID:              012001
Host:                10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Host PID:            030900
VI:                  20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:              012401
LUN number:          0x0
LUN serial number:   600507680185853FF0000000000000000
Rekey session number: 0
Percentage complete:  9
Rekey state:          Read Phase
Rekey role:           Primary/Active
Block size:           512
Number of blocks:     62914560
Current LBA:          5731793

Container name:      CTC_2
EE node:             10:00:00:05:1e:54:17:10
EE slot:             0
Target:              50:05:07:68:02:40:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:          030500
VT:                  20:02:00:05:1e:54:17:0c 20:03:00:05:1e:54:17:0c

```

```

VT PID:                012801
Host:                  10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Host PID:              030900
VI:                    20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:                012401
LUN number:            0x0
LUN serial number:     600507680185853FF0000000000000000
Rekey session number:  0
Percentage complete:   9
Rekey state:           Read Phase
Rekey role:            Primary/Redundant
Block size:            512
Number of blocks:      62914560
Current LBA:           5731793

Container name:        CTC_3
EE node:               10:00:00:05:1e:54:17:10
EE slot:               0
Target:                50:05:07:68:02:30:a7:be 50:05:07:68:02:00:a7:be
Target PID:            030600
VT:                    20:04:00:05:1e:54:17:0c 20:05:00:05:1e:54:17:0c
VT PID:                012c01
Host:                  10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Host PID:              030900
VI:                    20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:                012401
LUN number:            0x0
LUN serial number:     600507680185853FF0000000000000000
Rekey session number:  0
Percentage complete:   9
Rekey state:           Read Phase
Rekey role:            Primary/Redundant
Block size:            512
Number of blocks:      62914560
Current LBA:           5731793

Container name:        CTC_4
EE node:               10:00:00:05:1e:54:17:10
EE slot:               0
Target:                50:05:07:68:02:40:a7:be 50:05:07:68:02:00:a7:be
Target PID:            030700
VT:                    20:06:00:05:1e:54:17:0c 20:07:00:05:1e:54:17:0c
VT PID:                013001
Host:                  10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Host PID:              030900
VI:                    20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:                012401
LUN number:            0x0
LUN serial number:     600507680185853FF0000000000000000
Rekey session number:  0
Percentage complete:   9
Rekey state:           Read Phase
Rekey role:            Primary/Redundant
Block size:            512
Number of blocks:      62914560

```

```
Current LBA:          5731793
Operation succeeded.
SAN32B-E4-1:admin>
```

By viewing the Storwize V7000 Performance information window that is shown in Figure 4-8, we observe that we are averaging a volume of 65 MBps. Because no host workload is running at this time, we know that this volume is the first-time encryption (FTE) work.

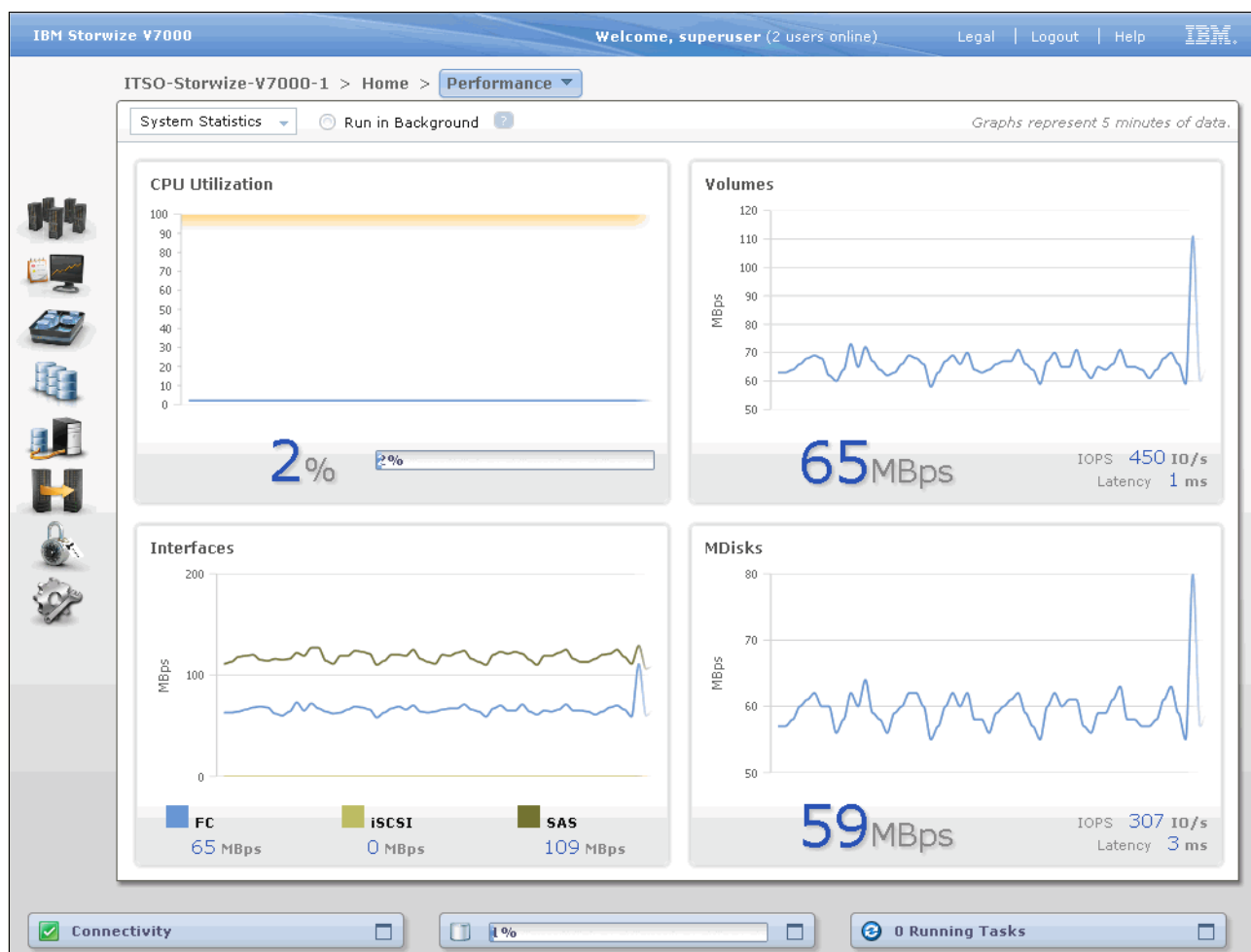


Figure 4-8 Observing FTE workload to the Storwize V7000

After the FTE completes, we confirm that there are no rekey sessions running, as shown in Example 4-24.

Example 4-24 Displaying the encryption process after its completion

```
SAN32B-E4-1:admin> cryptocfg --show -rekey -all
Number of rekey session(s):    0
Operation succeeded.
SAN32B-E4-1:admin>
```

Displaying the container status in Example 4-25 on page 110 shows our LUN is in an encryption-enabled state and no rekey operation is running. Therefore, this LUN is completely encrypted, and our task is complete.

Example 4-25 Completed encryption state of the LUN

```
SAN32B-E4-1:admin> cryptocfg --show -container CTC_1 -stat
```

```
Container name:      CTC_1
Type:               disk
EE node:            10:00:00:05:1e:54:17:10
EE slot:            0
EE hosting container: current
Target:             50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:         030400
VT:                 20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
VT PID:             012001
Number of host(s):  1
Number of rekey session(s): 0
Host:               10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Host PID:           030900
VI:                 20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:             012401
Number of LUN(s):   1
LUN number:         0x0
LUN type:           disk
LUN serial number:  600507680185853FF0000000000000000
Encryption mode:    encrypt
Encryption format:  native
Encrypt existing data: enabled
Rekey:              disabled
Internal EE LUN state: Encryption enabled
Encryption algorithm: AES256-XTS
Key ID state:       Read write
New LUN:            No
Key ID:             47:62:48:66:9d:f8:4a:76:91:28:6c:f1:9a:a4:cb:dd
Key creation time:  Tue Jul 26 17:13:46 2011
Operation succeeded.
SAN32B-E4-1:admin>
```

4.4 Data encryption key cluster config: Multiple path/dual fabric

In this section, we show how to build or extend SAN32B-E4 Encryption Switch solutions from a single encryption node setup to a multiple fabric/multiple path solution. Typically, a dual-fabric solution provides multiple independent paths from the host server to a disk storage subsystem.

4.4.1 Starting the configuration

Our starting point is the target configuration that we achieved in 4.1, “Configuring encryption on a single fabric” on page 90. Figure 4-9 on page 111 shows you the single-fabric EE configuration that we will use as our base. At this point, we already have two defined LUNs in the IBM Storwize V7000 disk storage system, which our host server, Host 1, accesses.

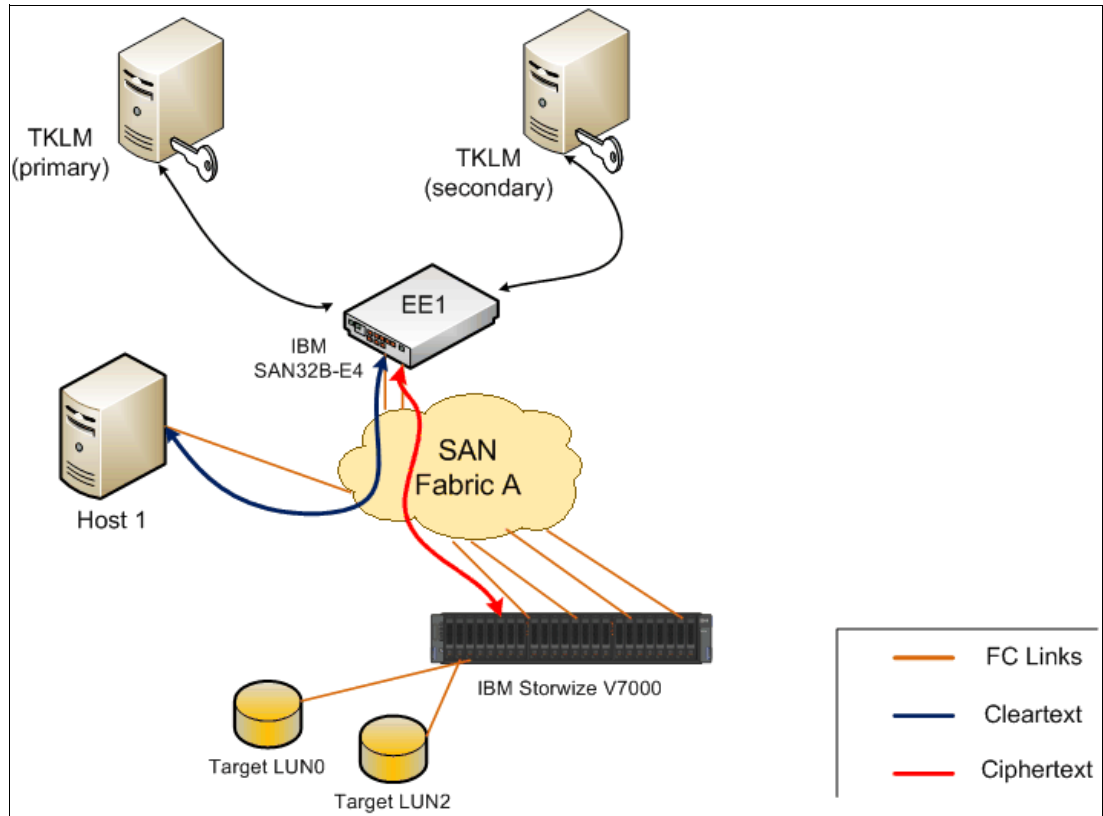


Figure 4-9 Single-fabric EG setup

4.4.2 Target configuration for a data encryption key cluster environment

A data encryption key (DEK) cluster (DEK cluster) is a cluster of EEs that can host all paths to a LUN and share the DEK set. The EEs can be in the same fabric or separate fabrics. DEK clusters also enable host multipath I/O (MPIO) failover.

Important: You configure LUN policies at the LUN level, but they apply to the entire high-availability (HA) or DEK cluster. You *must* configure the same LUN policies for multipath LUNs that are exposed through multiple target ports and thus configured on multiple CTCs on separate EEs in an HA cluster or a DEK cluster. A failure to configure the same policies results in unexpected behavior and might cause data corruption.

Figure 4-10 on page 112 shows the target configuration that we will create in this section.

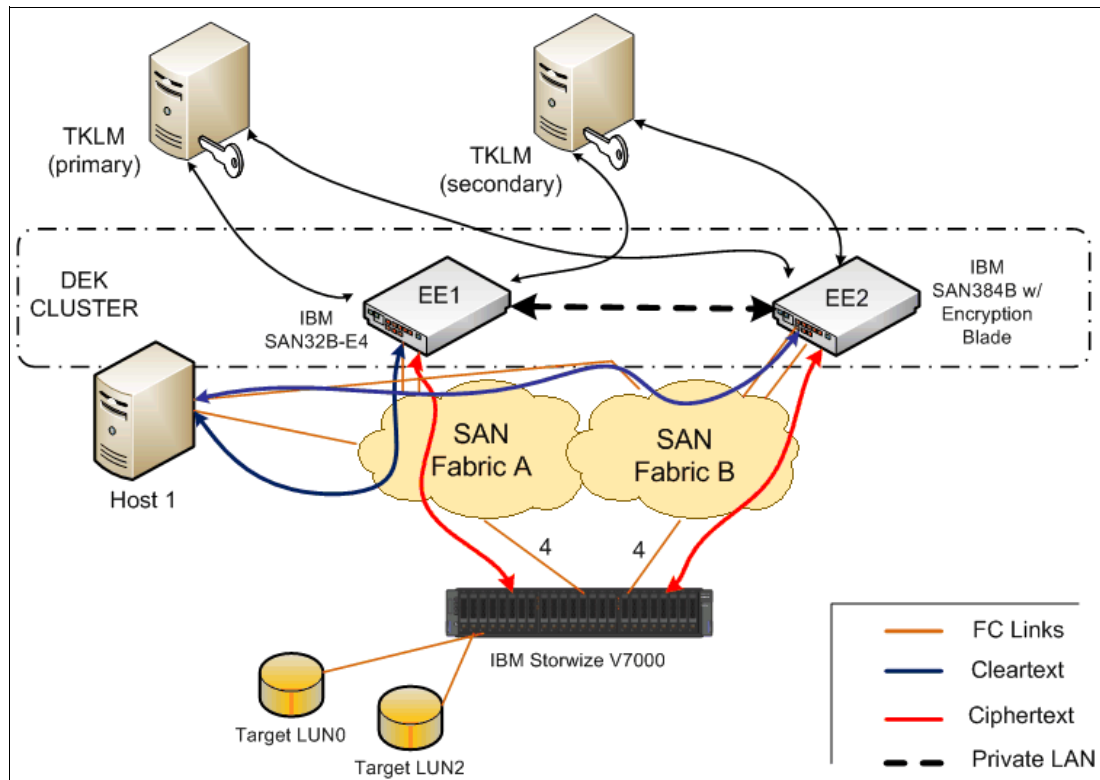


Figure 4-10 Dual-fabric encryption group setup forming a DEK cluster solution

Dual-fabric encryption group setup forming a DEK cluster

In our example, we have a multipath disk LUN set connecting to an initiator (host server), with a single HBA providing two SAN ports. The IBM Storwize V7000 provides two disk controllers. We connected two target ports from each disk controller to each fabric (for a total of eight target ports, or four target ports per fabric). Also, we used the IBM Subsystem Device Driver (SDD) multipath disk driver for the IBM Storwize V7000 disk storage subsystem support.

4.4.3 Steps to extend a single fabric setup to a DEK cluster environment

In 4.1, “Configuring encryption on a single fabric” on page 90, we showed in detail how to establish a single-fabric encryption group (EG) setup. In these steps, we show and explain installing a new DEK cluster SAN32B-E4 Encryption Switch setup from the beginning. We also show and explain how to enhance an existing single-fabric/single-encryption node setup to become a dual-cluster DEK cluster setup.

The major difference between these scenarios is that when you install a new DEK cluster environment, the associated LUNs in the disk storage subsystem have not been used (encrypted) yet.

4.4.4 Preparing the multipath/dual-fabric configuration

Configuring a LUN with multiple paths increases the risk of potentially catastrophic situations. For example, separate policies might exist for each path of the LUN. Or, one path might be exposed through the Encryption Switch, and the other path might have direct access to the device from a host that is outside of the secured realm of the encryption platform.

The failure to follow proper configuration procedures for multipath LUNs results in data corruption. To avoid the risk of data corruption, it is of the utmost importance that you observe the following rules when configuring multipath LUNs:

- ▶ During the initiator-target zoning phase, you must complete, in sequence, *all* zoning for *all* hosts that must gain access to the targets *before* committing the zoning configuration.
- ▶ Complete the CTC configuration for *all* target ports, in sequence, and add the hosts that must gain access to these ports *before* committing the container configuration.

Important: Upon the commit, the hosts lose access to all LUNs until the LUNs are explicitly added to the CTCs. This process is not a concurrent process.

- ▶ When configuring the LUNs, you must configure the same LUN policies for *all* paths of *all* LUNs. A failure to configure all LUN paths with the same LUN policies results in data corruption.

4.4.5 Verifying the current encryption group members

The next activity is to check the current status of all encryption member nodes in the EG. We use the DCFM Encryption Center window for that purpose. You can perform all configuration action on the Encryption Switch by using the Encryption Manager. To open the DCFM Encryption Manager, click **Configure** → **Encryption**. The Encryption Center window opens, showing all Encryption Switches that are discovered by DCFM and their status, as shown in Figure 4-11.

You can see that we have already added the new second member to the existing EG. To add a new encryption member node to an existing encryption node, refer to “Step 10: SAN32B-E4 Encryption Switch: Registering member nodes on the GL node” on page 60.

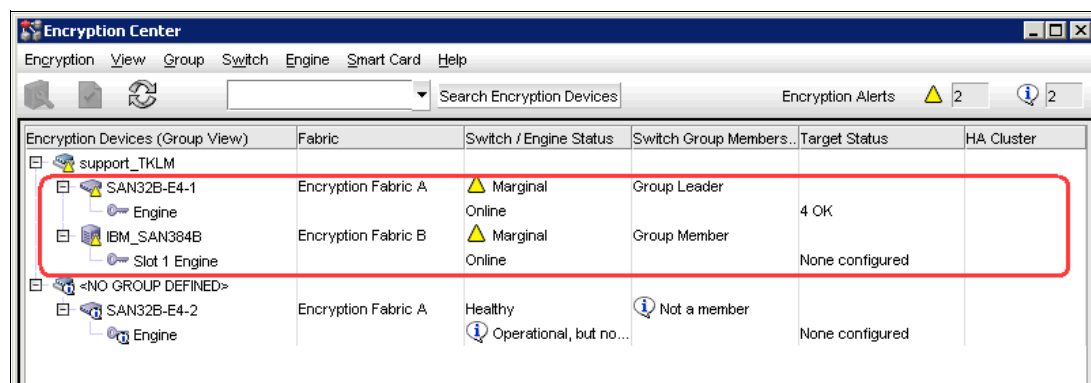


Figure 4-11 DCFM Encryption Center showing the current EG setup

In our lab setup for the second fabric, we used an IBM SAN384 Director with an Encryption Blade installed in slot 1.

Important: Our setup was already running four encryption paths from the host server to two LUNs in the disk subsystem. Therefore, we needed to prevent access from the host to the LUNs by way of the second fabric until we applied *all* configurations and settings successfully to the new encryption member node. We achieved this setup easily by disabling the new additional host server port at the SAN switch temporarily.

4.4.6 Adding a new host server Fibre Channel port to the existing LUNs

The first action starts at the IBM Storwize V7000 GUI to define a new host server Fibre Channel (FC) port for the existing LUNs. In our case, it was the second HBA port of the host server that was connected into the second fabric (we called it Fabric B).

On the IBM Storwize V7000 GUI, navigate to the Ports by Host view. Click **Add** to select **Fibre Channel Port**, as shown in Figure 4-12.

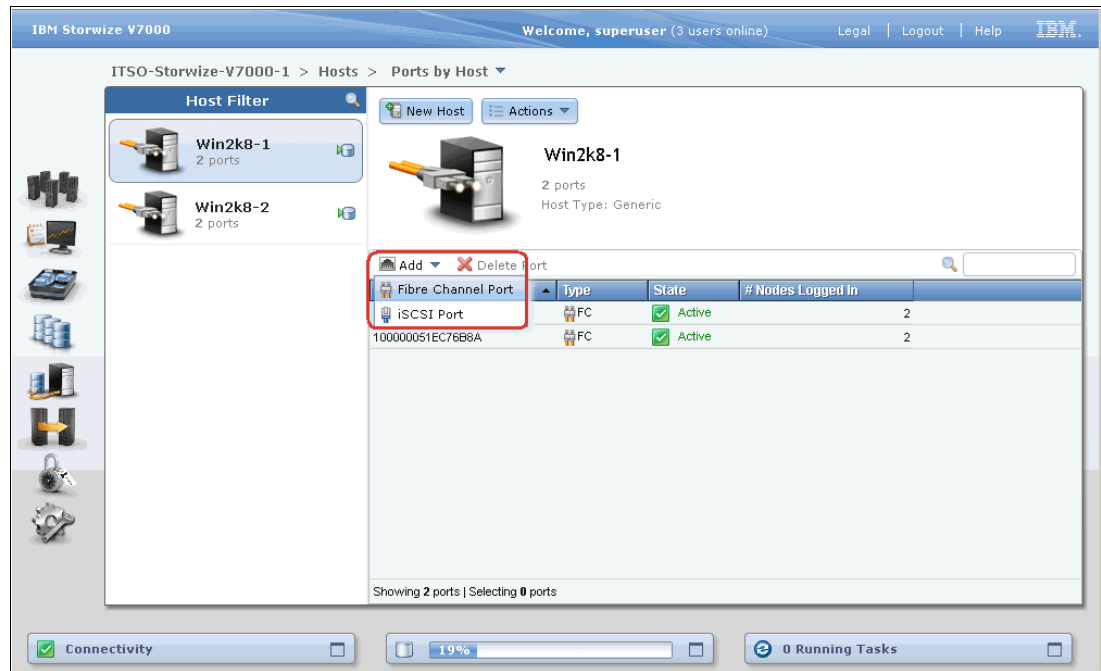


Figure 4-12 Add a new host server Fibre Channel port for a V7000 LUN

You will get a new window to enter the specific host server port information, as shown in Figure 4-13 on page 115.



Figure 4-13 Add Fibre Channel ports window for host server port details

Important: Because we have disabled the new host server port to the SAN (Fabric B), the IBM Storwize V7000 storage subsystem cannot see the new host server port at this point, which is good. It is important that the host cannot see the LUN on the second path directly during our configuration steps, because access might corrupt the existing encrypted data.

We enter the new host server FCI port by entering its WWPN manually. You can obtain the WWPN easily by using the HBA configuration tool at the host server or at the SAN switch. Although the port is disabled, it is already registered at the switch.

After entering the WWPN manually in Figure 4-13 and clicking **Add Ports to Host**, you see the final confirmation window. This window (Figure 4-14 on page 116) shows that a new host port for the selected host server profile (win2k8-1) has been added. Because the new host port is not visible from the V7000 subsystem, it shows Offline for the newly entered WWPN, which is good.

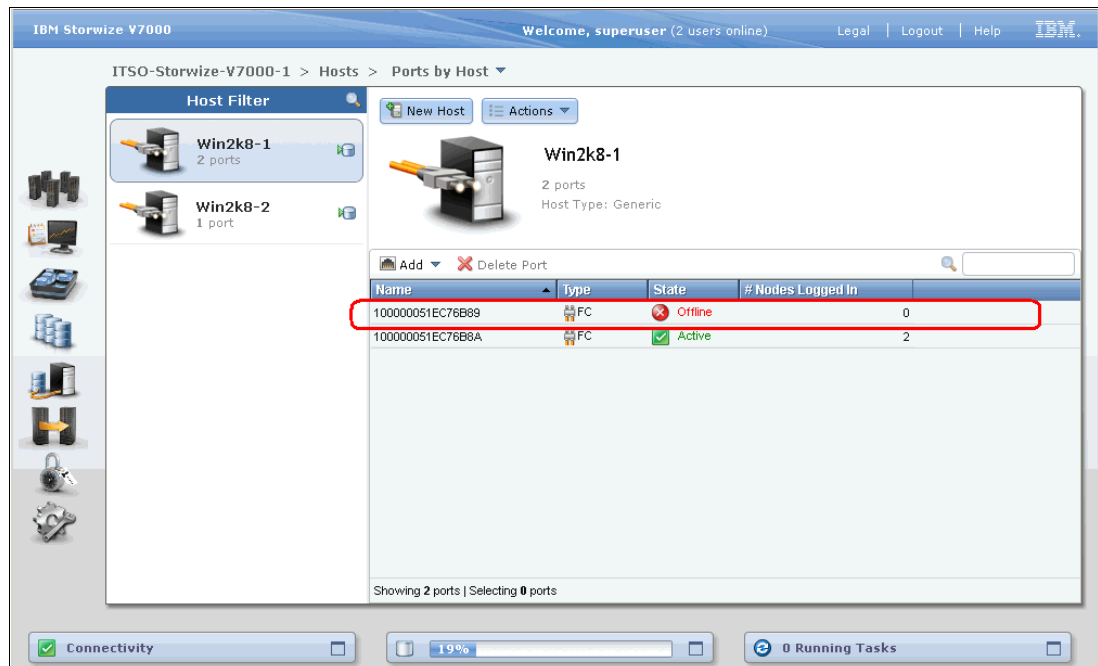


Figure 4-14 Confirmation that new Fibre Channel port has been added

4.4.7 Adding and defining the new CTCs for the second fabric via DCFM

After we have successfully added the new host initiator port (in this case, for the additional paths via the second fabric, Fabric B) within the disk storage subsystem, we continue to create the new CTCs at the second encryption node, which is the newly added member node. The new CTCs are necessary to enable the overall access to the disk storage subsystem target ports that attach to the second fabric.

If using the DCFM GUI, a wizard guides you through the necessary steps:

1. First, you need to open the DCFM Encryption Center windows (see Figure 4-57 on page 144) from the DCFM main window by selecting **Configure** → **Encryption**.
2. In the Encryption Center window, select **Switch** → **Targets**. To ensure that you selected the correct view, see Figure 4-15 on page 117.

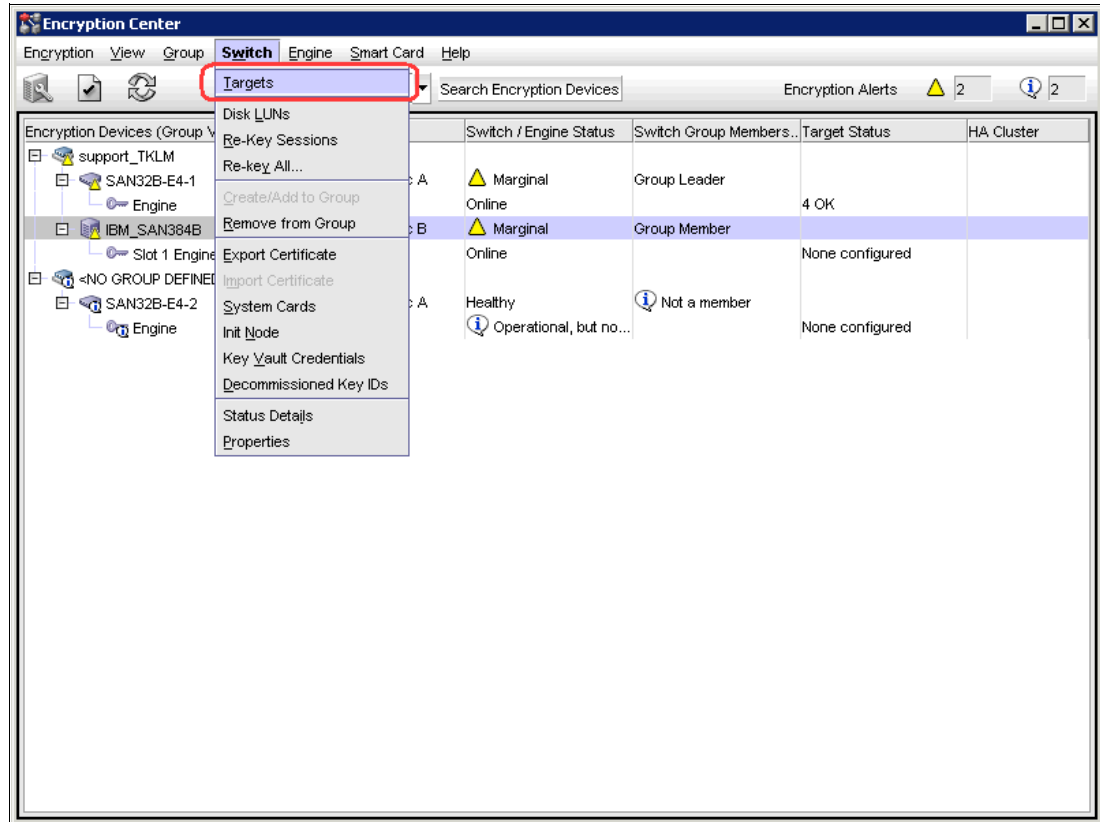


Figure 4-15 DCFM Encryption Center main window

The Encryption Targets window opens, as shown in Figure 4-16.

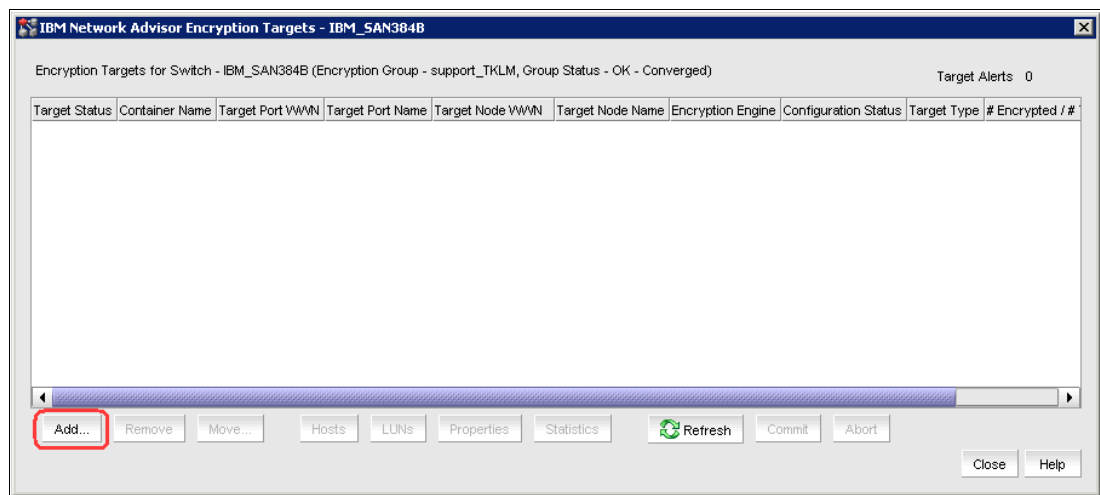


Figure 4-16 DCFM Encryption Center Encryption Targets window

3. Click **Add** to start the Configure Storage Encryption wizard to create the new CTC definitions, as shown in Figure 4-17 on page 118.

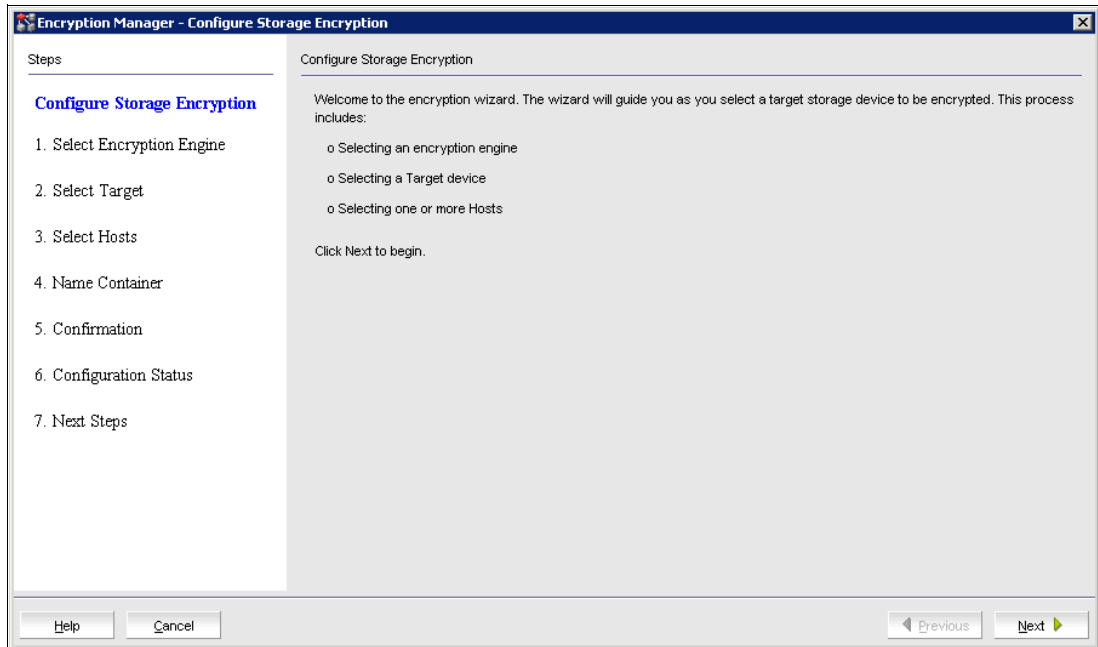


Figure 4-17 DCFM Configure Storage Encryption wizard to create new CTC definitions

4. Click **Next** to navigate through this wizard to create the new CTC definitions.

Important: If you have more than one target port connected to the SAN (in our case, connected to the second fabric, Fabric B), you must repeat this sequence until you have created an associated CTC entry for every new target port.

5. In the first step of the wizard, you need to select the EE on which to create the new CTC definitions. In our case, we select the newly added EE belonging to Fabric B, as shown in Figure 4-18. Select the appropriate EE in the panel, and click **Next**.

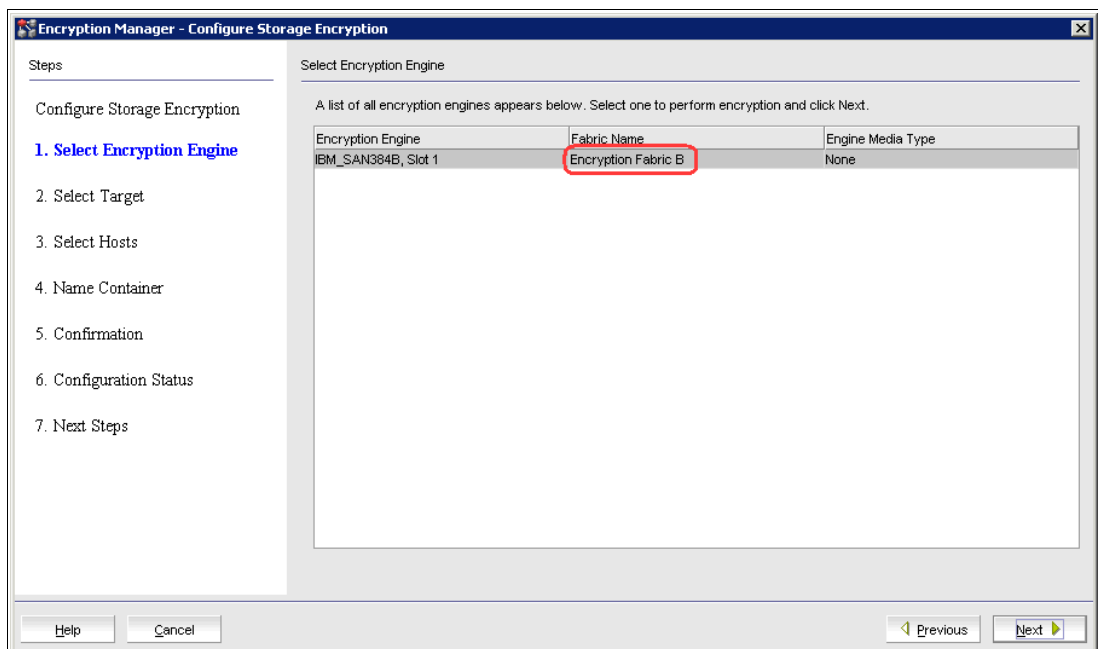


Figure 4-18 DCFM Configure Storage Encryption' wizard to create the new CTC definitions

- Next, you are prompted to select a target port (or enter the target port by entering its port and node WWNs at the bottom of the window) against which to create a new CTC. Ensure that you also select the correct type of encryption target from the lower-right pull-down list. For disk encryption, you must select **Disk**, as shown in Figure 4-19. You will see a list of the target ports that have already been discovered by the EE. Your target port must be on this list for you to select it.

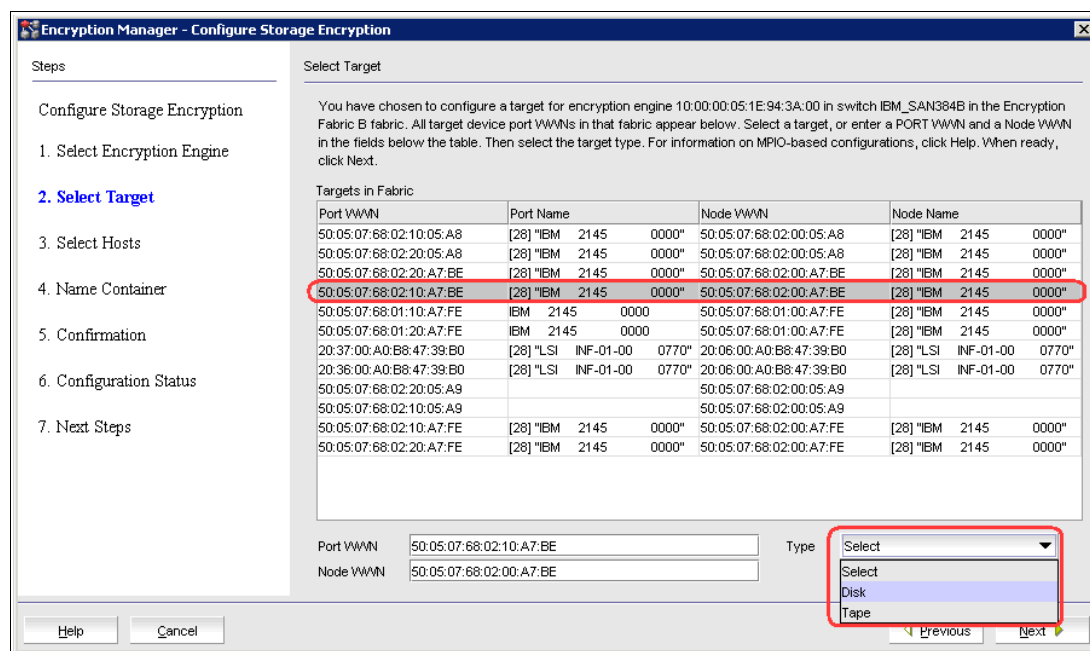


Figure 4-19 DCFM Configure Storage Encryption wizard to create new CTC definitions

- The Select Hosts window, as shown in Figure 4-20 on page 120, prompts you to select the initiator port (host server port) to add to this CTC. The Select Hosts window provides you with a list of initiator ports that have been discovered in the SAN by the EE.
- To block access to the existing disk volume (LUN) from the host server until we have created all the definitions in the EEs, we earlier disabled the host server initiator port in the SAN switch (as explained in 4.4.5, "Verifying the current encryption group members" on page 113). Therefore, we cannot find our new initiator in the list of already discovered and active initiators. So, we need to manually add the initiator in this step, as well. You can discover the necessary information (WWNN and WWPNN) easily by using an HBA tool at the host server, or by using the SAN (switch) administration utilities.

Figure 4-20 on page 120 and Figure 4-21 on page 120 illustrate manually adding the Port WWN and Node WWN of the host initiator within the DCFM wizard to create a new CTC.

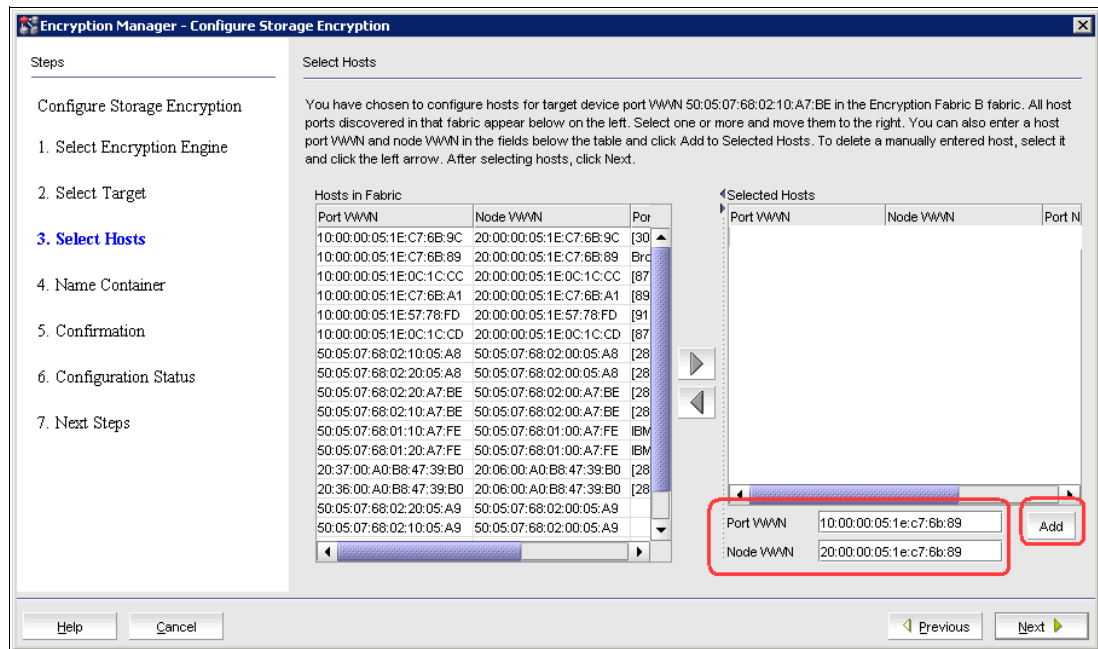


Figure 4-20 DCFM Configure Storage Encryption wizard to create new CTC definitions

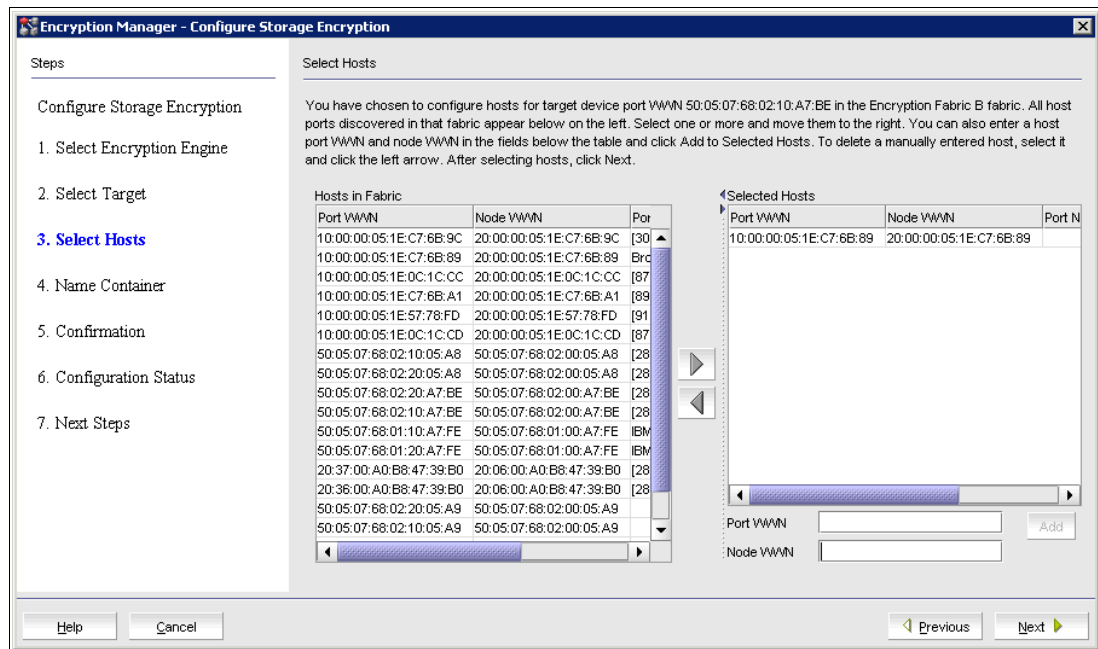


Figure 4-21 DCFM Configure Storage Encryption wizard to create new CTC definitions

- Figure 4-22 on page 121 displays the last step in the wizard where you need to enter information. Provide a name for the newly created CTC. In our setup, we used the same names that we used for the CTCs in Fabric A, except we added a B at the end of the name (for example, we named the first CTC CTC_1B).

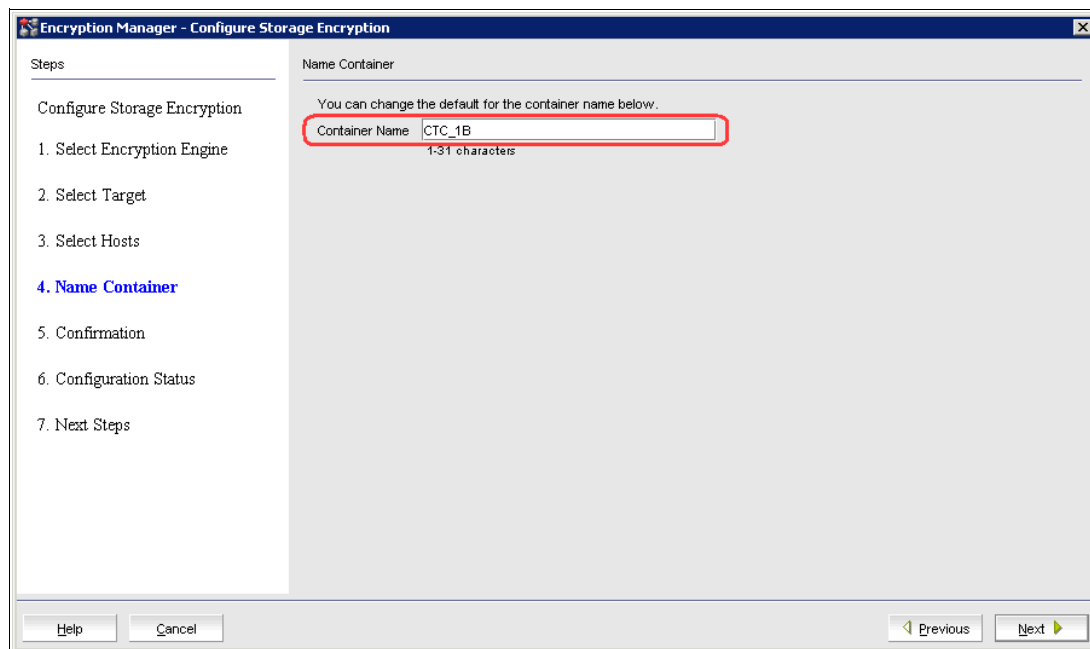


Figure 4-22 DCFM Configure Storage Encryption wizard to create new CTC definitions

10. After specifying the new CTC name, continue to click **Next** until the wizard finishes successfully. You will see a final confirmation of your configuration status, as shown in Figure 4-23.

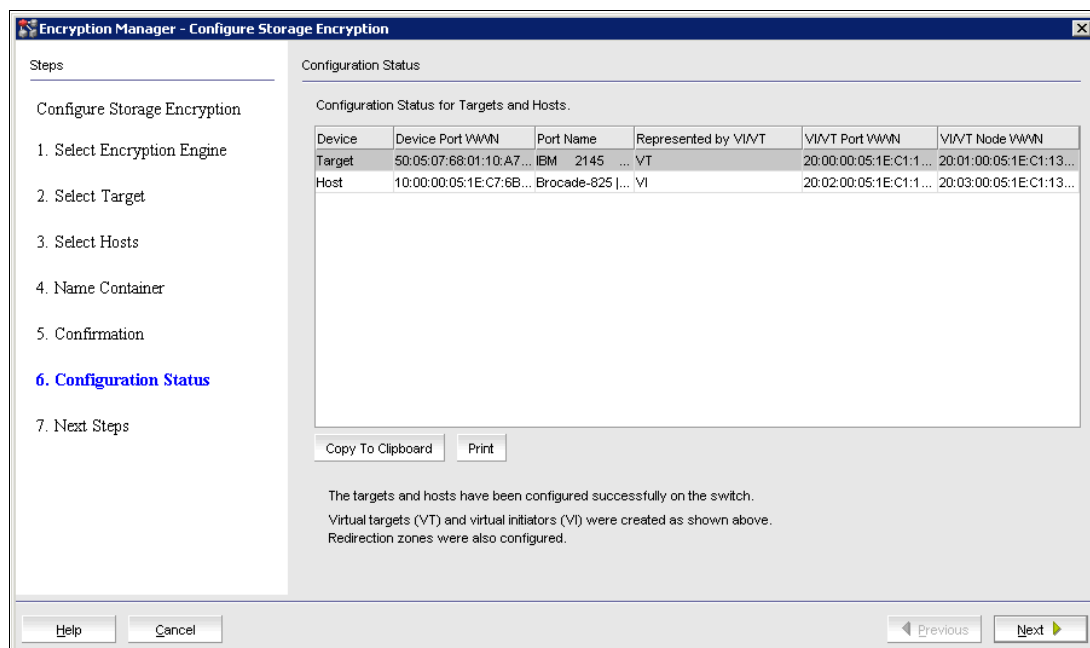


Figure 4-23 DCFM Configure Storage Encryption wizard to create new CTC definitions

11. After we have successfully created all the new CTCs for all the target ports to the disk storage subsystem via the second fabric (in our lab setup, we ran the wizard four times for four CTC definitions), the Encryption Targets window showed the following confirmation (see Figure 4-24 on page 122). At the bottom of the window, click **Commit** to store and activate your new CTC definitions.

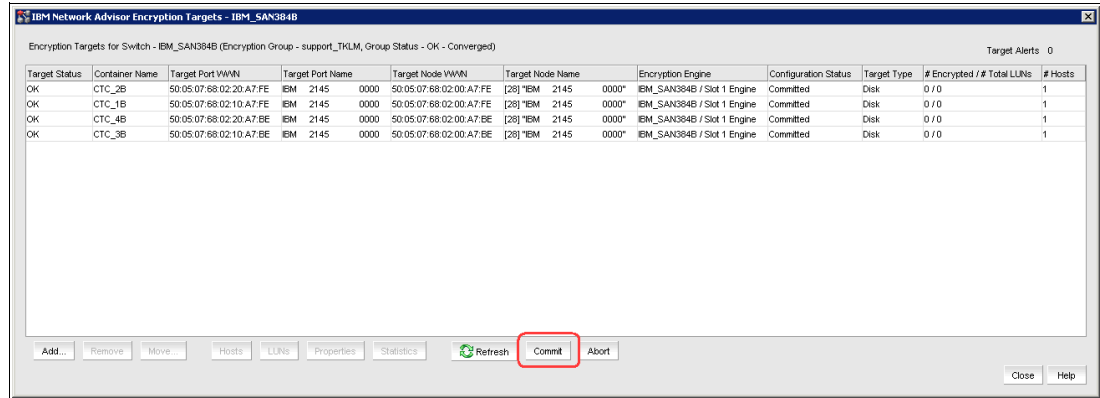


Figure 4-24 DCFM Encryption Targets window after creating the new CTCs

Important: Ensure that you click **Commit** to store and activate the new CTC definitions that you have created.

4.4.8 Adding the existing LUNs for the second fabric CTCs via DCFM

After we have successfully created the new CTC definitions from the initiator (host server) to the target ports (disk storage subsystem) through the additional paths (second fabric), we still need to specify to the EG *which* LUN to encrypt over these new paths. In certain cases, specific LUNs in the storage subsystem might *not* be eligible for encryption, and they must operate further in cleartext mode. Therefore, these LUNs must not be routed through the EE.

We start at the DCFM Encryption Center:

1. With a right-mouse click on the encryption node where we want to add the LUNs to the CTCs, from the pull-down list, we select **Disk LUNs**, as shown in Figure 4-25 on page 123.

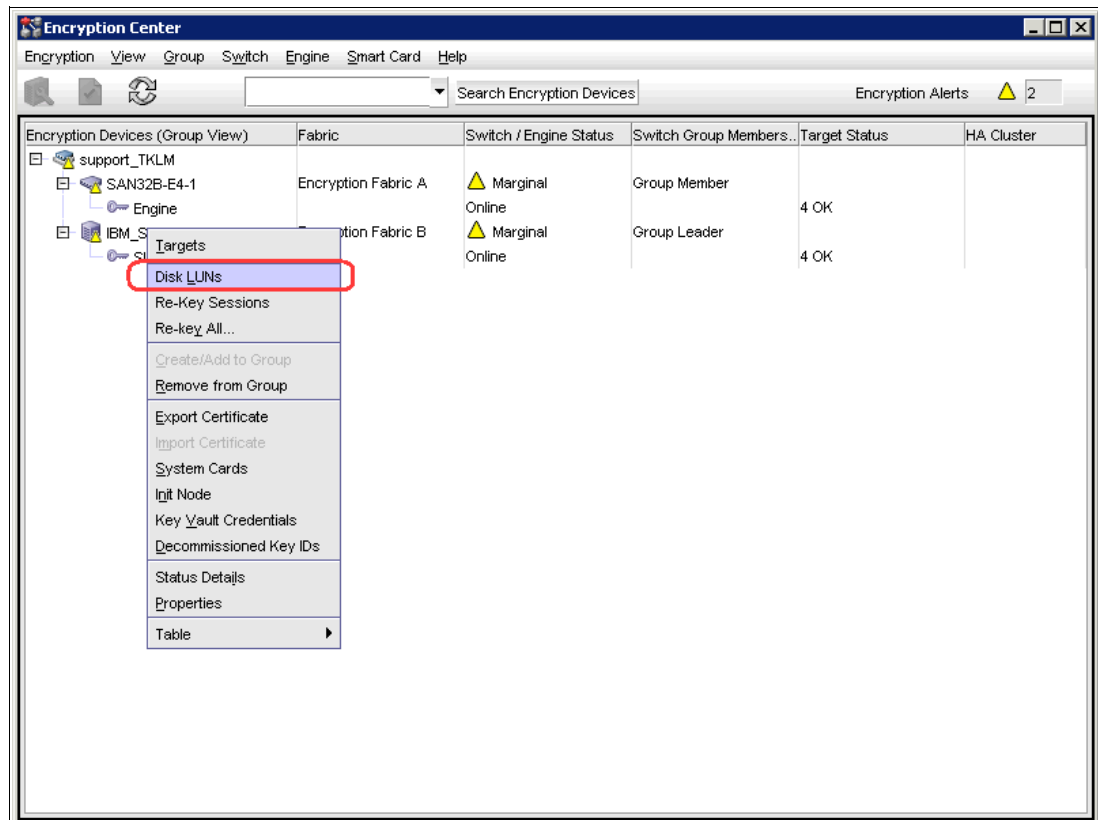


Figure 4-25 DCFM Encryption Center main window: Select Disk LUNs

2. The window that we use to select the disk LUNs (Figure 4-26) is similar to the window that we used to create the CTCs. In the upper-left corner under Storage Array, select the corresponding target port, and click **Add**. In our example, we continued with the default, which corresponded to CTC_3B.

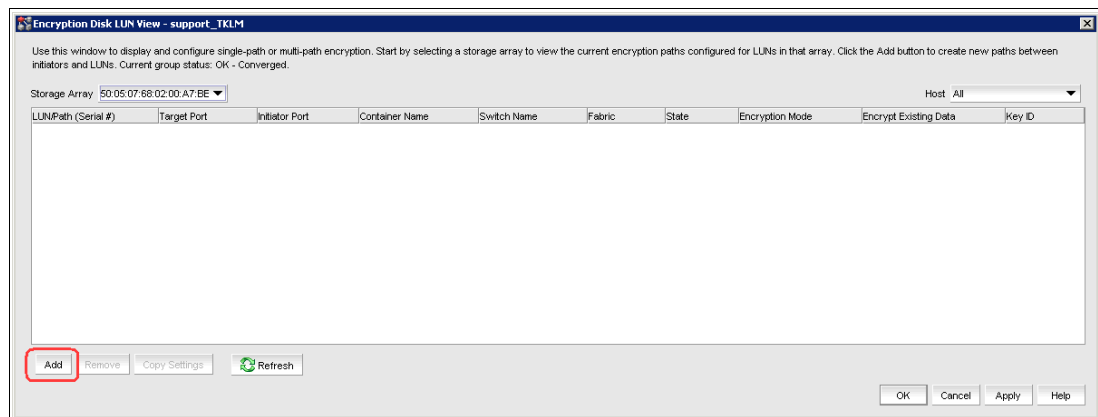


Figure 4-26 DCFM Disk LUNs main window

3. When you click **Add**, a wizard begins to guide you through adding the LUNs to each CTC. If you have multiple CTC definitions to which you want to grant access to the same LUN, you must *repeat this action* multiple times until the LUNs have been added to *each* CTC.

- The first part of the wizard to add a new LUN prompts you to select and confirm the target port from the LUN that you want to add. In our setup, we started with CTC_3B, because it was the default selected Storage Array in our main window. CTC_3B pointed to the target port of the disk subsystem to which CTC_3B was associated (Figure 4-27).

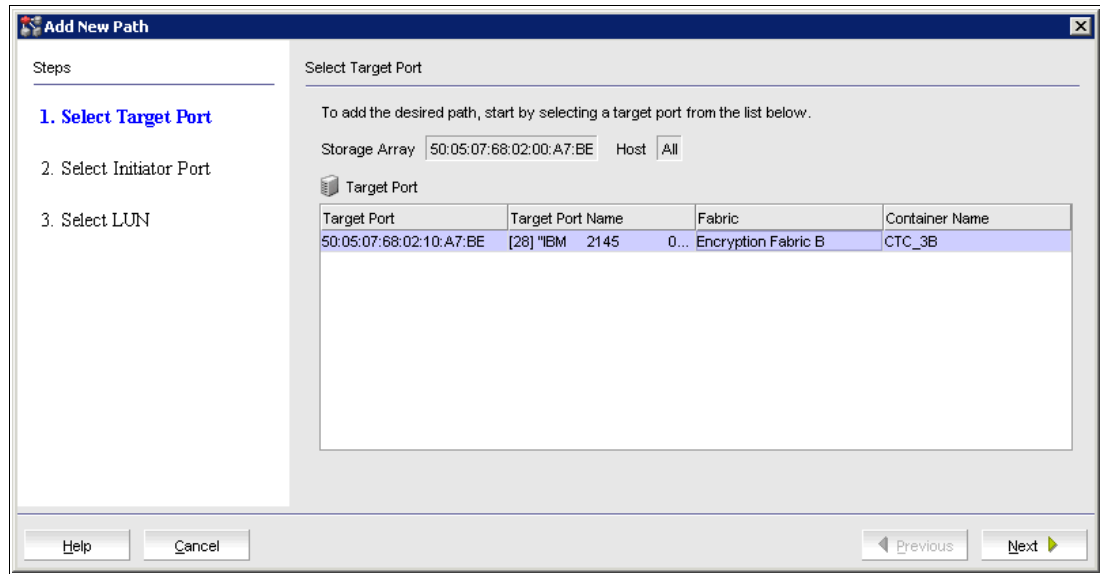


Figure 4-27 DCFM Add New Path wizard: Select Target Port

- When you click **Next**, you need to select the related initiator port, as shown in Figure 4-28.

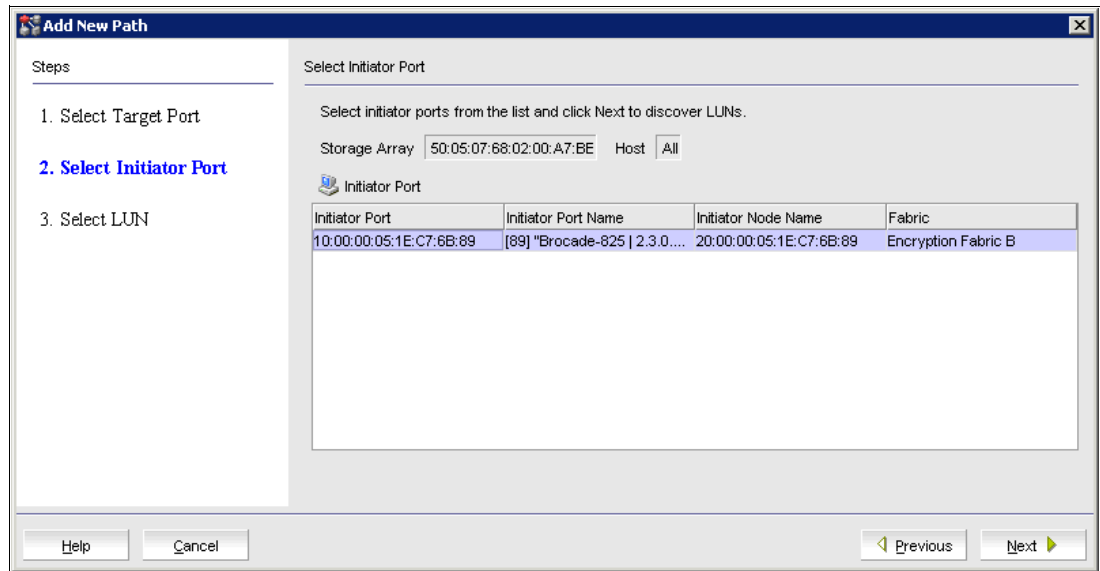


Figure 4-28 DCFM Add New Path wizard: Select Initiator Port

- Continue to click **Next** to finally select the LUNs. Figure 4-29 on page 125 shows the Select LUN start window.

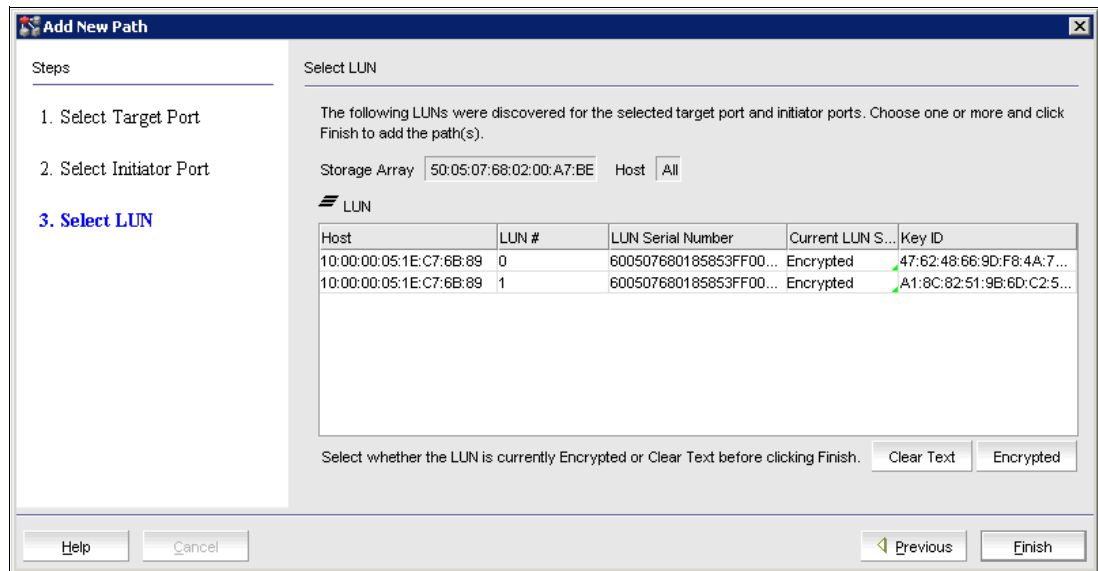


Figure 4-29 DCFM Add New Path wizard: Select LUN

- Because we created two LUNs previously for the selected target port, both LUNs are listed to add to each new CTC. If you have multiple LUNs at this point, you can select **All** in this window (as shown in Figure 4-30) if you want to add all of them to the CTC, and click **Finish**.

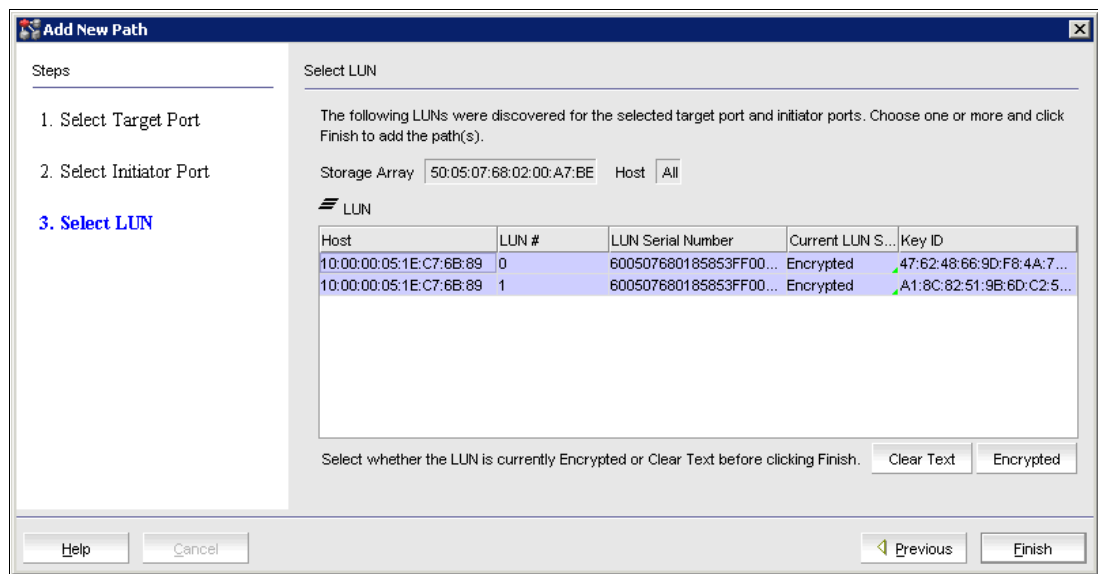


Figure 4-30 DCFM Add New Path wizard: Select LUN

- Next, you see an informational box (see Figure 4-31 on page 126) advising that a new definition of a new path has just been created, but it will *not be committed* until you click **OK** or **Apply** on the main Disk LUN window.

Important: You need to continue, at this point, to add the LUN (path) to *all* other required CTC definitions.

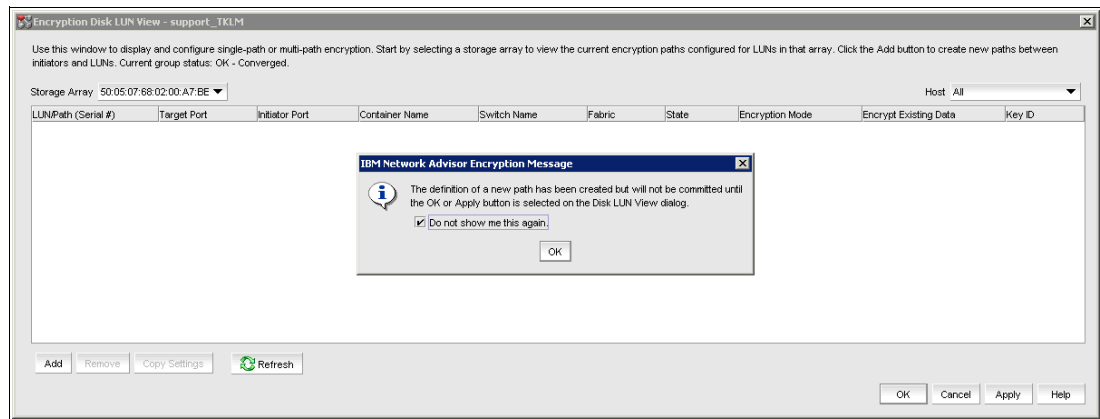


Figure 4-31 DCFM Add New Path wizard: Final reminder to commit and apply changes

4.4.9 Activating the DEK cluster CTC definitions

In the previous two sections, we performed these tasks:

- ▶ Created the new CTC definitions in the new encryption member in the second fabric (the DEK cluster)
- ▶ Added the existing LUNs (two volumes), which need be encrypted, to *all* the newly generated CTCs

The last informational window from the Add path/LUN wizard (shown in Figure 4-31) advised us that the new definition of a new path has just been created but will *not be committed* until **OK** or **Apply** is clicked in the main Disk LUN window. During the final activation, we show how to perform this commit step, and we show the errors that you might encounter, because there are still several parameters that you need to adjust correctly.

At this point we have clicked **Apply** in the main view of the Disk LUN window. Another box appears. In this box, you need to confirm your choice again by clicking **Yes** (see Figure 4-32).

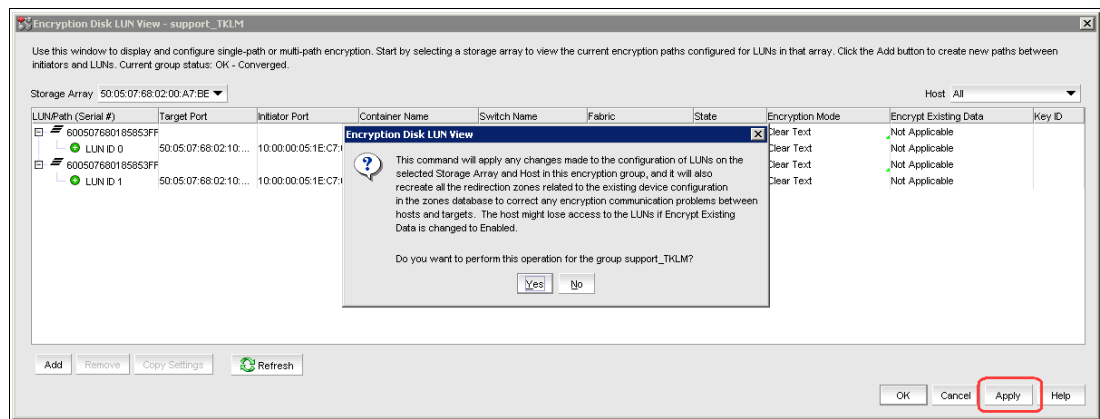


Figure 4-32 DCFM Add New Path wizard: Commit and apply changes

If you continue at this point, you might discover the following error message, as shown in Figure 4-33 on page 127.

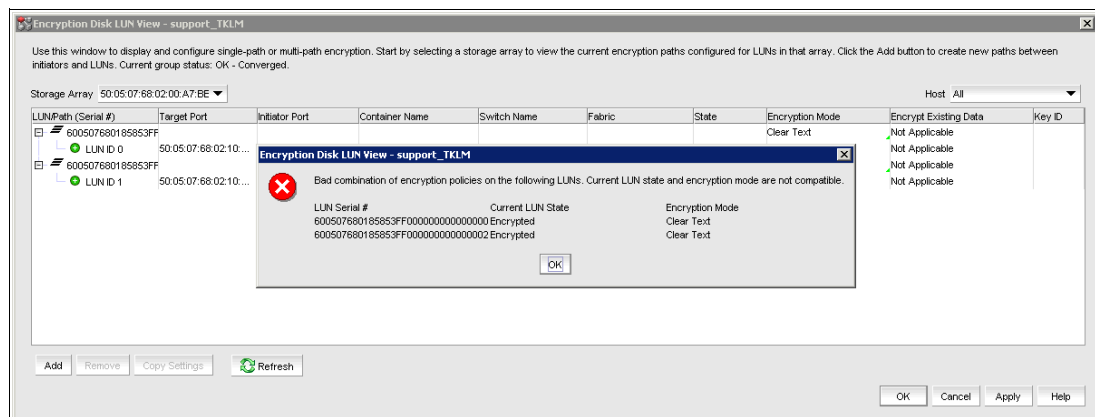


Figure 4-33 DCFM Add New Path wizard: Error on commit and apply changes

The error message states that we have a bad combination of encryption policies and LUNs (the two existing LUNs that we previously added in our example). The error message states that the current LUN state and encryption mode are incompatible.

This error is correct, because we added *existing, encrypted* LUNs to a new path (in this case, a second fabric to build a DEK cluster solution).

The clue is related to the prior window (Figure 4-32 on page 126), because, after adding the LUNs, by default, the entries in the Encryption Mode column are set to *Clear Text*, as shown in Figure 4-34.

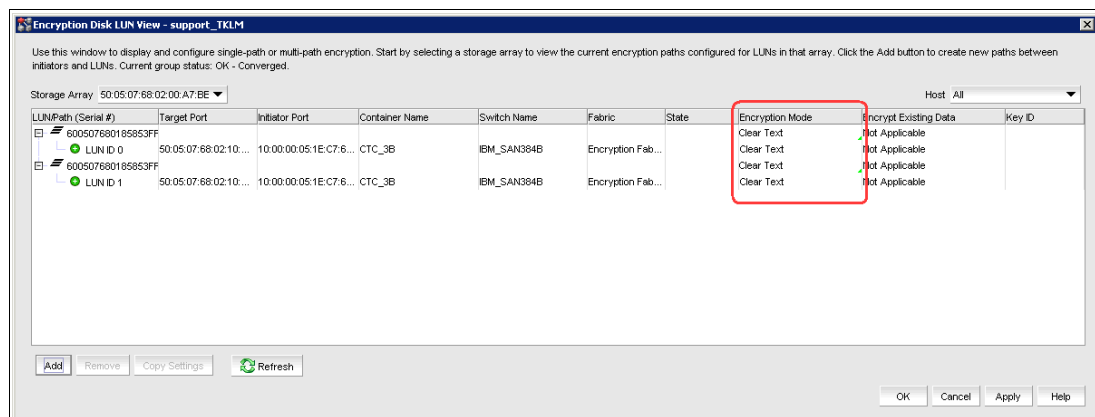


Figure 4-34 DCFM Disk LUNs main window to display and modify the encryption mode

Because the existing LUNs are *already encrypted*, we need to change the Encryption Mode for every LUN, in every CTC that we added, as shown in Figure 4-35 on page 128 and Figure 4-36 on page 128 to **Native Encryption**. We started with CTC_4B. To change the view to modify the other CTCs (if present, and, in our example, CTC_1B to CTC_3B), you need to use the upper-left pull-down Storage Array list again to switch the view to the other CTCs. See Figure 4-37 on page 128.

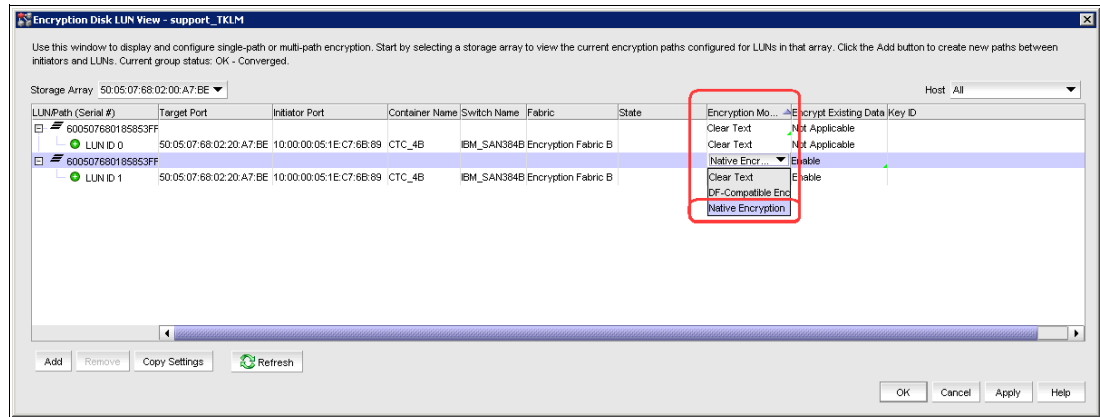


Figure 4-35 DCFM Disk LUNs main window to modify the encryption mode

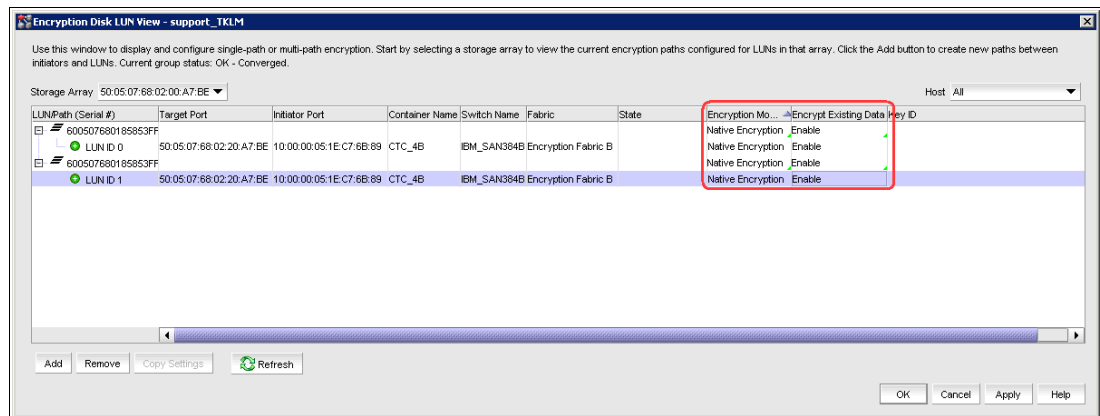


Figure 4-36 DCFM Disk LUNs main window to confirm the modified encryption modes

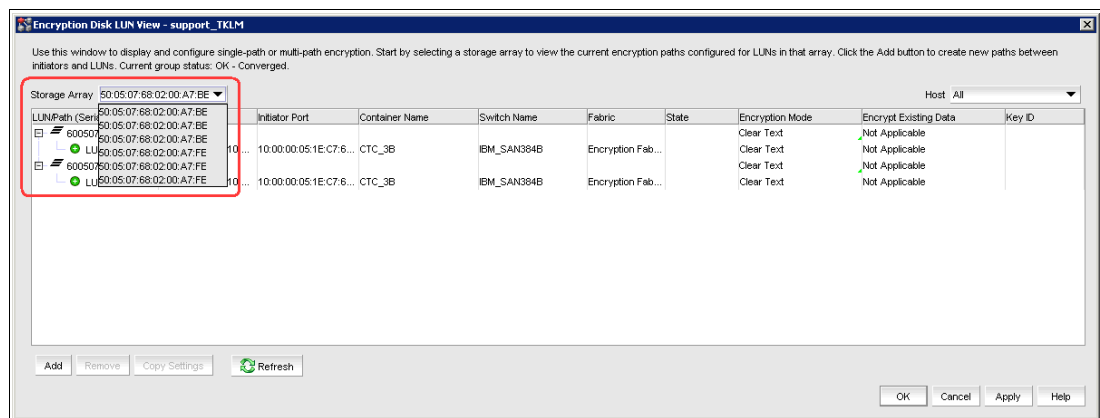


Figure 4-37 DCFM Disk LUNs main window to switch the view to the other CTCs

After we have modified the Encryption Mode for all existing LUNs, in all CTCs, from Clear Text to **Native Encryption**, we can try to commit and apply the changes again.

Unfortunately, another error message might display stating that the LUN cannot be added to the target. The details are “Invalid combination of LUN options specified”, as shown in Figure 4-38 on page 129.

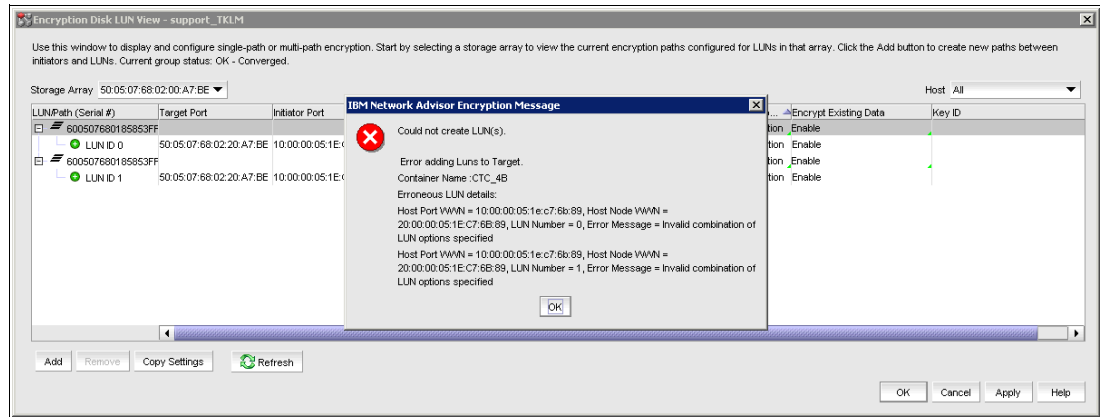


Figure 4-38 DCFM Add New Path wizard: Another error on commit and apply changes

The error text here might not explain the problem as well as the other error message did. But, if you refer back again to the previous window (Figure 4-36 on page 128), you can easily discover the error that we made there.

Look at the second column next to where we needed to adjust the values before. Because we previously changed the Encryption Mode from Clear Text to **Native Encryption**, the DCFM window assumes that the associated LUNs will perform a first-time encryption and automatically enables the entries in the next column for Encrypt Existing Data. This assumption, of course, does not match our scenario, because we added existing, encrypted LUNs. We do *not* need to encrypt these LUNs again. Therefore, we must modify all the LUNs in all the new CTCs to set this field to **Disable**, as shown in Figure 4-39 and Figure 4-40 on page 130.

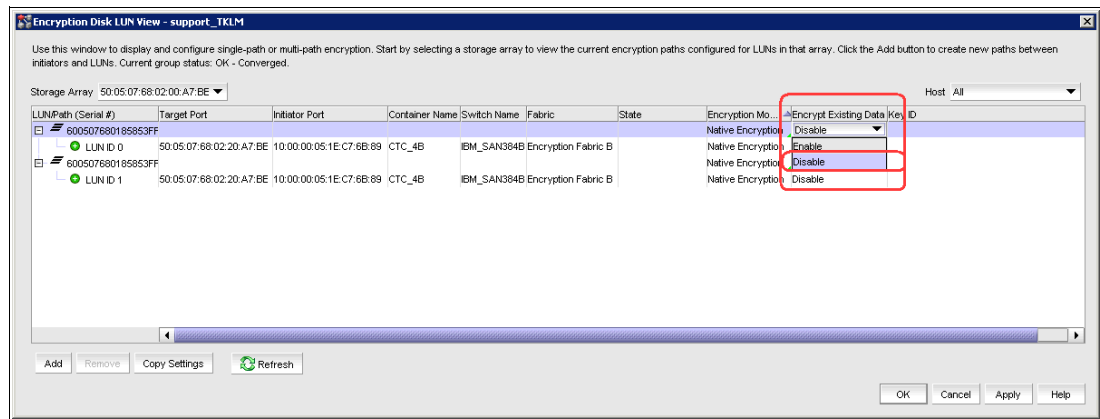


Figure 4-39 DCFM Disk LUNs main window to modify Encrypt Existing Data field to Disable

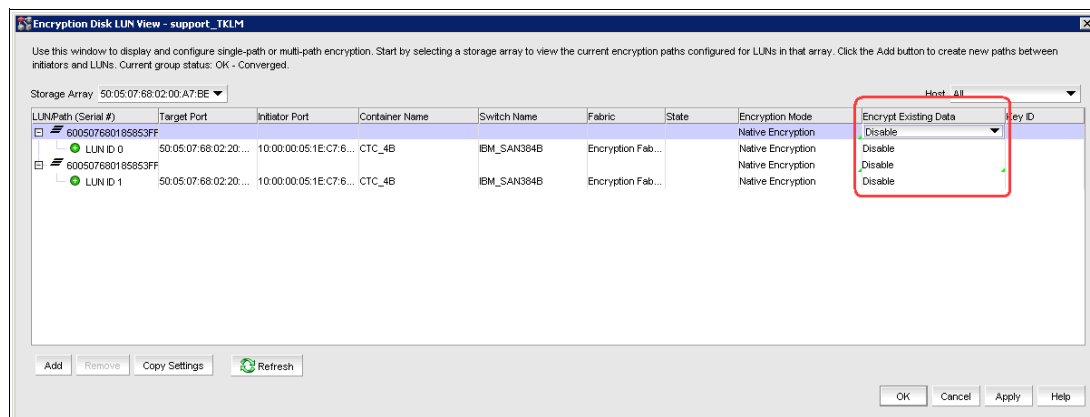


Figure 4-40 DCFM Disk LUNs main window to confirm updated Encrypt Existing Data field is Disable

With these two important modifications to the Encryption Mode and Encrypt Existing Data fields, you can successfully apply and commit the new CTC definitions.

Plan ahead: Whenever you add LUNs to the CTCs, carefully plan to perform this task. Know in advance whether you are adding a new and empty LUN, a used LUN in clear text, or an already encrypted LUN, and enter the appropriate settings in the DCFM Disk LUNs main window.

4.4.10 Final activation/confirmation of the new paths (via a second fabric)

It is important to prevent host server access to the existing LUNs via the second fabric (or new paths) as long as we are in the process of defining and preparing them at the SAN32B-E4 Encryption Switch layer. Therefore, we previously disabled the SAN port from the switch to the host server port, which we configured for the new paths.

Now that all the components are installed (DEK cluster) and all new definitions are in place (new CTCs and added LUNs), we can re-enable the host server initiator port in the SAN switch to which it is attached.

After the host server gets access to the SAN, it will discover the new, additional paths to the already existing LUNs. In our setup, we used a Microsoft Windows-based host server and the IBM Storwize V7000 disk storage subsystem. In this configuration, the IBM SDD multipath device driver is used in the Windows host server, and it automatically will detect the new paths. Other operating systems might need manual intervention to detect the new paths, for example, a rescan to detect the new paths. Refer to the manuals for your specific host operating system for information about discovering additional paths.

Example 4-26 and Example 4-27 on page 131 show the IBM SDD output before and after adding the additional paths via the second fabric. In our example, we see two LUNs, because we added four paths for the two existing LUNs via a new second fabric, Fabric B, to establish the overall DEK cluster configuration (dual fabric).

Example 4-26 IBM SDD output to display the paths to the storage system before adding the new paths

```
DEV#: 0 DEVICE NAME: Disk1 Part0 TYPE: 2145 POLICY: OPTIMIZED
SERIAL: 600507680185853FF0000000000000000
=====
Path# Adapter/Hard Disk State Mode Select Errors
0 Scsi Port12 Bus0/Disk1 Part0 OPEN NORMAL 13469057 0
```


1	Scsi Port12 Bus0/Disk1 Part0	OPEN	NORMAL	322	0
2	Scsi Port12 Bus0/Disk1 Part0	OPEN	NORMAL	13457053	0
3	Scsi Port12 Bus0/Disk1 Part0	OPEN	NORMAL	495	0

DEV#: 1 DEVICE NAME: Disk2 Part0 TYPE: 2145 POLICY: OPTIMIZED
 SERIAL: 600507680185853FF000000000000002

Path#	Adapter/Hard Disk	State	Mode	Select	Errors
0	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	25727	0
1	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	19567	0
2	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	111	0
3	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	386	0

Example 4-27 BM SDD output to display the paths to the storage system after adding the new paths

DEV#: 0 DEVICE NAME: Disk1 Part0 TYPE: 2145 POLICY: OPTIMIZED
 SERIAL: 600507680185853FF000000000000000

Path#	Adapter/Hard Disk	State	Mode	Select	Errors
0	Scsi Port12 Bus0/Disk1 Part0	OPEN	NORMAL	13469057	0
1	Scsi Port12 Bus0/Disk1 Part0	OPEN	NORMAL	322	0
2	Scsi Port12 Bus0/Disk1 Part0	OPEN	NORMAL	13457053	0
3	Scsi Port12 Bus0/Disk1 Part0	OPEN	NORMAL	495	0
4	Scsi Port11 Bus0/Disk1 Part0	OPEN	NORMAL	0	0
5	Scsi Port11 Bus0/Disk1 Part0	OPEN	NORMAL	0	0
6	Scsi Port11 Bus0/Disk1 Part0	OPEN	NORMAL	4613	0
7	Scsi Port11 Bus0/Disk1 Part0	OPEN	NORMAL	4435	0

DEV#: 1 DEVICE NAME: Disk2 Part0 TYPE: 2145 POLICY: OPTIMIZED
 SERIAL: 600507680185853FF000000000000002

Path#	Adapter/Hard Disk	State	Mode	Select	Errors
0	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	25727	0
1	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	19567	0
2	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	111	0
3	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	386	0
4	Scsi Port11 Bus0/Disk2 Part0	OPEN	NORMAL	0	0
5	Scsi Port11 Bus0/Disk2 Part0	OPEN	NORMAL	0	0
6	Scsi Port11 Bus0/Disk2 Part0	OPEN	NORMAL	231	0
7	Scsi Port11 Bus0/Disk2 Part0	OPEN	NORMAL	57	0

4.5 Adding a second Encryption Switch for high availability

In this section, we add a second SAN32B-E4 Encryption Switch to our Fabric A, which already has one installed Encryption Switch and actively encrypts target LUNs. The intent of adding a second encryption node to the fabric is to provide a high-availability (HA) setup. When two encryption nodes are attached to the same fabric, you can configure them to operate independently, providing a manually defined load split; that is, you define the first target port container on EE1 and the second port on the target to EE2. This setup is good for sharing the load, but if one of the EEs fails, we lose a path to the target, which causes degradation to all the hosts. However, you also can define these two EEs in an HA cluster. In this case, when one EE fails, that EE's CTCs will fail over to the other EE.

It is also possible to configure your CTCs so that only one EE in the HA cluster performs all the work and the second EE is purely standby in case the first EE fails or has its firmware updated. We will configure this active/standby scenario, because we already have one EE installed and running with all of our Fabric A CTCs. We will add a second EE for the purpose of HA.

Typically, the requirement for HA suggests that dual fabrics are also in use. We will only show the steps for one fabric, but you can repeat the same steps for a second fabric.

To add the second switch for HA, it already must be configured, registered with the TKLMs, and added to the EG. Because we already have described these tasks in previous sections, we will begin by assuming that these steps already have been performed and that the new SAN32B-E4 is FC-connected to the Fabric A switch. *However, ensure that you do not forget to set the NTP time server on the new switch.*

With the new SAN32B-E4 Encryption Switch fabric attached and already in our EG, we are ready to define an HA cluster. First, we determine which EE is the Group Leader (GL) by issuing **cryptocfg --show -groupmember -all** for any of the existing EEs. Then after logging on to the GL, we issue the **cryptocfg** command, as shown in Example 4-28, to create an HA cluster named **enc_ha_fabA**. We must specify the WWN of the GL Encryption Switch, which was determined from the **show groupmember** output.

You do not need to add the existing EE to the HA cluster. It is added automatically, because it is in the same fabric as the new EE that we have just added.

Example 4-28 Using the CLI to create an HA cluster

```
SAN32B-E4-1:admin> cryptocfg --create -hacluster enc_ha_fabA
10:00:00:05:1e:54:17:10
```

EE Node WWN	Slot Number	Local/Remote
10:00:00:05:1e:54:17:10	0	Local

```
Operation succeeded.
SAN32B-E4-1:admin>
```

Next, we add the new SAN32B-E4 EE to the HA cluster that we have just defined. We determine the new switch's WWN by issuing the **wwn** command on that switch. As shown in Example 4-29, the new switch WWN is added to the newly created HA cluster by using the CLI on the GL. By committing these definitions, our HA cluster is operational.

Example 4-29 Adding the new EE to the HA cluster

```
SAN32B-E4-1:admin> cryptocfg --add -haclustermember enc_ha_fabA
10:00:00:05:1e:54:16:53
```

EE Node WWN	Slot Number	Local/Remote
10:00:00:05:1e:54:16:53	0	Remote

```
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --commit
Crypto device configuration out of sync is detected during EG formation.
The commit operation distributes the configuration to member nodes
ARE YOU SURE (yes, y, no, n): [no] y
Operation succeeded.
SAN32B-E4-1:admin>
```

Using the **show hacluster** parameters, we can confirm the HA cluster state. As shown in Example 4-30, we can confirm that the HA cluster status is committed and converged. Also, we see that both EEs are online.

Example 4-30 Displaying the HA clusters

```
SAN32B-E4-1:admin> cryptocfg --show -hacluster -all
Encryption Group Name: support_TKLM
Number of HA Clusters: 1

HA cluster name: enc_ha_fabA - 2 EE entries
Status:          Committed
HAC State:       Converged

          WWN              Slot Number  Status
10:00:00:05:1e:54:17:10      0         Online
10:00:00:05:1e:54:16:53      0         Online
SAN32B-E4-1:admin>
```

The following steps show how to create the same HA cluster by using IBM DCFM. We first select **Configure** → **Encryption** to open the Encryption Devices (Group View) window, as shown in Figure 4-41. Note that our new SAN32B-E4-2 has already been defined to the EG.

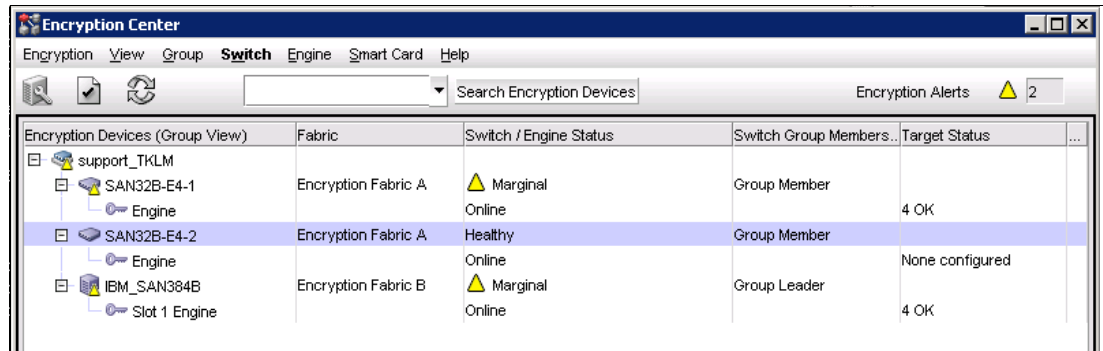


Figure 4-41 New EE has already been added to the EG

By clicking the EG’s line and right-clicking, a submenu is displayed. We select **HA Clusters**, as shown in Figure 4-42 on page 134.

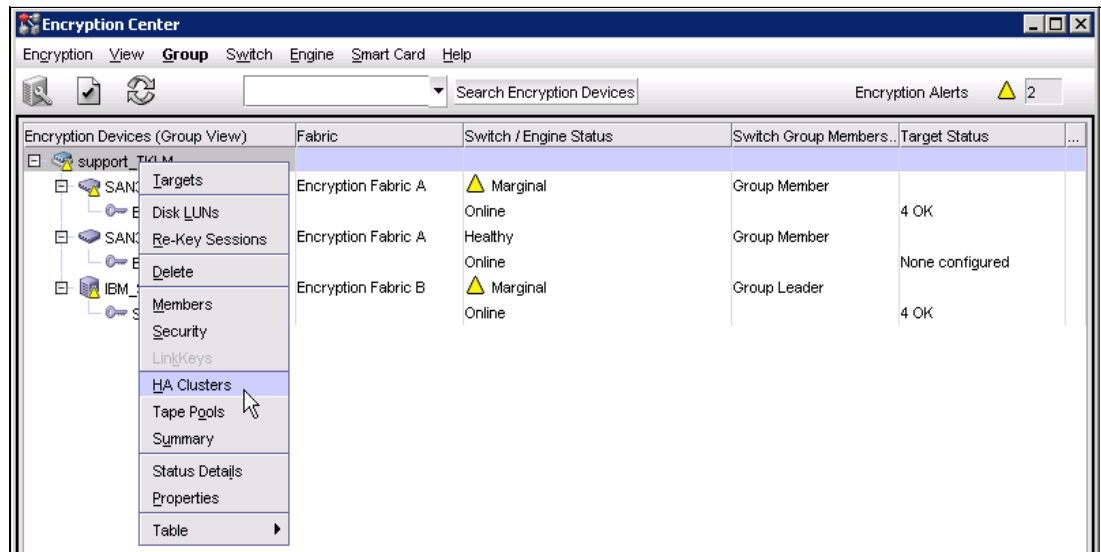


Figure 4-42 Selecting the HA Clusters option

A new HA Clusters tab opens, as shown in Figure 4-43. This window allows us to use the arrow icon to add selected EEs from the left side to the High-Availability Clusters definition on the right side.

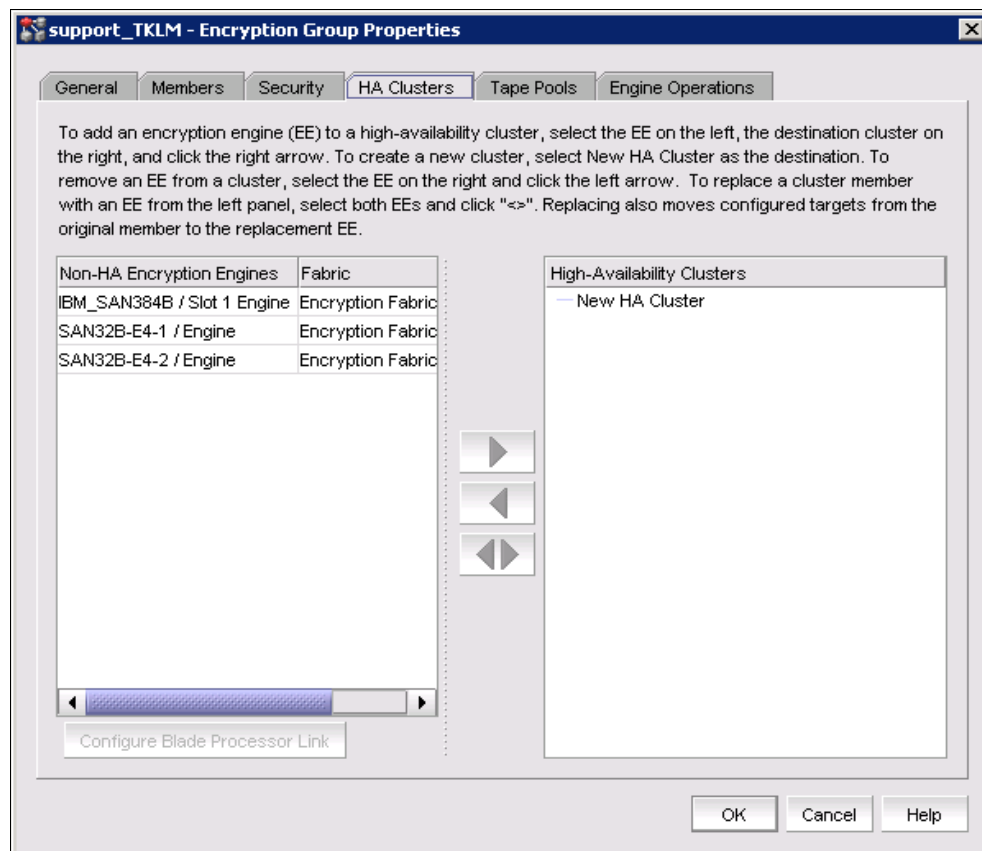


Figure 4-43 HA Clusters panel

In this case, we do not have an HA cluster defined yet, so we select our first EE from the left side and select **New HA Cluster** on the right side, as shown in Figure 4-44.

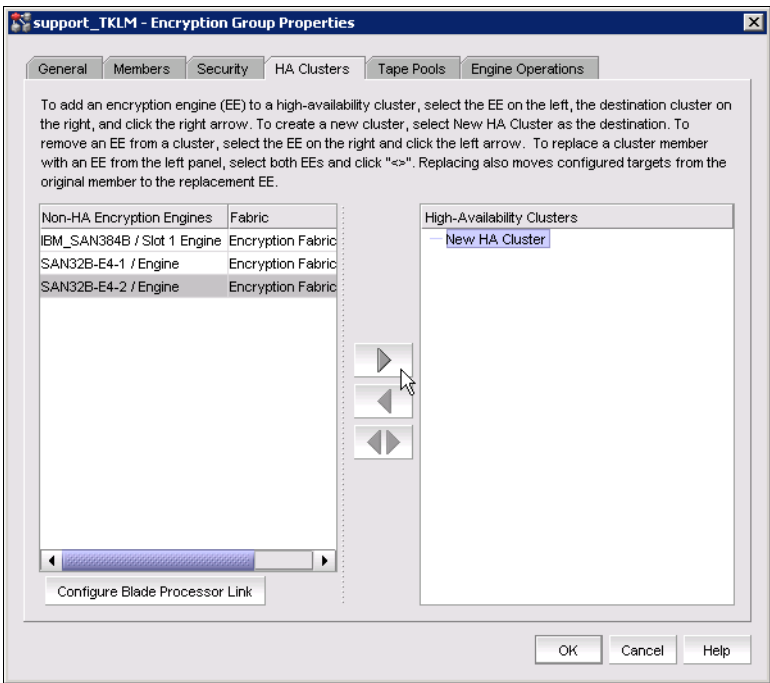


Figure 4-44 Selecting an EE for the HA cluster

By clicking the arrow icon (>) to move our selected non-HA EE to the right pane, we are prompted to name our new HA cluster, as shown in Figure 4-45.



Figure 4-45 Naming the HA cluster

After naming the HA cluster a name and clicking **OK**, we return to the previous window. If we click **OK** to complete our definitions, we are prompted with a message box, as shown in Figure 4-46 on page 136, that states that we must have more than one EE in an HA cluster. This behavior differs from the CLI method where it automatically added the other EE in the same fabric. In this process, we must define it.

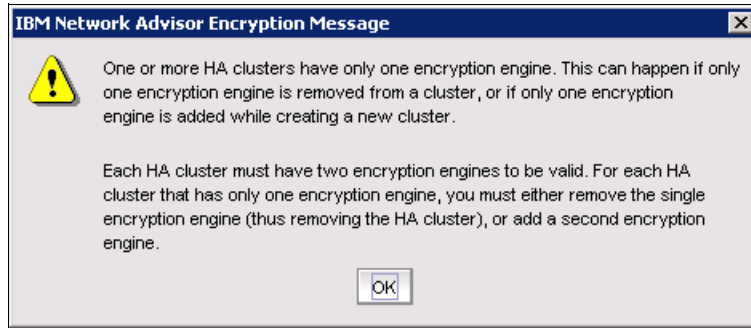


Figure 4-46 Notification of incomplete HA definitions

As shown in Figure 4-47, we have added the SAN32B-E4-1 to our HA cluster on the right panel. After we click **OK**, the definitions are committed to the switches.

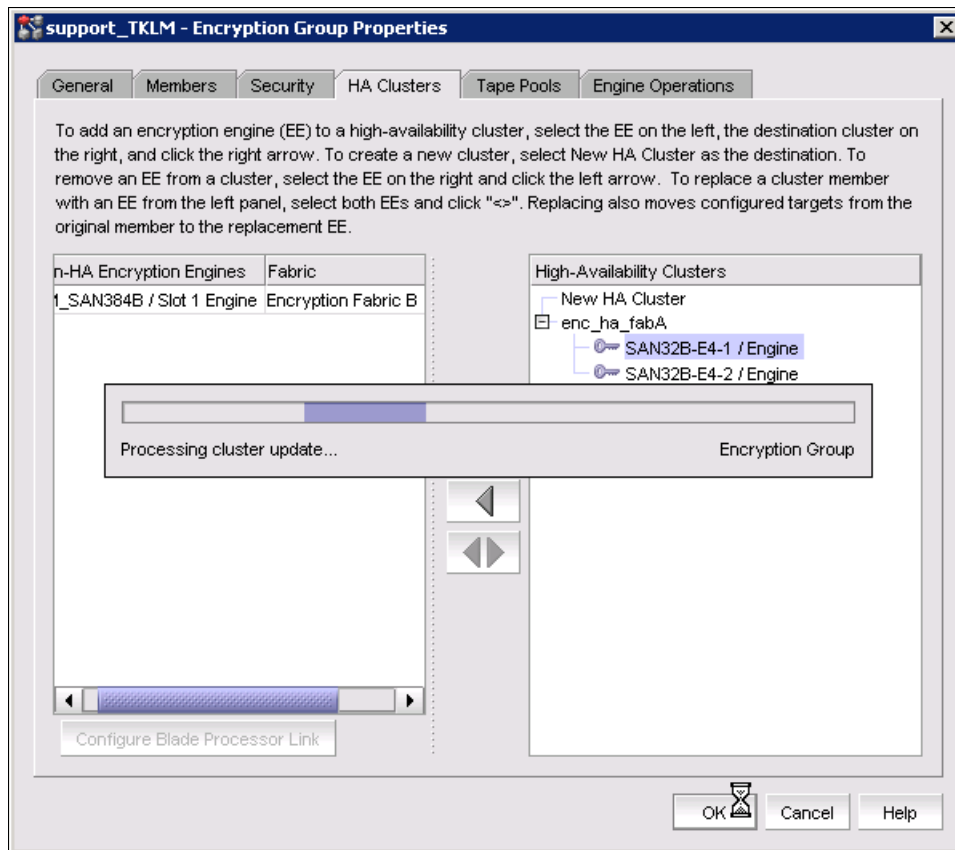


Figure 4-47 Completing the HA definitions

After the commit processing completes, we return to the Encryption Center view, which is shown in Figure 4-48 on page 137, where we now see both SAN32B-E4 Encryption Switches in the same HA cluster.

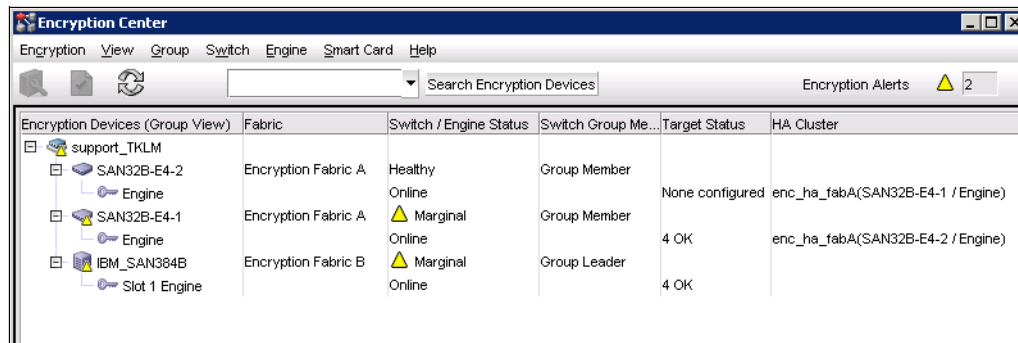


Figure 4-48 Encryption center showing the HA cluster

Now that we have an HA cluster, we test the failover behavior by disabling SAN32B-E4-1 while our host performs a large file copy from the Storwize V7000 LUN. Our initial test was a cause for concern, because we discovered that while the HA cluster was created correctly, we had neglected to define the NTP server to the new EE. Because the time on this EE was in the future, the certificate that was created at switch initialization time was seen as invalid on the TKLM servers. The effect of this situation is that, while the EE in the HA cluster failed over, no traffic was being passed and our file copy was not progressing. After correcting that situation and ensuring that all EEs were *connected* to the TKLM servers successfully, we performed the same test. This time, we observed a short pause in the file copy while the EE failed over, and the copy completed with the CTCs failed over to the SAN32B-E4-2 Encryption Switch. By making the failed EE available again, the CTCs automatically failed back without any intervention.

4.6 Creating and adding a new LUN to an existing EG

In this section, we show how easy it is to add a new LUN to an existing EG. We also show how simple and fast a new LUN can be created on the IBM Storwize V7000 to implement an encryption solution swiftly.

4.6.1 Creating and adding a new LUN in a single-fabric environment

In this section, we show the step-by-step process to create and add a new LUN to an encryption node within a single-fabric environment. Our starting configuration, at this point, refers to the configuration that we successfully established in 4.1, “Configuring encryption on a single fabric” on page 90. Figure 4-49 on page 138 shows the starting hardware setup for the upcoming tasks in this section.

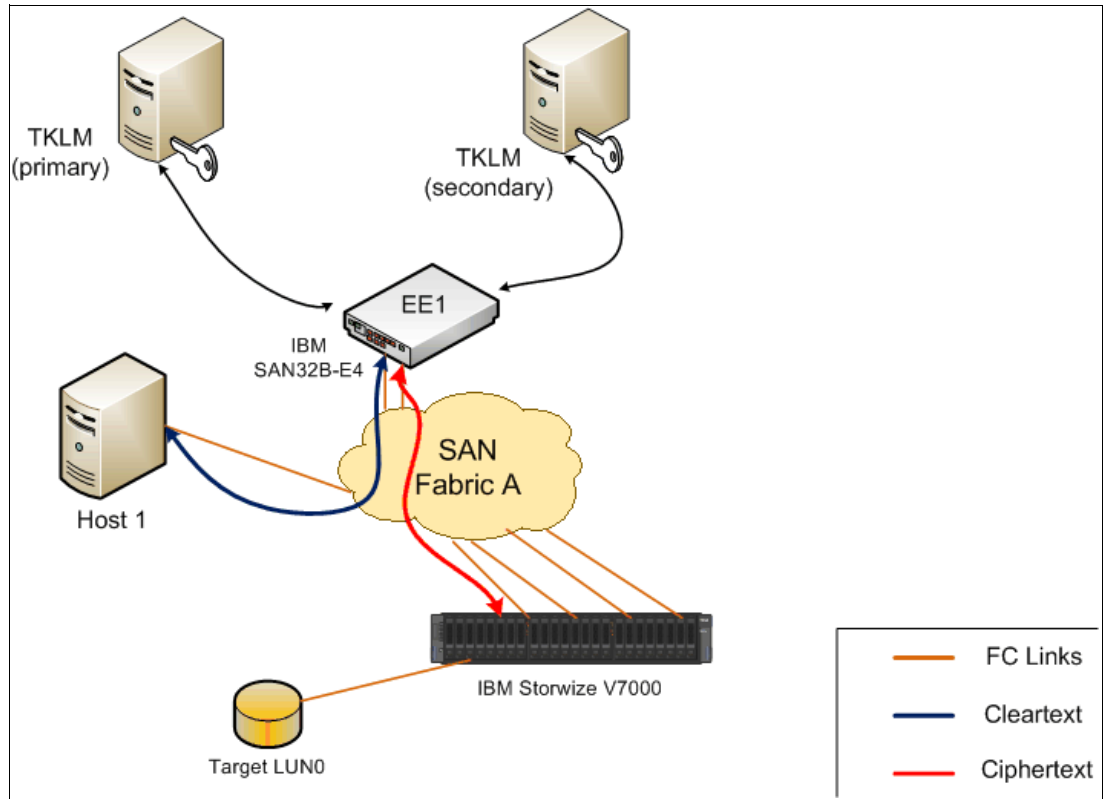


Figure 4-49 Starting configuration for creating and adding a new LUN to the EG

4.6.2 Creating a new LUN in the IBM Storwize V7000

To create a new LUN within the IBM Storwize V7000 disk subsystem, you typically use the web GUI of the IBM Storwize V7000 disk subsystem. As shown in Figure 4-50 on page 139, from the main window after the login, navigate to the **Volumes** icon and select **Volumes by host**.

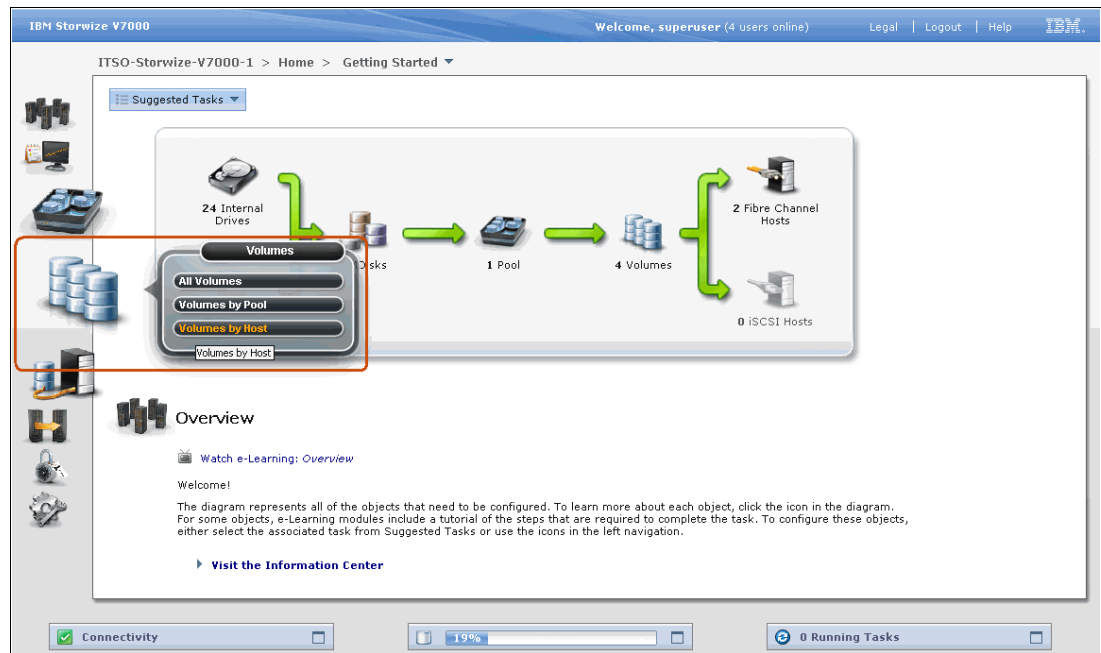


Figure 4-50 Creating a new LUN within the IBM Storwize V7000 subsystem

In Figure 4-51, click **New Volume**.

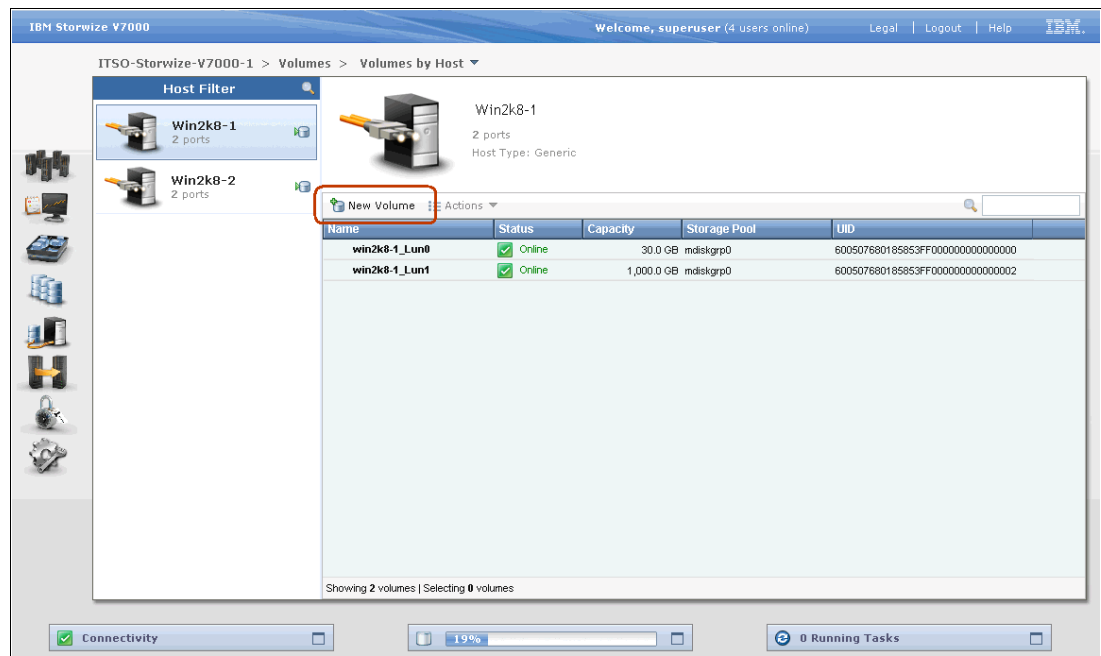


Figure 4-51 Creating a new LUN within the IBM Storwize V7000 subsystem

A new box appears for you to select the kind of new volume that you want to create. Select **Generic**, as shown in Figure 4-52.

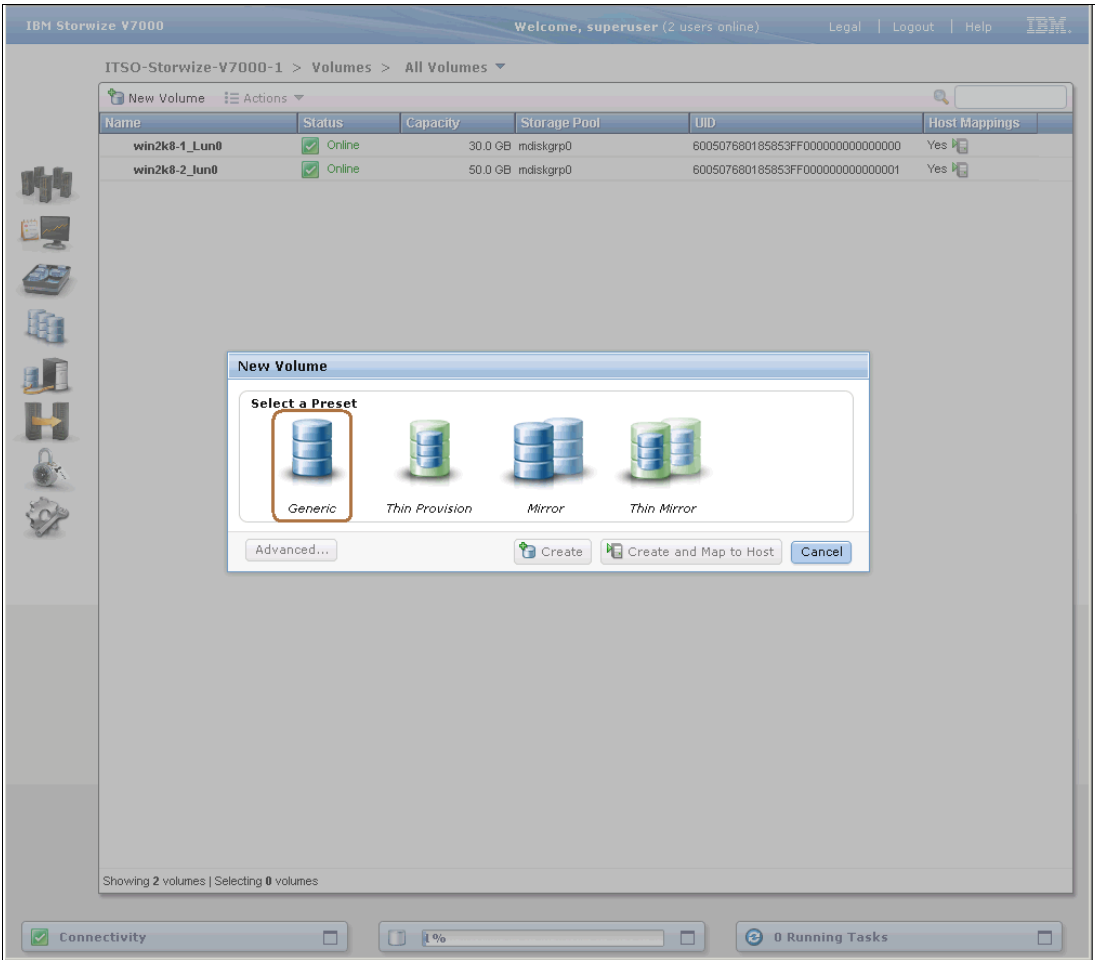


Figure 4-52 Creating a new LUN within the IBM Storwize V7000 subsystem

The following window asks you to enter a name and size for the new volume (LUN). Choose a name for the new volume, enter the volume size that you need, and click **Create and Map to host** (Figure 4-53).

The screenshot shows the 'New Volume' window with the following sections:

- Select a Preset:** Four icons representing different volume types: *Generic* (single blue disk), *Thin Provision* (single green disk), *Mirror* (two blue disks), and *Thin Mirror* (two green disks). A 'New Volume' label is positioned above the icons.
- Select a Pool:** A text field showing 'Primary Pool: mdiskgrp0' and an 'Edit' button.
- Select Names and Sizes:** A section containing a 'Volume Name' text field, a 'Size' text field, a unit dropdown menu set to 'GB', and a green plus icon for adding more volumes.
- Summary:** A line of text stating 'Summary: 1 volume, 0 bytes, 4.4 TB free in pool'.
- Buttons:** At the bottom, there are four buttons: 'Advanced...', 'Create' (with a green plus icon), 'Create and Map to Host' (with a green plus and disk icon), and 'Cancel'.

Figure 4-53 Creating a new LUN within the IBM Storwize V7000 subsystem

The next window opens (Figure 4-54). Select the host to which the new volume (LUN) will be mapped (Figure 4-54).

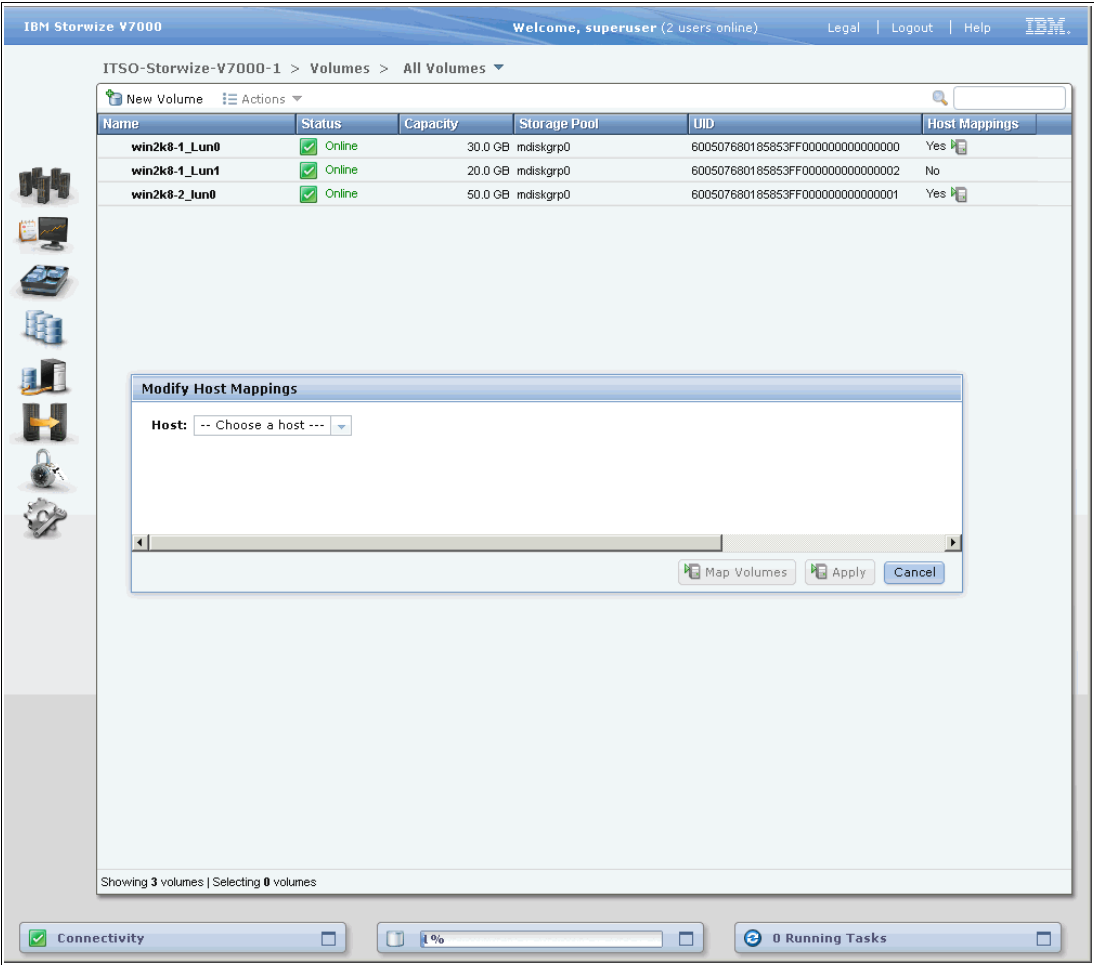


Figure 4-54 Creating a new LUN within the IBM Storwize V7000 subsystem

After you have selected the correct host to which you want to map the newly created volume (LUN), a new window opens (Figure 4-55). You need to associate the newly created LUN with the selected host.

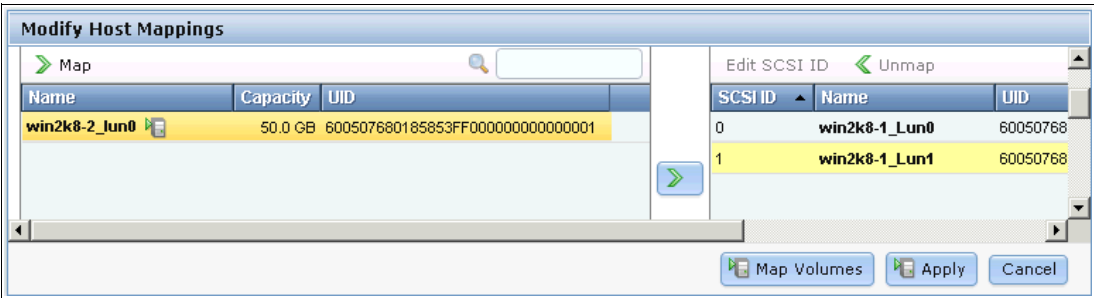


Figure 4-55 Creating a new LUN within the IBM Storwize V7000 subsystem

Important: The host server only appears in the list if it has already been defined and used within the IBM Storwize V7000 subsystem.

Select **Apply** and the final confirmation window opens, as shown in Figure 4-56. This window shows that the new volume (LUN) was created and is mapped to the host server that you selected.

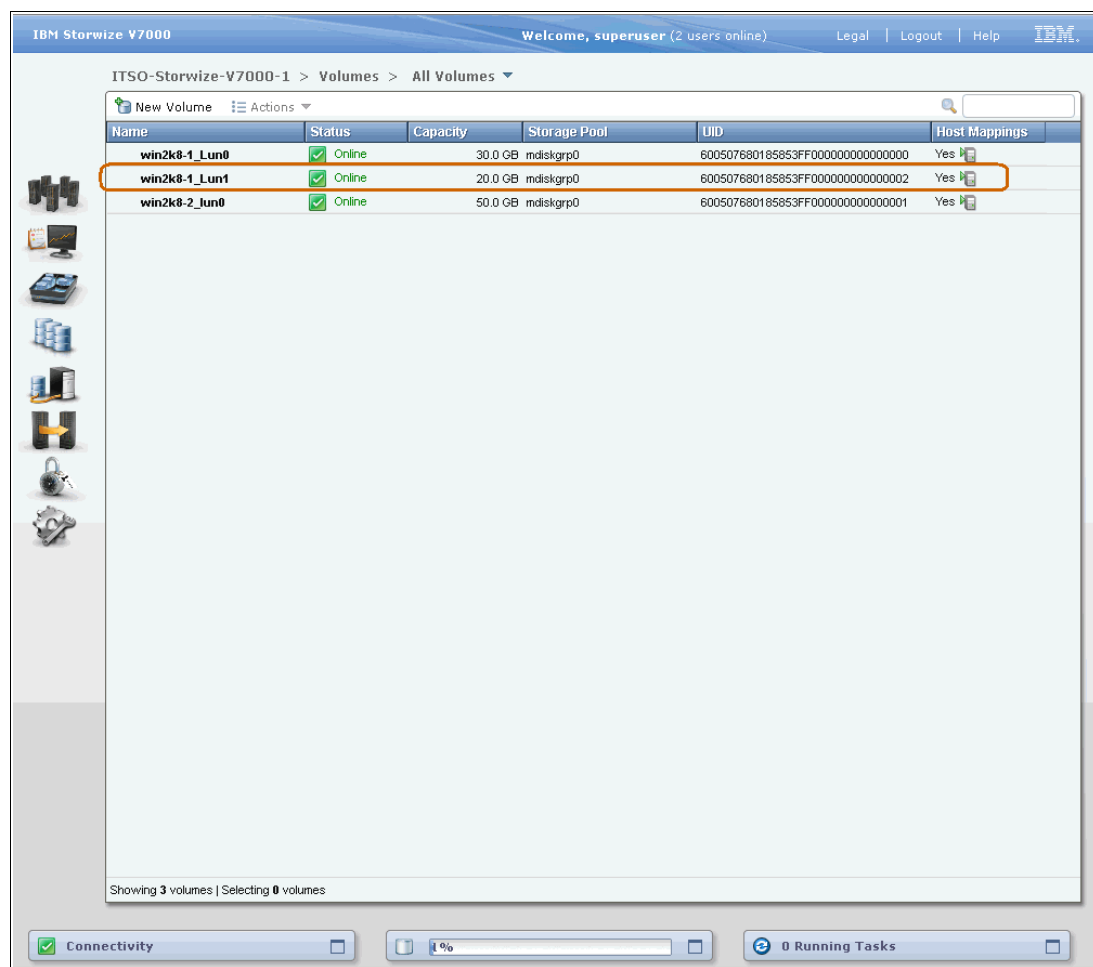


Figure 4-56 Creating a new LUN within the IBM Storwize V7000 subsystem confirmation

We have successfully created a new LUN in the IBM Storwize V7000 disk subsystem and mapped it to an existing host server.

Important: Do not scan at this point from the host server or servers to detect this newly created LUN. Before this newly created LUN can be used as an encrypted LUN, we first must add this new LUN into the existing SAN32B-E4 Encryption Switch setup (to the EG).

In the next section, we show how to add this newly created LUN into the existing SAN EG setup.

4.6.3 Adding a newly created LUN into an existing encryption group

In 4.6.5, “Creating a new LUN in the IBM Storwize V7000 (dual fabric)” on page 155, we have shown how easily a new LUN can be created in the IBM Storwize V7000 disk subsystem. Let us continue to incorporate this new LUN into the existing EG within the SAN32B-E4 Encryption Switch setup.

Remember these important definitions, which are explained in detail in Chapter 2, “Terminology and technology” on page 13:

- ▶ CTCs are target-port-centric to the storage subsystem.
- ▶ One unique CTC is needed for each path via the SAN down to each disk subsystem target port.
- ▶ LUNs are initiator-centric and defined within the CTCs.
- ▶ A CTC can have multiple initiators (host server ports) defined.
- ▶ A CTC also can have multiple target LUNs defined for the initiators:
 - Multiple LUNs for one initiator
 - Multiple LUNs for multiple initiators
 - A specific LUN for multiple initiators in a server clustered environment

Because we have made no changes at our disk subsystem target ports at this point, we do not need to change or add CTCs at this step. The only necessary task now is to add the newly created LUN into the existing CTCs.

Because in 4.1, “Configuring encryption on a single fabric” on page 90, we showed the steps to add a LUN to a CTC via the IBM Encryption Switch CLI interface, this time we will use the DCFM GUI to accomplish this task.

After logging in to the DCFM GUI, you need to open the DCFW Encryption Center via the DCFM menu. Navigate to **Configure** → **Encryption** to open the Encryption Center window (see figure Figure 4-57 and Figure 4-58 on page 145).

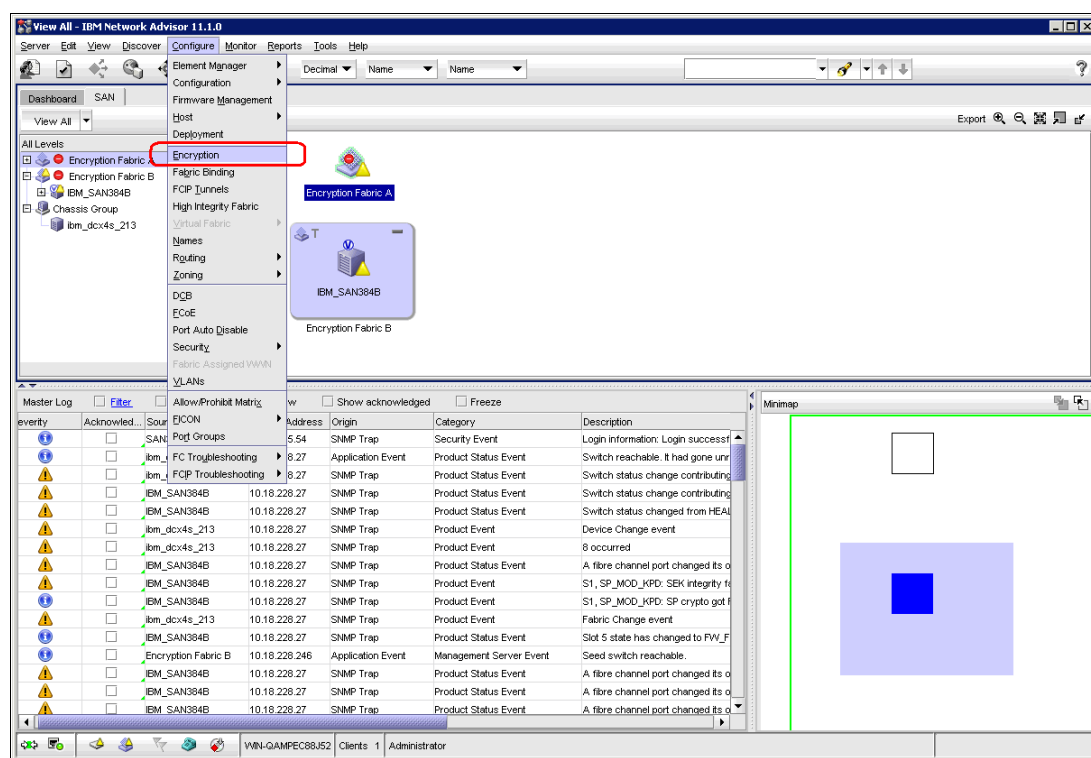


Figure 4-57 DCFM main GUI window

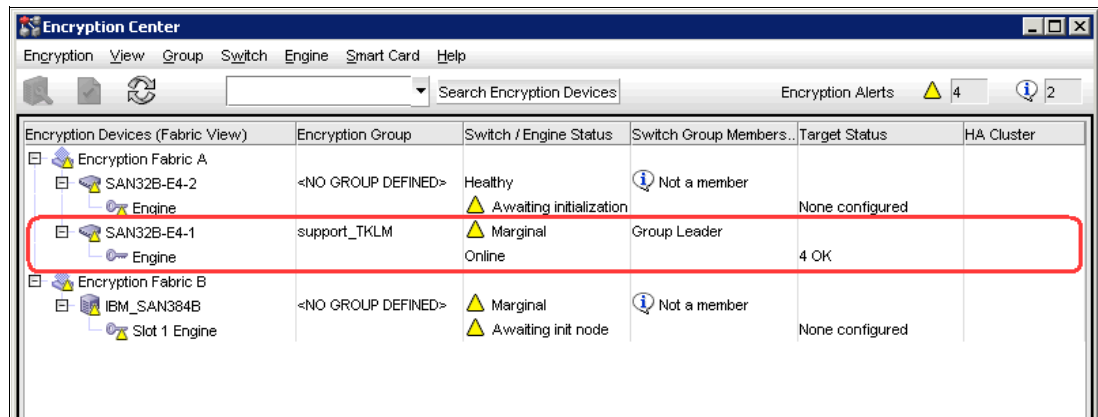


Figure 4-58 DCFM Encryption Center window

In Figure 4-58, you can review our setup and starting configuration. We discovered three encryption nodes. Two encryption nodes were IBM Encryption Switches (IBM SAN32B-E4) and one encryption node was an Encryption Blade in a director (IBM SAN384-3). At this point in time, you can see that only one SAN32B-E4 Encryption Switch was configured. It is part of an EG and acts as the Group Leader. Under the Target Status column, you can see that we have set up four CTCs, because we used four SAN paths to four target ports of the IBM Storwize V7000 subsystem. Ignore the Marginal warning icon (the reason for this informational message was because, in our lab setup, we only had one power supply configured in the IBM Encryption Switches that were being used).

After you open the DCFM Encryption Center, you can click by using the right-mouse button the encryption member to which you want to add the newly created LUN into the existing CTC definition. After the right-mouse click, select **Disk LUNs** from the option list.

You can also open the Disk LUNs window by selecting the encryption node in the Encryption Center window and use the main menu via **Group** → **Disk LUNs** to start the LUN configuration window, as shown in Figure 4-59.

Tip: If using the main menu in the Encryption Center and the pull-down options are all grayed out, you need to change to the correct view via **View** → **Fabrics** or **View** → **Group**, depending on your view and selection and which Encryption Center menu you want to use.

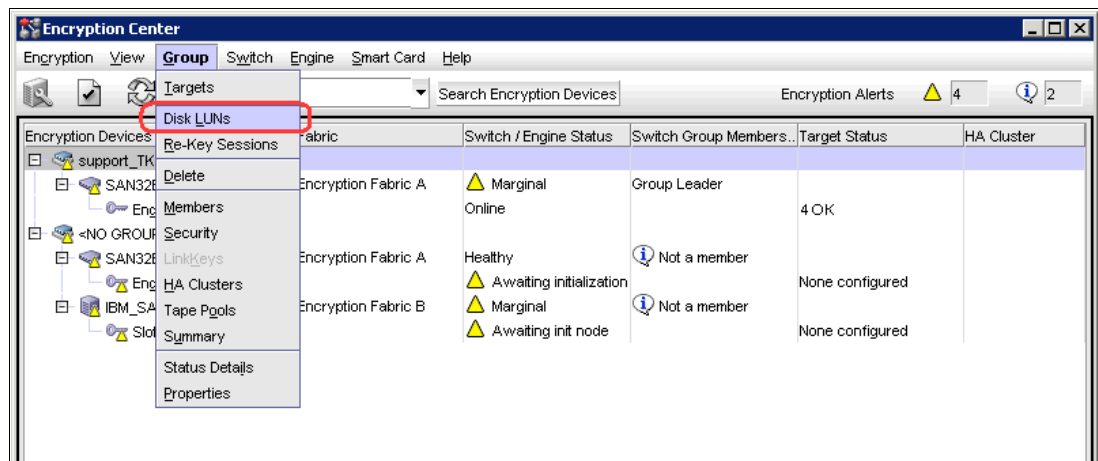


Figure 4-59 DCFM Encryption Center window: Opening the window to add disk LUNs

The Encryption Disk LUN View (refer to Figure 4-60) window opens to allow you to manage LUNs and add new LUNs.

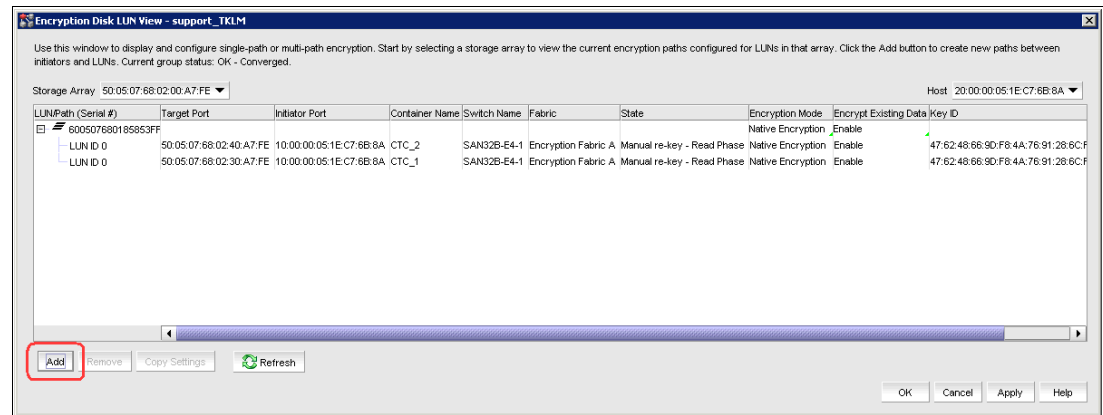


Figure 4-60 DCFM Encryption Center: Encryption Disk LUN View

In this window, you can see the previously defined LUNs and LUNs currently in use. In our setup, this window displays CTC_1 and CTC_2, which are associated to two separate physical target ports but are on one controller node within the IBM Storwize V7000 disk subsystem. The DCFM groups these two separate target ports together in this view, because they attach to the same disk subsystem controller node and end with the same WWNN (in this case, :A7:FE). With the Storage Array pull-down list in the upper-left corner, you can switch the view to the other target ports (CTCs).

We will continue to add the newly created LUN to the existing CTCs. Remember that we have four CTCs: CTC_1, CTC_2, CTC_3, and CTC_4. Click **Add** to open the wizard, as shown in Figure 4-61 on page 147.

Important: You need to repeat this step (adding the new LUN to the CTCs) multiple times, because you can add a new LUN within this wizard to only *one* CTC at a time. Therefore, you have to repeat this task as many times as the number of defined CTCs you have to access the LUN. In our setup, we perform this step four times.

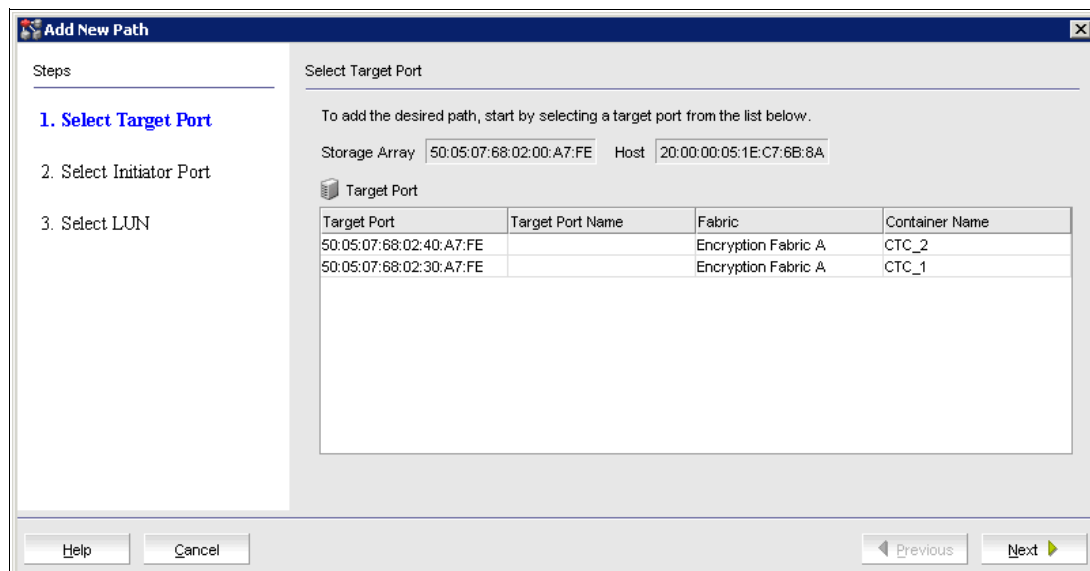


Figure 4-61 DCFM Encryption Center wizard to add a new LUN to a CTC

After the wizard to add a LUN to a CTC is displayed, you first select the Target Port (CTC) to which you want to add the LUN, and click **Next**. We started with the CTC called CTC_1, as shown in Figure 4-62.

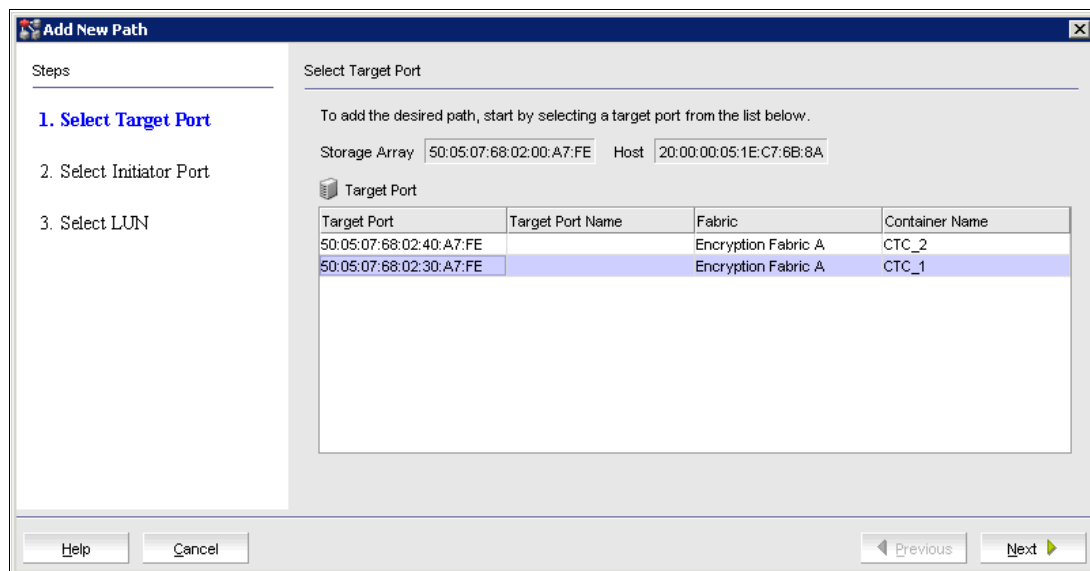


Figure 4-62 DCFM Encryption Center wizard to add a new LUN: Select Target Port

In the next window, as shown in Figure 4-63 on page 148, you need to select the associated initiator port, which is the port of the host server or servers that must have access to the newly created LUN.

Initiators: In our setup, we had only one host server with a single port HBA connected to the SAN as the initiator, and it had been previously defined to that CTC. So, we only had the option to select this one host port as the initiator for the new LUN. If you have more initiators that are defined in your CTC definition, you can select one or more initiators in this window.

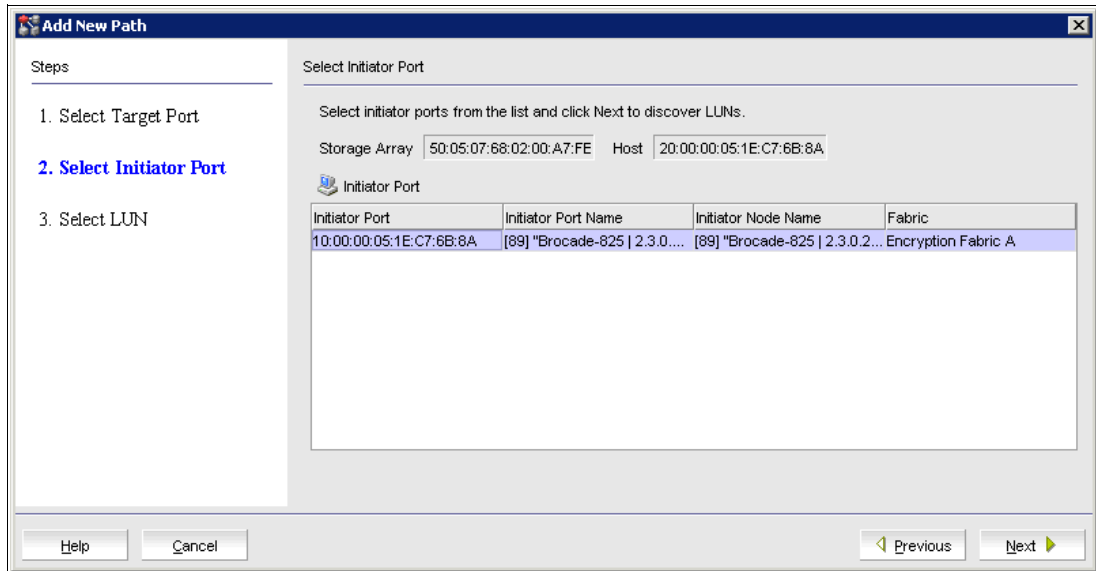


Figure 4-63 DCFM Encryption Center wizard to add a new LUN: Select Initiator Port

After clicking **Next**, the final window opens where you need to select the LUN. Figure 4-64 shows this window. This window shows all discovered LUNs that have not been assigned to any CTC yet. In our setup, we only see the new LUN that we have just created.

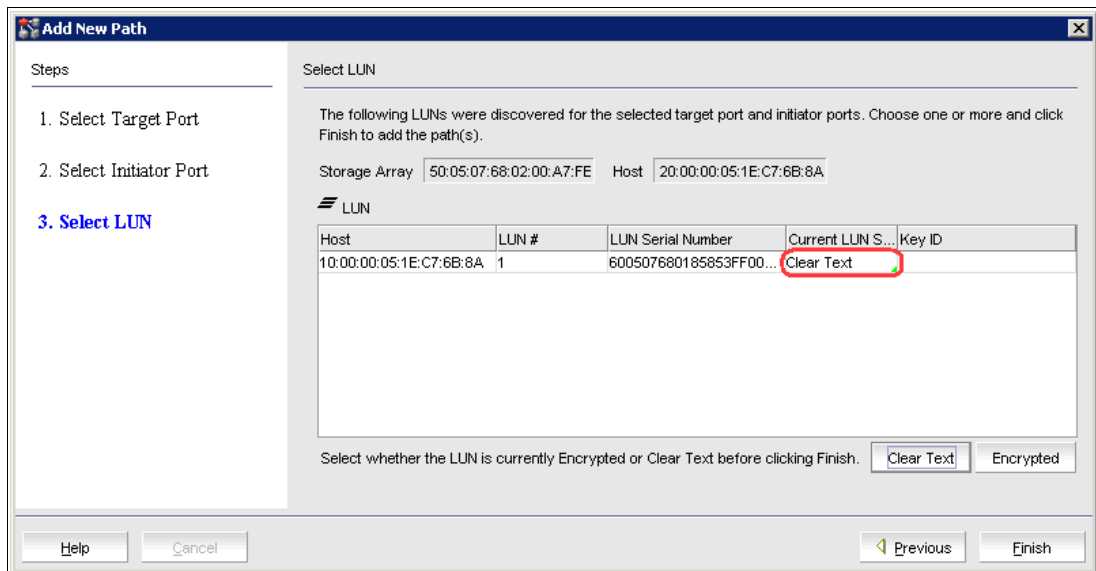


Figure 4-64 DCFM Encryption Center wizard to add a new LUN: Select LUN

Verifying the LUN: In our example, which is shown in Figure 4-64, under the LUN Serial Number heading, you also see the LUN UID (vdisk_UID in the terminology of an IBM Storwize V7000), which the disk storage subsystem generates and provides. The last digit of this number represents the real LUN number of the LUN. Unfortunately, you cannot see the real LUN number in this window, by default, because the column in this wizard's window is not wide enough to show it. However, you can adjust the view to display the last digit to confirm that it is the correct LUN that you want to add (see Figure 4-65 on page 149).

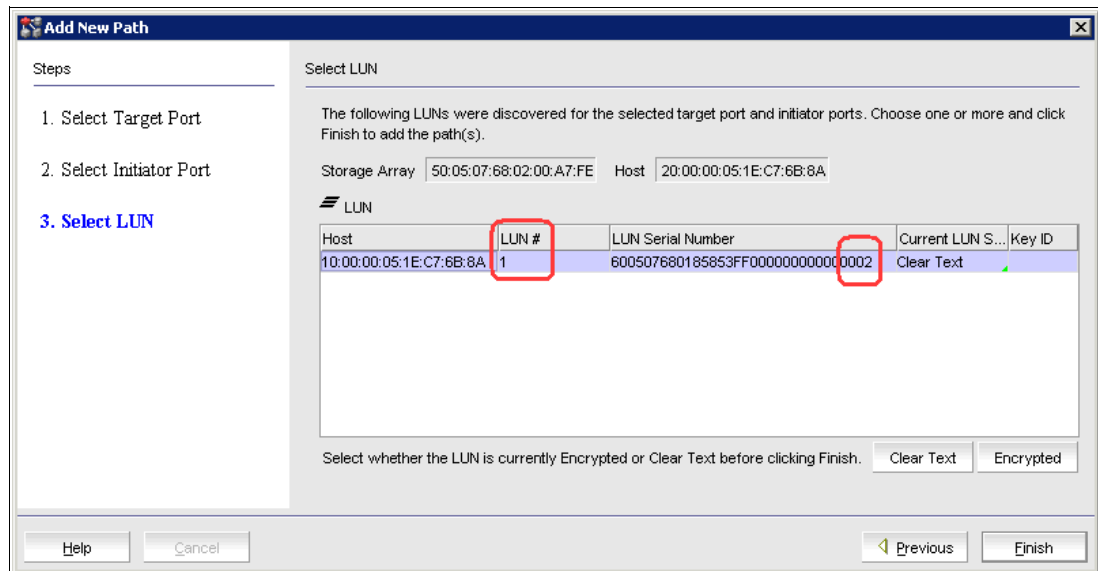


Figure 4-65 DCFM Encryption Center wizard to add a new LUN: Expanded LUN S/N view

EE-assigned LUN number: In our example, which is shown in Figure 4-65, under the LUN # heading, you see a LUN number that the EE has assigned. This number is not the actual physical LUN number that is created and assigned by the disk subsystem, as you can see. From the EE's point of view, this entry under LUN # reflects the appropriate SCSI ID (in decimal) that was defined during the creation of the new LUN in the disk storage subsystem. All configuration and definitions within the EEs will rely on this number.

You can only see the physical LUN number that was created and assigned by the disk storage subsystem at the end of the string under the heading LUN Serial Number (UID). This physical LUN number has no further usage within the EE setup tasks. You only need to pay attention to this LUN Serial Number (UID) to identify the LUN with which you want to work. If you want to learn more about this topic, see Chapter 5, "Advanced techniques and functionality" on page 171.

Important: The LUN # that is used by the EEs is displayed in the DCFM in decimal. If working with the CLI, this number is displayed, by default, as hexadecimal. When you enter a CLI command, you can choose whether to specify the decimal value or the hexadecimal value (for example, 0xFA translates to 250). Be aware that certain operating systems (for example, Microsoft Windows Server 2008) currently only support SCSI IDs up to 254, although the IBM Storwize V7000 disk subsystem can assign SCSI IDs up to 2047.

Before you finish the wizard to add the new LUN to the selected CTC, ensure that the entry under Current LUN State is Clear Text (as shown in Figure 4-64 on page 148). A newly created LUN will not contain encrypted data at this point in time, because it has just been created and logically is empty.

After you click **Finish** in the wizard, you see a final information box. This box informs you that the definition for this new LUN (path) has been created but will not be committed until you select **OK** or **Apply** on the Disk LUNs View window (see Figure 4-66 on page 150).

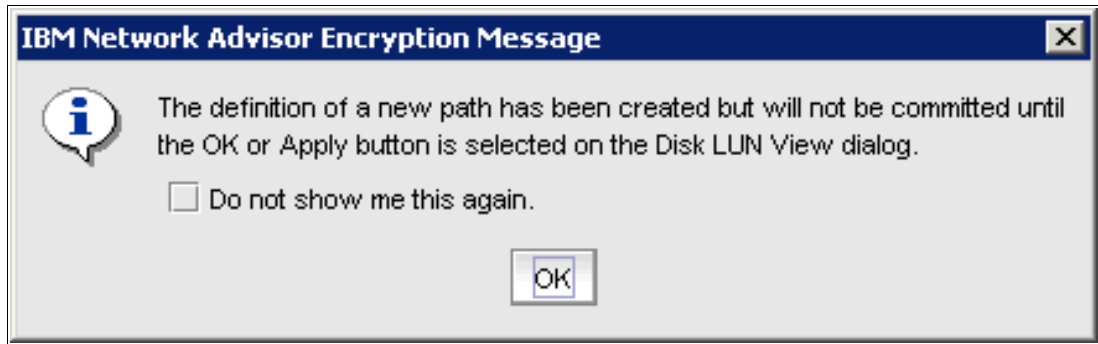


Figure 4-66 DCFM Encryption Center wizard to add a new LUN: Final user information

After you click **OK** on the confirmation message that is shown in Figure 4-66, you return to the Disk LUN View window. There, you can see the definition, which we just created to associate the new LUN with a CTC (CTC_1 in Figure 4-67). The marker, a green circle with a + in it, to the left of our new LUN shows that the LUN is there but not yet committed.

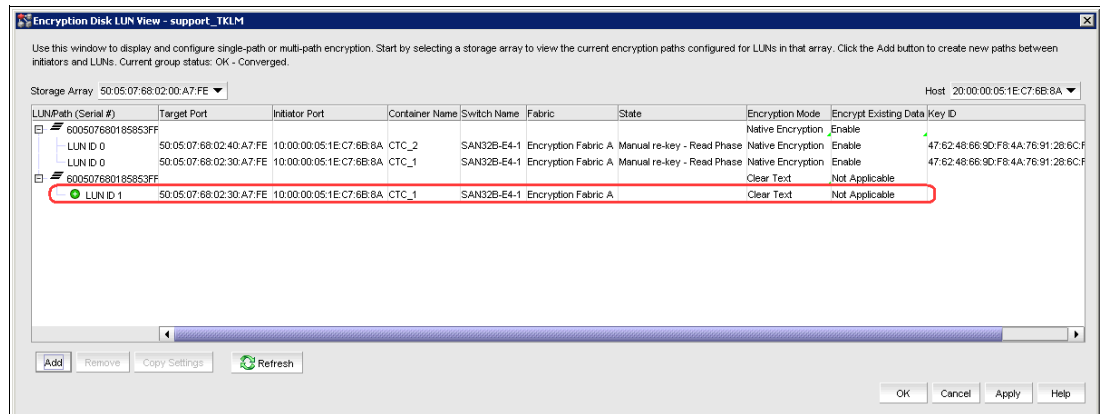


Figure 4-67 DCFM Encryption Center Disk LUN View window after adding the new LUN definition

Continue by clicking **Apply** to activate this definition in CTC_1. The green marker disappears (see Figure 4-68 on page 151 where we have already committed the second CTC, CTC_2). The path of CTC_1 is now provided to the host server. You will get a confirmation box that you must acknowledge.

At this point, you are able to scan and discover the LUN from a host server (the initiator). Nevertheless, you must continue at this point to add the new LUN to *all* the other CTCs (disk subsystem target ports) that are present and that present the newly created LUN to the SAN.

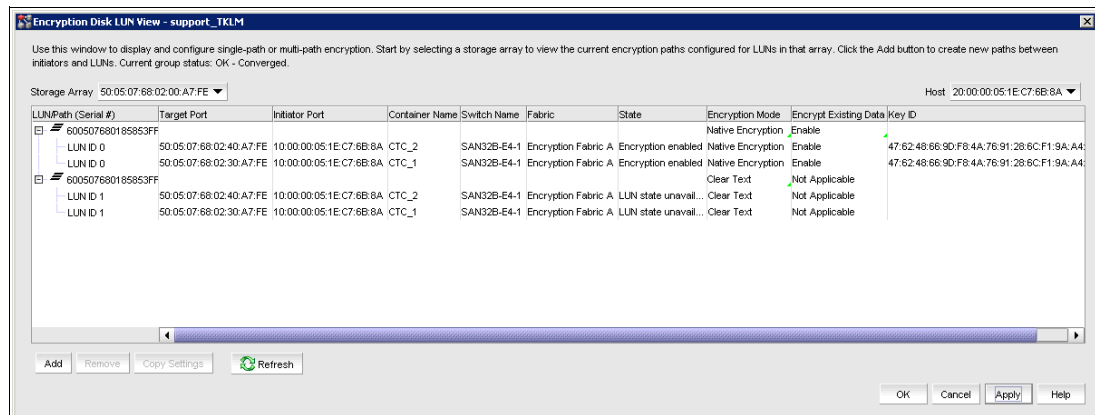


Figure 4-68 DCFM Encryption Center Disk LUN View window after committing the new LUN definition

Preferred practice: As shown in Figure 4-67 on page 150, the column for Encryption Mode still shows Clear Text for our new CTC definitions. This value is acceptable, because, before we finally encrypt the newly created LUN, all CTC definitions must be in place. Be aware that it will take several minutes for you to add the LUN to all the existing CTC definitions. During this time, the newly created LUN is presented natively to the SAN, because the redirection zones are not in place for all paths yet. To prevent, by a mistake, another host or user scanning or accessing the new LUN while we create the definitions, we keep the LUN unencrypted until all CTCs are updated and all redirection zones are in place.

Finally, after all CTCs have been updated with the new LUN information, we set the LUN to the encryption state. We use the Disk LUN View window of the DCFM Encryption Center. You also must perform this step for *every* CTC definition.

As shown in Figure 4-69, for our CTC_3 and CTC_4 definition, we need to change the setting in the column Encryption Mode from Clear Text to **Native Encryption** for the newly created LUN.

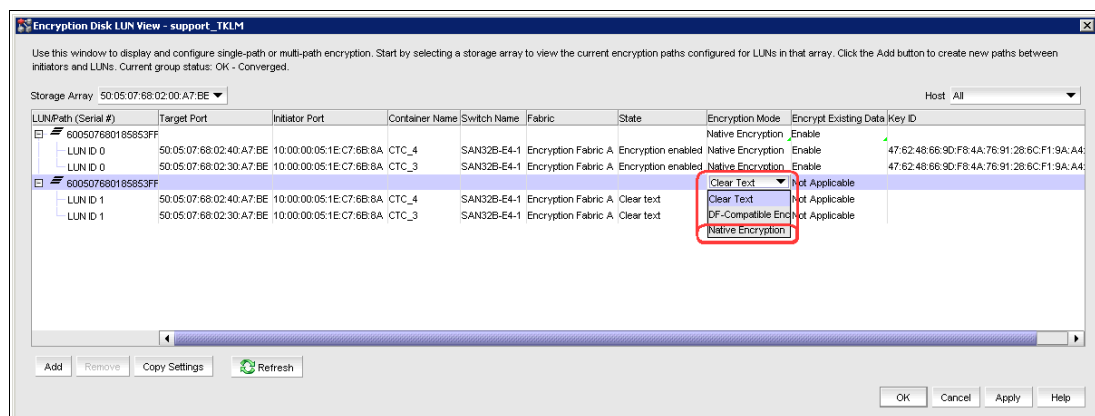


Figure 4-69 DCFM Encryption Center Disk LUN View window to change the Encryption Mode

After you confirm by clicking **Apply**, using the Storage Array upper-left pull-down, select the other target ports to perform the same update for all other CTCs, if more CTCs exist. In our example, we had four CTCs in place, so we also set CTC_1 and CTC_2 to **Native**

Encryption mode. Figure 4-70 shows the status after the final commit. You now see Enabled in the Encrypt Existing Data column.

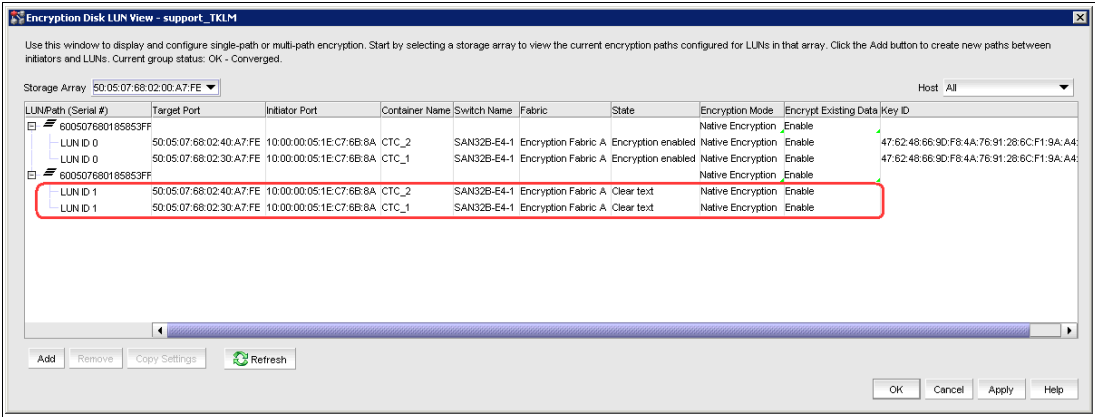


Figure 4-70 DCFM Encryption Center Disk LUN View window after changing the Encryption Mode

We have added a new LUN (in our example, LUN 2) to an existing EG successfully. Figure 4-71 shows the final configuration setup that we created in this section. You can see that LUN 2 is now in the EG setup.

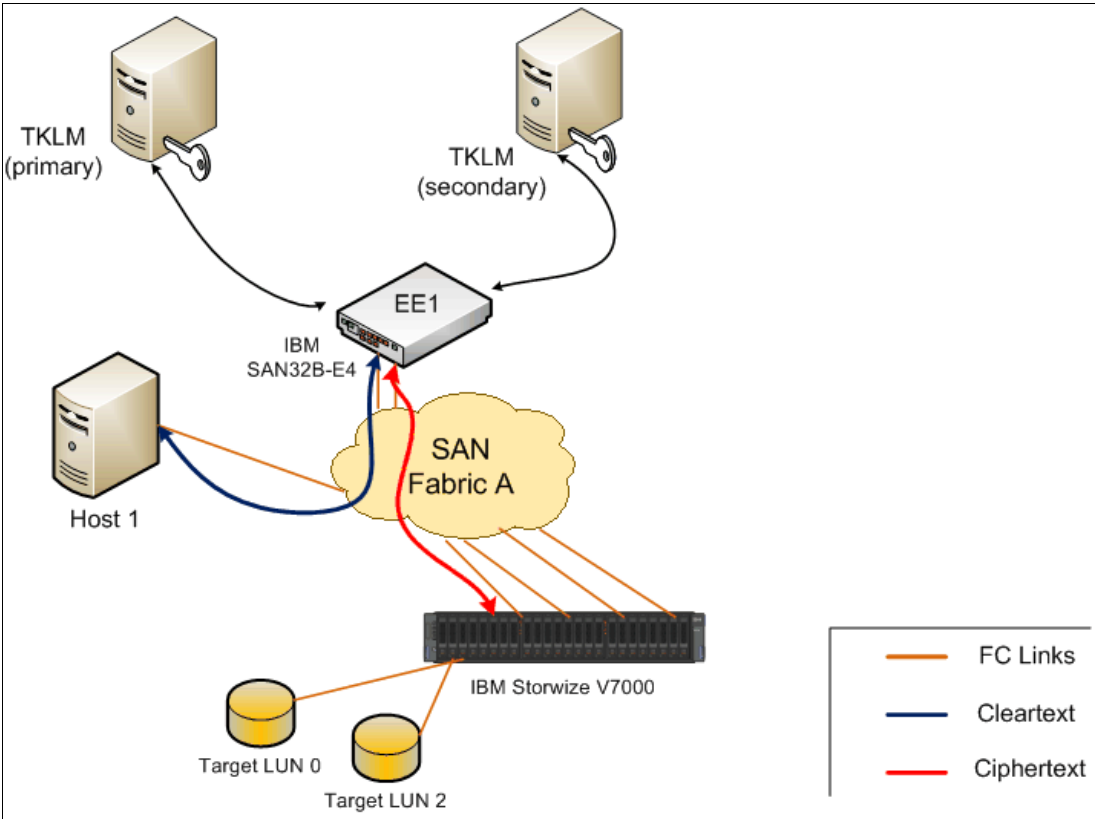


Figure 4-71 Target configuration for creating and adding a new LUN (LUN 2) to the EG

4.6.4 Creating and adding a new LUN in a dual-fabric environment

In this section, we provide additional information for you to consider when creating and adding a new LUN (volume) in a dual-fabric EE environment. A dual-fabric environment means that you have more than one SAN fabric to connect your host servers to the storage subsystems. This environment gives you at least two physically separated paths. Nevertheless, you still can have more paths within each fabric to the storage, which also will give you also additional paths. The overall maximum number of paths from a host server to a storage device also relates to the number of SAN ports that the host and the storage subsystem provide.

Whenever you plan on having more than one path involved in a SAN32B-E4 Encryption Switch setup, you need a DEK cluster configuration. To keep keys and configuration in synchronization on each path, both EEs connect together via an internal 1 Gbps Ethernet connection. As explained in Chapter 2, “Terminology and technology” on page 13, this kind of configuration is called a *DEK cluster*.

HA cluster: Do not confuse this issue with an HA cluster configuration. In a DEK cluster environment, both EEs operate independently in *separate* paths or fabrics and provide their service in their fabric and paths with encryption processing. If an encryption node fails, this specific path (to the second fabric) will fail.

In an HA cluster configuration, the HA encryption node is always located in the *same* path or fabric to provide HA functionality if the operational encryption node in the *same* path or fabric fails.

Figure 4-72 on page 154 shows you a typical DEK cluster layout. We have an IBM Storwize V7000 in place, which provides four physical paths (targets) to each fabric. Host 1 maintains only one path in each fabric. The key vaults (IBM Tivoli Key Lifecycle Manager) only have IP connection to the encryption nodes; they do not need access to the SAN, because they only store and serve the keys.

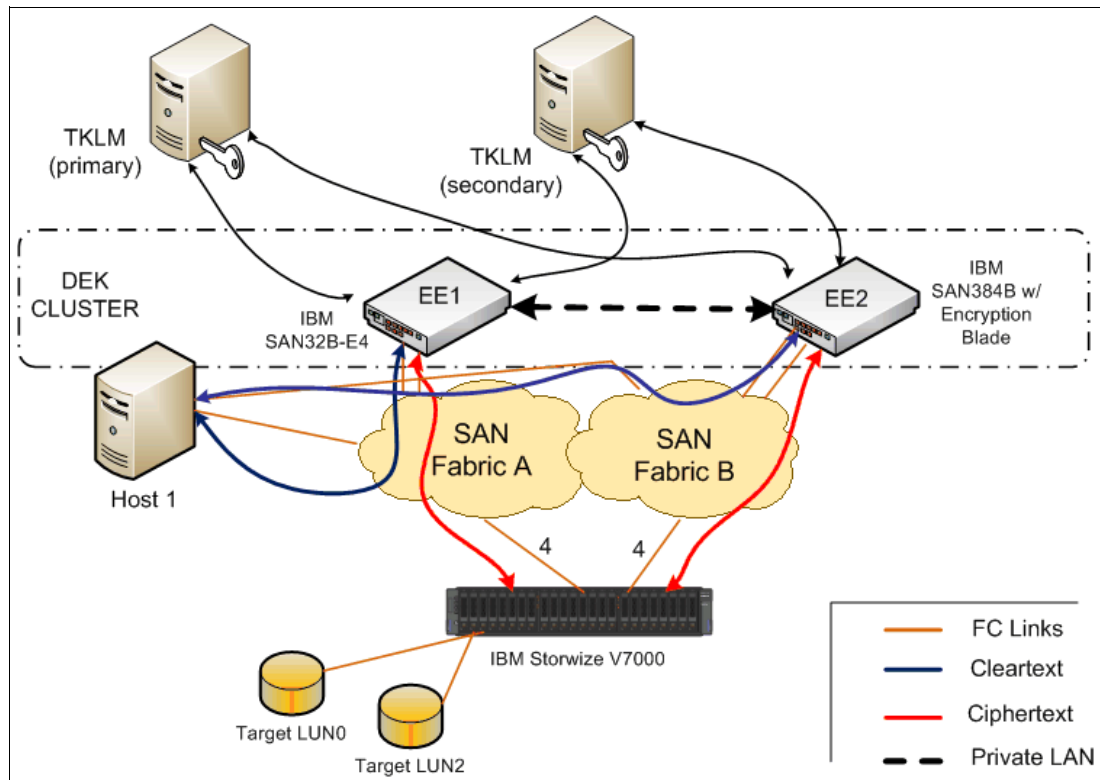


Figure 4-72 A typical DEK cluster IBM Encryption Switch layout

Refer also to 4.4, “Data encryption key cluster config: Multiple path/dual fabric” on page 110, where we have set up such an environment step-by-step.

In 4.6.1, “Creating and adding a new LUN in a single-fabric environment” on page 137, we described how to set up a newly added LUN in a single-fabric environment. We pointed out that we need to make sure that each CTC, which contains an initiator definition (host port) for the new LUN, must be modified and updated to handle the new LUN down to the target ports. We have a CTC definition in place for each target port on the storage subsystem, because the CTCs are target-centric.

We summarize the steps to create and add a new LUN in a single-fabric environment:

1. Create a new LUN in the disk storage subsystem.
2. Follow the Create LUN/Add path wizard in the DCFM Encryption Center window:
 - Select the target.
 - Select the initiator.
 - Select the LUN.

Repeat this sequence of steps for each CTC definition that you have in place.

We have two main tasks to accomplish. We first need to create the new LUN (volume) in the disk storage subsystem. Secondly, we will configure the EG members in a way that the new LUN is presented to the associated host servers and the data paths both up and down (from host to storage subsystem) and will be encrypted via the redirection of the zones through the encryption nodes. This basic sequence of tasks, in general, also applies for a dual-fabric layout.

4.6.5 Creating a new LUN in the IBM Storwize V7000 (dual fabric)

This task is identical to the task that was described in 4.6.1, “Creating and adding a new LUN in a single-fabric environment” on page 137 and in the section 4.6.2, “Creating a new LUN in the IBM Storwize V7000” on page 138.

4.6.6 Adding a new LUN into an existing encryption group (dual fabric)

The steps of this task are also identical to the steps that we described in 4.6.1, “Creating and adding a new LUN in a single-fabric environment” on page 137 and in the section 4.6.3, “Adding a newly created LUN into an existing encryption group” on page 143.

The only major, but important, difference is that you need to add the new LUN to *all* the CTC definitions of the second encryption node (the DEK cluster node) within the second path/second fabric. Therefore, you have to repeat the “Add new path” wizard in the DCFM Encryption Center view to manage Disk LUNs (typically twice as much) to add the new LUN to all the CTCs in the other encryption node (see Figure 4-60 on page 146).

Ensure that you select the **Disk LUN** view in the Group view in the DCFM Encryption Center window. In that case, it does not matter whether you have selected the group leader or the member node (refer to Figure 4-59 on page 145), because you selected the entire EG to add a new LUN.

Preferred practice: You might want to name the CTC definition in the group leader CTC_1A, CTC_2A, and so on. Then, name the CTC definition in the other encryption node (second path, DEK cluster) CTC_1B, CTC_2B, and so on.

4.7 Adding and removing a path to an initiator from a CTC

In this section, we add extra paths to an already existing and encrypted LUN for a specific host. This action might become a requirement as server load increases and extra bandwidth is required. After the paths are added and operating, we will then remove them again without causing disruption. Path removal might be required to reduce an overly configured environment that results from an insufficiently planned initial setup.

Important: It is critical to perform the steps in order as outlined in this section. By not following these procedures in sequence, you will corrupt the data on the target LUN.

4.7.1 Adding host paths

In Example 4-31, we display the current CTC_1 definitions. One host initiator is defined using WWPN 10:00:00:05:1e:c7:6b:8a (host#1) with two target LUNs.

Example 4-31 Displaying CTC_1 definitions with a single initiator

```
SAN32B-E4-1:admin> cryptocfg --show -container CTC_1 -stat
```

```
Container name:    CTC_1
Type:             disk
EE node:          10:00:00:05:1e:54:17:10
EE slot:          0
```

```

EE hosting container:  current
Target:              50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:         030400
VT:                 20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
VT PID:             012001
Number of host(s):  1
Number of rekey session(s): 0
Host:               10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Host PID:           030900
VI:                 20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:             012401
Number of LUN(s):   2
LUN number:         0x0
LUN type:           disk
LUN serial number:  600507680185853FF000000000000000
Encryption mode:    encrypt
Encryption format:  native
Encrypt existing data: disabled
Rekey:              enabled
Internal EE LUN state: Read only (Current key is in read only state)
Encryption algorithm: AES256-XTS
Key ID state:       Read only
New LUN:            No
Key ID:             47:62:48:66:9d:f8:4a:76:91:28:6c:f1:9a:a4:cb:e2
Key creation time:  Thu Jul 28 02:50:32 2011
Key life:           1 (days) 0 (minutes)
Time remaining to Auto Rekey: 3 (hours) 0 (minutes)
LUN number:         0x1
LUN type:           disk
LUN serial number:  600507680185853FF0000000000000002
Encryption mode:    encrypt
Encryption format:  native
Encrypt existing data: disabled
Rekey:              disabled
Internal EE LUN state: Encryption enabled
Encryption algorithm: AES256-XTS
Key ID state:       Read write
New LUN:            No
Key ID:             a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36
Key creation time:  Wed Jul 27 07:48:54 2011
Operation succeeded.
SAN32B-E4-1:admin>

```

In Example 4-32, we display the current CTC_2 definitions showing a separate host with WWPN 10:00:00:05:1e:c7:6b:a2 with one encrypted target LUN (host#2). We want to add extra paths to host#2. Because our CTC_1 and CTC_3 have defined paths to host#1, and CTC_2 and CTC_4 for host#2, we will add host#2 to CTC_1 and CTC_3.

Example 4-32 Displaying CTC_2 definitions with a single initiator

```

SAN32B-E4-1:admin> cryptocfg --show -container CTC_2 -stat

Container name:    CTC_2
Type:             disk
EE node:          10:00:00:05:1e:54:17:10
EE slot:          0

```

```

EE hosting container:  current
Target:              50:05:07:68:02:40:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:          030500
VT:                  20:02:00:05:1e:54:17:0c 20:03:00:05:1e:54:17:0c
VT PID:              012801
Number of host(s):    1
Number of rekey session(s): 0
Host:                10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Host PID:             030800
VI:                   20:0a:00:05:1e:54:17:0c 20:0b:00:05:1e:54:17:0c
VI PID:               012d01
Number of LUN(s):     1
LUN number:           0x0
LUN type:              disk
LUN serial number:    600507680185853FF000000000000005
Encryption mode:      encrypt
Encryption format:    native
Encrypt existing data: disabled
Rekey:                 disabled
Internal EE LUN state: Encryption enabled
Encryption algorithm: AES256-XTS
Key ID state:          Read write
New LUN:               No
Key ID:                a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36
Key creation time:    Wed Jul 27 07:48:54 2011
Operation succeeded.
SAN32B-E4-1:admin>

```

By viewing the **datapath query** output in Example 4-33 from host#2, we confirm that there are currently only two paths to the target LUN for this initiator.

Example 4-33 Displaying host#2 datapath output

```

C:\Program Files\IBM\SDDDSM>datapath query device

Total Devices : 1

DEV#: 0  DEVICE NAME: Disk1 Part0  TYPE: 2145          POLICY: OPTIMIZED
SERIAL: 600507680185853FF0000000000000005
=====
Path#      Adapter/Hard Disk      State  Mode      Select  Errors
  0         Scsi Port8 Bus0/Disk1 Part0  OPEN   NORMAL      0        0
  1         Scsi Port8 Bus0/Disk1 Part0  OPEN   NORMAL    260        0

C:\Program Files\IBM\SDDDSM>

```

We define the HBA WWPNN of our host#2 initiator to CTC_1 and CTC_3, as shown in Example 4-34.

Example 4-34 Adding initiator to CTCs

```

SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_1 10:00:00:05:1e:c7:6b:a2
20:00:00:05:1e:c7:6b:a2
Operation succeeded.

```

```
SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_3 10:00:00:05:1e:c7:6b:a2
20:00:00:05:1e:c7:6b:a2
Operation succeeded.
SAN32B-E4-1:admin>
```

The next step is to add the target LUN to the initiator in CTC_1 and CTC_3 that we just defined. From the previous **show container** output for CTC_2 in Example 4-32 on page 156, we know that the LUN number for the target LUN to our host#2 is 0x0. Therefore, we do not have to commit the changes thus far and discover LUNs; instead, we can just add LUN 0x0 to our host initiator, as shown in Example 4-35.

Example 4-35 Adding target LUN to the new initiator

```
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_1 0x0 10:00:00:05:1e:c7:6b:82
20:00:00:05:1e:c7:6b:82
Operation succeeded.
SAN32B-E4-1:admin>
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_3 0x0 10:00:00:05:1e:c7:6b:82
20:00:00:05:1e:c7:6b:82
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>
```

After committing the changes to our CTCs, we look at the defined zone configuration by using the **zoneshow** command. We see the new redirect zones defining virtual initiators and virtual targets for the changes that we have just committed. In Example 4-36, we show one of these automatically created redirect zones.

Example 4-36 Automatically created redirect zone

```
zone: red_1109_support_TKLM500507680230a7fe200a00051e54170c
      10:00:00:05:1e:c7:6b:a2; 50:05:07:68:02:30:a7:fe;
      20:0a:00:05:1e:54:17:0c; 20:00:00:05:1e:54:17:0c
```

At this point, the paths are not available for the server to use yet, because we have no actual zone in the active zone configuration to provide connectivity. We have created the redirect zone first, because if we modify the normal zone by adding the new target WWPNs as the first step, we give the server direct access to the encrypted target LUN. Therefore, any read will be unreadable, and any write will corrupt the volume.

In Example 4-37, we add the two Storwize V7000 target ports, which correspond to CTC_1 and CTC_3, to the host#2 zone. We then save and activate these zone changes.

Example 4-37 Modifying the existing zone using CLI

```
SAN32B-E4-1:admin> zoneadd "W2k8_2_to_V7000_1", "50:05:07:68:02:30:a7:fe"
SAN32B-E4-1:admin> zoneadd "W2k8_2_to_V7000_1", "50:05:07:68:02:30:a7:be"
SAN32B-E4-1:admin> cfsave
```

You are about to save the Defined zoning configuration. This action will only save the changes on Defined configuration. Any changes made on the Effective configuration will not take effect until it is re-enabled. Until the Effective configuration is re-enabled, merging new switches into the fabric is not recommended and may cause unpredictable results with the potential of mismatched Effective Zoning

```

configurations.
Do you want to save Defined zoning configuration only? (yes, y, no, n): [no] y
Updating flash ...
SAN32B-E4-1:admin> cfgenable "ITS0_SG247977"
You are about to enable a new zoning configuration.
This action will replace the old zoning configuration with the
current configuration selected. If the update includes changes
to one or more traffic isolation zones, the update may result in
localized disruption to traffic on ports associated with
the traffic isolation zone changes
Do you want to enable 'ITS0_SG247977' configuration (yes, y, no, n): [no] y
zone config "ITS0_SG247977" is in effect
Updating flash ...
SAN32B-E4-1:admin>

```

After activating the zone change, we can see that our host#2 now has four target ports in the effective zone configuration. Example 4-38 shows only the Effective configuration section from a **cfgshow** command.

Example 4-38 Updated zone configuration

```

Effective configuration:
cfg:  ITS0_SG247977
zone:  W2k8_1_to_V7000_1
      50:05:07:68:02:30:a7:be
      50:05:07:68:02:30:a7:fe
      10:00:00:05:1e:c7:6b:8a
zone:  W2k8_2_to_V7000_1
      10:00:00:05:1e:c7:6b:a2
      50:05:07:68:02:40:a7:fe
      50:05:07:68:02:40:a7:be
      50:05:07:68:02:30:a7:fe
      50:05:07:68:02:30:a7:be

```

By issuing the **datapath query** on our host#2, we display four paths to our LUN, as shown in Example 4-39.

Example 4-39 Displaying the additional paths on the host

```

C:\Program Files\IBM\SDDDSM>datapath query device

Total Devices : 1

DEV#:  0  DEVICE NAME: Disk1 Part0  TYPE: 2145          POLICY: OPTIMIZED
SERIAL: 600507680185853FF0000000000000005
=====
Path#      Adapter/Hard Disk      State  Mode      Select      Errors
  0      Scsi Port8 Bus0/Disk1 Part0  OPEN   NORMAL         0         0
  1      Scsi Port8 Bus0/Disk1 Part0  OPEN   NORMAL      1444         0
  2      Scsi Port8 Bus0/Disk1 Part0  OPEN   NORMAL         0         0
  3      Scsi Port8 Bus0/Disk1 Part0  OPEN   NORMAL         0         0

C:\Program Files\IBM\SDDDSM>

```

In Example 4-40, we have piped the output of the **show container** command to **grep** so that we only show specific details about our CTCs. By using this filtered output, we easily can see the number of hosts defined in a CTC and how many LUNs that host can access via that CTC.

For our host#2 (wwpn 10:00:00:05:1e:c7:6b:a2), we see it listed under CTC_1 and CTC_3, as well as CTC_2 and CTC_4 to which it was defined originally.

Example 4-40 Displaying filtered container status

```

SAN32B-E4-1:admin> cryptocfg --show -container -all -stat| grep 'LUN number\|LUN
serial number\|Host:\|Container name\|Number of host\|Number of LUN'
Container name:          CTC_1
Number of host(s):      2
Host:                   10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Number of LUN(s):       2
LUN number:             0x0
LUN serial number:      600507680185853FF0000000000000000
LUN number:             0x1
LUN serial number:      600507680185853FF0000000000000002
Host:                   10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):       1
LUN number:             0x0
LUN serial number:      600507680185853FF0000000000000005
Container name:          CTC_2
Number of host(s):      1
Host:                   10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):       1
LUN number:             0x0
LUN serial number:      600507680185853FF0000000000000005
Container name:          CTC_3
Number of host(s):      2
Host:                   10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Number of LUN(s):       2
LUN number:             0x0
LUN serial number:      600507680185853FF0000000000000000
LUN number:             0x1
LUN serial number:      600507680185853FF0000000000000002
Host:                   10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):       1
LUN number:             0x0
LUN serial number:      600507680185853FF0000000000000005
Container name:          CTC_4
Number of host(s):      1
Host:                   10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):       1
LUN number:             0x0
LUN serial number:      600507680185853FF0000000000000005
SAN32B-E4-1:admin>

```

4.7.2 Removing host paths

Now that we have four paths to our target LUN for our host through one fabric, we will remove the paths that we have just added in the previous steps. The order of performing these steps is extremely important to avoid corrupting the data on the target LUN.

The paths that we will remove are in CTC_1 and CTC_3, leaving two paths to the LUN via CTC_2 and CTC_4. First, we modify the zone definition for our host zone by removing the two target WWPNs that are associated to CTC_1 and CTC_3, as shown in Example 4-41.

Example 4-41 Removing target ports from a host zone

```

SAN32B-E4-1:admin> zoneremove "W2k8_2_to_V7000_1", "50:05:07:68:02:30:a7:fe;
50:05:07:68:02:30:a7:be"
SAN32B-E4-1:admin> cfgsave
You are about to save the Defined zoning configuration. This
action will only save the changes on Defined configuration.
Any changes made on the Effective configuration will not
take effect until it is re-enabled. Until the Effective
configuration is re-enabled, merging new switches into the
fabric is not recommended and may cause unpredictable
results with the potential of mismatched Effective Zoning
configurations.
Do you want to save Defined zoning configuration only? (yes, y, no, n): [no] y
Updating flash ...
SAN32B-E4-1:admin> cfgenable "ITS0_SG247977"
You are about to enable a new zoning configuration.
This action will replace the old zoning configuration with the
current configuration selected. If the update includes changes
to one or more traffic isolation zones, the update may result in
localized disruption to traffic on ports associated with
the traffic isolation zone changes
Do you want to enable 'ITS0_SG247977' configuration (yes, y, no, n): [no] y
zone config "ITS0_SG247977" is in effect
Updating flash ...
SAN32B-E4-1:admin>

```

After the zone modification is saved and enabled in the fabric, we immediately notice that the host only has two paths to the target LUN, as shown in Example 4-42.

Example 4-42 Displaying paths to a LUN on the host

```

C:\Program Files\IBM\SDDDSM>datapath query device

Total Devices : 1

DEV#: 0  DEVICE NAME: Disk1 Part0  TYPE: 2145          POLICY: OPTIMIZED
SERIAL: 600507680185853FF0000000000000005
=====
Path#      Adapter/Hard Disk      State  Mode      Select  Errors
  0        Scsi Port8 Bus0/Disk1 Part0  OPEN   NORMAL      0        0
  1        Scsi Port8 Bus0/Disk1 Part0  OPEN   NORMAL    7773        0

```

```

C:\Program Files\IBM\SDDDSM>

```

While our goal of reducing the host paths to the target LUN appears to have been achieved from the output in Example 4-42, our job is not yet complete. We must also remove the LUN and initiator definitions in each CTC. We prove these CTC definitions still exist by using our *filtered show container* command, as shown in Example 4-43 on page 162. Note that CTC_1 and CTC_3 still have two hosts defined, one of which is host#2.

Example 4-43 Displaying the filtered current CTC status

```
SAN32B-E4-1:admin> cryptocfg --show -container -all -stat | grep 'LUN number\|LUN
serial number\|Host:\|Container name\|Number of host\|Number of LUN'
Container name:      CTC_1
Number of host(s):  2
Host:               10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Number of LUN(s):   2
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000000
LUN number:         0x1
LUN serial number:  600507680185853FF0000000000000002
Host:               10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):   1
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000005
Container name:      CTC_2
Number of host(s):  1
Host:               10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):   1
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000005
Container name:      CTC_3
Number of host(s):  2
Host:               10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Number of LUN(s):   2
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000000
LUN number:         0x1
LUN serial number:  600507680185853FF0000000000000002
Host:               10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):   1
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000005
Container name:      CTC_4
Number of host(s):  1
Host:               10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):   1
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000005
SAN32B-E4-1:admin>
```

We remove the LUN that is associated to host#2 and remove the host#2 definitions from both CTC_1 and CTC_3, as shown in Example 4-44. Committing the change removes the associated redirect zones from the defined zone configuration.

Example 4-44 Removing LUN and initiator from CTCs

```
SAN32B-E4-1:admin> cryptocfg --remove -LUN CTC_1 0x0 10:00:00:05:1e:c7:6b:a2
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --remove -initiator CTC_1 10:00:00:05:1e:c7:6b:a2
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --remove -LUN CTC_3 0x0 10:00:00:05:1e:c7:6b:a2
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --remove -initiator CTC_3 10:00:00:05:1e:c7:6b:a2
Operation succeeded.
```



```
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>
```

Using our filtered **show container** command, we note that CTC_1 and CTC_3 only have one defined host initiator, and that host is not the WWPN/WWNN of host#2, as shown in Example 4-45.

Example 4-45 Displaying the cleaned up CTCs with a filtered command

```
SAN32B-E4-1:admin> cryptocfg --show -container -all -stat | grep 'LUN number\|LUN
serial number\|Host:\|Container name\|Number of host\|Number of LUN'
Container name:      CTC_1
Number of host(s):  1
Host:               10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Number of LUN(s):   2
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000000
LUN number:         0x1
LUN serial number:  600507680185853FF0000000000000002
Container name:      CTC_2
Number of host(s):  1
Host:               10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):   1
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000005
Container name:      CTC_3
Number of host(s):  1
Host:               10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Number of LUN(s):   2
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000000
LUN number:         0x1
LUN serial number:  600507680185853FF0000000000000002
Container name:      CTC_4
Number of host(s):  1
Host:               10:00:00:05:1e:c7:6b:a2 20:00:00:05:1e:c7:6b:a2
Number of LUN(s):   1
LUN number:         0x0
LUN serial number:  600507680185853FF0000000000000005
SAN32B-E4-1:admin>
```

With our LUN and host initiator ports removed from the relevant CTCs, our path removal is complete. The path removal did not affect any host.

4.8 Changing a host HBA

Occasionally, we need to replace an HBA in a host either due to a failure or an upgrade to a newer model. Either way, this relatively simple task requires a good understanding of the encryption environment so that a simple HBA replacement does not result in the data corruption of the host's LUNs.

In 4.2, “General prerequisites for encryption” on page 95, we mentioned that a prerequisite to implementing encryption using the SAN32B-E4 Encryption Switch is that WWPN zoning must be used. Of course, when we replace an HBA, we also change the WWPN/WWNN that is associated with that HBA and its ports. Without encryption in the configuration, we will be required to update the host zone by replacing the old HBA WWPN with the new HBA WWPN. However, if we replace the old HBA WWPN with the new HBA WWPN as a first step in an encryption environment, we create a direct path to the target LUN, which means that any data that is read from the LUN via this path will be unreadable. And, any data that is written via this path to the LUN will corrupt the target LUN, because we are bypassing the EE.

As described in 4.7, “Adding and removing a path to an initiator from a CTC” on page 155, the order of performing these steps is critical when adding or removing paths to target LUNs. Think of replacing an HBA in a host as both removing and adding paths, because the same steps of removing and then adding the HBA WWPN/WWNN to the CTCs need to be performed.

Our recommendation is to persistently disable (**portcfgpersistentdisable**) the switch port to which the host HBA being replaced attaches until all the definition changes are complete. By persistently disabling the port, we are also safe in the event that the switch might reboot during our change. Although this reboot is extremely unlikely, if it occurs and we had only disabled the port (**portdisable**), we will corrupt the target LUNs.

In Example 4-46, we show the **switchshow** output from the fabric switch. The output is truncated to show our host that is attached to port 9 persistently disabled after issuing **portcfgpersistentdisable 9** to which host#1 is attached.

Example 4-46 Displaying persistently disabled port

```
IBM_2498_R06:admin> switchshow
switchName:      IBM_2498_R06
.
.
.
Index Port Address Media Speed State      Proto
=====
```

0	0	030000	id	N8	Online	FC	E-Port	(Trunk port, master is
Port 1)								
1	1	030100	id	N8	Online	FC	E-Port	10:00:00:05:1e:54:17:10
"SAN32B-E4-1" (downstream)(Trunk master)								
2	2	030200	id	N8	No_Light	FC		
3	3	030300	id	N8	No_Light	FC		
4	4	030400	id	N8	Online	FC	F-Port	50:05:07:68:02:30:a7:fe
5	5	030500	id	N8	Online	FC	F-Port	50:05:07:68:02:40:a7:fe
6	6	030600	id	N8	Online	FC	F-Port	50:05:07:68:02:30:a7:be
7	7	030700	id	N8	Online	FC	F-Port	50:05:07:68:02:40:a7:be
8	8	030800	id	N8	Online	FC	F-Port	10:00:00:05:1e:c7:6b:a2
9	9	030900	id	N8	No_Light	FC	Disabled	(Persistent)
10	10	030a00	--	N8	No_Module	FC		
. . .								
15	15	030f00	id	N8	No_Light	FC		

After the port is disabled, we are safe to proceed with physically replacing the HBA in the server, followed by modifying the CTC and zone definitions.

Tip: To modify the zone and CTC definitions, we need the WWPN and WWNN of the new HBA. If you do not have an HBA utility on your host to display this detail or it is not printed on the card anywhere, we temporarily re-enable the switch port. Re-enabling the switch port allows the HBA to log in to the switch so that we can see the details in the **switchshow** and **nscamshow** output, as explained in 4.1.1, “Current configuration” on page 90. After noting the WWPN and WWNN, we persistently disable the port again.

At this point, we assume that the old HBA with WWPN 10:00:00:05:1e:c7:6b:8a has been physically replaced. We have noted the new HBA WWPN of 10:00:00:05:1e:a7:ab:aa with WWNN 20:00:00:05:1e:a7:ab:aa details.

The next step is to modify CTC_1 and CTC_3 with the new WWPN/WWNN. In Example 4-47, we begin by adding the new initiator.

Example 4-47 Adding the new initiator HBA to the container

```
SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_1 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_3 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin>
```

We have to define the LUNs to these new initiators in each CTC. In Example 4-48, we add the LUNs 0x0 and 0x1 exactly as they are defined for the previous HBA, as seen in the Example 4-43 on page 162 output.

Example 4-48 Defining the LUNs to the new HBA initiator

```
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_1 0x0 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_1 0x1 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_3 0x0 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_3 0x1 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin>
```

Next, we remove the old HBA from the containers, ensuring that we remove the LUN definitions, as well. In Example 4-49, we show the removal of both LUN 0x0 and 0x1 from the old HBA initiator, then remove that HBA definition from each CTC, and commit the changes.

Example 4-49 Removing old HBA LUNs and initiator definitions

```
SAN32B-E4-1:admin> cryptocfg --remove -LUN CTC_1 0x0 10:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --remove -LUN CTC_1 0x1 10:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --remove -initiator CTC_1 10:00:00:05:1e:c7:6b:8a
```

```

Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --remove -LUN CTC_3 0x0 10:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --remove -LUN CTC_3 0x1 10:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --remove -initiator CTC_3 10:00:00:05:1e:c7:6b:8a
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>

```

In Example 4-50, we define the new HBA WWPN and WWNN for our initiator to both CTC_1 and CTC_3. We also add the LUNs as they were previously defined to the old HBA, committing our changes again to complete the container changes.

Example 4-50 Adding the new HBA to the CTCs

```

SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_1 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -initiator CTC_3 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_1 0x0 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_1 0x1 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_3 0x0 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --add -LUN CTC_3 0x1 10:00:00:05:1e:a7:ab:aa
20:00:00:05:1e:a7:ab:aa
Operation succeeded.
SAN32B-E4-1:admin> cryptocfg --commit
Operation succeeded.
SAN32B-E4-1:admin>

```

The next step is to alter the zone definition. In Example 4-51, we first show the current active zone configuration, clearly showing that the old HBA is defined and not the new HBA WWPN. We then add the new HBA WWPN to the existing alias for our host and remove the previous HBA WWPN. By using the **alishow** command, we confirm that the alias for our affected host is now correct, before saving the updated configuration and making it active with the **cfgenable** command. The final **cfgshow** command is truncated for documentation purposes to show that our effective configuration is changed with the new HBA in the host#1 zone.

Example 4-51 Updating the zoning

```

Effective configuration:
cfg:  ITS0_SG247977
zone:  W2k8_1_to_V7000_1
        50:05:07:68:02:30:a7:be
        50:05:07:68:02:30:a7:fe
        10:00:00:05:1e:c7:6b:8a
zone:  W2k8_2_to_V7000_1

```

```
10:00:00:05:1e:c7:6b:a2
50:05:07:68:02:40:a7:fe
50:05:07:68:02:40:a7:be
```

```
IBM_2498_R06:admin> aliadd "Win2k8_1", "10:00:00:05:1E:A7:AB:AA"
IBM_2498_R06:admin> aliremove "Win2k8_1", "10:00:00:05:1E:C7:6B:8A"
IBM_2498_R06:admin> alishow Win2k8_1
alias: Win2k8_1
10:00:00:05:1e:a7:ab:aa
```

```
IBM_2498_R06:admin> cfigsave
You are about to save the Defined zoning configuration. This
action will only save the changes on Defined configuration.
Any changes made on the Effective configuration will not
take effect until it is re-enabled. Until the Effective
configuration is re-enabled, merging new switches into the
fabric is not recommended and may cause unpredictable
results with the potential of mismatched Effective Zoning
configurations.
Do you want to save Defined zoning configuration only? (yes, y, no, n): [no] y
Updating flash ...
IBM_2498_R06:admin> cfigenable ITS0_SG247977
You are about to enable a new zoning configuration.
This action will replace the old zoning configuration with the
current configuration selected. If the update includes changes
to one or more traffic isolation zones, the update may result in
localized disruption to traffic on ports associated with
the traffic isolation zone changes
Do you want to enable 'ITS0_SG247977' configuration (yes, y, no, n): [no] y
zone config "ITS0_SG247977" is in effect
Updating flash ...
IBM_2498_R06:admin> cfigshow
.
.
Effective configuration:
cfg: ITS0_SG247977
zone: W2k8_1_to_V7000_1
50:05:07:68:02:30:a7:be
50:05:07:68:02:30:a7:fe
10:00:00:05:1e:a7:ab:aa
zone: W2k8_2_to_V7000_1
10:00:00:05:1e:c7:6b:a2
50:05:07:68:02:40:a7:fe
50:05:07:68:02:40:a7:be

IBM_2498_R06:admin>
```

With the crypto containers and active fabric zoning now updated with the new HBA, we are safe to unblock the switch port to which this HBA is connected. We unblock the switch port by using the **portcfgpersistentenable 9** command.

Finally, we need to update the host definition in the Storwize V7000 to remove the old HBA and define the new WWPN to the host definition. Using the Storwize V7000 GUI, we navigate to the Ports by Host window, as shown in Figure 4-73 on page 168. We click the old adapter

WWPN to select it and then click **Delete Port**. After a confirmation box is displayed, the Storwize V7000 will commit our change.

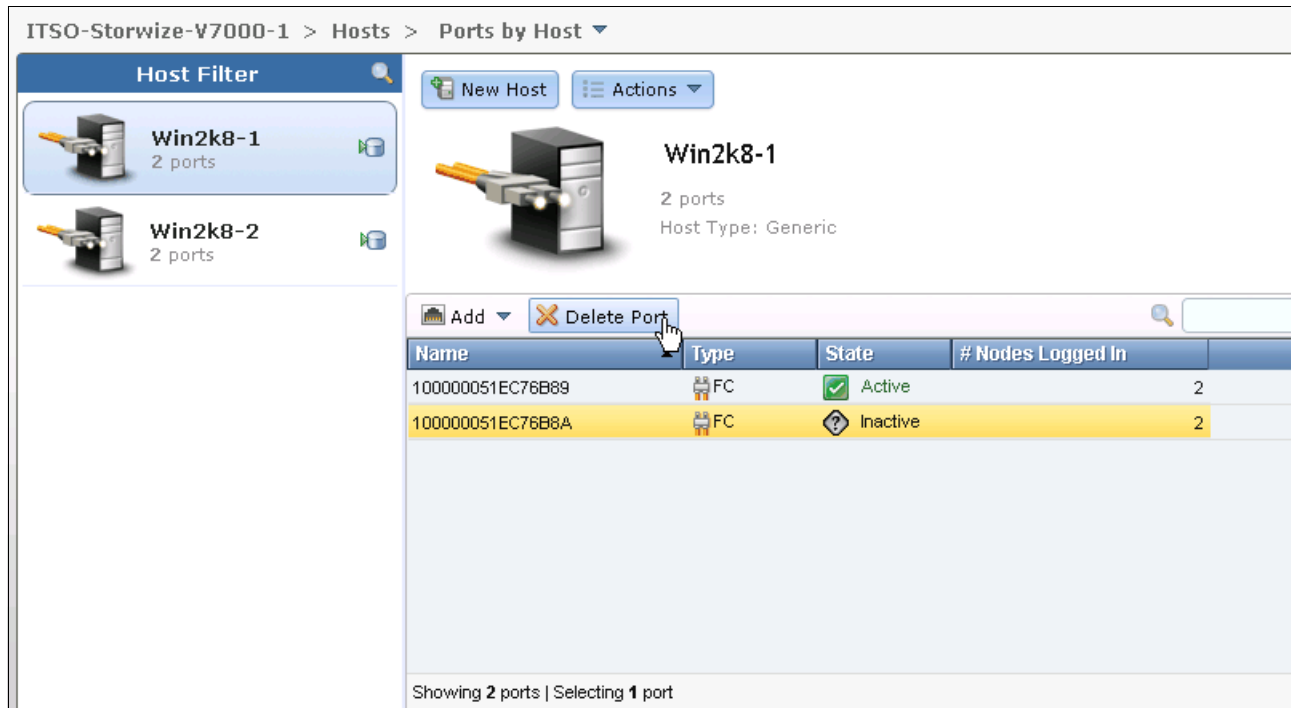


Figure 4-73 Removing old HBA host definition

To add the new HBA to the host, we click **Add** and select **Fibre Channel Port** to get the window that is shown in Figure 4-74. By clicking the drop-down list, we see our new HBA WWPN listed, which allows us to select it easily. If it is not listed, we manually enter the WWPN in the same box, click **Add Port to List**, and then click **Add Ports to Host** to commit the definition to the Storwize V7000.

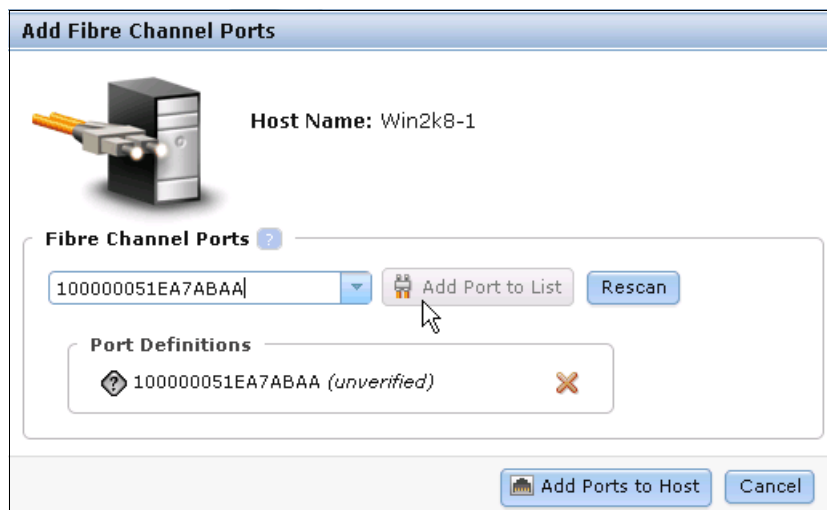


Figure 4-74 Adding host HBA definition

A rescan on the host might be required to pick up the new paths. A **datapath query device** command shows all the paths as online, as shown in Example 4-52 on page 169.

Example 4-52 Checking that all paths are operational

C:\Program Files\IBM\SDDDSM>**datapath query device**

Total Devices : 2

DEV#: 0 DEVICE NAME: Disk1 Part0 TYPE: 2145 POLICY: OPTIMIZED
SERIAL: 600507680185853FF000000000000000

=====

Path#	Adapter/Hard Disk	State	Mode	Select	Errors
0	Scsi Port11 Bus0/Disk1 Part0	OPEN	NORMAL	0	0
1	Scsi Port11 Bus0/Disk1 Part0	OPEN	NORMAL	0	0
2	Scsi Port12 Bus0/Disk1 Part0	OPEN	NORMAL	0	0
3	Scsi Port12 Bus0/Disk1 Part0	OPEN	NORMAL	0	0

DEV#: 1 DEVICE NAME: Disk2 Part0 TYPE: 2145 POLICY: OPTIMIZED
SERIAL: 600507680185853FF0000000000000002

=====

Path#	Adapter/Hard Disk	State	Mode	Select	Errors
0	Scsi Port11 Bus0/Disk2 Part0	OPEN	NORMAL	0	0
1	Scsi Port11 Bus0/Disk2 Part0	OPEN	NORMAL	0	0
2	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	0	0
3	Scsi Port12 Bus0/Disk2 Part0	OPEN	NORMAL	0	0

C:\Program Files\IBM\SDDDSM>



Advanced techniques and functionality

In this chapter, we describe advanced methods of working with IBM SAN32B-E4, IBM SAN768/384 with Encryption Blades, and IBM Storwize V7000. We provide an in-depth explanation of the technology of the encryption process, point to the key areas, and show how to avoid several potential errors.

We describe the following topics in this chapter:

- ▶ Frame redirection zone
- ▶ Performance effect of the rekey and first-time encryption process
- ▶ Encryption solution sizing and planning
- ▶ Implementation of copy services with the encryption solution

5.1 Frame redirection zone in detail

In Chapter 2, “Terminology and technology” on page 13, we briefly described the frame redirection zone, which is a major mechanism for encryption technology. An essential part of the frame redirection zone is the Crypto Target Container (CTC). If there are no defined CTCs, the frame redirection process cannot be enabled. When encryption is implemented, frames that are sent between a host and a target LUN are redirected to a virtual target within an IBM SAN32B-E4 switch or IBM DCX768/384 Encryption Blade. *Redirection zones* are created to route these frames. When redirection zones are in effect, direct access from host to target must not be allowed to prevent data corruption, as shown in Figure 5-1.

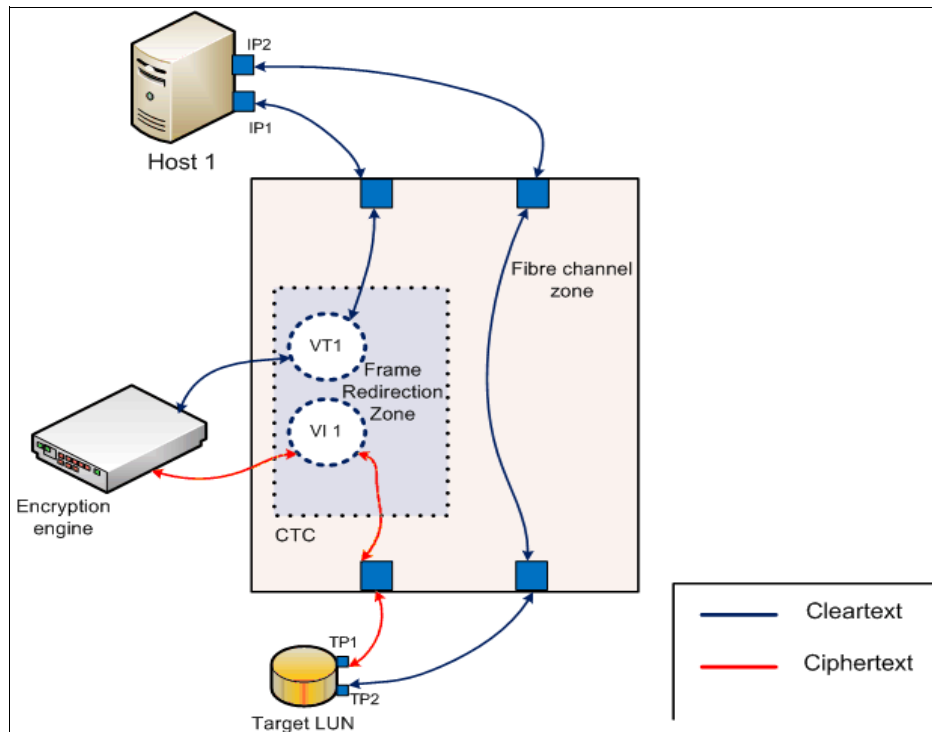


Figure 5-1 Examples of frame redirection zone and traffic that is not redirected

In Figure 5-1, Host 1 has two Fibre Channel (FC) ports, and there are two paths to the Target LUN. Traffic from port IP1 to port TP1 is redirected. From port IP2 to port TP2, traffic is not redirected. This example is shown for example purposes only. You *must* avoid situations similar to this one.

Zone before encrypting: Zone the hosts and targets together before you configure them for encryption. Redirection zones are automatically created to redirect the host-target traffic through the encryption engine (EE), but redirection zones can only be created if the host and target are already zoned.

Frame redirection starts after the CTC creation has completed and has been confirmed. A CTC contains the following components:

- ▶ Target port
- ▶ Initiator ports
- ▶ Target logical unit numbers (LUNs)
- ▶ EE worldwide node (WWN)

Having the EE WWN in the CTC is important, because it is responsible for the encryption itself, although all configuration changes are done on the encryption group leader (GL).

Note: You perform all changes on the GL, but to get the status of a specific CTC, run the `cryptocfg --show -container CTC_name -stat` command on the node on which the CTC is hosted. Running this command on the node is explained by the strict link between the CTC and the EE, which is only responsible for the status of the CTC.

5.1.1 Name service

Name server (NS)-based frame redirection enables the Encryption Switch or Encryption Blade to be deployed transparently to hosts and targets in the fabric. NS-based frame redirection is enabled in the following manner:

- ▶ You first create a zone that includes the host (H) and target (T). This process adds records of the aliases, the port, and the node WWNs to the name server database and adds zones to the effective configuration, as well. Sometimes, this process can cause temporary traffic disruption to the host.
- ▶ You then create a CTC for the target and configure the container to allow access to the initiator. This process adds additional records to the name service database.
- ▶ When you commit the transaction, a special zone called a “*redirection zone*” is generated automatically. The redirection zone includes the host (H), the virtual target (VT), the virtual initiator (VI), and the target (T). The redirection zone is not included in the effective fabric configuration and is used only for reference of the EEs.
- ▶ When configuring multipath LUNs, do not commit the CTC configuration before you have performed the following steps in sequence to prevent data corruption. We explain the details of these steps in Chapter 4, “Implementation scenarios and recommendations for managing the SAN32B-E4 Encryption Switch” on page 89.

The EE uses the fabric’s name service to look for the target’s and initiator’s WWNs. It also creates the additional records in the name service database with the assigned VT’s and VI’s WWNs. You can discover these records with the `nsshow` command, as shown in Example 5-1.

Example 5-1 The nsshow command output (output is truncated)

```
IBM_SAN384B:FID128:admin> nsshow
{
N    012501;      3;20:08:00:05:1e:c1:13:13;20:09:00:05:1e:c1:13:13; na
FC4s: FCP
PortSymb: [21] "Crypto Virtual Target"
NodeSymb: [55] "Cntr Name: CTC_4B Tgt Port wwn: 50:05:07:68:02:20:A7:BE"
Fabric Port Name: 20:25:00:05:1e:94:3a:00
Permanent Port Name: 20:08:00:05:1e:c1:13:13
Port Index: 37
Share Area: No
Device Shared in Other AD: No
Redirect: Yes virtual
Partial: No
N    011d01;      3;20:0a:00:05:1e:c1:13:13;20:0b:00:05:1e:c1:13:13; na
FC4s: FCP
PortSymb: [24] "Crypto Virtual Initiator"
NodeSymb: [37] "Ini Port wwn: 10:00:00:05:1E:C7:6B:A1"
Fabric Port Name: 20:1d:00:05:1e:94:3a:00
Permanent Port Name: 20:0a:00:05:1e:c1:13:13
```

Port Index: 29
Share Area: No
Device Shared in Other AD: No
Redirect: Yes virtual
Partial: No

Example 5-1 on page 173 shows the two types of additional records in the name server database. One record is for the VT, and the other record is for the VI. Both records have the Yes virtual attribute, which identifies that they are virtual, that is, you will not find real devices with assigned virtual WWNs.

OS levels: To enable frame redirection, the host and target edge switches must run Fabric OS v6.1.1 and Fabric OS v5.3.1.b or later firmware to ensure host and target connectivity with existing platforms. In McDATA fabrics, the hosts and the switches that are hosting the targets require firmware versions M-EOSc 9.8 and M-EOSn 9.8 or later.

5.1.2 Crypto Target Container

We described the process to create Crypto Target Containers (CTCs) in 4.3.1, “Creating containers” on page 98. Here, we focus on other important areas after the CTC has been created.

Initially, the default zoning for IBM SAN switches is set to All Access. The All Access setting allows the IBM SAN32B-E4 and IBM SAN768/384 to join the fabric and be discovered before zoning is applied. If a difference exists in this setting within the fabric, the fabric will segment. Before committing an encryption configuration in a fabric, you must set the default zoning to No Access within the fabric. The No Access setting ensures that no two devices on the fabric can communicate with one another without going through a regular zone or a redirection zone.

Check the default zoning setting. Usually, it is set to All Access, as shown in Example 5-2.

Example 5-2 Checking the default zone status

```
switch:admin> defzone --show
```

Default Zone Access Mode

committed - All Access

transaction - No Transaction

From any configured primary switch, change the default zone setting to No Access, as shown in Example 5-3.

Example 5-3 Setting the default zone status to No Access

```
switch:admin> defzone --noaccess
```

```
switch:admin> cfgfsave
```

The change will be applied within the entire fabric.

Consider these key points when creating the CTC:

- ▶ Estimating the number of required CTCs
- ▶ Estimating the multipathing requirements for the LUN
- ▶ Adding cleartext and encrypted LUNs to the container
- ▶ Understanding the components and the structure of the CTC

We do not describe the creation process itself, but we point to areas of focus within it.

5.1.3 Estimating the number of required CTCs

Note the rule that one CTC exists for every target port. So, for the number of target ports that you have in your configuration, you create the same number of CTCs.

The command output in Example 5-4 shows the current effective fabric configuration with the **cfgshow** command.

Example 5-4 The output of the cfgshow command (output is truncated)

```
SAN32B-E4-1:admin> cfgshow
Effective configuration:
  cfg:   ITS0_SG247977
  zone:  W2k8_1_to_V7000_1
          50:05:07:68:02:30:a7:be
          50:05:07:68:02:30:a7:fe
          10:00:00:05:1e:c7:6b:8a
  zone:  W2k8_2_to_V7000_1
          10:00:00:05:1e:c7:6b:a2
          50:05:07:68:02:40:a7:fe
          50:05:07:68:02:40:a7:be
```

Example 5-4 shows that Server W2k8_1_to_V7000_1 has one initiator port (10:00:00:05:1e:c7:6b:8a) and two target ports (50:05:07:68:02:30:a7:be and 50:05:07:68:02:30:a7:fe). So, you need to create two CTCs for this server. Each CTC will contain one target port and one initiator port. Use the **cryptocfg --show -container -all -cfg** command. Example 5-5 shows the CTCs that have been created.

Example 5-5 Output of the cryptocfg --show -container -all -cfg command

```
SAN32B-E4-1:admin> cryptocfg --show -container -all -cfg
Encryption group name: support_TKLM
Number of Container(s): 2
Container name:       CTC_1
Type:                 disk
EE node:              10:00:00:05:1e:54:17:10
EE slot:              0
Target:               50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
VT:                   20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
Number of host(s):    1
Configuration status: committed
Host:                 10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
VI:                   20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
Number of LUN(s):     0
Container name:       CTC_3
Type:                 disk
EE node:              10:00:00:05:1e:54:17:10
```

```

EE slot:                0
Target:                50:05:07:68:02:30:a7:be 50:05:07:68:02:00:a7:be
VT:                   20:04:00:05:1e:54:17:0c 20:05:00:05:1e:54:17:0c
Number of host(s):      1
Configuration status:   committed
Host:                 10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
VI:                   20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
Number of LUN(s):       2

```

We use the **cryptocfg --show -container CTC_NAME -cfg** command to display the configuration of the CTC, as shown in Example 5-6.

Example 5-6 Output of the cryptocfg --show -container CTC_1 -cfg command

```

SAN32B-E4-1:admin> cryptocfg --show -container CTC_1 -cfg
Container name:        CTC_1
Type:                 disk
EE node:              10:00:00:05:1e:54:17:10
EE slot:              0
Target:               50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
VT:                   20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
Number of host(s):    1
Configuration status: committed
Host:                 10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
VI:                   20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
Number of LUN(s):     2
Operation succeeded.

```

Note the following values in Example 5-6:

- ▶ 50:05:07:68:02:30:a7:fe is the *real* target port WWN.
- ▶ 10:00:00:05:1e:c7:6b:8a is the *real* initiator port WWN.
- ▶ 20:00:00:05:1e:54:17:0c is the *virtual* target port WWN.
- ▶ 20:08:00:05:1e:54:17:0c is the *virtual* initiator port WWN.

During the commit process of the changes to the encryption group (EG) with the **cryptocfg --commit** command, the changes are propagated to the entire EG, and frame redirection zones are created.

Important: When making configuration changes using the CLI, you are required to use the **cryptocfg --commit** command. If this command has not been issued *after* making changes to the configuration, *no changes will apply*. All commands that are run *before* are kept in the temporary memory of the FC switch, and it is similar to zoning configuration changes. This memory is volatile, and if a reboot of the switch occurs, *all changes are lost*. When making configuration changes with the DCFM software, all changes are committed when you click **Apply**. No special commit operation is available when using the wizards.

Use the **nsshow** and **cfgshow** commands to see the changes in the Name Service database. Example 5-7 shows the **cfgshow** output.

Example 5-7 Output of the cfgshow command after creating the CTC (output is truncated)

```

SAN32B-E4-1:admin> cfgshow
Defined configuration:
cfg:   ITS0_SG247977

```

```

W2k8_1_to_V7000_1; W2k8_2_to_V7000_1
cfg:  r_e_d_i_r_c_fg
      red_1109_support_TKLM500507680230a7fe200800051e54170c;
      red_____base
zone:  W2k8_1_to_V7000_1
      V7000_1_p3; Win2k8_1
zone:  W2k8_2_to_V7000_1
      Win2k8_2; V7000_1_p4
zone:  red_1109_support_TKLM500507680230a7fe200800051e54170c
      10:00:00:05:1e:c7:6b:8a; 50:05:07:68:02:30:a7:fe;
      20:08:00:05:1e:54:17:0c; 20:00:00:05:1e:54:17:0c
zone:  red_____base
      00:00:00:00:00:00:00:01; 00:00:00:00:00:00:00:02;
      00:00:00:00:00:00:00:03; 00:00:00:00:00:00:00:04
alias: V7000_1_p3
      50:05:07:68:02:30:A7:BE; 50:05:07:68:02:30:A7:FE
alias: V7000_1_p4
      50:05:07:68:02:40:A7:FE; 50:05:07:68:02:40:A7:BE
alias: Win2k8_1
      10:00:00:05:1E:C7:6B:8A
alias: Win2k8_2
      10:00:00:05:1E:C7:6B:A2
Effective configuration:
cfg:  ITS0_SG247977
zone:  W2k8_1_to_V7000_1
      50:05:07:68:02:30:a7:be
      50:05:07:68:02:30:a7:fe
      10:00:00:05:1e:c7:6b:8a
zone:  W2k8_2_to_V7000_1
      10:00:00:05:1e:c7:6b:a2
      50:05:07:68:02:40:a7:fe
      50:05:07:68:02:40:a7:be
SAN32B-E4-1:admin>

```

As shown in Example 5-7, the red_1109_support_TKLM500507680230a7fe200800051e54170c redirection zone has appeared. This newly created zone is not in the effective configuration of the fabric, and it is used only by the EEs to provide frame redirection. The Fabric Name Service database keeps those records in the defined configuration state.

Important: From this point in time, no traffic goes through the initiator ports participating in the frame redirection except the encrypted traffic. This situation causes disruption in LUN access until the LUN is added to the CTC and the encryption policy is defined for it. To prevent data loss and an application crash, stop any I/O on the ports that is intended for encryption. *We advise that you unmount the appropriate LUNs also.*

Initiator ports that are intended for encrypted traffic cannot participate in any other zones, and they cannot host any other traffic. If there are not enough FC ports in the server, you must add a new FC adapter to resolve the issue.

5.1.4 Estimating the multipathing requirements for the LUN

A single LUN can be accessed over multiple paths. A multipath LUN is exposed and configured on multiple CTCs that are located on the same Encryption Switch or Encryption Blade or on separate Encryption Switches or Blades.

Important: All paths of the LUN that you intend to encrypt must be associated with the appropriate CTCs. Paths that are not intended to be used for encryption must be removed from access to the LUN. *Not following this rule, and leaving access in place to the LUN for paths that are not using EE will lead to data corruption.*

To avoid the risk of data corruption, you must observe the following rules when configuring multipath LUNs:

- ▶ During the initiator-target zoning phase, complete in sequence all zoning for *all* hosts that must gain access to the targets before committing the zoning configuration.
- ▶ Complete the CTC configuration for *all* target ports in sequence and add the hosts that must gain access to these ports before committing the container configuration. Upon commit, the hosts lose access to all LUNs until the LUNs are explicitly added to the CTCs.
- ▶ When configuring the LUNs, the same LUN policies must be configured for *all* paths of *all* LUNs. Failure to configure all LUN paths with the same LUN policies will result in data corruption.

Figure 5-2 shows the CTCs that have been created for the multipath solution.

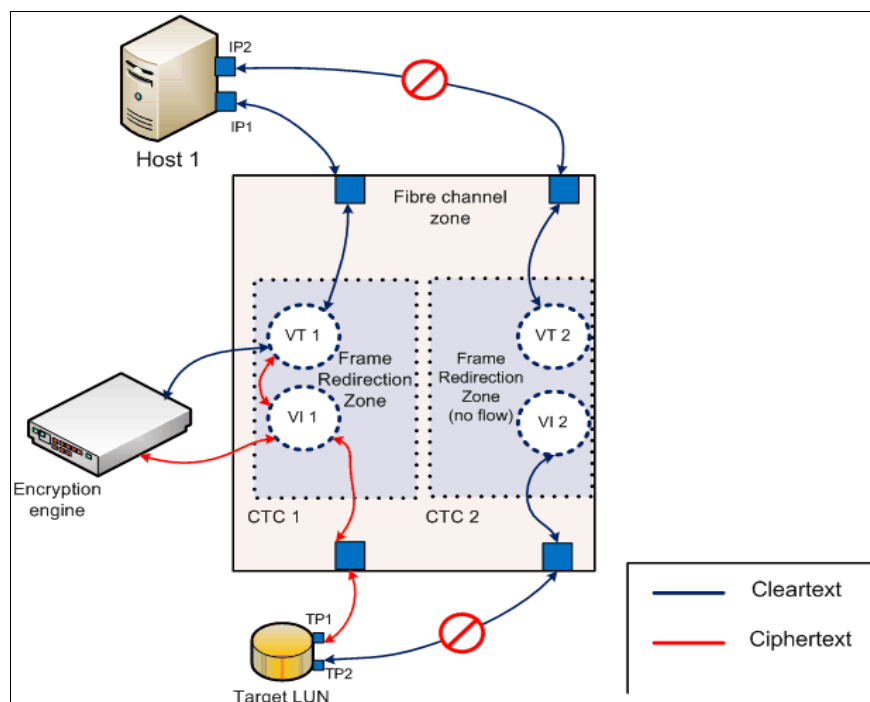


Figure 5-2 Two paths to one target with one active path

The host has two FC ports. The LUN is connected using two target ports. CTC 1 is active and redirects traffic through the EE for one path. CTC 2 is created, but it has no LUN specified, so no traffic flows through it. This situation is correct for a multipath solution when it is not fully configured. You must avoid a situation when one path is configured to use encryption and the

other path is not, because this situation will lead to data corruption. Figure 5-1 on page 172 is the situation that you are trying to avoid.

In our setup, server W2k8_1_to_V7000_1 had to have two paths to the target from one initiator port. See Example 5-8 where we check the number of paths to the target.

Example 5-8 Checking the number of paths to the target

```
zone:  W2k8_1_to_V7000_1
      50:05:07:68:02:30:a7:be
      50:05:07:68:02:30:a7:fe
      10:00:00:05:1e:c7:6b:8a
```

IBM Storwize V7000: IBM Storwize V7000 always provides multiple paths to the LUNs.

We have created two CTCs. Remember the rule of one CTC for each target port, as shown in Example 5-9.

Example 5-9 Configuration of the CTCs (output is truncated)

```
IBM_SAN384:admin> cryptocfg --show -container -all -cfg
Container name:      CTC_1
Type:                 disk
EE node:              10:00:00:05:1e:54:17:10
EE slot:              0
Target:             50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
VT:                  20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
Number of host(s):    1
Configuration status: committed
Host:               10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
VI:                  20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
Container name:      CTC_3
Type:                 disk
EE node:              10:00:00:05:1e:54:17:10
EE slot:              0
Target:             50:05:07:68:02:30:a7:be 50:05:07:68:02:00:a7:be
VT:                  20:04:00:05:1e:54:17:0c 20:05:00:05:1e:54:17:0c
Number of host(s):    1
Configuration status: committed
Host:               10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
VI:                  20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
Operation succeeded.
```

5.1.5 Adding LUNs to the CTCs

No traffic exists on the initiator port until a LUN (any LUN) is added to the CTC. Prior to adding the LUN, the LUN must be discovered to define its number. The LUN number here is the number in hexadecimal of the Small Computer System Interface (SCSI) ID value in decimal, which was assigned to the appropriate LUN when it was being mapped to the host in the IBM Storwize V7000 host mapping window. Example 5-10 on page 180 shows how we discover the LUNs in CTC_1.

Example 5-10 Discovering the LUNs in CTC_1

```

SAN32B-E4-1:admin> cryptocfg --discoverLUN CTC_1
Container name:      CTC_1
Number of LUN(s):    2
Host:                10:00:00:05:1e:c7:6b:8a
LUN number:       0x0
LUN serial number: 600507680185853FF000000000000000
LUN connectivity state: Connected
Key ID state:        Read only
Key ID:              47:62:48:66:9d:f8:4a:76:91:28:6c:f1:9a:a4:cb:e2
Host:            10:00:00:05:1e:c7:6b:8a
LUN number:       0x1
LUN serial number: 600507680185853FF000000000000002
LUN connectivity state: Connected
Key ID state:        Read write
Key ID:              a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36
  
```

Example 5-10 shows the LUNs that were discovered for CTC_1. Two LUNs were discovered:

- The first LUN that was discovered was LUN serial number 600507680185853FF000000000000000 and LUN number 0x0
- The second LUN that was discovered was LUN serial number 600507680185853FF000000000000002 and LUN number 0x2

To avoid mistakes in the configuration, it is better to identify the discovered LUNs using the LUN serial number, which is the same as the LUN UID. You can obtain this number in the IBM Storwize V7000 Volumes Mapped to the Host window, as shown in Figure 5-3.

Volumes Mapped to the Host		
Edit SCSI ID ← Unmap 🔍 <input type="text"/>		
SCSI ID	Name	UID
0	win2k8-1_Lun1_01	600507680185853FF000000000000005
1024	win2k8-2_Lun0	600507680185853FF000000000000001

Figure 5-3 Volumes Mapped to the Host window

Figure 5-3 shows the LUN UID 600507680185853FF000000000000001 and SCSI ID 1024 mapped to the host. Example 5-11 shows the discovery of the LUN UID.

Example 5-11 Discovery of LUN UID 600507680185853FF000000000000001

```

IBM_SAN384B:FID128:admin> cryptocfg --discoverLUN CTC_2B
Container name:      CTC_2B
Number of LUN(s):    2
Host:                10:00:00:05:1e:c7:6b:a1
LUN number:          0x0
LUN serial number:    600507680185853FF000000000000005
LUN connectivity state: Connected
Key ID state:         Read write
  
```

Key ID: a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36
Host: 10:00:00:05:1e:c7:6b:a1
LUN number: 0x400
LUN serial number: 600507680185853FF000000000000001
LUN connectivity state: Connected

Example 5-11 on page 180 shows the LUN UID 600507680185853FF000000000000001 found under the LUN serial number 600507680185853FF000000000000001 and the LUN number 0x400, which is the hexadecimal representation of decimal number 1024. The SCSI ID of the LUN can be changed during LUN remapping, but the LUN UID and LUN serial number remain the same. The discovered LUN can be added to the CTC with the following options, as shown in Table 5-1.

Table 5-1 Options for adding LUN to the CTC

Option	Description
-lunstate <encrypted cleartext >	Sets the encryption state of a specified disk LUN. When set to encrypted, the metadata on the LUN containing the key ID of the data encryption key (DEK) that was used for encrypting the LUN is used to retrieve the DEK from the key vault. If the LUN state is not specified, the default state is cleartext.
-encryption_format <native DF_compatible>	Specifies the LUN encryption format. Two encryption formats are supported: native and DF_compatible. We use only the native metadata format and algorithm for the encryption and decryption of data. This is the default mode.
-encrypt -cleartext	Enables or disables the LUN for encryption. By default, cleartext is enabled (no encryption). When the LUN policy is changed from encrypt to cleartext, the following policy parameters become disabled (default) and generate errors when executed: -enable_encexistingdata , -enable_rekey , and -key_lifespan .
-enable_encexistingdata -disable_encexistingdata	Specifies whether existing data must be encrypted. The Encryption policy must be enabled on the LUN before the -enable_encexistingdata parameter can be set and the LUN state must be set to -cleartext . By default, the encryption of the existing data is disabled. If the LUN policy is set to -encrypt , the encryption of the existing data must be enabled, or existing data is not preserved.
-enable_rekey <time period> -disable_rekey	Enables or disables the automatic rekeying capability on the specified disk LUN. By default, the automatic rekey feature is disabled. Enabling automatic rekeying is valid only if the LUN policy is set to encrypt. You must specify a time period in days when enabling automatic rekeying to indicate the interval at which automatic rekeying must take place.
-key_lifespan <time in days>	Specifies the lifespan of the encryption key in days. The key will expire after the specified number of days. Accepted values are integers from 1 to 2982616. The default value is none, which means that the key does not expire. This operand is valid only for tape LUNs. The key life span cannot be modified after it is set.

Option	Description
-keyID <key ID>	Specifies the key ID. Use this operand only if the LUN was encrypted but does not include the metadata containing the key ID for the LUN.

The following options are important and used most frequently:

- ▶ **-lunstate**
- ▶ **-encrypt**
- ▶ **-enable_encexistingdata**

Option **-lunstate** indicates the status of the LUN that is being added. Unencrypted LUNs are added with the **cleartext** option, and encrypted with the **encrypted** option.

Important: LUNs, which are encrypted already, must always have **-lunstate encrypted** set while they are added. Not following this guideline can cause DEK blocking or the destruction of the data.

The policy option **-encrypt** indicates that encryption must be turned on when the LUN is added. For the LUNs added in cleartext, encryption turns on after the first-time encryption or the rekey operation.

You must specify the option **-enable_encexistingdata** for the cleartext LUNs, which contain user data. This option is extremely important, because otherwise, the first-time encryption process will not be initiated and the LUN is considered empty. This state leads to data corruption. LUNs that do *not* contain any user data can skip this option to save time.

After you add LUNs to the CTC and commit the operation forming the CTCs, and the frame redirection zone completes, normal traffic flow is restored.

Commit when using the CLI: Remember to commit all the configuration changes with the **cryptocfg --commit** command when you use the CLI. Wizards in DCFM commit the changes automatically when you click **Apply**.

5.1.6 Understanding the structure of the CTC

You can display the structure of the CTC with the **cryptocfg --show -container CTCNAME -stat** command, as shown in Example 5-12.

Example 5-12 Output of the cryptocfg --show -container CTC_1 -stat command

```

SAN32B-E4-1:admin> cryptocfg --show --container CTC_1 -stat
Container name:      CTC_1
Type:                disk
EE node:             10:00:00:05:1e:54:17:10
EE slot:             0
EE hosting container: current
Target:              50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:          030400
VT:                  20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
VT PID:              012001
Number of host(s):   1
Number of rekey session(s): 0
Host:                10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a

```

```

Host PID:                030900
VI:                    20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:                  012401
Number of LUN(s):      2
LUN number:           0x0
LUN type:                disk
LUN serial number:    600507680185853FF000000000000000
Encryption mode:         encrypt
Encryption format:    native
Encrypt existing data:    disabled
Rekey:                   disabled
Internal EE LUN state: Read only (Current key is in read only state)
Encryption algorithm: AES256-XTS
Key ID state:            Read only
New LUN:                 No
Key ID:              47:62:48:66:9d:f8:4a:76:91:28:6c:f1:9a:a4:cb:e2
Key creation time:       Thu Jul 28 09:50:32 2011
LUN number:           0x1
LUN type:                disk
LUN serial number:    600507680185853FF0000000000000002
Encryption mode:      encrypt
Encryption format:       native
Encrypt existing data:    disabled
Rekey:                   disabled
Internal EE LUN state: Encryption enabled
Encryption algorithm: AES256-XTS
Key ID state:         Read write
New LUN:                 No
Key ID:              a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36
Key creation time:       Wed Jul 27 14:48:54 2011
Operation succeeded.

```

Example 5-12 on page 182 shows the output of the query of the CTC status. The main entries of the output are shown in bold in the example. Read the CTC in the following way.

The CTC name is CTC_1. It is hosted and serviced by the EE on node 10:00:00:05:1e:54:17:10 (EE node: 10:00:00:05:1e:54:17:10). This CTC is made for target 50:05:07:68:02:30:a7:fe, which has virtual target WWN 20:00:00:05:1e:54:17:0c.

This CTC has one host that is specified with the port WWN 10:00:00:05:1e:c7:6b:8a and the virtual initiator WWN 20:08:00:05:1e:54:17:0c. This host has access to 2 LUNs, 0x0 and 0x1. The LUN 0x0 UID is 600507680185853FF000000000000000. The LUN 0x1 UID is 600507680185853FF0000000000000002.

Both LUNs are encrypted and online. One LUN, 0x0, is encrypted with DEK ID 47:62:48:66:9d:f8:4a:76:91:28:6c:f1:9a:a4:cb:e2, which is now in Read Only status. The other LUN, 0x1, is in Read write mode and is encrypted with DEK ID a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36.

No background operations, such as rekeying, are running now. If they were, the Number of rekey sessions entries: 0 and Rekey: disabled have another status.

Example 5-13 on page 184 shows the CTC with two hosts connected.

Example 5-13 Output of the CTC with two hosts (output is truncated)

```
IBM_SAN384B:FID128:admin> cryptocfg --show -container CTC_1B -stat
Container name:          CTC_1B
Type:                     disk
EE node:                10:00:00:05:1e:94:3a:00
EE slot:                1
EE hosting container:      current
Target:                 50:05:07:68:02:10:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:               010000
VT:                    20:04:00:05:1e:c1:13:13 20:05:00:05:1e:c1:13:13
VT PID:                   011101
Number of host(s):       2
Number of rekey session(s): 0
Host:                  10:00:00:05:1e:c7:6b:89 20:00:00:05:1e:c7:6b:89
Host PID:                 010900
VI:                    20:02:00:05:1e:c1:13:13 20:03:00:05:1e:c1:13:13
VI PID:                   011402
Number of LUN(s):       2
LUN number:               0x0
LUN type:                 disk
LUN serial number:     600507680185853FF0000000000000000
Key ID:                47:62:48:66:9d:f8:4a:76:91:28:6c:f1:9a:a4:cb:e2
LUN number:               0x1
LUN type:                 disk
LUN serial number:     600507680185853FF0000000000000002
Key ID:                a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36
Key creation time:         Wed Jul 27 14:48:54 2011
Host:                  10:00:00:05:1e:c7:6b:a1 20:00:00:05:1e:c7:6b:a1
Host PID:                 010800
VI:                    20:0a:00:05:1e:c1:13:13 20:0b:00:05:1e:c1:13:13
VI PID:                   012501
Number of LUN(s):       1
LUN number:            0x1
LUN type:                 disk
LUN serial number:     600507680185853FF0000000000000003
Encryption mode:       cleartext
Encryption format:         native
Encrypt existing data:     disabled
Rekey:                     disabled
Internal EE LUN state: Disabled (Found metadata while LUN is clear text)
Encryption algorithm:  None
Key ID state:          Key ID not Applicable
New LUN:                   No
Operation succeeded.
```

Example 5-13 shows the output of the CTC with name CTC_2B, which has two hosts defined: One host with PWWN 10:00:00:05:1e:c7:6b:89 and the other host with PWWN 10:00:00:05:1e:c7:6b:a1. Host 1 has two LUNs, and host 2 has one LUN. LUN 0x1 (UID 600507680185853FF0000000000000003) of host 2 has encryption status **-cleartext**, encryption disabled, and the EE was unable to read the DEK ID from the LUN. Also, there are errors with adding this LUN, so the KEY ID state is Key ID not Applicable.

5.1.7 Changing the fabric configuration

During production activity, certain changes might occur. Those changes can include adding or removing FC ports from the fabric, or adding or removing LUNs. Implementing encryption makes these procedures complex, and you must remember that changing the fabric configuration and *not* updating the encryption configuration in accordance with the changes might cause I/O interruption to the encrypted LUNs. This interruption might take place until the incorrect configuration has been eliminated. Certain configuration changes are disruptive.

If the port exists in the encryption configuration, it must be removed from the configuration of the EG first, and then removed from the fabric.

Next, we show the sequence of the necessary steps for removing the initiator port from the fabric:

1. Stop the application.
2. Unmount the LUN that is related to this initiator port.
3. Remove the initiator port from the CTCs.
4. Commit the changes to the EG.
5. Change the zoning configuration and remove the initiator.
6. Commit the changes.
7. Check the statuses of the CTCs.
8. Mount the LUNs again if everything is in order.

For removing the LUN from the EG, perform the following steps:

1. Stop the application.
2. Unmount the LUN.
3. Remove the LUN from the CTCs.
4. Commit the changes to the EG.
5. Change the host mapping settings and remove the LUN from mapping.
6. Commit the changes.
7. Check the statuses of the CTCs.

Important: For LUNs with multiple paths, which is always true for the IBM Storwize V7000, the configuration of all CTCs that relate to the same LUN must remain identical. Ensure that the CTC configurations are identical *before* committing the changes to the EG. Not following this guideline might result in data corruption.

Example 5-14 shows removing the initiator port from the CTC and configuration changes are verified before being committed.

Example 5-14 Removing initiator port from the CTC

```
IBM_SAN384B:FID128:admin> cryptocfg --show -container CTC_3 -cfg
Container name:      CTC_3
Type:                disk
EE node:             10:00:00:05:1e:54:17:10
EE slot:             0
Target:              50:05:07:68:02:30:a7:be 50:05:07:68:02:00:a7:be
VT:                  20:04:00:05:1e:54:17:0c 20:05:00:05:1e:54:17:0c
```

```

Number of host(s):      1
Configuration status:   committed
Host:                   10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
VI:                     20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
Number of LUN(s):      2
Operation succeeded.

IBM_SAN384B:FID128:admin>cryptocfg --remove -initiator CTC_3
10:00:00:05:1e:c7:6b:8a
Operation succeeded.

IBM_SAN384B:FID128:admin> cryptocfg --show -container CTC_3 -cfg
Container name:         CTC_3
Type:                   disk
EE node:                10:00:00:05:1e:54:17:10
EE slot:                0
Target:                 50:05:07:68:02:30:a7:be 50:05:07:68:02:00:a7:be
VT:                     20:04:00:05:1e:54:17:0c 20:05:00:05:1e:54:17:0c
Number of host(s):      0
Configuration status:   committed
Operation succeeded.

IBM_SAN384B:FID128:admin>cryptocfg --commit
Operation succeeded.

```

5.2 First-time encryption and rekey operation in detail

We describe first-time encryption and rekey operations in Chapter 2, “Terminology and technology” on page 13. In this section, we go deeper into the details of these operations. We explain several additional points that might be valuable for planning, implementing, and sizing the IBM Storwize V7000 and SAN32B-E4 Encryption Switch solution.

5.2.1 Performance effect of the first-time encryption and rekey operations

As mentioned in Chapter 2, “Terminology and technology” on page 13, there might be an effect on performance on the current I/O while first-time encryption or rekey operations occur. We performed several simple tests to help you understand the effect of these operations.

The most sensitive workload is the online transaction processing (OLTP) workload. OLTP usually runs in small blocks (up to 64 KB), consists of mostly reads, and is extremely sensitive to the response times, which are usually kept short. Any additional workload that is placed on the OLTP-designated volumes, especially with large blocks for writing, can seriously reduce the performance of the OLTP operations and increase the response times. First-time encryption and rekey operations are always 256 KB blocks for read and write. The large block sequential workload can affect the current OLTP workload. The preferred practice then is to start these operations at a time of minimal I/O to the data. If that is not possible, it is important to understand the potential effect on the performance and to be able to plan this activity.

For example, in our setup, it took approximately 14 hours to rekey a 1 TB LUN; however, lab tests on the busy volumes showed approximately 24 hours for the 1 TB LUN. The numbers mostly depend on the ability of the disk system. You might have an effect on performance to the LUN because of the length of time that the rekey operation takes.

5.2.2 The nature of the first-time encryption and rekey operations

We said before that from a performance point of view, consider the first-time encryption and rekey operations as the same operations. They both have the same workload pattern and the same running time, because actually they do the same work - reading the block from the LUN, encrypting it, and then returning it back. The only difference between these operations is that first-time encryption runs on the cleartext LUN and the rekey operation runs on the encrypted LUN. So, when we discuss the rekey operation, the same information is also true for the first-time encryption operation.

There are two types of rekey operations:

- ▶ Manual rekey
- ▶ Automatic rekey

Manual rekey is invoked manually by administrators to have a new DEK for the LUN that is generated.

Automatic rekey is invoked by the policy setting, which can be done while adding the LUN to the CTC or modified later. This setting is disabled by default.

Table 5-2 explains the main parameters for the rekeying options.

Table 5-2 Main command parameters and the description

Cryptocfg command parameter	Description
-enable_encexistingdata -disable_encexistingdata	Specifies whether existing data must be encrypted. The encryption policy must be enabled on the LUN before the -enable_encexistingdata parameter can be set, and the LUN state must be set to -cleartext . By default, the encryption of existing data is disabled. If the LUN policy is set to -encrypt , the encryption of existing data must be enabled, or existing data is not preserved.
-enable_rekey time_period -disable_rekey	Enables or disables the automatic rekeying capability on the specified disk LUN. By default, the automatic rekey feature is disabled. Enabling automatic rekeying is valid only if the LUN policy is set to encrypt. You must specify a time_period in days when enabling automatic rekeying to indicate the interval at which automatic rekeying must take place.
-key_lifespan time_in_days none	Specifies the life span of the encryption key in days. The key will expire after the specified number of days. Accepted values are integers from 1 to 2982616. The default value is none , which means that the key does not expire.

Cryptocfg command parameter	Description
-manual_rekey	Performs a manual rekeying of a specified LUN that is associated with a specified CTC. Manual rekeying is performed in both an online and off-line manner, depending on whether the host is online or host I/O is present. If any policy-based rekeying operation is currently in progress, this command aborts with a warning message. This command is valid only on the group leader.

The rekey operation has read and write cycles, which change each other sequentially. You can check these cycles with the command **cryptocfg --show -rekey CTCNAME** or **cryptocfg --show -rekey -all** for the all rekey operations, as shown in Example 5-15. We demonstrate this function later.

Example 5-15 Output of cryptocfg --show -rekey CTC_4 (output is truncated)

```

Container name:      CTC_4
VI:                 20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:             012401
LUN number:         0x0
LUN serial number:  600507680185853FF000000000000000
Rekey session number: 0
Percentage complete: 1
Rekey state:        Write Phase
Rekey role:         Primary/Redundant
Block size:         512
Number of blocks:   62914560
Current LBA:        1075841

```

Example 5-15 shows CTC_4, which has one rekey session running with 1% complete and it is in the Write Phase now. The block size of 512 defines the geometry of the LUN, which is 512 byte blocks. It is currently on the 512 block with number 1075841 of the total number of 62914560 blocks.

Example 5-16 shows the Read Phase of the rekey operation, and it is 3% complete.

Example 5-16 Output of cryptocfg --show -rekey CTC_4 (output is truncated)

```

Container name:      CTC_4
Encryption mode:     encrypt
Encryption format:   native
Encrypt existing data: enabled
Rekey:               disabled
Encryption algorithm: AES256-XTS
Key ID state:        Re-key
Rekey session number: 0
Percentage complete: 3
Rekey state:         Read Phase
Rekey role:         Primary/Redundant
Block size:         512
Number of blocks:   62914560
Current LBA:        3227253

```

The rekey operation with phase switching works this way:

- ▶ The EE reads 200 MB - 800 - 900 MB into the memory.
- ▶ The EE runs the encryption process on that read data.
- ▶ The EE writes the encrypted data back to the LUN.
- ▶ It continues with the next data on the LUN.
- ▶ If the rekey process encounters a block that is being written to by a host, it will wait for the host to complete its I/O.
- ▶ If the host I/O attempts to write to the block that is being encrypted, the host I/O is paused for the amount of time that is needed to encrypt this block.
- ▶ After every 5-7% of the LUN has been processed, the EE performs the synchronization of the LUN status within the EG.

The rekey operation can have the following statuses:

- ▶ Read Phase
- ▶ Write Phase
- ▶ HA Sync Phase
- ▶ LUN Cleanup
- ▶ Rekey Setup
- ▶ LUN Prep
- ▶ Key Update

You can discover the status during the rekey operation, as shown in Example 5-17 and Example 5-18.

Example 5-17 Synchronizing the changes within the encryption group (output is truncated)

Rekey session number:	0
Percentage complete:	7
Rekey state:	Cluster Sync After Read Phase
Rekey role:	Primary/Redundant
Block size:	512
Number of blocks:	62914560
Current LBA:	4615793

Example 5-17 shows the status of the changes that are being synchronized with the EG.

Example 5-18 Rekey process waits for the host I/O completion (output is truncated)

Rekey session number:	0
Percentage complete:	62
Rekey state:	Waiting for Host Cmd completion
Rekey role:	Primary/Redundant
Block size:	512
Number of blocks:	62914560
Current LBA:	39556017

Example 5-18 shows a situation when the rekey process wanted to write the changes to the block to which the host was writing. The rekey operation has been paused until the host has completed its operation.

Performance effect

There will always be an effect on performance during a rekey operation. It is important to schedule a rekey operation correctly and estimate the potential trade-offs. We used the following test environment:

- ▶ One Microsoft Windows Server 2008 R2 Datacenter edition server with 2-port Brocade 8 Gbps FC host bus adapter (HBA)
- ▶ One IBM Storwize V7000 system
- ▶ One EE was used
- ▶ IBM Storwize V7000 had 24 drives organized in 3 x RAID 5 giving us 20 effective drives
- ▶ 30 GB LUN has been created on those drives and presented to the host through the EE
- ▶ Free workload generator Iometer was used to emulate the OLTP workload
- ▶ Workload patterns were 8, 16, and 32 KB block size, 70% read, 90% random
- ▶ 30 GB LUN has been formatted with New Technology File System (NTFS)
- ▶ Performance statistics have been collected on the Windows Server 2008 server with the Performance monitor tool for the object - Physical disk
- ▶ Performance values were collected every one second

The following parameters were the most interesting:

- ▶ Overall response time increase
- ▶ Read response time increase
- ▶ Decrease in the number of I/Os

Figure 5-4 shows the performance graph of the rekey operation.

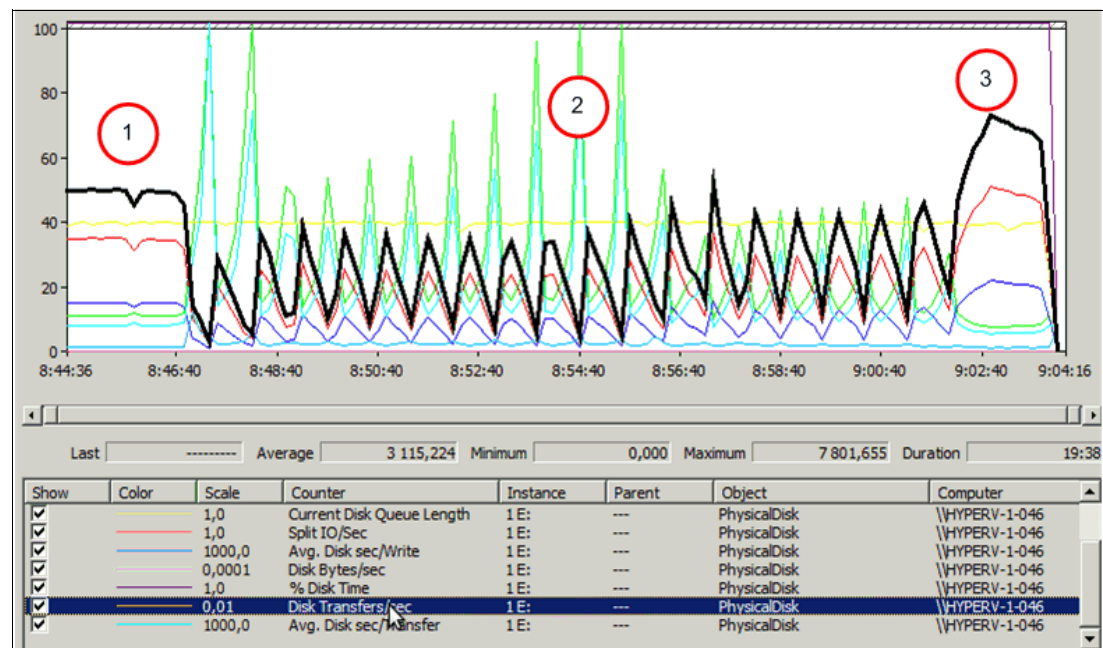


Figure 5-4 Performance graph of the rekey operation showing the number of I/Os per second

Focus on the three areas on that graph:

- ▶ Area 1 is the normal I/O flow with no rekey session started
- ▶ Area 2 is the rekey session started
- ▶ Area 3 is the rekey session finished and normal operation has resumed

The graph view shows significant fluctuation with many peaks and troughs. This fluctuation is explained by the phased nature of the rekey operation. There are multiple graphs in one

figure to demonstrate that all values were fluctuating, not only one. In our case, it took approximately 16 minutes to rekey the 30 GB LUN.

The black highlighted graph in Figure 5-4 on page 190 shows the I/Os to the LUN that is being encrypted. It is clear that approximately a two-time decrease exists during the rekey session. You can see a slight increase in the I/Os closer to the end of the operation, which can be explained by the ability of IBM Storwize V7000 to adapt to the workload. This capability is true for all types of workload patterns. We will demonstrate this capability later.

Figure 5-5 shows the fluctuations of the I/O response time during the rekey operation. There are three defined areas: before, during, and after the rekey operation. The graph shows the response time was approximately 10 ms in the beginning and had risen to an average of 25 - 30 ms during the rekey process.

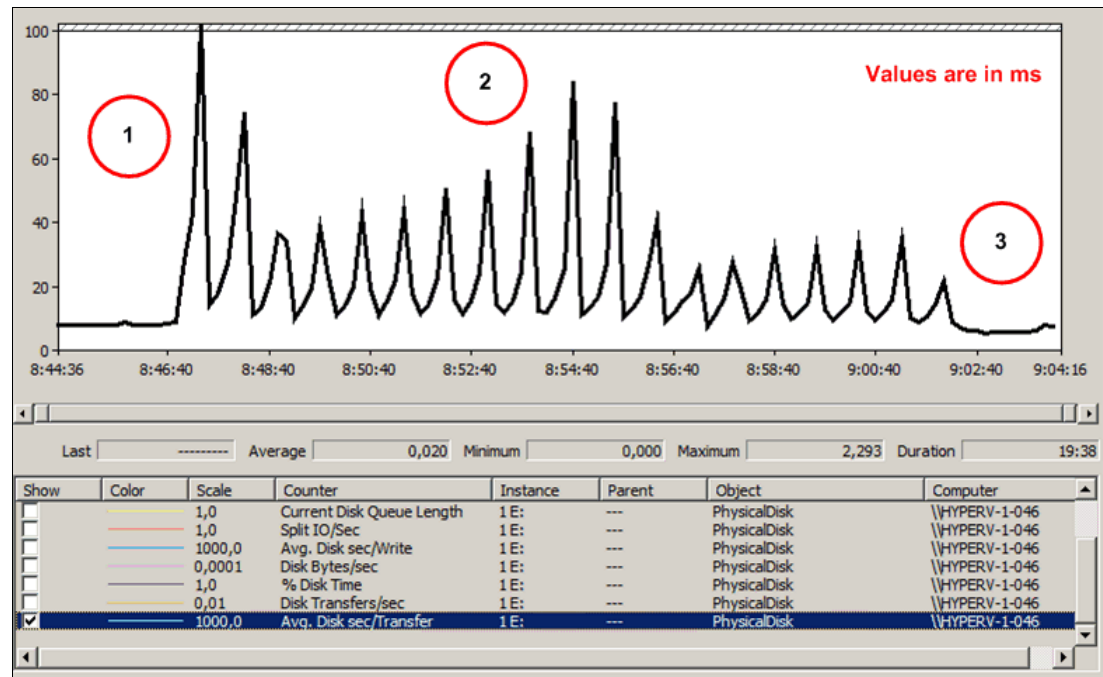


Figure 5-5 I/O response time of 8 KB OLTP workload during the rekey operation

Figure 5-6 on page 192 shows that, unlike the read operations, the response time of the write operations was not affected greatly. This performance can be explained by the excellent work of the cache algorithms of the IBM Storwize V7000. On the other graphs, there are clear areas of the rekey process running and not running.

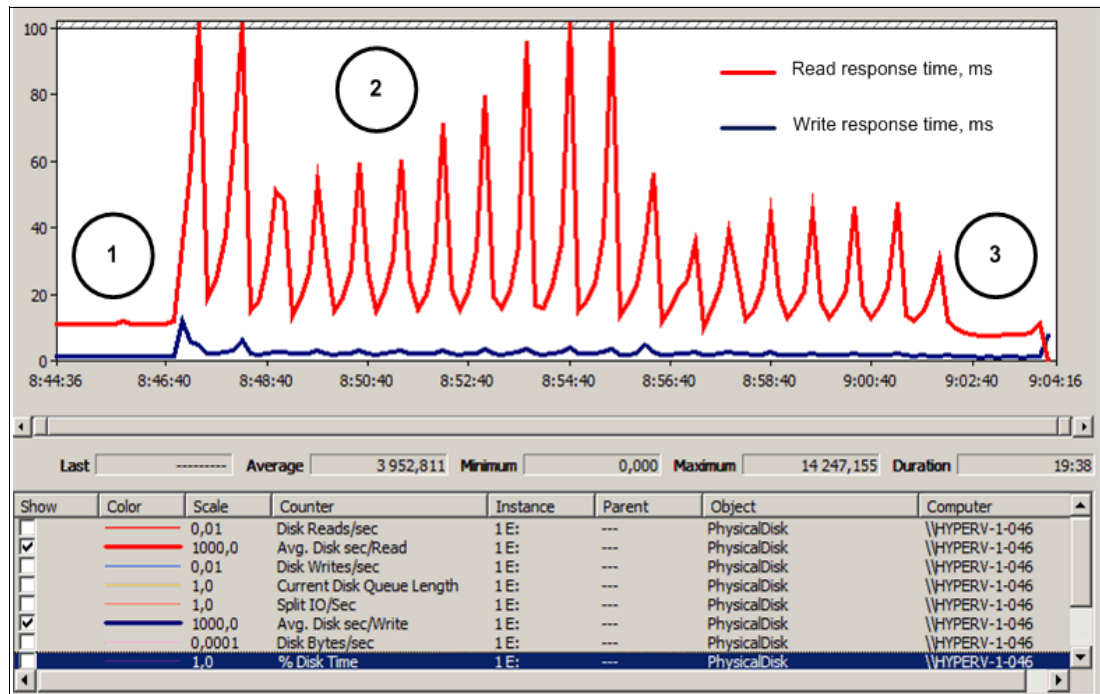


Figure 5-6 I/O read and write response time during the rekey operation with an 8 KB block size

Figure 5-7 shows the response time against the number of I/Os per second at the same time. It is clear that when the response time goes up, the number of I/Os goes down. This result happens when the EE writes the encrypted data to the LUN. Because the amount of data is sufficiently large, it takes time to destage it to the disk. At this time, the LUN blocks are busy, and the number of I/Os drops. In our case, every I/O drop was approximately 30 - 40 seconds.

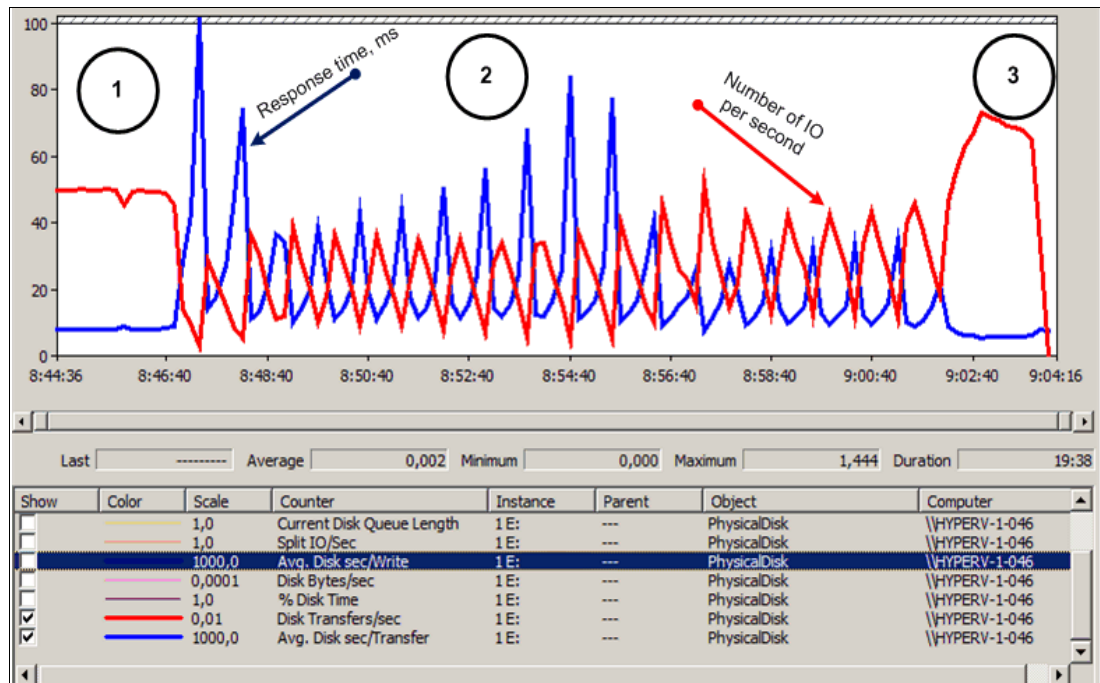


Figure 5-7 Number of I/Os and the response time with 8 KB blocksize

The workload with 16 KB block size had the same behavior during the rekey operation as the 8 KB workload, as shown in Figure 5-8. The absolute numbers differ slightly, but the fluctuating pattern remained. The rekey process took the same amount of time, so it can be considered an extremely stable and constant workload.

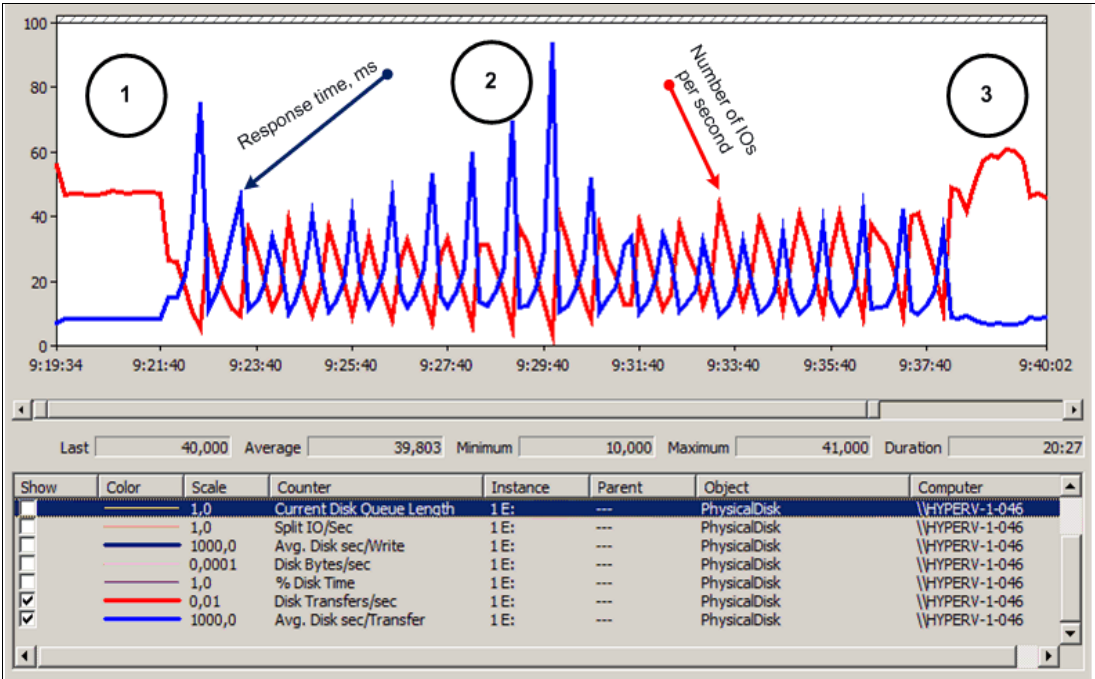


Figure 5-8 Response time and the number of I/Os per second for the 16 KB workload

In Figure 5-8, the 16 KB block size workload write operations were not affected by the rekey process and looked stable. Read response times fluctuated as before. There was also a slight drop in response times closer in the second half of the test, which can confirm the ability of the IBM Storwize V7000 to adapt to the workload during the same period of time.

Figure 5-9 on page 194 shows the read and write response times of the I/O during the rekey process with a 16 KB block size workload.

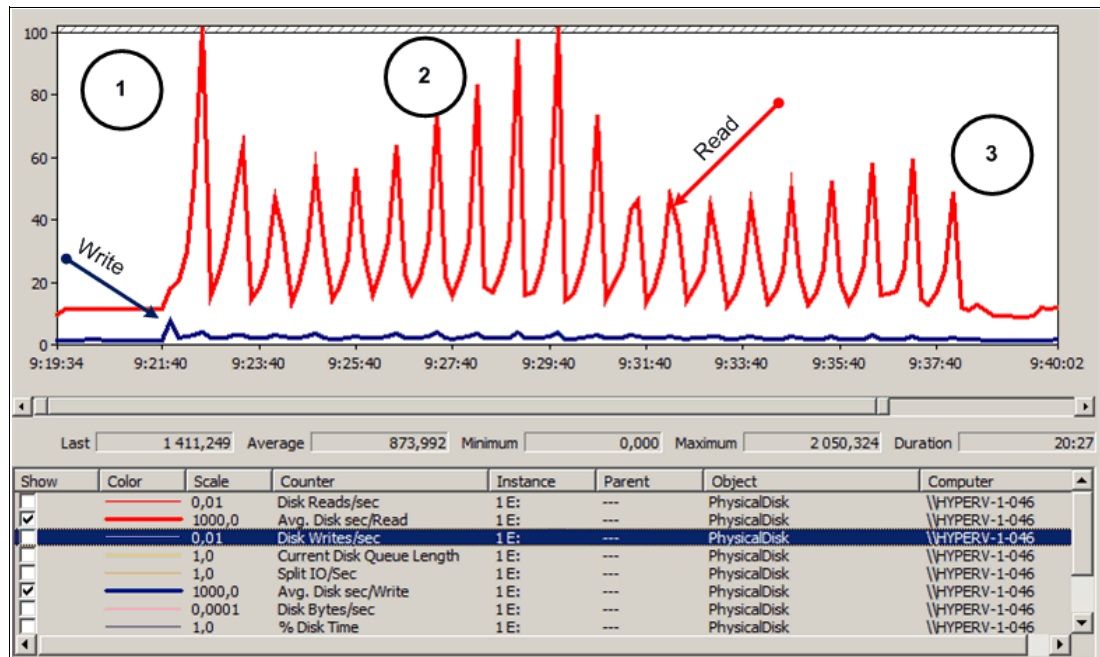


Figure 5-9 Read and write response times of the I/O during the rekey process with a 16 KB block size workload

The 64 KB workload pattern is typical for OLTP workloads. As shown in Figure 5-10, the behavior of the workload during the rekey process remained the same. As before, there are three areas and similar sequential fluctuations, and although the numbers have changed a little, the workload pattern is mostly the same.

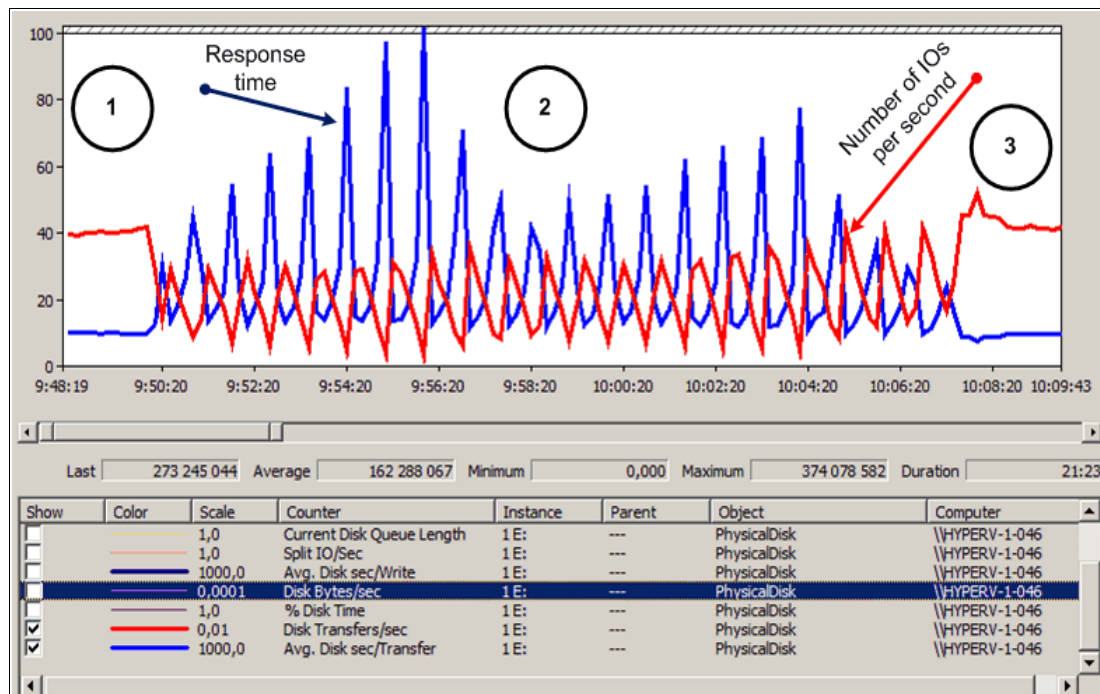


Figure 5-10 Response time and the number of I/Os for the 64 KB workload

As shown in Figure 5-11 on page 195, it is also true for the 64 KB workload pattern that write operations were not affected much. However, you can see the slight increase of the average

response time for read operations compared to the 8 KB and 16 KB workload, but it was higher in the beginning of the test without the rekey operation. It is more important to see that response times go back to the normal values right after the rekey process has completed. Notice that there are no delays in going back to the normal values right after the rekey process has completed, which means that IBM Storwize V7000 is capable of managing such a mixture of workloads on one LUN.

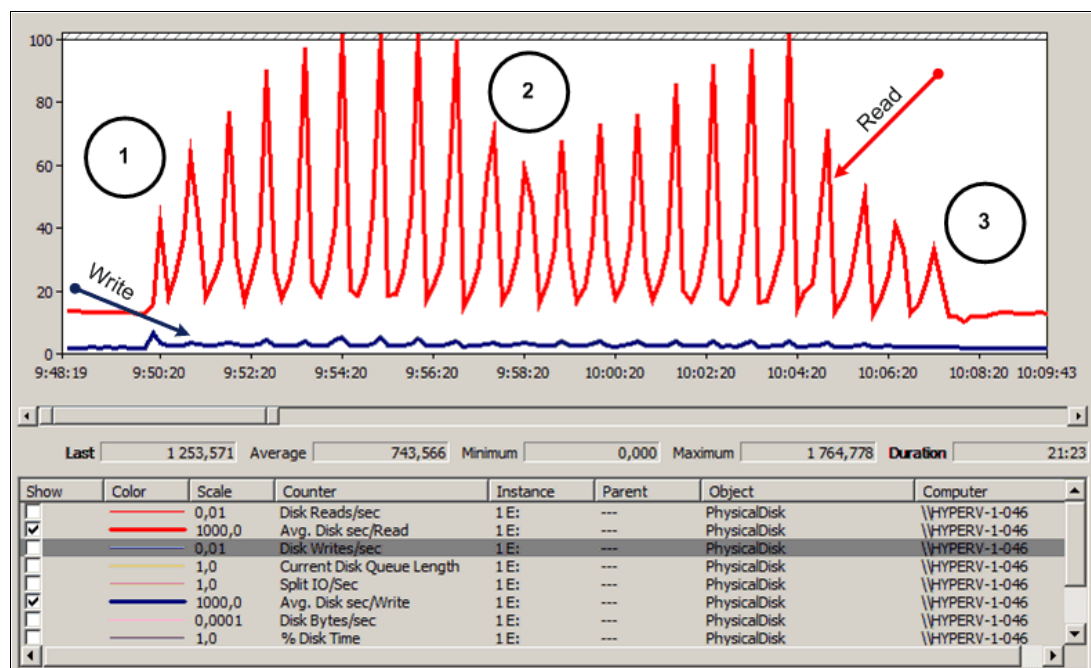


Figure 5-11 Read response time and write response time in 64 KB workload

We summarize our performance tests:

- ▶ A performance effect exists on the LUN that is being rekeyed.
- ▶ It is better to schedule rekey operations for a time with lower I/O activity on the LUN.
- ▶ Remember that the rekey operation can be a lengthy operation.
- ▶ Perform several tests in your environment *prior* to implementing the encryption solution to understand the influence of the rekey operations on your applications.
- ▶ New LUNs with no data can skip the first-time encryption operation to save time by either specifying **-disable_encexistingdata** or not specifying **-enable_encexistingdata** as the parameters for the **cryptocfg** command. But, remember that this is true for *new* LUNs only. If you do so for the LUNs with existing data, they will be corrupted.

5.3 Designing the encryption solution

Encrypting large amounts of data in a short time period requires significant processing power. New encryption algorithms are optimized for that, but there are still certain limitations that exist, which must be considered while designing the solution. Poor solution design can seriously affect the current data flow and can degrade the performance of current operations. Solutions that are based on the IBM Storwize V7000 disk system with IBM SAN32B-E4 switch or IBM384/768 directors are about encrypting a large amount of data in as short a period of time as possible.

5.3.1 Performance effect of encryption operations

We have described the performance influence of the rekey and first-time encryption operations. After implementing the encryption solution, the data starts to go through the EEs, which you need to consider. The expected latency of the response time increases by approximately 1 - 3 milliseconds (ms). So, if your current response times are quite long, consider either adding more disk drives to the existing LUNs, moving to a better performing disk system, or splitting the workload among several disk systems. For the OLTP workloads, the critical response times are 20 ms for reading operations and 5 ms for writing operations.

If you have response times that are close to these numbers and those LUNs are to be encrypted, optimize the workload and the configuration of the LUNs first.

5.3.2 Defining the correct number of encryption engines

The IBM SAN32B-E4 has these known limitations:

- ▶ Maximum of 96 Gbps for disk encryption
- ▶ 20 microseconds latency for the FC frame
- ▶ 32 x 8 Gbps FC ports

The IBM SAN 768\384 Encryption Blade has these known limitations:

- ▶ Maximum of 96 Gbps for disk encryption
- ▶ 20 microseconds latency for the FC frame
- ▶ 16 x 8 Gbps FC ports

The EG has these known limitations:

- ▶ Maximum of four encryption nodes per one EG
- ▶ Maximum of 16 EEs per one EG

Note that one IBM SAN768/384 director is *one* encryption *node* and can host four EEs. So, the maximum EG size is four IBM SAN768/384 directors, each hosting four Encryption Blades.

The EE has these known limitations:

- ▶ Maximum of 10 concurrent rekey or first-time encryption operations
- ▶ Maximum of two EEs per HA cluster
- ▶ Two Encryption Blades in one encryption node cannot be in one HA cluster

The following guidelines might be useful in sizing the encryption solution:

- ▶ Maximum encryption bandwidth of IBM SAN32B-E4 is 96 Gbps, which means that it can provide 12 full-speed 8 Gbps FC ports for encryption purposes. It is possible to have 12 LUNs running close to the full speed in one EE. If you plan to have at least six full-speed LUNs, it is better to increase the number of EEs.
- ▶ Maximum of 10 concurrent rekey and first-time encryption operations can take place in one EE. Remember this rule when you plan the DEK lifetime for the LUNs and add LUNs to the CTCs. A guideline is approximately 30 LUNs per one EE with the minimal DEK lifetime.
- ▶ It is better to use IBM SAN384/768 directors with Encryption Blades if you plan to have more than two EEs. This design simplifies the management, eliminates unnecessary cabling, and allows easy upgrades. Solutions with an HA cluster become easier in this case, and problem determination and resolution are simpler, too. And, you can have more EEs in one EG if you use IBM SAN directors.

5.3.3 Connecting the encryption engine

Because the EE maintains the FC traffic and performs the encryption, it can become a bottleneck and add an unnecessary inter-switch link (ISL) for the frames. Connecting the EE correctly is extremely important. The main rule is to connect the EE to the core switches or directors in the fabric, as shown in Figure 5-12.

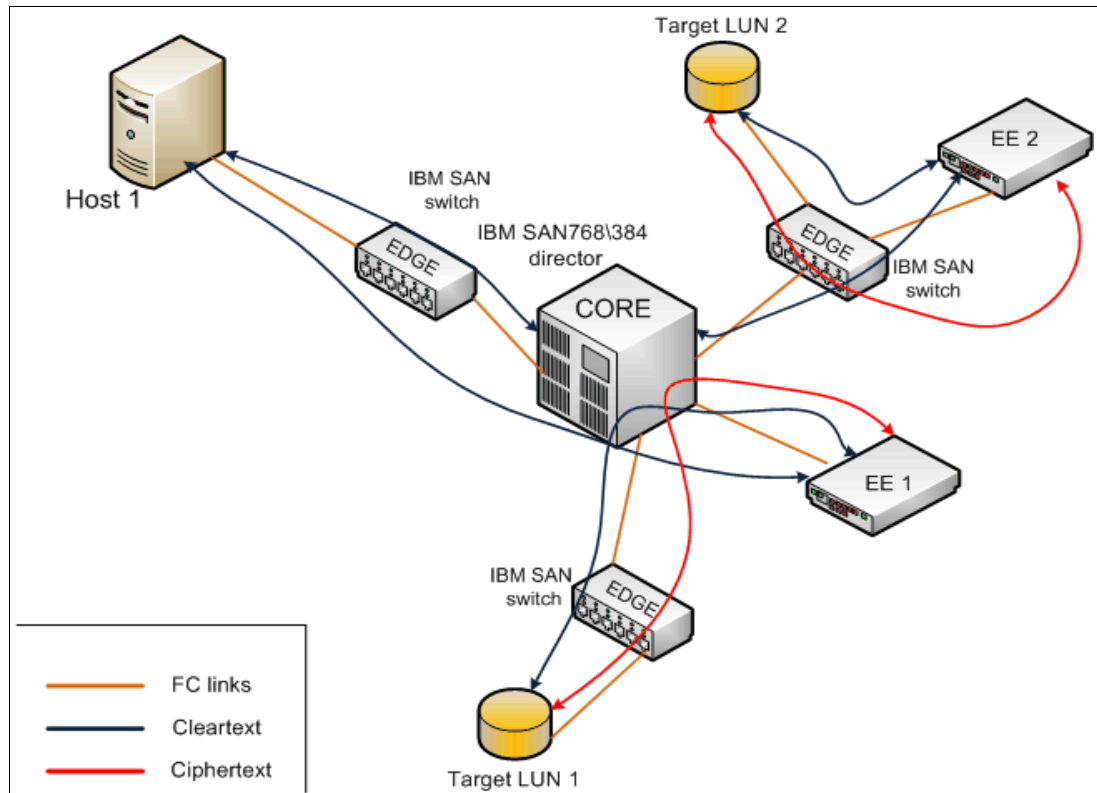


Figure 5-12 Correct and incorrect placement of the EE

Figure 5-12 shows two EEs that connect to the SAN. Encryption engine 1 (EE1) connects correctly to the core director of the SAN. The route from the host to EE1 is 3 hops and the edge switch is not overloaded with traffic to Target LUN 1. Encryption engine 2 (EE 2) connects to the edge switch and that makes the route to the host 4 hops. This connection overloads the edge switch with unnecessary traffic to Target LUN 2. Estimate the number of hops when deciding how you are going to connect the EEs.

5.4 Copy services in the encryption environment

IBM Storwize V7000 has feature-rich copy services capabilities. It is important to study the implementation of them with the IBM SAN32B-E4 and IBM SAN768/384 encryption. We describe the following copy services:

- ▶ IBM FlashCopy®
- ▶ Metro/Global Mirror solutions

We do not describe virtual disk mirroring in this section, because it is the internal state of the LUN and it does not interact with the EEs.

FlashCopy

There are three types of FlashCopies in IBM Storwize V7000:

- ▶ Snapshot
- ▶ Clone
- ▶ Backup

For more information about the process of invoking FlashCopy, refer to *Implementing the IBM Storwize V7000*, SG24-7938.

No matter which type of FlashCopy you use, there are two possible ways to use FlashCopy from the encryption point of view:

- ▶ Map the target volume using the existing CTCs, which means that you use the same target ports as you use for the source volume.
- ▶ Map the target volume using a newly created CTC, which means that you use other target ports.

Figure 5-13 shows the layout of the solution with FlashCopy connected through the existing CTC. To create this solution, you must add an additional initiator to the existing CTC, and the FlashCopy LUN has been discovered and assigned to it.

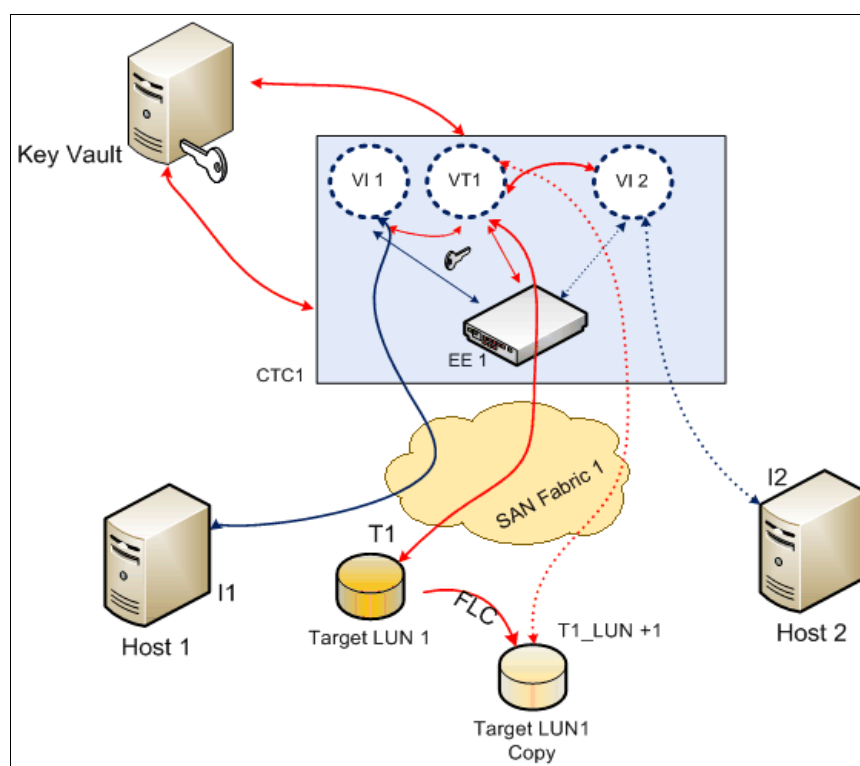


Figure 5-13 FlashCopy connected through the existing CTC

Figure 5-14 on page 199 shows an example of connecting the FlashCopy target LUN through the new CTC. FlashCopy does not use the existing target ports. FlashCopy uses the new CTC host, target port, and initiator port. The LUN is discovered through the new initiator port and connects to it.

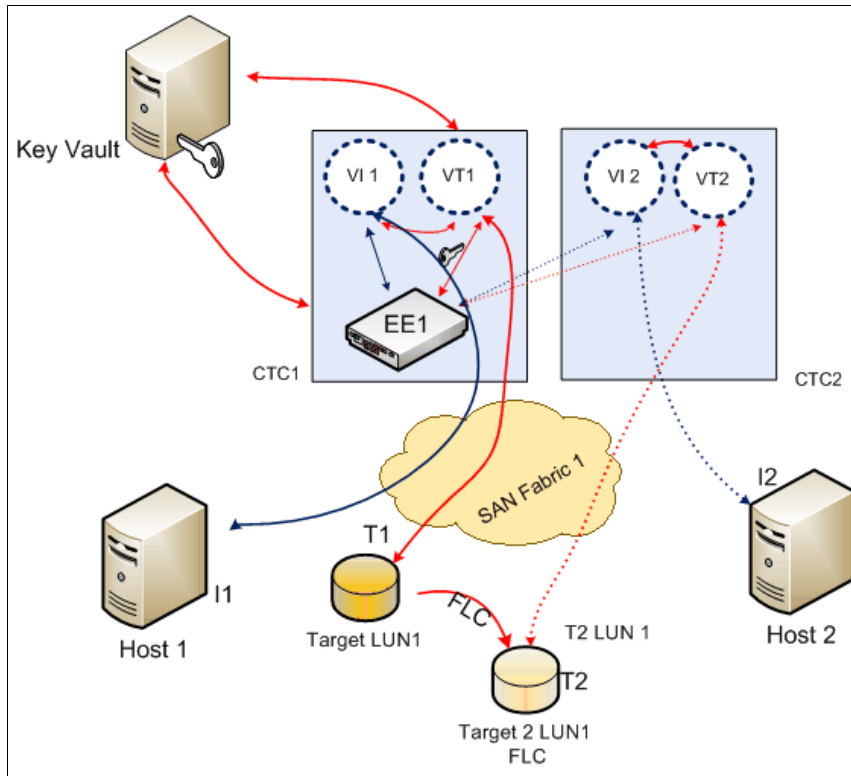


Figure 5-14 FlashCopy connected to the newly created CTC

We have used the second scenario in our environment. The FlashCopy target LUN has been mapped using the other target ports of the IBM Storwize V7000 and to another host.

We created a snapshot of the one of existing volumes. Figure 5-15 shows this snapshot.

Name	Status	Capacity	Storage Pool	UID
win2k8-1_Lun0	✓ Online	30.0 GB	mdiskgrp0	600507680185853FF000000000000000
win2k8-1_Lun1	✓ Online	1,000.0 GB	mdiskgrp0	600507680185853FF0000000000000002
win2k8-1_Lun1_01	✓ Online	1,000.0 GB	mdiskgrp0	600507680185853FF0000000000000005
win2k8-1_Lun1_02	✓ Online	1,000.0 GB	mdiskgrp0	600507680185853FF0000000000000007
win2k8-2_Lun0	✓ Online	50.0 GB	mdiskgrp0	600507680185853FF0000000000000001
win2k8-2_Lun0_01	✓ Online	50.0 GB	mdiskgrp0	600507680185853FF0000000000000006

Figure 5-15 FlashCopy target volume created

The FlashCopy target volume is mapped to the host, and the snapshot volume is mapped to the other host (Win2k8-2). It is highlighted on Figure 5-16.

Host Name	SCSI ID	Volume Name	Volume Unique Identifier	I/O Group ID
Win2k8-1	0	win2k8-1_Lun0	600507680185853FF0000000000000000	0
Win2k8-1	1	win2k8-1_Lun1	600507680185853FF0000000000000002	0
Win2k8-2	250	win2k8-2_Lun0	600507680185853FF0000000000000001	0
Win2k8-2	0	win2k8-1_Lun1_01	600507680185853FF0000000000000005	0

Figure 5-16 FlashCopy target LUN is mapped to the host

You will get an error message if you try to access the snapshot of the encrypted LUN without it being added to the EE. See Figure 5-17 as an example. At this point, the LUN is accessed directly and is not going through the EE. Because the target volume is the byte-to-byte block-level copy of the source volume, it is encrypted and cannot be accessed directly without merging it into the EE.

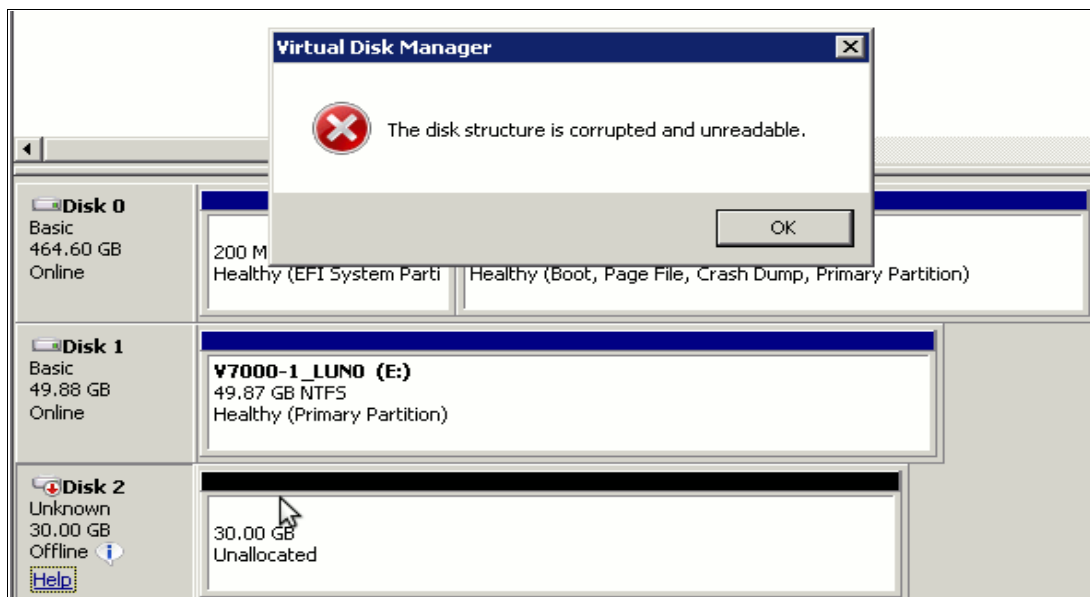


Figure 5-17 Error when connecting the snapshot of the encrypted LUN with direct access

In Example 5-19, the FlashCopy target LUN is discovered in the newly created CTC with the name CTC_2B. At this time, the LUN has been discovered with an Unknown key ID state of the encryption key, which is a common situation for any discovered and encrypted LUNs. However, this LUN is encrypted already, because the ID of the encryption key is also discovered. If the LUN is not encrypted, the key ID state is Key ID not available. You can read this state in the current example for the LUN numbered 0xfa.

Example 5-19 Discovering the FlashCopy target LUN

```

IBM_SAN384B:FID128:admin> cryptocfg --discoverLUN CTC_2B
Container name:          CTC_2B
Number of LUN(s):       2
Host:                   10:00:00:05:1e:c7:6b:a1
LUN number:             0x0
LUN serial number:      600507680185853FF0000000000000005
LUN connectivity state: Connected
Key ID state:           Unknown
Key ID:                 a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36
Host:                   10:00:00:05:1e:c7:6b:a1
LUN number:             0xfa
LUN serial number:      600507680185853FF0000000000000001
LUN connectivity state: Connected
Key ID state:           Key ID not available
Operation succeeded.
  
```

Example 5-20 shows the command for adding the FlashCopy target LUN. It is similar to adding the regular LUN, but with two important parameters:

- ▶ **-lunstate encrypted**
- ▶ **-encrypt**

You *must* always use these parameters to add an already encrypted LUN to the CTC. Table 5-1 on page 181 describes these parameters.

Example 5-20 Adding the FlashCopy target LUN to the CTC

```
IBM_SAN384B:FID128:admin> cryptocfg --add -LUN CTC_2B 0x0 10:00:00:05:1e:c7:6b:a1
20:00:00:05:1e:c7:6b:a1 -lunstate encrypted -encrypt
Operation succeeded.
```

Important: Always use the **-lunstate encrypted** and **-encrypt** parameters when adding FlashCopy target or Metro\Global Mirror target LUNs. If you do not use these parameters, encryption key blocking can occur in the Read-only state and prevent further access to the source LUN.

In Example 5-21, we show the CTC view after the FlashCopy target has been added.

Example 5-21 CTC view after the FlashCopy target has been added

```
IBM_SAN384B:FID128:admin> cryptocfg --show -container CTC_2B -stat
Container name:      CTC_2B
Type:               disk
EE node:            10:00:00:05:1e:94:3a:00
EE slot:            1
EE hosting container: current
Target:             50:05:07:68:02:20:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:         010100
VT:                20:00:00:05:1e:c1:13:13 20:01:00:05:1e:c1:13:13
VT PID:            011801
Number of host(s):  1
Number of rekey session(s): 0
Host:           10:00:00:05:1e:c7:6b:a1 20:00:00:05:1e:c7:6b:a1
Host PID:          010800
VI:               20:0a:00:05:1e:c1:13:13 20:0b:00:05:1e:c1:13:13
VI PID:           011d01
Number of LUN(s):  1
LUN number:     0x0
LUN type:          disk
LUN serial number: 600507680185853FF000000000000005
Encryption mode: encrypt
Encryption format: native
Encrypt existing data: disabled
Rekey:             disabled
Internal EE LUN state: Encryption enabled
Encryption algorithm: AES256-XTS
Key ID state:    Read write
New LUN:        No
Key ID:         a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36
Operation succeeded.
```

Table 5-3 shows the important points to check to ensure that everything is correct in Example 5-21 on page 201.

Table 5-3 Key points to check

Key points to check	Expected status
LUN number and LUN serial number	The same as they are in the IBM Storwize V7000 host mapping
Encryption mode	Encrypt
Internal EE LUN State	Encryption enabled
Key ID state	Read write
New LUN	No
Key ID	The same as the key ID of the source LUN. See Example 5-22 on page 202.

As stated in Table 5-3, the key ID of the FlashCopy target LUN must be the same as the key ID of the source LUN. Example 5-22 shows the output of the CTC with the FlashCopy source LUN. It is mapped to a separate host port and it has a separate LUN serial number, but it has the same key ID.

Example 5-22 Discovering the key ID of the source LUN (output is truncated)

```

SAN32B-E4-1:admin> cryptocfg --show -container CTC_1 -stat
Container name:      CTC_1
Type:               disk
EE node:            10:00:00:05:1e:54:17:10
EE slot:            0
EE hosting container: current
Target:             50:05:07:68:02:30:a7:fe 50:05:07:68:02:00:a7:fe
Target PID:         030400
VT:                20:00:00:05:1e:54:17:0c 20:01:00:05:1e:54:17:0c
VT PID:            012001
Number of host(s):  1
Number of rekey session(s): 0
Host:            10:00:00:05:1e:c7:6b:8a 20:00:00:05:1e:c7:6b:8a
Host PID:          030900
VI:               20:08:00:05:1e:54:17:0c 20:09:00:05:1e:54:17:0c
VI PID:           012401
LUN number:       0x1
LUN type:         disk
LUN serial number: 600507680185853FF0000000000000002
Encryption mode:  encrypt
Encryption format: native
Encrypt existing data: disabled
Rekey:           disabled
Internal EE LUN state: Encryption enabled
Encryption algorithm: AES256-XTS
Key ID state:      Read write
New LUN:         No
Key ID:           a1:8c:82:51:9b:6d:c2:59:7e:45:cd:59:34:74:83:36
Operation succeeded.

```

If your output is similar to the highlighted output in Example 5-23 in the status of the CTC, adding the FlashCopy target LUN has completed, but with errors. If your output is similar, remove the LUN from all the CTCs immediately.

Example 5-23 FlashCopy target LUN has been added incorrectly (output is truncated)

```
IBM_SAN384B:FID128:admin> cryptocfg --show -container CTC_1B -stat
VI PID:                012501
Number of LUN(s):      1
LUN number:            0x1
LUN type:              disk
LUN serial number:     600507680185853FF000000000000003
Encryption mode:    cleartext
Encryption format:     native
Encrypt existing data: disabled
Rekey:                 disabled
Internal EE LUN state: Disabled (Found metadata while LUN is clear text)
Encryption algorithm: None
Key ID state:      Key ID not Applicable
New LUN:          No
Operation succeeded.
```

Never try to enable encryption on a FlashCopy target LUN that been added to the CTC with cleartext status. This status will lock the existing encryption key of the source LUN to the read-only state, as shown in Example 5-24.

Example 5-24 Encryption key has been locked to the read-only state

```
LUN number:            0x0
LUN type:              disk
LUN serial number:     600507680185853FF0000000000000000
Encryption mode:       encrypt
Encryption format:     native
Encrypt existing data: disabled
Rekey:                 disabled
Internal EE LUN state: Read only (Current key is in read only state)
Encryption algorithm:  AES256-XTS
Key ID state:          Read only
New LUN:               No
Key ID:            47:62:48:66:9d:f8:4a:76:91:28:6c:f1:9a:a4:cb:e2
```

The encryption key with ID 47:62:48:66:9d:f8:4a:76:91:28:6c:f1:9a:a4:cb:e2 has been placed in read-only mode because of the manipulation of the FlashCopy target LUNs. This mode prevents any future access to the LUN from most of the supported operating systems. If this LUN is the part of the Windows Server virtual disk, AIX volume group, or any other virtual group of any kind, this situation will be treated as a disk failure by the virtual disk group engine and the failover policies of the group will apply.

Important: The rekey policy of the FlashCopy target LUN must *always* be set to **disabled**. Never try to rekey the FlashCopy target LUN. Rekeying the FlashCopy target LUN will lock the source LUN encryption key to a read-only state and make the source LUN inaccessible.

Important: The correct way is to set the rekey policy of the FlashCopy source LUN to the manual (disabled) state before making any snapshots or clones. The rekey process can be initiated manually when required. Before starting the rekey operation on the FlashCopy source volume, remove all existing FlashCopy target LUNs from the hosts and CTCs, remove the FlashCopy mappings in the IBM Storwize V7000, and then, perform the rekey. FlashCopy mappings must be created again after the rekey operation, and host mappings and the CTC configuration must be done again. If you do not follow this guideline, it might result in locking the encryption keys of the target LUNs, and likely the source LUNs to a read-only state, which prevents any further access to the data. During the rekey operation, the existing keys are placed in a read-only state and the new keys generated.

You must treat FlashCopy target LUNs as LUNs with multiple paths in the IBM Storwize V7000 environment, and you must follow the same guidelines. See 4.2, “General prerequisites for encryption” on page 95 for reference. After successfully adding the FlashCopy target LUN to all CTCs, remember to commit the changes with the **cryptocfg --commit** command.

5.4.1 Metro/Global Mirror

You use the Metro/Global Mirror copy functions of the IBM Storwize V7000 to build disaster recovery solutions. Managing the Metro/Global Mirror target and source volumes is the same as managing the FlashCopy volumes. The difference between the Metro/Global Mirror solutions is in the usage of the key vault. You must design the solution to protect the key vault and make it available after a disaster occurs. The surviving part of the disaster recovery solution must always be able to access the key vault. There are two ways to achieve this result:

- ▶ Using a copy of the key vault on the recovery site
- ▶ Using a copy of the key vault at a third site

The choice of the scenario depends on the disaster recovery solution to be implemented. Usage of the copy of the key vault on the recovery site means that all data activity is moved completely to the recovery site in the case of a disaster.

In Figure 5-18 on page 205, there are two independent key vaults in the solution: one key vault on the main site and the other key vault, which is a copy of the main key vault, on the recovery site. Key vault data must be backed up, moved to the recovery site, and restored in the recovery key vault. This process must happen every time that any DEK changes have taken place on the LUNs that are to be protected. EEs on the main site and on the recovery site can be in one EG forming the DEK or HA cluster configuration, but there must be *separate* EGs on the main and recovery sites.

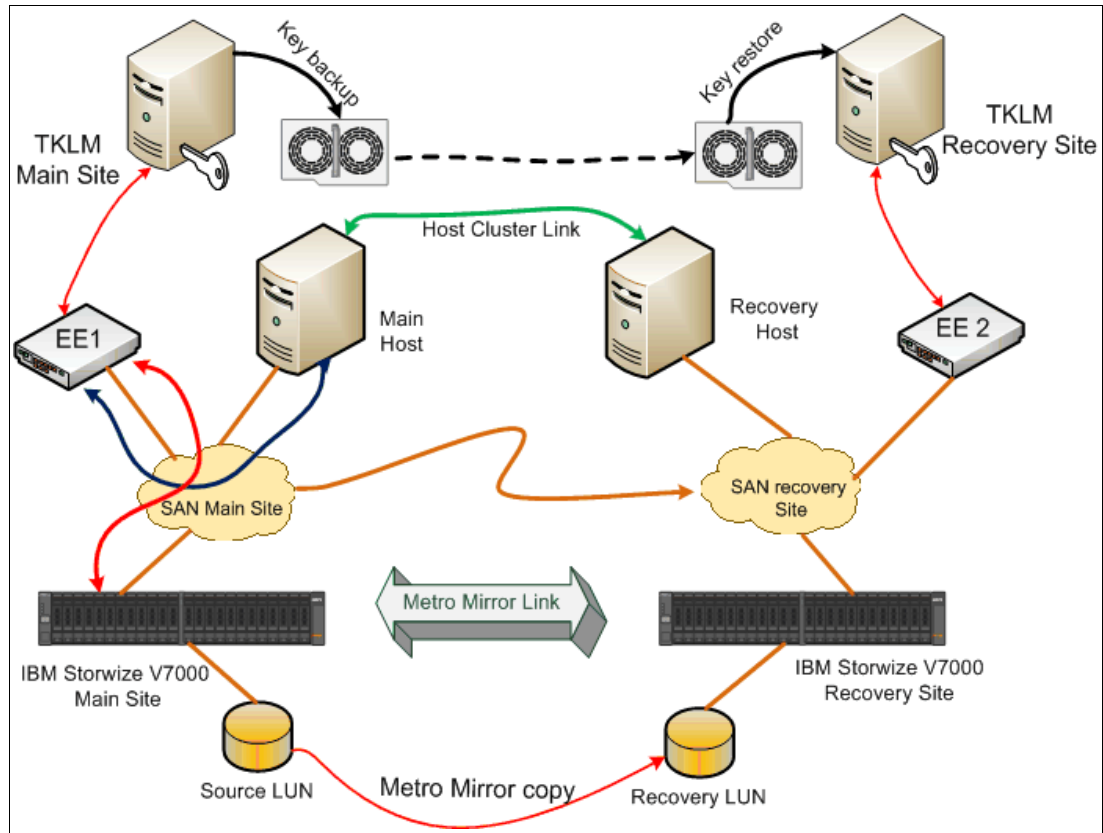


Figure 5-18 Metro Mirror disaster recovery solution in an encryption environment

If a disaster occurs at the main site, all activities move to the recovery site. This move is either automatic or manual, depending on the recovery solution that has been implemented, as shown in Figure 5-19 on page 206.

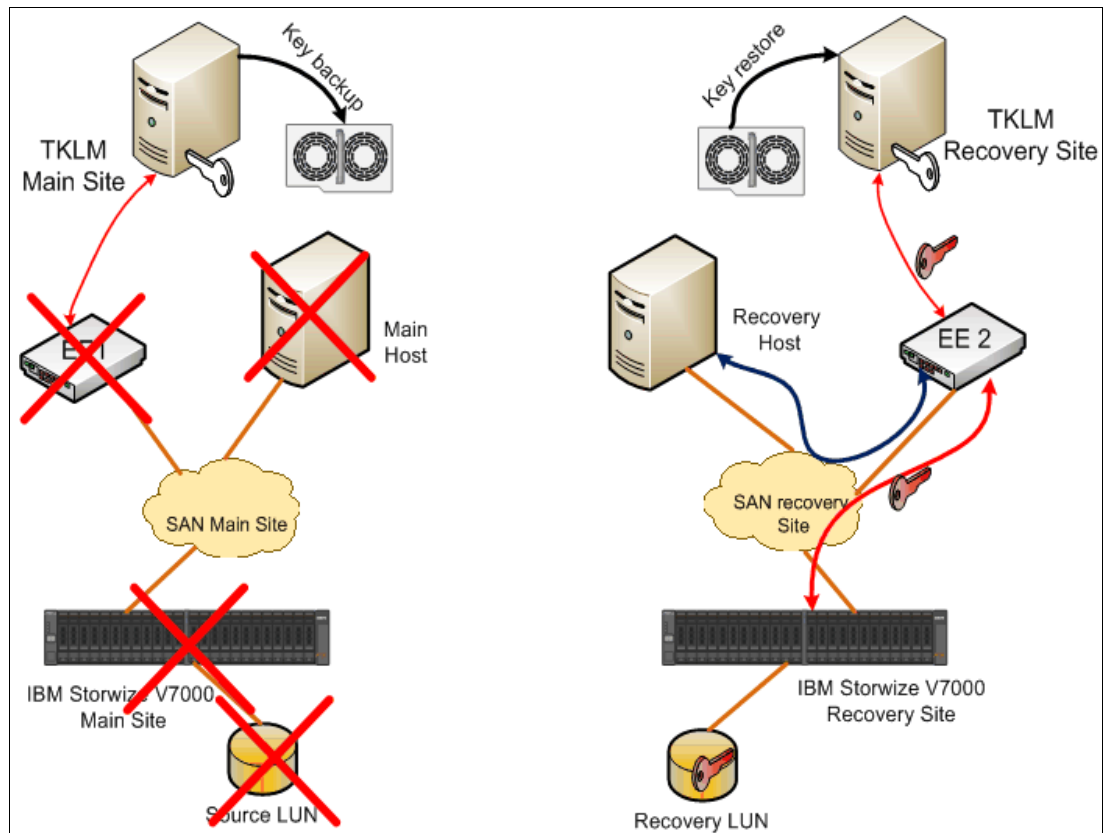


Figure 5-19 Failover of the main site in an encryption environment

Disaster recovery: The disaster recovery software and the implementation of the disaster recovery solutions are beyond the intended scope of this book.

When the data starts to be managed at the recovery site, the EEs read the encrypted LUNs, retrieve the information about the key IDs, request the key IDs from the key vault, and use the key IDs to access the data.

Because the design and implementation of any disaster recovery solution with the IBM SAN32B-E4 and IBM SAN768/384 can be complex, we advise that you engage IBM Professional Services.

5.5 External storage virtualization

IBM Storwize V7000 is capable of virtualizing not only internal drives but externally connected disk systems from third-party vendors. The implementation of this type of a solution does not differ from an encryption point of view; however, you must consider one important area.

You must always connect the LUNs from the third-party storage systems to the IBM Storwize V7000, virtualize them, and then perform encryption on the IBM Storwize V7000 virtual volumes. If you perform the encryption of the LUN from the third-party storage system first and then attach it to the IBM Storwize V7000, it will be corrupted if it is attached in managed mode, or not accepted if it is attached in image mode. Figure 5-20 on page 207 shows the correct attachment.

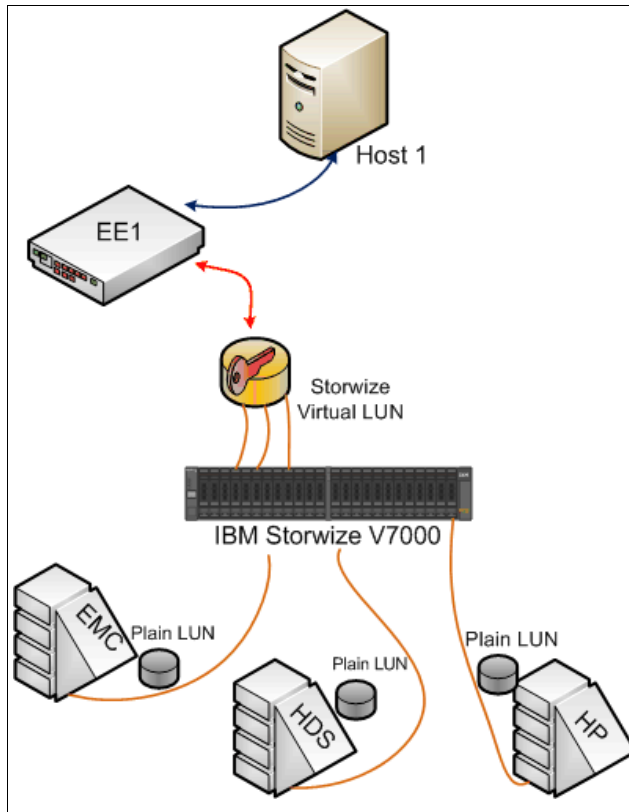


Figure 5-20 Correct attachment of external storage

Figure 5-20 shows the correct implementation of the encryption solution with the third-party storage systems. Logical volumes (plain LUNs) are connected to the IBM Storwize V7000 first, and then, the virtual volume is created and encrypted.

Figure 5-21 on page 208 shows an *incorrect* implementation of the encryption solution with third-party storage systems. Logical volumes are incorrectly encrypted first and then connected to the IBM Storwize V7000. This kind of implementation leads to data corruption and must never be used.

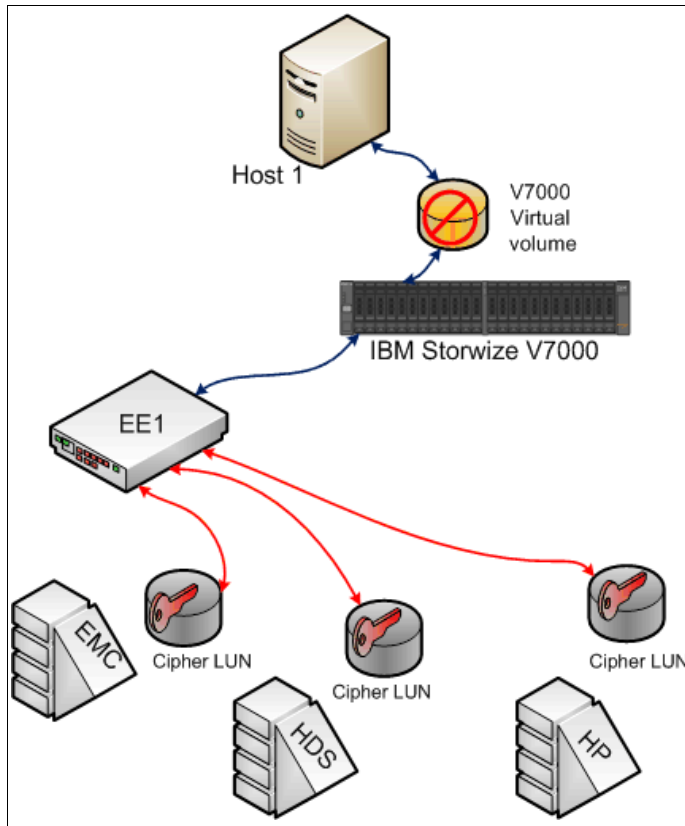


Figure 5-21 Incorrect connection of the external storage

5.5.1 Thin-provisioned volumes

At the time of writing this book, Fabric Operating System 7.0 does not support encryption on thin-provisioned volumes. Because of the nature of the first-time encryption and rekey operations, thin-provisioned volumes will become fully provisioned.



Maintenance and troubleshooting

In this chapter, we describe the major aspects of the maintenance of and troubleshooting the IBM SAN32B-E4 switch and IBM SAN768/384 director in the solution with IBM Storwize V7000. We explain the encryption part of the functionality. It is not our intent to describe the regular maintenance tasks. We assume that you are experienced in IBM storage area network (SAN) switches and directors. If you need additional reference for the procedures that are mentioned in this chapter, use the Fabric OS Administration guides.

We describe the following points:

- ▶ Firmware upgrades
- ▶ Encryption engine (EE) removal and replacement
- ▶ Master key backup and restore
- ▶ Adjustments and settings
- ▶ Troubleshooting guidelines

6.1 Firmware upgrades

The firmware upgrade process of the IBM SAN32B-E4 switch or IBM SAN768/384 director is the same as on any other IBM SAN b-type Fibre Channel (FC) switch. However, because of the encryption capabilities of the IBM SAN32B-E4 and IBM SAN768/384, there are several peculiarities and items to which you need to pay attention. In this section, we assume that you are familiar with the regular firmware upgrade process, and we will not explain the basics of it. Use the IBM Support site documentation for reference for the regular firmware upgrade process. We only focus on the important points, which can be vital to the data and the I/O flow:

- ▶ The IBM SAN32B-E4 and IBM SAN768/384 with Encryption Blades firmware upgrade is an I/O disruptive upgrade, because the EEs and backplanes are reset after the firmware download/upgrade.
- ▶ We suggest that you perform the firmware upgrade in a high-availability (HA) cluster environment so that the encryption I/O traffic is not disrupted. With only one EE maintaining the encryption I/O, you will always have I/O interruption during a firmware upgrade. To avoid this situation, you must back up the existing EE with the other EE, forming an HA cluster.
- ▶ In an HA cluster, perform the firmware upgrade one node at a time, so the other node will take over the Crypto Target Containers (CTCs)/logical unit numbers (LUNs) and I/O during the firmware upgrade.
- ▶ In the data encryption key (DEK) cluster, manually move the CTC to another node in the cluster before upgrading the firmware, and then move it back.

The following general guidelines are for a firmware upgrade of the IBM SAN32B-E4 and IBM SAN768/384 with Encryption Blades in encryption groups (EGs), HA clusters, and DEK clusters:

- ▶ Upgrade one node at a time.
- ▶ Do not perform a firmware upgrade when rekey operations and first-time encryption operations are underway.
- ▶ Do not start any manual rekey operations and first-time encryption operations during the firmware upgrade process for all the nodes in the HA/DEK cluster.

Follow these guidelines for the firmware upgrade of IBM SAN32B-E4 and IBM SAN768/384 with Encryption Blades that are deployed in a DEK cluster with two HA clusters:

- ▶ Upgrade nodes in one HA cluster at a time.
- ▶ Within an HA cluster, upgrade one node at a time.

Follow these guidelines for the firmware upgrade of IBM SAN32B-E4 and IBM SAN768/384 with Encryption Blades that are deployed in a DEK cluster with no HA cluster (each node hosting one path):

- ▶ Upgrade one node at a time.
- ▶ In the case of active/passive arrays, upgrade the node, which hosts the passive path first. Upgrade the node that hosts the active path next. The host multipath I/P (MPIO) ensures that I/O fails over and fails back from active to passive and back to active during this firmware upgrade process.
- ▶ In the case of active/active arrays, the upgrade order of nodes does not matter, but you still must upgrade one node at a time. The host MPIO ensures that I/O fails over and fails back from one active path to another active path during this firmware upgrade process.

Important: All nodes in an EG must be at the same firmware level before starting a rekey or first-time encryption operation.

A firmware consistency check for Fabric OS V6.4.0(x) and later is enforced in an EG if any of the Version 6.4.0(x) features are enabled. If any Fabric OS V6.4.0(x) feature is in an enabled state, any firmware download to Fabric OS v6.3.x or earlier is blocked:

- ▶ Do not try registering a node running Fabric OS v6.3.x or earlier to an EG when all nodes are running Fabric OS V6.4.0(X) with one or more Fabric OS V6.4.0(X) features enabled.
- ▶ Disable all Fabric OS V6.4.0(X) features before ejecting a node running Fabric OS V6.4.0(X) and registering the node as a member of an EG with nodes running Fabric OS v6.3.x or earlier.

A firmware consistency check for Fabric OS V6.4.1 or later is enforced if the key vault is set to Tivoli Key Lifecycle Manager (TKLM). When firmware consistency check is enabled, any firmware download to any Fabric OS V6.4.0 or earlier is blocked.

- ▶ Do not try registering a node running Fabric OS V6.4.0 or earlier to an EG when all nodes are running Fabric OS v6.4.1 or later and the key vault is set to TKLM.
- ▶ After ejecting a node running Fabric OS v6.4.1 or later, set the key vault to any key vault other than TKLM before registering the node as a member of an EG with nodes running Fabric OS v6.4.0 or earlier.

Reboot: Changing the key vault type (`cryptocfg --set -key vault`) is a disruptive operation that requires you to reboot the node, or both Control Processors in the case of an IBM SAN768/384.

6.2 Master key maintenance

The master key is a Key Encryption Key (KEK) that is used to encrypt and decrypt DEKs when storing DEKs in the key vaults. One master key exists per EG. Therefore, all node EEs within an EG use the same master key to encrypt and decrypt the DEKs. A master key must be generated by the group leader EE. The master key can be generated once by the group leader and then propagated to the other members of an EG.

Back up the master key: It is important to back up the master key, because if the master key is lost, none of the DEKs can be restored and none of the encrypted data can be decrypted.

Only the active master key can be backed up, and multiple backups are advised. The master key can be backed up to any of the following devices or locations:

- ▶ A file as an encrypted key.
- ▶ The key vault as an encrypted key record.
- ▶ A set of recovery smart cards. This option is only available if the switch is managed by the Data Center Fabric Manager (DCFM) workstation, and if a card reader is available for attachment to the workstation.

6.2.1 Backing up the master key to a file

Example 6-1 describes the process of saving the master key to the file.

Example 6-1 Saving the master key to the file

```
SAN32B-E4-1:admin> cryptocfg --exportmasterkey -file
Enter passphrase:
Confirm passphrase:
Master key file generated.
```

Example 6-1 shows that the master key has been saved to the file and that a passphrase is necessary to protect the master key. Be sure to remember to keep the passphrase documented, because it is needed to restore the master key later. You can check it with the command in Example 6-2.

Example 6-2 Checking the master key ID that is ready for export

```
SAN32B-E4-1:admin> cryptocfg --show -mkexported_keyids\
e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5

e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5
Operation succeeded.
```

Example 6-2 shows that the master key with the key ID of e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5 is ready for export. The key ID must be in the format that is displayed in the output of the **cryptocfg --show -localEE** command. This command is valid on any node that connects to the key vault.

After saving the master key to a file, you must transfer it to the host or to the certified USB stick to be safely stored.

Important: File Transfer Protocol (FTP) is not supported for transferring the master key file.

Export the master key to a Secure Copy Protocol (SCP)-capable external host, as shown in Example 6-3.

Example 6-3 Exporting the master key to the host

```
SAN32B-E4-1:admin> cryptocfg --export -scp -currentMK 10.18.228.46 administrator
bes1_mk.mk
administrator@10.18.228.46's password:
Operation succeeded.
```

Example 6-3 shows the process of exporting the saved master key to the host:

- ▶ The IP address of the host master key file to be stored is 10.18.228.46.
- ▶ The administrator is the login of the user that is allowed to have SCP connections.
- ▶ The bes1_mk.mk file is the exported master key to be stored on the remote host.
- ▶ The password is entered in interactive mode.

After the master key file is exported to the host, it can be viewed with a text editor. The contents of the file are hardly human-readable and depend on the text editor that is used, but you can find the name of the EG where the file was generated.

6.2.2 Backing up the master key to the smart cards

A card reader must be attached to the SAN Management application PC to complete this procedure. Recovery cards can only be written once to back up a single master key. Each master key backup operation requires a new set of previously unused smart cards.

OS: Microsoft Windows operating systems do not require smart card drivers to be installed separately; the driver is bundled with the operating system. However, you must install a smart card driver for Linux and Solaris operating systems.

The key is divided among the cards in the card set, up to 10. The quorum of cards that is required to restore the master key must be less than the total number of the cards in the set, and no greater than five cards. For example, when the master key is backed up to a set of three cards, a quorum of any two cards can be used together to restore the master key. When the master key is backed up to a set of 10 cards, a quorum size of up to five cards can be configured for restoring the master key. Backing up the master key to multiple recovery cards is the recommended and most secure option.

Important: When you write the key to the card set, ensure that you write the full set without canceling. If you cancel, all previously written cards become unusable, and you will need to discard them and create a new set.

Saving the master key to the smart cards is supported with DCFM only. Follow these steps:

1. Select **Configure** → **Encryption** from the menu task bar. The Encryption Center window opens.
2. Select a group from the Encryption Center Devices table. Then, select **Group** → **Security** from the menu task bar, or right-click a group and select **Security**. The Encryption Group Properties window opens with the Security tab selected.
3. Select **Backup Master Key** as the Master Key Action.

The Backup Master Key for Encryption Group window opens, as shown in Figure 6-1 on page 214.

Backup Master Key for Encryption Group - support_TKLM

Backup Destination: A Recovery Set of Smart Cards

You will need a card reader attached to the management Station

1) Select Recovery Card Set Size: 1

2) Insert each card, in turn, into the card reader and wait for its ID to appear below.

Card Serial #:

3) Enter card assignment information. First and Last names are required.

First Name: Last Name: Notes:

4) Enter card password below.

Card Password: Case sensitive, 6-64 characters

Re-type Password:

5) Write Card

Completed	Group Card #	Card ID	First Name	Last Name	Notes

Status: Connecting to the smart card ...

OK Cancel Help

Figure 6-1 Backing up the master key to smart cards

4. Enter this information:
 - a. For Backup Destination, select **A Recovery Set of Smart Cards**.
 - b. Select the Recovery Card Set size.
 - c. Insert the first blank card and wait for the card serial number to appear in the Card Serial # field. Run the additional cards that are needed for the set through the reader. As you read each card, the card ID displays in the Card Serial # field. Be sure to wait for the ID to appear.
 - d. Enter the mandatory First Name and Last Name of the person to whom the card is assigned.
 - e. Enter a Card Password and re-enter the password for verification.
 - f. Record and store the password in a secure location.
 - g. Click **Write Card**.
5. The window prompts you to insert the next card, up to the number of cards specified in the set. Repeat step 4 for each card in the set.
6. After the last card is written, click **OK** in the Backup Master Key for Encryption Group window to finish the operation.

6.2.3 Restoring the master key from a file

To restore the master key to the EE, transfer the master key from the host first, see Example 6-4 on page 215.

Important: Ensure that you restore the master key to the group leader node only.

Example 6-4 Transferring the master key file from the host

```
SAN32B-E4-2:admin> cryptocfg --show -file -all
No files found

SAN32B-E4-2:admin> cryptocfg --import -scp mk_restore.mk 10.18.228.46
administrator bes1_mk.mk

Available Space:28672
Make sure your file size is not greater than 28672.
The switch will be unstable or the operation will fail if you exceed this limit.
Do you want to proceed?
ARE YOU SURE (yes, y, no, n): [no] y
administrator@10.18.228.46's password:
Operation succeeded.

SAN32B-E4-2:admin> cryptocfg --show -file -all
File name: mk_restore.mk, size: 848 bytes
```

Example 6-4 shows the procedure for transferring the master key file from the host. Before the transfer, there are no files at the default location on the switch.

- ▶ The `mk_restore.mk` file is the local file name on the switch where the master key is to be stored.
- ▶ The IP address of the host keeping the master key file is 10.18.228.46.
- ▶ The administrator is the login that is allowed to connect by SCP.
- ▶ The `bes1_mk.mk` file is the name of the file on the host.

Important: FTP is not supported for transferring the master key file.

After the transfer has succeeded, the file with the name `mk_restore.mk` appears in the default location of the switch. The restored master key must be applied to the Encryption Switch and it must be applied on the EG leader node. Example 6-5 shows the master key from the file that has been recovered as the current master key. The passphrase, which has been used to export the master key, must be used for recovery.

Example 6-5 Activating the restored master key

```
SAN32B-E4-2:admin> cryptocfg --recovermasterkey currentMK -srcfile mk_restore.mk

Enter passphrase:

Operation succeeded
```

After the master key recovery, check the active master key with the `cryptocfg --show -groupmember -all` command to verify that the recovered master key has been applied.

6.2.4 Restoring the master key from the smart cards

A card reader must be attached to the SAN Management application PC to complete this procedure.

To restore the master key from a set of smart cards, complete the following steps:

1. Select **Configure** → **Encryption** from the menu task bar. The Encryption Center window opens.
2. Select a group from the Encryption Center Devices table. Then, select **Group** → **Security** from the menu task bar, or right-click a group and select **Security**.
3. The Encryption Group Properties window opens with the Security tab selected. Select **Restore Master Key** as the Master Key Action.
4. The Restore Master Key for Encryption Group window opens, as shown in Figure 6-2.

Select a Master Key to Restore

☒ Active Master Key - The resulting key will be used for all new data encryption.

☐ Alternate Master Key - The resulting key can be used for reading old tapes.

Restore From: A Recovery Set of Smart Cards ▼

You will need a card reader attached to the management station and recovery cards.

1) Insert each card, in turn, into the card reader and wait for its ID to appear below.

Card ID:

2) Enter card password below.

Card Password:
Case sensitive.

3)

Completed	Group Card #	Card ID	First Name	Last Name	Notes

Status: Connecting to the smart card ...

Figure 6-2 Restoring the master key file from the smart cards

5. Follow these steps:
 - a. Choose the Active Master Key or the Alternate Master Key for restore, as appropriate.
 - b. Select **A Recovery Set of Smart Cards** as the Restore From location.
 - c. Insert the recovery card containing a share of the master key that was backed up earlier, and wait for the Card ID (card serial number) to appear.
 - d. Enter the password that was used to create the card. After five unsuccessful attempts to enter the correct password, the card becomes locked and unusable.
 - e. Click **Restore**.

6. The window prompts you to insert the next card, if needed. Repeat step 5 until all the cards in the set have been read.
7. Click **OK**.

6.3 Configuration upload and download

You must back up the switch configuration periodically. You must also back up the switch configuration after the new node has been added to the EG, new LUNs have been discovered, and so on. The process is similar to the regular configuration upload and download, but because of the nature of EEs, there are certain areas on which you need to focus.

6.3.1 Uploading the configuration

The configuration file from the EE differs from the regular configuration file and depends on the place where it was generated. Certain extra sections and entries appear in the configuration file from the EE. See Example 6-6.

Example 6-6 Output of the configuration file (only selected entries are shown)

```
[Configuration]
encryptiongroup.0:clusName=support_TKLM,numDefinedNodes=3,hbtCount=3,hbtInterval=2
,glPollPeriod=0,glPollRetries=0
encryptiongroup.0.defnode.0:nodeName=10:00:00:05:1e:54:17:10,ipAddr=10.18.235.54,c
ertFile=10.18.235.54_my_cp_cert.pem,maxEEs=16,discMode=1
encryptiongroup.0.defnode.1:nodeName=10:00:00:05:1e:94:3a:00,ipAddr=10.18.228.27,c
ertFile=10.18.228.27_my_cp_cert.pem,maxEEs=16,discMode=1
encryptiongroup.0.defnode.2:nodeName=10:00:00:05:1e:54:16:53,ipAddr=10.18.235.56,c
ertFile=10.18.235.56_my_cp_cert.pem,maxEEs=16,discMode=1
encryptiongroup.glnode:glwnn=10:00:00:05:1e:54:17:10,glname=10:00:00:05:1e:54:17:1
0,glvip=10.18.235.54,glpip1=,glpip2=
[Zoning]
cfg.CFGNAME:W2k8_1_to_V7000_1;W2k8_2_to_V7000_1
cfg.r_e_d_i_r_c_fg:red_1109_support_TKLM500507680230a7fe200800051e54170c;
red_1109_support_TKLM500507680240a7fe200a00051e54170c;red_1109_support_TKLM5005076
80230a7be200800051e54170c;
red_1109_support_TKLM500507680240a7be200a00051e54170c;red_1109_support_TKLM5005076
80210a7fe200200051ec11313;
red_1109_support_TKLM500507680220a7fe200a00051ec11313;red_1109_support_TKLM5005076
80210a7be200200051ec11313;
red_1109_support_TKLM500507680220a7be200a00051ec11313;red_____base
defzone:noaccess
[cryptoDev]
cryptoDev.swEncFeatureMode.tklm:Enabled
cryptoDev.hacGroup.1.name:enc_ha_fabA
cryptoDev.hacGroup.1.member.1:10:00:00:05:1e:54:16:53_0
cryptoDev.hacGroup.1.member.2:10:00:00:05:1e:54:17:10_0
cryptoDev.hacGroup.1.recEnd:
cryptoDev.ct.1.name:CTC_1
cryptoDev.ct.1.frontEndPort:0
cryptoDev.ct.1.mediaType:Disk
cryptoDev.ct.1.ee:10:00:00:05:1e:54:17:10_0
cryptoDev.ct.1.tgtNWWN:50:05:07:68:02:00:a7:fe
```

```

cryptoDev.ct.1.tgtNVWWN:20:01:00:05:1e:54:17:0c
cryptoDev.ct.1.tgtPWWN:50:05:07:68:02:30:a7:fe
cryptoDev.ct.1.tgtPVWWN:20:00:00:05:1e:54:17:0c
cryptoDev.ct.1.itr.1.NWWN:20:00:00:05:1e:c7:6b:8a
cryptoDev.ct.1.itr.1.NVWWN:20:09:00:05:1e:54:17:0c
cryptoDev.ct.1.itr.1.PWWN:10:00:00:05:1e:c7:6b:8a
cryptoDev.ct.1.itr.1.PVWWN:20:08:00:05:1e:54:17:0c
cryptoDev.ct.1.itr.1.disk.1.lunID:00:00:00:00:00:00:00:00
cryptoDev.ct.1.itr.1.disk.1.encAdminStatus:Encrypt
cryptoDev.ct.1.itr.1.disk.1.encOnExistingData:Disabled
cryptoDev.ct.1.itr.1.disk.1.enableRekey:Enabled
cryptoDev.ct.1.itr.1.disk.1.rekeyTimePeriod:1440
cryptoDev.ct.1.itr.1.disk.1.allowDecrypt:Disabled
cryptoDev.ct.1.itr.1.disk.1.lunState:Encrypt
cryptoDev.ct.1.itr.1.disk.1.newLUN:Disabled
cryptoDev.ct.1.itr.1.disk.1.opMode:native
cryptoDev.ct.1.itr.1.disk.2.lunID:00:01:00:00:00:00:00:00
cryptoDev.ct.1.itr.1.disk.2.encAdminStatus:Encrypt
cryptoDev.ct.1.itr.1.disk.2.encOnExistingData:Disabled
cryptoDev.ct.1.itr.1.disk.2.enableRekey:Disabled
cryptoDev.ct.1.itr.1.disk.2.rekeyTimePeriod:0
cryptoDev.ct.1.itr.1.disk.2.allowDecrypt:Disabled
cryptoDev.ct.1.itr.1.disk.2.lunState:Encrypt
cryptoDev.ct.1.itr.1.disk.2.newLUN:Disabled
cryptoDev.ct.1.itr.1.disk.2.opMode:native
cryptoDev.ct.1.recEnd:

```

Example 6-6 on page 217 shows the output of the configuration file. There are several additions:

- The Configuration section contains the EG description for each node.
- The Zoning section contains entries about frame redirection zones and also indicates that the default zone has no access.
- The CryptoDev section now appears and contains information about the EG members and HA cluster members. It also shows the configuration of all defined CTCs with LUN description and numbers, all target worldwide nodes (WWNs), and initiator WWNs.

The security information is not included when you upload a configuration from an Encryption Switch or Encryption Blade. Extra steps are necessary before and after the download to re-establish that information. The following sections describe the information that is included in an upload from an EG leader and EG member load, the information that is not included, and the steps to re-establish the information.

Configuration upload at an encryption group leader node

A configuration upload that is performed at an EG leader node contains the following information:

- Local switch configuration
- EG-related configuration
- EG-wide configuration of Crypto Targets, disk and tape LUNs, tape pools, HA clusters, security, and key vaults

Configuration upload at an encryption group member node

A configuration upload that is performed at an individual EG member node contains the following information:

- ▶ Local switch configuration
- ▶ EG-related configuration
- ▶ EG-wide configuration of Crypto Targets, disk LUNs, HA clusters, security, and key vaults

Information that is not included in an upload

The following certificates are not present when the configuration is downloaded:

- ▶ External certificates that were imported on the switch:
 - Key vault certificate
 - Peer node/switch certificate
 - Authentication card certificate
- ▶ Certificates that were generated internally:
 - KAC certificate
 - CP certificate
 - Federal Information Processing Standards (FIPS) officer and user certificates

The Authentication Quorum size is included in the configuration upload for read-only purposes, but it is not set by a configuration download.

6.3.2 Configuration download

Use the following steps to download the configuration.

Prerequisites before the configuration download

The configuration download does include any certificates, public or private keys, master key, or link keys. Perform the following steps prior to the configuration download to generate and obtain the necessary certificates and keys:

1. Use the following commands to initialize the EE:
 - `cryptocfg --InitNode`
 - `cryptocfg --initEE`
 - `cryptocfg --regEE`
2. Initializing the switch generates the following internal certificates:
 - KAC certificate
 - CP certificate
 - FIPS officer and user certificates
3. Import the certificates of the peer nodes/switches onto the switch prior to the configuration download.
4. Import the key vault certificates onto the switch prior to the configuration download.
5. Create an EG with the same name as the name in the configuration upload information for the EG leader node.
6. Import the Authentication Card Certificates onto the switch prior to the configuration download.

Configuration download at the encryption group leader

The configuration download contains the EG-wide configuration information about Crypto Targets, disk LUNs, HA clusters, security, and key vaults. The EG leader first applies the EG-wide configuration information to the local configuration database and then distributes the configuration to all members in the EG. Also, any layer 2 and switch-specific configuration information is applied locally to the EG leader.

Configuration download at an encryption group member

Switch-specific configuration information pertaining to the member switch or blade is applied. Information that is specific to the EG leader is filtered out.

Performing certain steps after the configuration download

For all key vaults, restore or generate and back up the master key. In a multiple-node EG, the master key is propagated from the group leader node. Perform these steps:

1. Use the following command to enable the EE:

```
cryptocfg --enableEE [slot num]
```

2. Commit the configuration:

```
cryptocfg --commit
```

3. If there are containers that belonged to the old Encryption Switch or Encryption Blade, after you run **con fig download**, use the following command to change the ownership of the containers to the new Encryption Switch or Encryption Blade, assuming that the host and target physical zone exist:

```
cryptocfg --replace <old EE WWN> <new EE WWN>
```

4. Commit the configuration:

```
cryptocfg --commit
```

5. Use the command that is shown in Example 6-7 to check if the Encryption Switch or Encryption Blade has the master key:

```
cryptocfg --show -groupmember <switch WWN>
```

Example 6-7 Checking the master key status of the groupmember

```
SAN32B-E4-1:admin> cryptocfg --show -groupmember 10:00:00:05:1e:54:17:10
Node Name:                10:00:00:05:1e:54:17:10 (current node)
State:                    DEF_NODE_STATE_DISCOVERED
Role:                     GroupLeader
IP Address:               10.18.235.54
Certificate:              10.18.235.54_my_cp_cert.pem
Current Master Key State:  Saved
Current Master
KeyID:e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5
Alternate Master Key State: Not configured
Alternate Master KeyID:    00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
EE Slot:                  0
SP state:                 Online
Current Master KeyID: e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5
Alternate Master KeyID: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
HA Cluster Membership:    enc_ha_fabA
```

Example 6-7 shows that the master key of the group leader node has been discovered and saved.

6. If a master key is not present, restore the master key from a backed up copy. Procedures will differ depending on the backup media that was used (from recovery smart cards, the key vault, a file on the network, or a file on a USB-attached device).
7. If a new master key needs to be generated, generate the master key and back it up. If authentication cards are used, set the authentication quorum size from the EG leader node after importing and registering the necessary number of Authentication Card certificates.

6.4 Adjusting heartbeat signaling values

EG nodes use *heartbeat signaling* to communicate to one another and to their associated key vaults. A configurable threshold of heartbeat misses determines how long an EG leader will wait before declaring a member node unreachable. The default heartbeat signaling values are three heartbeat misses, each followed by a two-second heartbeat timeout. If three consecutive heartbeats are missed (by default, a time interval of six seconds without a heartbeat signal), the EG leader node declares a member node as unreachable, resulting in an EG split scenario (EG split).

If the management network becomes congested or unreliable, resulting in excessive auto-recovery processing or the need for manual recovery from EG splits, it is possible to set larger heartbeat and heartbeat timeout values to mitigate the chances of having the EG split while the network issues are being addressed. You issue the following commands from the EG leader nodes to change the heartbeat signaling values:

```
switch:admin->cryptocfg --set -hbmisses <number>
```

```
switch:admin->cryptocfg --set -hbtimeout <time>
```

Table 6-1 shows the parameter definitions.

Table 6-1 Parameter descriptions

Value	Description
<number>	Sets the number of heartbeat misses allowed in a node that is part of an EG before the node is declared unreachable. This value is set in conjunction with the timeout value. It must be configured at the group leader node and is distributed to all member nodes in the EG. The value that is entered specifies the number of heartbeat misses. The default value is 3. The range is 1 - 15 in integer increments only.
<time>	Sets the timeout value for the heartbeat. This parameter must be configured at the group leader node and is distributed to all member nodes in the EG. The value entered specifies the heartbeat timeout in seconds. The default value is 2 seconds. Valid values are integers in the range between 1 - 30 seconds.

Limit: The collective time that is allowed (the heartbeat timeout value multiplied by the heartbeat misses) cannot exceed 30 seconds (this rule is enforced by Fabric OS).

6.5 Removing and replacing the IBM SAN768/384 Encryption Blade

The following procedure uses an IBM SAN 768/384 Encryption Blade that is installed in IBM SAN768\384 slot 4 as an example. If an Encryption Blade fails and it must be replaced, complete the following steps:

1. From the group leader, enter the following command to reclaim the virtual initiator (VI)/virtual target (VT) WWN base for the EE to be removed from the EG:

```
cryptocfg --reclaimWWN -EE <EB_1 WWN> 4
```

When prompted, enter Yes.

Important: Do *not* execute the previous command with the **-list** option between steps 1 and 4.

2. Enter the following command to propagate the change throughout the EG:

```
cryptocfg --commit
```

3. Remove the I/O links and FC cables from the Encryption Blade, noting where each I/O link and FC cable was attached.

4. Remove the failed Encryption Blade.

5. Insert the new Encryption Blade in the same slot (in this example, slot 4) in the chassis.

6. Connect the I/O sync ports to the same private LAN as the I/O sync ports of the failed Encryption Blade. Confirm that the IP addresses of the I/O sync ports (Ge0 and Ge1) are the same as the previous IP addresses.

7. Zeroize the new encryption engine:

```
cryptocfg --zeroizeEE 4
```

The new EE will power off and power on again automatically.

8. If a system card authentication is needed to enable the EE, re-register the system card through the Management application client for the new EE.

9. Initialize the new EE:

```
cryptocfg --initEE 4
```

10. Register the new EE:

```
cryptocfg --regEE 4
```

11. Enable the new EE:

```
cryptocfg --enableEE 4
```

12. Verify that this Encryption Blade EE has the same master key as the rest of the EEs in the EG by using the **cryptocfg --show -groupmember -all** command, as shown in Example 6-8.

Example 6-8 Checking the master key on the replaced Encryption Blade

```
SAN32B-E4-1:admin> cryptocfg --show -groupmember -all
```

```
NODE LIST
```

```
Total Number of defined nodes: 3
```

```
Group Leader Node Name: 10:00:00:05:1e:54:17:10
```

```
Encryption Group state: CLUSTER_STATE_CONVERGED
```

```
Node Name: 10:00:00:05:1e:54:17:10 (current node)
```

```

        State:                                DEF_NODE_STATE_DISCOVERED
        Role:                                GroupLeader
        IP Address:                          10.18.235.54
        Certificate:                          10.18.235.54_my_cp_cert.pem
        Current Master Key State:  Saved
    Current Master KeyID:  e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5
    Alternate Master Key State: Not configured
        Alternate Master KeyID: 0:00:00:00:00:00:00:00:00:00:00:00:00:00:00
    EE Slot: 0
    SP state: Online
        Current Master KeyID: e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5
    Alternate Master KeyID: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
    HA Cluster Membership:  enc_ha_fabA
        Node Name: 10:00:00:05:1e:94:3a:00
        State: DEF_NODE_STATE_DISCOVERED
        Role: MemberNode
        IP Address: 10.18.228.27
        Certificate: 10.18.228.27_my_cp_cert.pem
        Current Master Key State: Saved
        Current Master KeyID: e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5
        Alternate Master Key State: Not configured
        Alternate Master KeyID: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
    EE Slot: 1
    SP state: Online
    Current Master KeyID: e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5
        Alternate Master KeyID: 0:00:00:00:00:00:00:00:00:00:00:00:00:00:00
        No HA cluster membership
        Node Name: 10:00:00:05:1e:54:16:53
        State: DEF_NODE_STATE_DISCOVERED
        Role: MemberNode
        IP Address: 10.18.235.56
        Certificate: 10.18.235.56_my_cp_cert.pem
        Current Master Key State: Saved
        Current Master KeyID: e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5
        Alternate Master Key State: Not configured
        Alternate Master KeyID: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00

```

Example 6-8 on page 222 shows the EG members' properties. You can see that all three members use the same master key. Two nodes are in an HA cluster, and the third node is not. All the nodes are in the same EG and use the same key, and all the nodes have exchanged their certificates.

13. Check the EE state using the **cryptocfg --show -localEE** command to ensure that the EE is online, as shown in Example 6-9.

Example 6-9 Checking the status of the EE

```

SAN32B-E4-1:admin> cryptocfg --show -localEE
    EE Slot: 0
    SP state: Online
    Current Master KeyID: e5:ec:ce:12:c8:33:a2:04:a7:17:ab:73:2c:83:9f:a5
    Alternate Master KeyID: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
    HA Cluster Membership:  enc_ha_fabA
    EE Attributes:
    Link IP Addr  : 192.168.0.110
    Link GW IP Addr : 0.0.0.0
    Link Net Mask  : 255.255.255.0

```

```

Link MAC Addr      : 00:05:1e:54:17:0c
Link MTU           : 1500
Link State         : UP
Media Type         : DISK
Rebalance Recommended: NO
System Card        : Disabled
System Card Label  :
System Card CID    :
Remote EE Reachability :
Node WWN/Slot      EE IP Addr   EE State IO Link State
10:00:00:05:1e:94:3a:00/1  192.168.0.130  EE_STATE_ONLINE Reachable
10:00:00:05:1e:54:16:53/0  192.168.0.120  EE_STATE_ONLINE Reachable

```

Example 6-9 on page 223 shows the status of the node with the enabled EE. The status of the engine is OnLine. You can find the IP address of the Ge0 interface and the status of the other members of the EG and the HA cluster.

14. Check the EG state by using the **cryptocfg --show -groupcfg** command to ensure that the entire EG is in the CLUSTER_STATE_CONVERGED and In Sync states.

Example 6-10 shows the status of the EG. There are three nodes and all are in the In Sync state. The EG is in the CLUSTER_STATE_CONVERGED state, which is the normal condition. You can see that authentication cards are not configured for use by this group.

Example 6-10 Checking the EG status (output is truncated)

```

Key Vault configuration and connectivity checks successful, ready for key
operations.
Authentication Quorum Size:      0
Authentication Cards not configured
NODE LIST
Total Number of defined nodes:  3
Group Leader Node Name:         10:00:00:05:1e:54:17:10
Encryption Group state:         CLUSTER_STATE_CONVERGED
Crypto Device Config state:      In Sync
Encryption Group Config state:   In Sync
Node Name                      IP address      Role
10:00:00:05:1e:54:17:10        10.18.235.54    GroupLeader (current node)
    EE Slot:                    0
    SP state:                    Online
10:00:00:05:1e:94:3a:00        10.18.228.27    MemberNode
    EE Slot:                    1
    SP state:                    Online
10:00:00:05:1e:54:16:53        10.18.235.56    MemberNode
    EE Slot:                    0
    SP state:                    Online

```

Ownership: Because the Encryption Blade was inserted into the same slot as the previous Encryption Blade, no change of the HA cluster container ownership is required. The HA cluster configuration is retained as is. If manual failback had been set on the HA cluster, user intervention is required to manually fail back the LUNs that are owned by the newly replaced EE. No change occurs in the CTC ownership. The container ownerships are retained as is.

6.5.1 Encryption node removal and replacement in a multinode group

The following procedure uses IBM SAN32B-E4 3 (ES3) as the Encryption Switch to remove from an EG with the group leader designated as Encryption Switch 1 (ES1). We describe two scenarios:

- ▶ When the Encryption Switch has failed
- ▶ When the Encryption Switch has not failed

When the ES3 has failed

Complete the following steps:

1. Deregister ES3 from the EG:

```
cryptocfg --dereg -membnode <switchWWN>
```

2. Reclaim the WWN base of ES3:

```
cryptocfg --reclaimWWN -membnode <switchWWN> [-list]
```

3. Synchronize the crypto configurations across all member nodes:

```
cryptocfg --commit
```

Brocade Encryption Switch: When attempting to reclaim a failed Brocade Encryption Switch, do not execute **cryptocfg --transabort**. Executing this command will cause subsequent reclaim attempts to fail.

When ES3 has not failed

Complete the following steps:

1. From the group leader, enter the following command to reclaim the VI/VT WWN base for the node to be removed:

```
cryptocfg --reclaimWWN -membnode <ES3-WWN>
```

When prompted, enter Yes.

Important: Do not use the previous command with the **-list** option between steps 1 and 4.

2. From the group leader, enter the following command to propagate the change to all nodes in the EG:

```
cryptocfg --commit
```

3. Enter the following command in ES1 to eject the ES3 node from the EG:

```
cryptocfg --eject -membnode <ES3-WWN>
```

4. Enter the following command on ES1 to deregister the ejected node from the EG:

```
cryptocfg --dereg -membnode <ES3-WWN>
```

5. Execute the following command on the ES3:

```
cryptocfg --reclaimWWN -cleanup
```

6. Replace the old IBM SAN32B-E4 switch with the new IBM SAN32B-E4 switch and reattach the I/O link cables and Mgmt Link. Connect the I/O sync ports to the same private LAN as the I/O sync port of the failed node.

7. Run the following command on the ejected member node:

```
cryptocfg --reclaimWWN -cleanup
```

Important: Do not reconnect the FC cables yet.

8. Power on the new Brocade Encryption Switch.
9. Set the IP address for the new IBM SAN32b-E4 switch using the **ipaddrset** command for the Mgmt Link and I/O link. Check that the switch name and the domain ID that are associated with the replacement switch match the switch name and the domain ID of the original switch.
10. Zeroize the new IBM SAN32B-E4 switch:

```
cryptocfg --zeroizeEE
```

The IBM SAN32B-E4 switch reboots automatically.
11. If the EG has a system card authentication enabled, you must re-register the system card through the Management application client for the new EE.
12. Initialize the new IBM SAN32B-E4 switch node:

```
cryptocfg --initnode
```
13. From the new IBM SAN32B-E4 switch node, run the following command to export the CP certificate of the new IBM SAN32B-E4 switch:

```
cryptocfg --export -scp -CPcert <host IP> <host user> <host file path>
```
14. From the group leader node, run the following command to import the new IBM SAN32B-E4 node certificate on the group leader node:

```
cryptocfg --import -scp <Certificate file name> <host IP> <host user> <host file path>
```
15. From the group leader node, run the following command to register the new IBM SAN32B-E4 switch node as a member node on the group leader:

```
cryptocfg --reg -memberrnode <New BES WWN> <Cert file Name> <Old IP address>
```
16. Initialize the new EE:

```
cryptocfg --initEE [slotnumber]
```
17. Register the new EE:

```
cryptocfg --regEE [slotnumber]
```
18. Enable the new EE:

```
cryptocfg --enableEE [slotnumber]
```
19. Check that the EE state is online:

```
cryptocfg --show -localEE
```
20. Export the KAC certificate from the new node.
21. Register the switch KAC certificate on TKLM using the TKLM GUI (for the primary and secondary TKLMs).
22. From the new IBM SAN32B-E4 switch, run the following command to set the default zone as All Access, so that the configuration from the existing fabric is pushed to the new SAN32B-E4 Encryption Switch:

```
defzone -allaccess
```
23. Run the following command on the new IBM SAN32B-E4 switch:

```
cfsave
```
24. Replace the FC cables to the new IBM SAN32B-E4 switch.

25. Run the **cfgsave** command on any switch in that fabric. The fabric configuration from the existing fabric is merged into the new IBM SAN32B-E4 switch. Verify that defzone is now set as no access.

26. This step will vary depending on whether HA cluster (HAC) membership for the old SAN32B-E4 Encryption Switch exists.

If HAC membership for the old IBM SAN32B-E4 switch was in place, perform the following steps to move the container to the new IBM SAN32B-E4 switch:

- a. From the group leader, replace the old EE with the new EE:

```
cryptocfg --replace <WWN of Old ES> <WWN of new ES>
```

- b. Issue the commit:

```
cryptocfg --commit
```

- c. From the group leader, replace the HAC membership from the old EE to the new EE:

```
cryptocfg --replace -haclustermember <HA cluster name> <WWN of Old ES> <WWN of New BES>
```

- d. Issue the commit:

```
cryptocfg --commit
```

- e. If manual failback was set on the HAC, user intervention is required to manually fail back the LUNs that are owned by the newly replaced IBM SAN32B-E4 switch.

If HAC membership for the old IBM SAN32B-E4 switch was not in place, complete the following steps to move the container to the new IBM SAN32B-E4 switch:

- a. From the group leader, replace the old EE with the new EE:

```
cryptocfg --replace <WWN of Old ES> <WWN of new ES>
```

- b. Issue the commit:

```
cryptocfg --commit
```

27. Check the EG state to ensure that the entire EG is in the CLUSTER_STATE_CONVERGED and In Sync states:

```
cryptocfg --show -groupcfg
```

Single-node encryption group replacement

Follow these steps to replace the IBM SAN32B-E4 switch:

1. Upload the configuration that is stored on the IBM SAN32B-E4 switch that you are replacing by using the **FOS configupload** command.
2. Power off the IBM SAN32B-E4 switch.
3. Remove the Mgmt Link, I/O links, and FC cables from IBM SAN32B-E4 switch, carefully making note of where each one was attached so that the replacement IBM SAN32B-E4 switch can be cabled properly.
4. Power on the new IBM SAN32B-E4 switch. Note that the FC cables have not been plugged in yet.
5. Set the IP address for the new IBM SAN32B-E4 switch using the **ipaddrset** command both for the Mgmt and I/O links. Check that the switch name and domain ID that are associated with the replacement switch match the switch name and domain ID of the original switch.
6. Recreate the EG with the same name as before:

```
cryptocfg --create -encgroup <EG Name>
```

7. Download the configuration from the previous uploaded configuration.
8. Zeroize the new IBM SAN32B-E4 switch:

```
cryptocfg --zeroizeEE
```

The IBM SAN32B-E4 switch reboots automatically.
9. If system card authentication was enabled, you must re-register the system card through the Management application client for the new EE.
10. Initialize the new IBM SAN32B-E4 switch node by using following command:

```
cryptocfg --initnode
```
11. Initialize the new EE:

```
cryptocfg --initEE [slotnumber]
```
12. Register the new EE:

```
cryptocfg --regEE [slotnumber]
```
13. Enable the new EE:

```
cryptocfg --enableEE [slotnumber]
```
14. Check the EE state using following command to ensure that the EE is online:

```
cryptocfg --show -localEE
```
15. Export the KAC certificate from the new node.
16. Import the switch KAC certificate into TKLM using the TKLM GUI (for the primary and secondary TKLMs).
17. Export the TKLM server certificate (for the primary and secondary TKLMs).
18. Import the TKLM certificate into the switch (import the primary and secondary TKLM certificates into the GL).
19. Register the TKLM certificate as the key vault certificate on the switch (GL only).
20. Remove the old node certificates from TKLM (for primary and secondary TKLMs).
21. If a master key is not present, restore the master key from a backed up copy. Procedures will differ depending on the backup media that is used (from recovery smart cards, the key vault, a file on the network, or a file on a USB-attached device).
22. Set the defzone as All Access on the new IBM SAN32B-E4 switch, so that the configuration from the fabric is pushed to the new IBM SAN32B-E4 switch.
23. Run the following command on the new IBM SAN32B-E4 switch:

```
cfigsave
```
24. Connect the FC cables to the new IBM SAN32B-E4 switch.
25. Run the **cfigsave** command on any switch in that fabric. The fabric configuration from the existing fabric is merged into the new IBM SAN32B-E4 switch. Verify that defzone is now set as no access.
26. If the previous uploaded configuration is available, run the following command on the new IBM SAN32B-E4 switch to transfer the ownership of the containers to the new IBM SAN32B-E4 switch:

```
cryptocfg --replace <old node WWN> <new node WWN>
```
27. If the uploaded configuration is not available, you must re-create the container.
28. Issue the commit:

```
cryptocfg --commit
```

6.6 Removing stale rekey information for a LUN

To clean up stale rekey information for a LUN, complete one of the following procedures:

- Procedure 1:
 - a. Modify the LUN policy from “encrypt” to cleartext, and commit. The LUN will become disabled.
 - b. Enable the LUN using the following command:

```
cryptocfg --enable -LUN
```
 - c. Modify the LUN policy from “cleartext” to encrypt with the **enable_encexistingdata** command to enable the first-time encryption, and then, commit. This process will clear the stale rekey metadata on the LUN, and the LUN can be used again for encryption.
- Procedure 2:
 - a. Remove the LUN from the CTC and commit.
 - b. Add the LUN back to the CTC with LUN State=clear-text, policy=encrypt, and enable_encexistingdata set for enabling the first-time encryption, and then, commit. This process will clear the stale rekey metadata on the LUN, and the LUN can be used again for encryption.

6.7 Troubleshooting

Several basic troubleshooting steps follow.

6.7.1 Errors when adding a switch to an existing group

Table 6-2 lists configuration task errors that you might encounter while adding a switch to an existing group, and it describes how to troubleshoot them.

Table 6-2 Error recovery instructions for adding a switch to an existing group

Configuration task	Error description	Recovery steps
Initialize the switch	Unable to add the switch to the EG. The switch is no longer a group leader or does not contain a group.	Manual option: To add a switch to the group on the leader switch: 1. Relaunch the Configure Switch Encryption wizard and create a new EG on the leader switch. 2. When that group is created, launch the Configure Switch Encryption wizard again and add the switch to the group.
Initialize the switch	The switch was not properly initialized and was aborted because it is unavailable.	Rerun the Configure Switch Encryption wizard for the switch.
Add the switch to the encryption group	Adding the switch to the EG failed.	Rerun the Configure Switch Encryption wizard for the switch.

Configuration task	Error description	Recovery steps
Enable the EEs	A failure occurred while attempting to enable EEs on the switch.	<p>Follow these steps:</p> <ol style="list-style-type: none"> 1. Remove the switch from the group using the Group Members tab on the Encryption Group Properties window. 2. Rerun the Configure Switch Encryption wizard for the switch. <p>Manual option:</p> <ol style="list-style-type: none"> 1. Save the switch's public key certificate to a file using the Switch Encryption Properties window. 2. Follow the Key Vault instructions for RSA/Decru/Other key vault.
Save the switch's public key certificate to a file.	You are unable to save the switch's public key certificate to a file. Verify that the path name and the file name that you are using are both valid and that you have write permissions for the file.	<p>Follow these steps:</p> <ol style="list-style-type: none"> 1. Remove the switch from the group using the Group Members tab on the Encryption Group Properties window. 2. Rerun the Configure Switch Encryption wizard for the switch. <p>Manual option:</p> <ol style="list-style-type: none"> 1. Save the switch's public key certificate to a file using the Switch Encryption Properties window. 2. Follow the Key Vault instructions.

6.7.2 Errors related to adding a switch to a new group

Table 6-3 on page 231 explains errors when adding a switch to a new EG.

Table 6-3 Error recovery instructions for adding a switch to a new group

Configuration task	Error description	Instructions
Initialize the switch	Unable to initialize the switch due to an error response from the switch.	Diagnose the problem using standard switch CLI commands.
Initialize the switch	The switch was not properly initialized and was aborted, because it is unavailable.	Rerun the Configure Switch Encryption wizard for the switch.
Create EG on the switch	A failure occurred while attempting to create a new EG on the switch.	Follow these steps: <ol style="list-style-type: none"> 1. Click Refresh on the Configure Switch Encryption window to synchronize the data and the database. 2. Rerun the Configure Switch Encryption wizard for the switch.
Register one or more key vaults	A failure occurred while attempting to register one or more key vaults for a group on the switch.	Follow these steps: <ol style="list-style-type: none"> 1. Remove the switch from the group using the Group Members tab on the Encryption Group Properties window. 2. Rerun the Configure Switch Encryption wizard for the switch. Manual option: <ol style="list-style-type: none"> 1. Launch the Encryption Group Properties window and click the General tab. 2. From the General window, click Load from File to install key vault certificates, and then click OK to save the information on to the switch. 3. Follow the Key Vault instructions.

Configuration task	Error description	Instructions
Enable the EEs	A failure occurred while attempting to enable EEs on the switch.	<p>Follow these steps:</p> <ol style="list-style-type: none"> 1. Remove the switch from the group using the Group Members tab on the Encryption Group Properties window. 2. Rerun the Configure Switch Encryption wizard for the switch. <p>Manual option:</p> <ol style="list-style-type: none"> 1. Launch the Switch Encryption Properties window. 2. Save the switch's public key certificate to a file using the Switch Encryption Properties window. 3. Follow the Key Vault instructions for the key vault.
Create a new master key	A failure occurred while attempting to create a new master key.	<p>Follow these steps:</p> <ol style="list-style-type: none"> 1. Remove the switch from the group using the Group Members tab on the Encryption Group Properties window. 2. Rerun the Configure Switch Encryption wizard for the switch. <p>Manual option:</p> <ol style="list-style-type: none"> 1. Launch the Encryption Group Properties window, and click Security. 2. Click Master Key Action and select Create New Master Key to generate a new master key.
Save the switch's public key certificate to a file.	You are unable to save the switch's public key certificate to a file. Verify that the path name and the file name that you are using are both valid and that you have write permissions for the file.	<p>Follow these steps:</p> <ol style="list-style-type: none"> 1. Remove the switch from the group using the Group Members tab on the Encryption Group Properties window. 2. Rerun the Configure Switch Encryption wizard for the switch. <p>Manual option:</p> <ol style="list-style-type: none"> 1. Save the switch's public key certificate to a file using the Switch Encryption Properties window. 2. Follow the Key Vault instructions.

6.7.3 LUN policy troubleshooting

Use Table 6-4 as an aid in troubleshooting problems that relate to LUN policies.

Table 6-4 LUN policy troubleshooting

Reasons for the Encryption Switch disabling the LUN	Action taken	If you do not need to save the data	If you need to save the data
The LUN was modified from the encrypt policy to the cleartext policy, but metadata exists.	LUN is disabled. Reason code: Metadata exists but the LUN policy is cleartext.	Issue the cryptocfg --enable -LUN command on one path of the LUN. This command erases the metadata on the LUN. Then, the LUN is enabled with the cleartext policy. Issue the cryptocfg --discoverLUN command on the other paths of the LUN in the DEK cluster to enable the LUN.	Modify the LUN back to encrypt.
The LUN was set up with an encrypt policy, and the LUN was encrypted (metadata is present on the LUN), but the DEK for the key ID that is present in the metadata does not exist in the key vault.	LUN is disabled. Reason code: Metadata exists but the DEK for the key ID from the metadata does not exist.	Modify the LUN policy to cleartext.	Make sure that the key vault has the DEK, and when the DEK gets restored to the key vault, perform one of the following tasks on one of the paths of the LUN to enable the LUN: <ul style="list-style-type: none">► Issue the cryptocfg--discoverLUN command.► Remove the LUN from the container and then add it back.► Bounce the target port. Then, issue the cryptocfg --discoverLUN command on the other paths of the LUN in the DEK cluster.

Reasons for the Encryption Switch disabling the LUN	Action taken	If you do not need to save the data	If you need to save the data
The LUN was set up with an encrypt policy, and the LUN was encrypted (metadata is present on the LUN), but the current state of the LUN is cleartext instead of encrypted.	The LUN is disabled. Reason code: Metadata exists, but the LUN policy is indicated as cleartext.	Modify the LUN policy to cleartext. The subsequent handling is the same as in case 1.	Remove the LUN from the container, and then add the LUN back with the LUN state as encrypted, or issue the cryptocfg --enable -LUN command on one of the paths of the LUN, which will enable the LUN by using the appropriate key. Then, issue the cryptocfg --discoverLUN command on other paths of the LUN in the DEK cluster to enable the LUN.

Related publications

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

IBM Redbooks publications

The following IBM Redbooks publications provide additional information about the topic in this document. Note that several publications referenced in this list might be available in softcopy only.

- ▶ *Implementing the IBM Storwize V7000*, SG24-7938
- ▶ *Introduction to Storage Area Networks*, SG24-5470
- ▶ *Implementing the IBM System Storage SAN32B-E4 Encryption Switch*, SG24-7922
- ▶ *IBM System Storage Data Encryption*, SG24-7797
- ▶ *IBM System Storage Tape Encryption Solutions*, SG24-7320

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

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM System Storage Open Software Family SAN Volume Controller: Planning Guide*, GA22-1052
- ▶ *IBM System Storage Master Console: Installation and User's Guide*, GC30-4090
- ▶ *Subsystem Device Driver User's Guide for the IBM TotalStorage Enterprise Storage Server and the IBM System Storage SAN Volume Controller*, SC26-7540
- ▶ *IBM System Storage Open Software Family SAN Volume Controller: Installation Guide*, SC26-7541
- ▶ *IBM System Storage Open Software Family SAN Volume Controller: Service Guide*, SC26-7542
- ▶ *IBM System Storage Open Software Family SAN Volume Controller: Configuration Guide*, SC26-7543
- ▶ *IBM System Storage Open Software Family SAN Volume Controller: Command-Line Interface User's Guide*, SC26-7544
- ▶ *IBM System Storage Open Software Family SAN Volume Controller: CIM Agent Developers Reference*, SC26-7545
- ▶ *IBM TotalStorage Multipath Subsystem Device Driver User's Guide*, SC30-4096
- ▶ *IBM System Storage Open Software Family SAN Volume Controller: Host Attachment Guide*, SC26-7563

- ▶ *IBM System Storage SAN Volume Controller Model 2145-CF8 Hardware Installation Guide*, GC52-1356
- ▶ *IBM System Storage SAN Volume Controller Model 2145-8A4 Hardware Installation Guide*, GC27-2219
- ▶ *IBM System Storage SAN Volume Controller Model 2145-8G4 Hardware Installation Guide*, GC27-2220
- ▶ *IBM System Storage SAN Volume Controller Models 2145-8F2 and 2145-8F4 Hardware Installation Guide*, GC27-2221
- ▶ *IBM System Storage SAN Volume Controller V5.1.0 - Host Attachment Guide*, SG26-7905-05
- ▶ *Command Line Interface User's Guide*, SG26-7903-05
- ▶ *IBM Tivoli Key Lifecycle Manager v2.0 Installation and Configuration Guide*, SC27-2741-00
- ▶ Brocade Communications Systems publications:
 - *Fabric OS Administrators Guide v7.0*, 53-1002148-02 (7.0.0_FOS_AdminGd_v700.pdf)
 - *Fabric OS Command Reference Guide v7.0*, 53-1002147-01 (7.0.0_FOS_CmdRef_v700.pdf)
 - *Fabric OS Encryption. Administrator's Guide Supporting Tivoli Key Lifecycle Manager (TKLM)*, 53-1002162-02 (7.0.0_FOS_Enc_AdminGd_TKLM_v7.0.0)

Online resources

These websites are also relevant as further information sources:

- ▶ IBM System Storage home page:
<http://www.storage.ibm.com>
- ▶ SAN Volume Controller supported platform:
<http://www-1.ibm.com/servers/storage/support/software/sanvc/index.html>
- ▶ Tivoli Key Lifecycle Manager v2.0 Information Center:
<http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=%2Fcom.ibm.tk1m.doc%2Fwelcome.htm>
- ▶ Download site for Windows Secure Shell (SSH) freeware:
<http://www.chiark.greenend.org.uk/~sgtatham/putty>
- ▶ IBM site to download SSH for AIX:
<http://oss.software.ibm.com/developerworks/projects/openssh>
- ▶ Open source site for SSH for Windows and Mac:
<http://www.openssh.com/windows.html>
- ▶ Cygwin Linux-like environment for Windows:
<http://www.cygwin.com>
- ▶ IBM Tivoli Storage Area Network Manager site:
<http://www-306.ibm.com/software/sysmgmt/products/support/IBMTivoliStorageAreaNetworkManager.html>

- ▶ Microsoft Knowledge Base Article 131658:
<http://support.microsoft.com/support/kb/articles/Q131/6/58.asp>
- ▶ Microsoft Knowledge Base Article 149927:
<http://support.microsoft.com/support/kb/articles/Q149/9/27.asp>
- ▶ Sysinternals home page:
<http://www.sysinternals.com>
- ▶ Subsystem Device Driver download site:
<http://www-1.ibm.com/servers/storage/support/software/sdd/index.html>
- ▶ IBM TotalStorage Virtualization home page:
<http://www-1.ibm.com/servers/storage/software/virtualization/index.html>
- ▶ SVC support page:
<http://www-947.ibm.com/systems/support/supportsite.wss/selectproduct?taskind=4&brandind=5000033&familyind=5329743&typeind=0&modelind=0&osind=0&psid=sr&continue.x=1>
- ▶ SVC online documentation:
<http://publib.boulder.ibm.com/infocenter/svcic/v3r1m0/index.jsp>
- ▶ IBM Redbooks publications about SVC:
<http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=SVC>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

1024 41

2048 41

A

access control 17

activated 53

active group leader 16

active master key 25, 211

active/active 210

active/passive 210

active/standby 132

Active/Standby failover 27

AES 3, 71

AES256 6

AES256-XTS 23

alert window 43

alias 57

All Access 99, 174

alternate master key 25

architectural 13

asymmetric encryption algorithms 3

asymmetric key algorithms 4

asymmetric key encryption 3–4

asymmetric keys 37, 40

attack 2

audit log 40

authenticating 59

authentication cards 18

authenticity 5

automatic fallback mode 31

Automatic Rekey 187

Automatic re-keying 22

autorekeying 22

availability 42

B

backplanes 210

backup 37

bandwidth 155

block-by-block 23

blocking 182

Blowfish 3

bottleneck 197

bundled 213

C

CA 43

card reader 18, 25, 47, 211, 213

card set 213

CAST5 3

centralized key management 7, 35

certificate 15, 40, 137

Certificate Authority 43

certificate expiration 8

certificates 37

certification creation 54

certified USB stick 212

Ciphertext 16

ciphertext 3, 20, 36

clean 58

Clear Text 127

clear text 3, 130

Cleartext 16

cleartext 19, 21–22, 104, 122, 203

cleartext data 90

cleartext LUN 104, 187

cleartext LUNs 182

cluster definition 134

cluster manager 59

cluster of encryption engines 111

clusters 16

cluster-wide 16

commit 128, 130

commit operation 27

commit processing 136

committed 133, 150

communication paths 36

compression 9

compromised key 22

concurrent rekey 196

confidential 3

configuration file 37, 40

configuration files 58

configure encryption 97

configuring multi-path LUNs 113

container 97

container type 98

containers 22

converged 133

corrupted 206

CP Certificate 60

CP certificate 59

CP certificate file 59

CP Certificates 50

CP certificates 59

CP failover 27

CPcert 17

critical TKLM files 37

crypto keys 12

Crypto Target Container 18, 43, 45, 154, 172

Crypto Targets 27

cryptographic 41

cryptographic operations 18

cryptographically erased 36

CryptoModule 15

CTC 26, 43–45, 172, 187

current key 21
cycle 39

D

Data 14
data corruption 12, 105, 111, 178
Data Encryption Key 187
Data Encryption Key Cluster 111
Data Encryption Key cluster 32
data encryption keys 16–17
data keys 42
Data-at-Rest 46
data-at-rest 8
database 2
dataflow 195
DB2 tables 40
DCFM 77, 81
DCX configuration 61
debug log 40
decrypt information 37
default mode 31
default zone 99
default zoning 174
DEK 16, 43
DEK Cluster 32, 46, 111, 153
DEK creation 16
DEK lifetime 196
designing 195
destage 192
destroying 182
detect the new paths 130
device configuration 16
Device Group 52
Diffie-Hellman 4
digital certificates 5
digital document 5
disabled 187, 204
disaster recovery 204
discrete device 61
discrete device serial number 61, 83
disruptive upgrade 210
drive table 40
dual fabric encryption engine 153
dual fabric environment 153
duplicate Tivoli Key Lifecycle Managers 42
Dynamic Tracking 97
DYNTRK 97

E

ECC 4
edge switches 174
EE 16
effective configuration 100
EG splits 221
EKM 8
ElGamal 4
Elliptic curve cryptography 4
enable 61
Enabling encryption 105

encrypt information 37
encrypt target LUNs 10
encrypted block 23
encrypted data 3
encrypted key 47, 211
encrypted key record 47, 211
encrypted LUN 143, 187
encrypted traffic 177
encrypting devices 36
encrypting target LUNs 131
encryption 13
encryption algorithms 195
encryption deadlock 7
encryption enabled 109
encryption engine 16
Encryption Engine WWN 173
encryption engines 15, 134
Encryption Group 16
encryption group 12, 59
Encryption Group Leader 173
encryption group split 221
encryption key 3, 36
encryption key blocking 201
encryption Key ID 104
Encryption Key Manager 8
encryption keys 6, 37, 204
encryption mode 104
encryption node 17
encryption nodes 16
encryption operations 16, 196
encryption policies 127
encryption policy 177
encryption solution 195
encryption state 104
encryption target 119
ertext 44
export 43, 56
Export Self-Signed Server Certificate 56
exported 53
external arrays 10
externally connected disk systems 206
extra paths 155

F

Fabric 14
failback 31
failback capabilities 27
failback mode 31
failback policy 31
failover 29, 137
failover policies 203
Failover policy 32
FC adapter 177
FC traffic 197
Federal Information Processing Standards 15
file copy 137
file space 68
file systems 2
file transfer 67
file transfer protocol 64, 67

- FIPS 15, 219
- FIPS 140-2 certification 41
- firmware 132
- firmware consistency check 211
- firmware download 211
- firmware level 211
- firmware upgrade 210
- First Time Encryption 182, 186, 196
- first time encryption 23, 109, 210
- first-time encryption 21
- FlashCopy mappings 204
- flat files 40
- Frame Redirection 172, 177
- frame redirection zone 24
- frames 172
- FTP protocol 212
- fully-provisioned 208

G

- GbE 27
- generic serial number 61
- Gigabit Ethernet 27
- GL 45, 48
- Group Leader 45, 173
- group leader 16
- group leader node 16–17
- group leader succession 16
- group manager 59
- groupleader 48
- group-wide 16

H

- HA 16, 27
- HA Cluster 196, 210
- HA cluster 27, 132
- HA Clusters Tab 134
- HA reboots 27
- HA solution 46
- HA/DEK cluster 210
- hackers 2
- harm 2
- HBA 163
- HBA replace 163
- Heartbeat misses 32
- heartbeat misses 221
- heartbeat signaling 221
- Heartbeat timeout 32
- heartbeat time-out 221
- heartbeat time-out value 221
- high availability 16, 131
- high availability cluster 27
- hops 197
- host MPIO 210
- host MPIO failover 111

I

- IBM DB2® 40
- IBM Network Analyzer 18

- IBM Security Key Lifecycle Manager 38
- IBM Tivoli Key Lifecycle Manager 6–7
- IBMJCEFIPS 41
- IDEA 3
- identical keys 3
- import 43
- import node certificates 59
- imported 53
- In-Band 39
- independent 42
- independent key vaults 204
- independent paths 110
- in-flight 43
- information exchange process 15
- initialization 81
- initialization process 17
- initialize 61
- initialize the encryption node 59
- initiator port 119, 147
- in-place encryption 21
- integrity 5
- internal arrays 10
- intracluster communication 15
- IP addresses 44
- IP communication 39
- ISL 46
- ISL bandwidth 97

J

- Java Cryptographic Extension 41
- Java Cryptography Extension 40
- Java Cryptography Extension Key Store 17
- Java Runtime Environment 40
- Java Virtual Machine 41
- JCE 40
- JCEKS 18, 52
- JCEKS keystore 40
- Jython 56

K

- KAC 66
- KAC certificate 59
- KAC certificates 50
- KEK 24, 43
- Key 15
- key 3, 15
- key algorithm 71
- Key availability 6
- key availability 36
- key encryption key 24, 43
- key exchange 43
- key group 40
- key holders 47
- key id 43
- key label 43
- key management system 47
- Key security 6
- key security 36
- key server 6, 36

- key servers 36
- key sizes 3, 40
- key synchronisation 12
- key vault 12, 15–17, 19, 21, 23, 27, 67–68, 204
- key vault certificate distribution 16
- key vault connection status 83
- key vault type 60
- key vaults 44, 153
- keys 15, 36
- key-serving function 8
- keystore 37, 40, 56, 68
- keystore type 40
- keystores 40

L

- latency 196
- LBA 21–22
- legal requirements 44
- life cycle management 6, 20
- lifecycle management 44
- links 46
- load split 131
- lock 203
- logical block address 21–22
- Logical Unit Number 14
- LUN 14, 44
- LUN id 99
- LUN inaccessible 203
- LUN masking 26
- LUN number 98
- LUN provisioning 45
- LUN UID 148

M

- man in the middle attack 53
- management network 15
- management ports 15
- manual 204
- manual failback 31
- Manual Rekey 187
- Manual re-keying 22
- mappings 204
- Master Boot Record 19, 43
- Master Key 6, 43
- master key 17–18, 25, 36, 47, 52, 84, 216
- master key actions 25
- Master key backup 47
- Master key generation 47
- Master Key maintenance 211
- master key status 25
- Master Keystore 51
- master keystore 51–52
- MBR 19, 23, 43
- member node 16
- metadata 40
- MK 36, 43, 47
- MPIO 27, 111
- MPIO failover 32
- MPIO functionality 33

- multi fabric 110
- multi path 110
- multipath I/O 27, 111
- multi-path LUN 178
- multi-path LUNs 26
- multipathing requirements 178
- multiple CTC 123
- multiple paths 26, 178
- mutual authentication 17

N

- Name 177
- name server 24
- name server database 173
- Name Service 173
- name service 173
- National Institute of Standards and Technology 6
- Native Encryption 127
- new paths 130
- NIST 6
- No Access 174
- Node CP Certificate 59
- node initialization 59
- Notice 95
- NTP server 42, 48, 96
- NTP time server 132

O

- Offline encryption 21
- Offline re-keying 22
- OLTP 186, 196
- Online encryption 21
- Online re-keying 22
- Online Transaction Protection 186
- Opaque key vault 17
- out of resources 22
- Out-of-Band 43
- Out-Of-Band managed keys 39
- overloaded 197
- overloads 197

P

- passive path 210
- passphrase 70
- password 52
- performance influence 196
- persistently disable 164
- physical target 25
- PID 96, 104
- Pid 95
- plain 3
- plain text 16
- policies 112
- policy configuration 32
- policy setting 187
- Port identifier 95
- port identifiers 96
- power cycles 27

- primary 67–68
- private key 4
- Private network 15
- private network connections 32
- process terminology 13
- protects 37
- protocol layers 2
- public key 4
- public key encryption 4
- public-private key 4

Q

- queue 22
- quorum 47, 213
- quorum size 18

R

- RC4 3
- read-only mode 203
- read-only state 203–204
- read-verification 43
- records 174
- recovery card 18
- recovery card set 25
- recovery cards 213
- recovery key 6
- recovery purposes 70
- recovery site 204
- recovery smart cards 211
- Redbooks website 235
 - Contact us xi
- redirect zone 100
- redirect zones 158
- redirecting data flow 105
- redirection zone 24, 26, 173–174
- redirection zones 172
- redundancy 27, 42
- register 61
- register member node 60
- register member nodes 81
- register the discrete device serial number 62
- registered state change notification 24
- Rekey 182, 186, 196
- rekey 196
- re-key functions 44
- re-key operation 96
- rekey operation 190
- rekey operations 210
- rekey session 190
- re-keying 22
- rekeying options 187
- rescan 130
- response time 196
- response times 186, 196
- restart 54
- restore 37
- Rivest-Shamir-Adleman 4
- route 172
- RSA 4

- RSCN 24
- rules 16

S

- safe key 15
- safe key exchange 70
- SAN 14
- secondary 67–68
- secure 3
- secure exchange 17
- security exposure 36
- segment 174
- Self-Signed Server Certificate 53
- self-signed-certificates 46
- sequential process 23
- sequential workload 186
- serial number 61–62, 83
- Serpent 3
- server load 155
- serving of keys 7
- simplified key management solution 37
- single fabric encryption process 90
- slot number 61
- smart card 70
- smart card drivers 213
- smart cards 18, 25, 47, 213, 216
- Smartcard Reader 6
- snapshot 199
- snapshot volume 199
- sniff 2
- sniffed 2
- solution design 195
- special node 16, 59
- SSRE 38
- stand-alone 46
- standby 132
- state synchronization 46
- status indicator 67
- status information 27
- stores 37
- switch reboots 27
- symmetric key 71
- symmetric key encryption 3
- symmetric keys 37, 40
- synchronization 43
- synchronization of keys 96
- synchronized 42
- System Shell Runtime Environment 38

T

- target 173
- target container 22
- target LUN 25, 96–97
- target port 18, 25, 123
- target port containers 97
- target ports 119
- TDES 3
- terminology 13
- thin-provisioned volumes 208

- third site 204
- third-party 206–207
- third-party storage 207
- threat 2
- time zone 96
- TIP 8, 37, 39–40
- Tivoli Integrated Portal 8, 37, 40, 51
- Tivoli Key Lifecycle Manager 17, 35, 40–41
- TKLM 7, 13, 17, 35
- TKLM initial configuration 51
- TKLM key vaults 96
- TKLM Self Signed server certificates 50
- TKLM Server 23
- TKLM server 43
- TKLM server certificates 67
- TKLM server file system 67
- TKLM servers 137
- traffic 26, 97, 137, 179, 197
- traffic disruption 173
- traffic flow 182
- transfer method 64
- trunks 46
- trust 66
- trusted 53
- Trusted key vault 17
- trusted link 17
- Twofish 3

U

- unauthorized agent 36
- unencrypted data 16
- unprotected 3
- upgrade 210
- UUID 57

V

- vdisk_UID 148
- VI 25–26, 96, 104
- virtual devices 18
- Virtual Initiator 96, 104, 174
- virtual initiator 25, 173
- virtual initiators 158
- virtual IP address 46
- virtual PIDs 96
- Virtual Target 96, 104
- virtual target 25, 104, 172–173
- virtual targets 158
- virtual volume 207
- virtual WWNs 174
- virtualized server 38
- virtualizing 206
- VMware 39
- VT 25–26, 96, 104

W

- WebSphere Application 56
- WebSphere® Application Server 40
- workload 196

- World-Wide-Nodename 60
- wrapping key 6
- write activity 105
- wsadmin 56
- WWNN 60

Y

- Yes virtual 174

Z

- z/OS 38
- zeroize 58
- zeroized 25



Implementing the Storwize V7000 and the IBM System Storage SAN32B-E4 Encryption Switch

(0.5" spine)

0.475" <-> 0.873"

250 <-> 459 pages



Implementing the Storwize V7000 and the IBM System Storage SAN32B-E4 Encryption Switch



Redbooks®

Learn how to combine the power of virtualization and encryption

See the steps that are involved in a successful implementation

Read about the fundamentals of encryption

In this IBM Redbooks publication, we describe how these products can be combined to provide an encryption and virtualization solution:

- ▶ IBM System Storage SAN32B-E4 Encryption Switch
- ▶ IBM Storwize V7000
- ▶ IBM Tivoli Key Lifecycle Manager

We describe the terminology that is used in an encrypted and virtualized environment, and we show how to implement these products to take advantage of their strengths.

This book is intended for anyone who needs to understand and implement the IBM System Storage SAN32B-E4 Encryption Switch, IBM Storwize V7000, IBM Tivoli Key Lifecycle Manager, and encryption.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks