OpenVMS Technical Journal V16

Data Encryption Using Archive Backup System

Nagendra Shastry

About this document

This document provides information about the following topics:

- Using software encryption with the Archive Backup System (ABS)
- Using hardware encryption with the ABS
- Best practices and limitations

Intended audience

This document is intended for:

- Administrators who have implemented the ABS backup solution in their environment.
- Administrators who need a software or hardware based encryption solution in their ABS environment.

Prerequisite for using this document

- Knowledge of ABS and MDMS objects (Refer: ABS Operation Guide and Reference Guide).
- Knowledge of taking backup using the ABS saves and restoring data using the ABS restore operations (Refer: ABS Operation Guide).

Introduction

Encryption overview

Encryption is derived from the Greek word kryptós, which means hidden. It is the process of concealing information from unauthorized parties by means of a mathematical cipher. It is an algorithm that disguises the underlying text unless the reader has the de-cipher code. In encryption technology, the cipher is a complex mathematical algorithm that is applied to the unencrypted data, also known as "plain text", to produce encrypted data known as "cipher text".

A simple example of using a cipher is alphabetical substitution such as replacing each letter of the alphabet with a different letter. For example, A=D, B=I, C=J, and D=Z. Unfortunately this method is very limited, as a quick analysis of the letter frequency would easily reveal the substitution code used. More complex ciphers change the substitution each time it is used. For example, AAAA is encrypted as DFGT or similar random set of characters instead of HHHH, the simplified A=H substitution. The complex class of cipher is known as poly-alphabetical. Clearly far more complex mathematical algorithm ciphers are used today to ensure that the code cannot be broken by force.

To decrypt a message, a key is required and the correct mathematical algorithm. When the key is changed, the cipher completely alters the substitution sequence used. The correct key is required for recovering the original plain text.

Encryption is a way to make data unreadable to others while still allowing authorized users to access it. Encryption requires the user or system to have a specific key and software to encrypt and decrypt the data. Encryption uses various mathematical algorithms for transforming plain text into cipher text and back again. The strength of the encryption depends on the type of algorithm and the key used to encrypt the data.

There are two types of algorithm:

- Defense Encryption Standard (DES)
- Advanced Encryption Standard (AES)

For more information about the DES and AES supported algorithm, see the section "Support matrix" of this document.

Encryption in ABS

This section provides information about the supported encryption types in ABS. It also provides information on how the key management facility is used to generate and maintain the used key for software-based encryption and hardware-based encryption.

ABS Version V4.5 and later supports the following types of encryption:

- Software encryption
- Hardware encryption

Note:
The key management facility is supported only on software encryption. For hardware encryption, you need to use the supported hardware for maintaining the key.

Software encryption

Software encryption is a method where the data is encrypted before leaving a server for writing on the targeted storage medium such as a disk or a tape. In ABS, data is encrypted by underlying the BACKUP utility. Since software encryption happens in the host system, ABS requires extra CPU cycles apart from reading and writing the data.

ABS supports data encryption in the following ways:

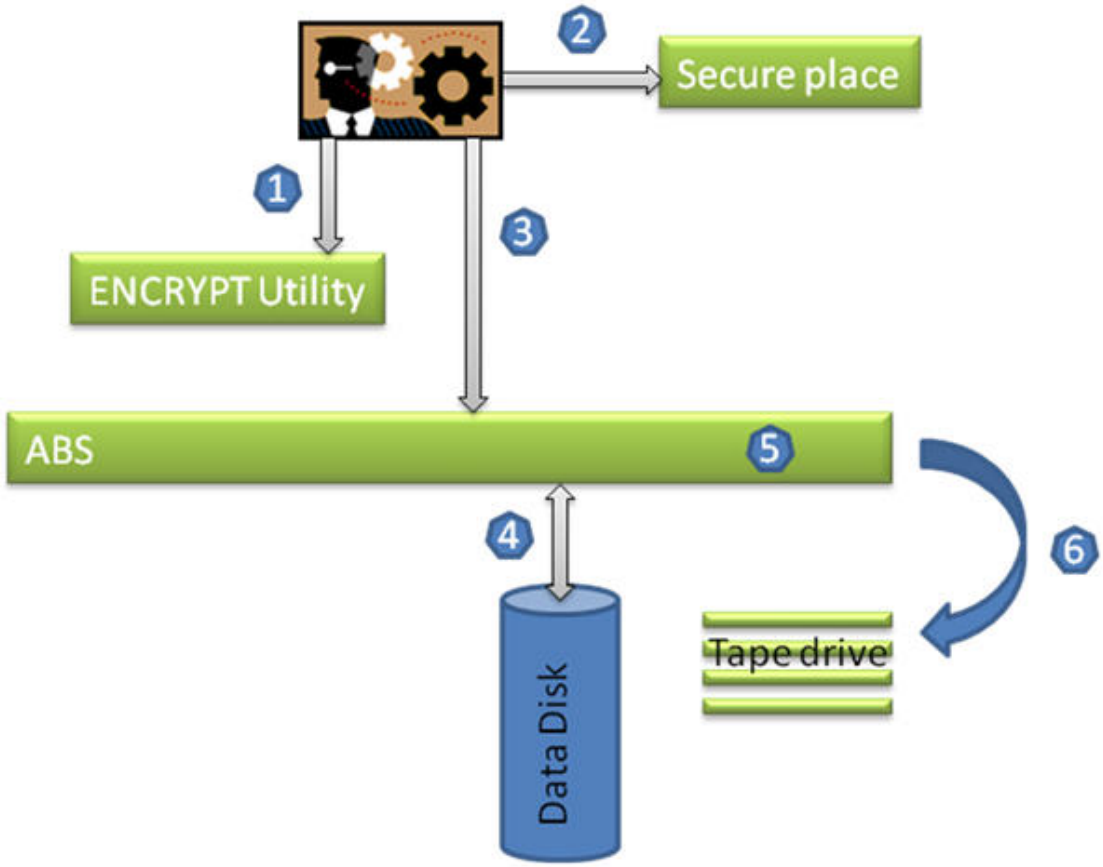
- Encrypting data without key management
- Encrypting data with key management

How to use software encryption without key management?

In this method, you have the flexibility of creating a key and passing the key to ABS to encrypt the data, using the required encryption algorithm. The key used for encryption has to be stored securely by you for future restore operation. ABS does not maintain encryption key. If you lose the key, ABS will not be able to restore the data. If you choose to pass the key name or key value, then the key must be created using the ENCRYPT utility on an OpenVMS node/host system where the key is being used with the ABS. Alternatively, you can pass the key value directly to the ABS without creating the key name using the ENCRYPT utility.

Note:
HP recommends using key name to pass a key value to ABS for encryption instead of directly passing the key value to ABS, because the key value gets displayed in the ABS save/restore log file.

The following figure demonstrates how encryption happens without key management.



1. Generates key by using the ENCRYPT utility.
2. Stores generated key in a secure place.
3. Passes the generated key to ABS.
4. Reads the data from the Data Disk.
5. Encrypts the data using the key passed by the user.
6. Passes the encrypted data to tape drive for writing on to the tape.

The following example explains how to use ABS to encrypt the data without key management. There are two methods of encrypting data without using the ABS key management. In both the methods you need to manually pass the key or key name to ABS.

1. Specify the key to use for encrypting in the selection object. This makes the key visible to any user.

Or

2. Create a key and associate a name to that key so that the key name can be used to specify in the selection object.

Note:
HP recommends using the second method because this is more secure to create a key. Hence, only the second method is explained in this document.

The following example explains how to create a key and associate the key with a key name, and how to use the key name to encrypt the data using ABS.

1. Create an encryption key and associate a key name to the key.
   ```
   $ ENCRYPT/CREATE_KEY/PROCESS DATADISK123_KEY_NAME  01234567890
   ```
   This commands provides you an example of how the ENCRYPT utility is used to create a key by assigning a key name for encrypting the data by hiding the key value. You can create the key name either in the JOB, PROCESS, or SYSTEM logical table depending on your security and backup policy. Specify the following qualifiers to control the key name:

   - /JOB The key name is created in the JOB wide table.
   - /PROCESS The key name is created in the PROCESS wide table.
   - /SYSTEM The key name is created in the SYSTEM wide table.

2. Specify the key name DATADISK123_KEY_NAME in selection object along with the algorithm to be used to encrypt the data using the key.
   ```
            Selection: ENCRYPTION
          Description:
       Access Control: NONE
                Owner: TEST_NODE::TEST
     Agent Qualifiers:/ENCRYPT=(NAME=DATADISK123_KEY_NAME, ALGO=DESCBC)
          Before Date: NONE
     Conflict Options: RETAIN_VERSION
     Data Select Type: VMS_FILES
            Date Type: MODIFIED
              Exclude:
              Include:
           Since Date: NONE
          Source Node:
   ```
3. Use this modified selection object to run the save or restore operation.

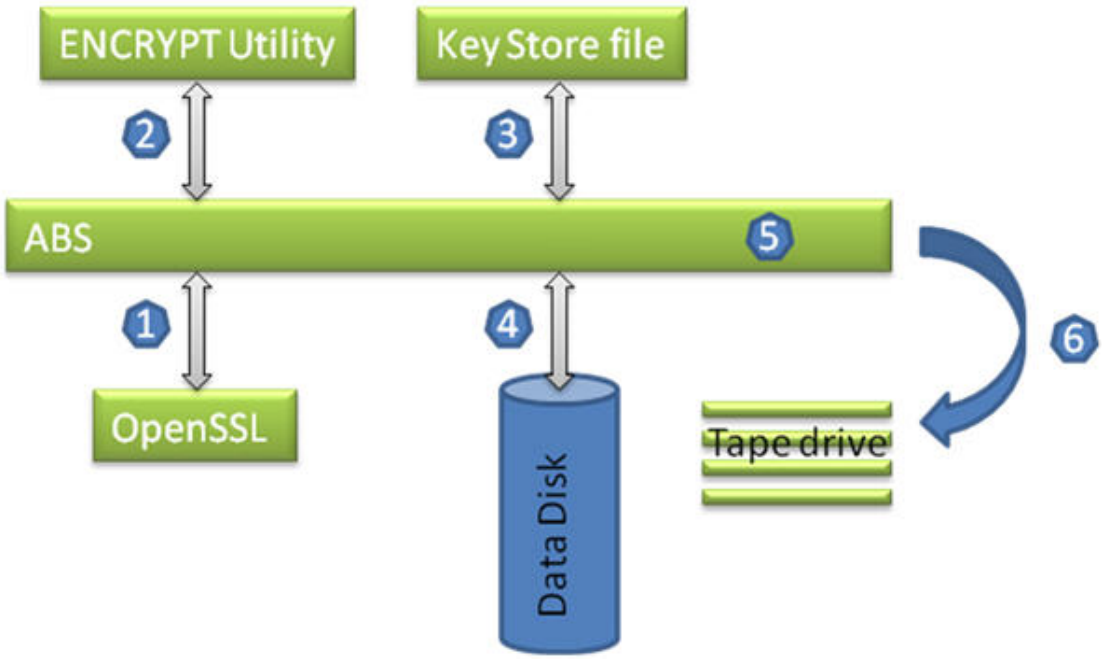
Note:
In this method you have to maintain a key in a secure place for future restore operation. ABS does not maintain the key in its database. If a key is lost, then the data cannot be restored.

How to use software encryption with key management?

It is difficult to keep track of the all the keys used and to create keys manually for every save and associating the key for restore operation is cumbersome. ABS provides an interface to automatically encrypt the data. This interface provides the following functions:

- Generating a key.
- Using the generated key for save operation without user intervention.
- Storing the key in a secure place for future restore operation.
- Retrieving correct key from the key file while restoring encrypted data.

The following figure demonstrates how encryption happens with key management.



1. Makes request to OpenSSL to generate a unique key for encryption.
2. Passes a unique random number to the ENCRYPT utility to create a key for encrypt operation and assign a key name to the key.
3. Stores the generated unique random number in the *key store file*.
4. Reads data from the Data Disk.
5. Encrypts the data using the key.
6. Passes the encrypted data to the Tape drive for writing on to the tape.

The *key store file* name ABS$ENCRYPTION_<node_name>.DAT. <node_name>, is the name of the node/host where the ABS is installed. This file is placed in the ABS$SYSTEM directory.

Steps for running save operation

1. Define a logical "ABS_MDMS_ENCRY_ALGO" with encryption algorithm name for encrypting a data.
   ```
   $ DEFINE/SYS ABS_MDMS_ENCRY_ALGO DESCBC
   ```
2. Include the following line in the save prologue, to generate and store the key:
   ```
   $ RUN SYS$SYSTEM:ABS$ENCRYPT_SAVE.EXE
   ```

3. Modify the selection object to include the /ENCRYPT qualifier with a key name and the algorithm to use for encryption, as given in the following:

```
         Selection: ENCRYPTION
       Description:
    Access Control: NONE
             Owner: XION::NAGENDRA
  Agent Qualifiers:/ENCRYPT=(NAME=ABS_MDMS_KEY,
ALGO='ABS_MDMS_ENCRY_ALGO_TYPE)
       Before Date: NONE
  Conflict Options: RETAIN_VERSION
  Data Select Type: VMS_FILES
         Date Type: MODIFIED
           Exclude:
           Include:
        Since Date: NONE
       Source Node:
```

Note:
" 'ABS_MDMS_ENCRY_ALGO_TYPE" is a symbol and not a logical name. Do not forget to include the character " ' " in the symbol.

4. Ensure that this modified selection is specified in the save object before running a save.
5. Run the save operation.

Steps for running restore operation

Note:
For restoring an encrypted data, you do not need to define a logical "ABS_MDMS_ENCRY_ALGO". This is taken from the *key store file* along with the key.

1. Include the following line in the restore prologue to retrieve the required decryption key from the *key store file*:
   ```
   $ RUN SYS$SYSTEM:ABS$ENCRYPT_RESTORE.EXE
   ```
2. Modify the selection object and include the /ENCRYPT qualifier with a key name and the algorithm to use for encryption.

```
         Selection: ENCRYPTION
       Description:
    Access Control: NONE
             Owner: XION::NAGENDRA
  Agent Qualifiers:/ENCRYPT=(NAME=ABS_MDMS_KEY,
ALGO='ABS_MDMS_ENCRY_ALGO_TYPE)
       Before Date: NONE
  Conflict Options: RETAIN_VERSION
  Data Select Type: VMS_FILES
         Date Type: MODIFIED
           Exclude:
           Include:
        Since Date: NONE
```

```
Source Node:
```

3. Ensure that this selection is specified in the restore object before running a restore operation.

Hardware encryption

Like software encryption, hardware encryption can also be used to encrypt data that is stored on a tape medium to secure the data. Unlike software encryption, the data in the hardware encryption is encrypted on a tape drive. Hence, the encryption process overhead will not be there in the host processor and the server CPU cycles can be used for other processes.

Prerequisites for using hardware encryption with ABS

- Tape drive and cartridge must be LTO-4
- ABS supports the following hardware as key generator and manager:
    - Secure Key Manager (SKM)
    - Product Level Key (PLK)

Note:
In hardware encryption method, the encryption happens in the LTO-4 tape drive. ABS does not know that the data is encrypted and hence, ABS does not maintain the key used for encryption. The SKM/PLK is responsible for storing and retrieving a key. If a key is lost, then ABS will not be able to restore the data.

An overview of the Linear Tape Open-4

Linear Tape-Open (LTO) is a magnetic tape technology developed as an open alternative to the proprietary Digital Linear Tape (DLT). LTO-4 is the fourth generation of standardization for tape drive and tape media.

LTO-4 tape drive features

The LTO-4 tape drive supports the following features:

- Reads and writes data to an LTO–4 tape medium.
- Encrypts data using the 256-bit AES-GCM mode or drive-based encryption, which is currently supported only on an LTO–4 tape medium.
- Reads and writes data to an LTO–3 medium without encryption.
- Reads data from an LTO–2 medium.

For more information on the LTO-4 tape drive features, see
http://h71028.www7.hp.com/ERC/downloads/4AA1-4878ENW.pdf.

LTO-4 tape medium feature

The LTO-4 tape medium supports 800 GB of native capacity. You can store data in the range of 800 GB to 1600 GB (1:2 compressions) by compressing the data before it is stored on the tape medium.

An overview of the Secure Key Manager

Secure Key Manager (SKM) is a key generator with a secure and centralized encryption key management solution for HP LTO4 enterprise-tape libraries. Using SKM, the key generation and

management can be automated according to the security policies for multiple libraries. All the process in SKM occurs transparent to ISV backup applications. SKM provides reliable lifetime key archival with automatic multi-site key replication, high availability, clustering, and failover capabilities.

SKM features

The SKM supports the following features:

- Centralized encryption key management for HP LTO4 enterprise tape libraries:
    - Automatic policy-based key generation and management supporting key/cartridge granularity
    - ISV transparent key archival and retrieval for multiple libraries
    - Extensible to emerging open standards
- Strong auditable security for encryption keys:
    - Hardened server appliance
    - Secure identity-based access, administration, and logging
    - Designed for FIPS 140-2 validation
- Reliable lifetime key archival:
    - Automatic multi-site clustering, key replication, and failover
    - Comprehensive backup and restore functionality for keys
    - Redundant device components and active alerts

SKM encryption kit requirement

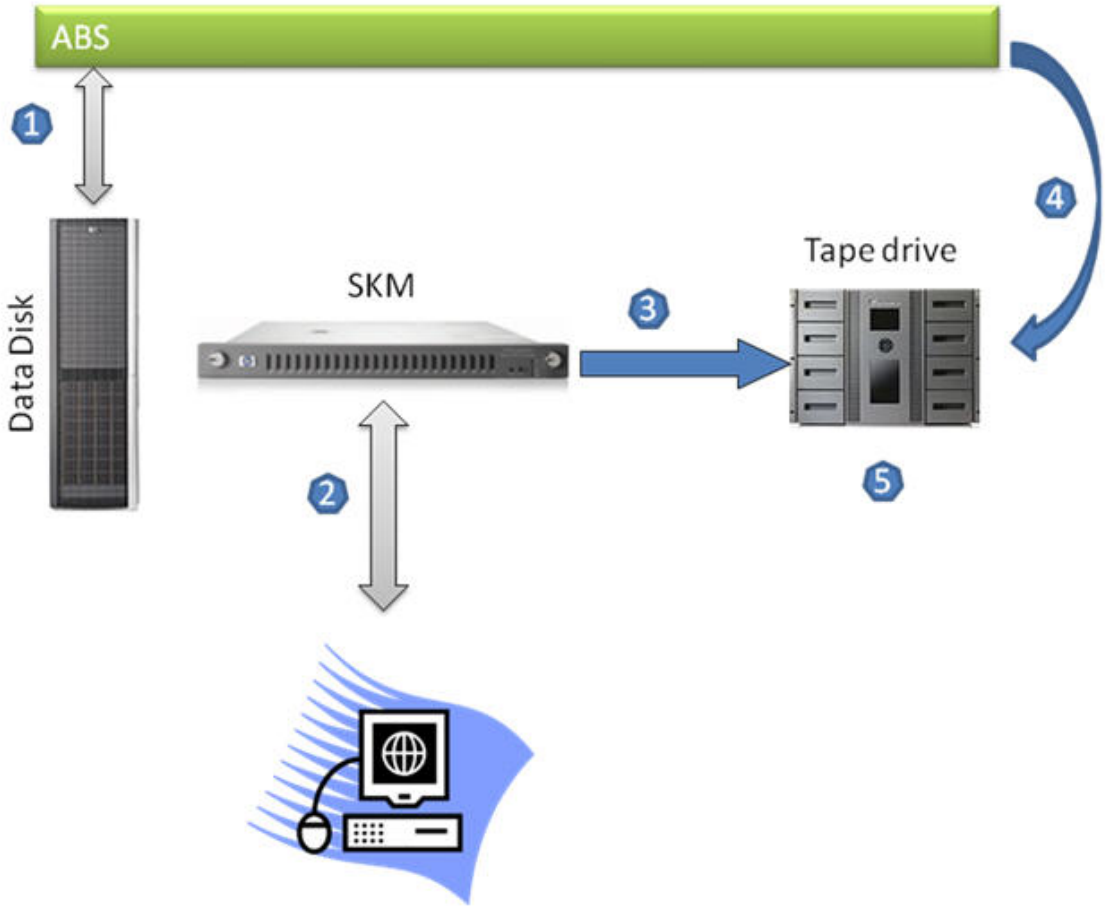
- Compatible Tape Library (CTL) with supported library firmware
- LTO-4 tape drive with supported firmware

Note:

Configuration of SKM and how to use SKM according to the security policy is beyond the scope of this document. For more information, see the *SKM User Guide*.

Hardware encryption using the SKM and LTO-4 tape drive

The following figure demonstrates how hardware encryption happens with LTO-4 and SKM.



1. ABS reads data from the Data Disk for backup operation.
2. Configures SKM for key generation according to the security policy.
3. SKM passes the unique key generated to LTO-4 on request for encrypting data passed by ABS before writing the data on an LTO-4 tape media. SKM stores the key used for encryption for future restore operation.
4. ABS passes data to LTO-4 tape drive in library for writing on to LTO-4 tape cartridge.
5. LTO-4 tape drive first encrypts the data received from ABS using the key passed by SKM and then writes encrypted data on LTO-4 cartridge.

Product Level Key (PLK)

Product Level Key (PLK) is a USB device, which is capable of generating and storing a key used for encrypting data on the LTO-4 tape drive. PLK provides an encryption solution to MSL libraries and Autoloader.

PLK features

The PLK supports the following features:

- Provides access to encrypt and decrypt capabilities of LTO-4 drives.
- Provides secure generation and storage for up to 100 LTO-4 encryption keys on a token.
- Uses random number generator, strong password authentication, and digital envelopes.
- Provides a method for backing up, restoring, and transferring keys.

Operational overview

- With the PLK Encryption Kit, keys are stored on a key server token and are protected by a password (the token's PIN).
- Only one encryption key is used on a cartridge.
- To write encrypted data, you must have the key server token and the password for the key server token.
- If the cartridge contains previously-encrypted data, the key server token with the key for the tape must be in the Autoloader or library.
- To read encrypted data, you must have the key server token with the key for the tape and the password for the key server token.
- Neither the key server token nor the cartridge tapes track the association between the encryption key and the tape. Therefore, it is important to know which token was used with each tape.

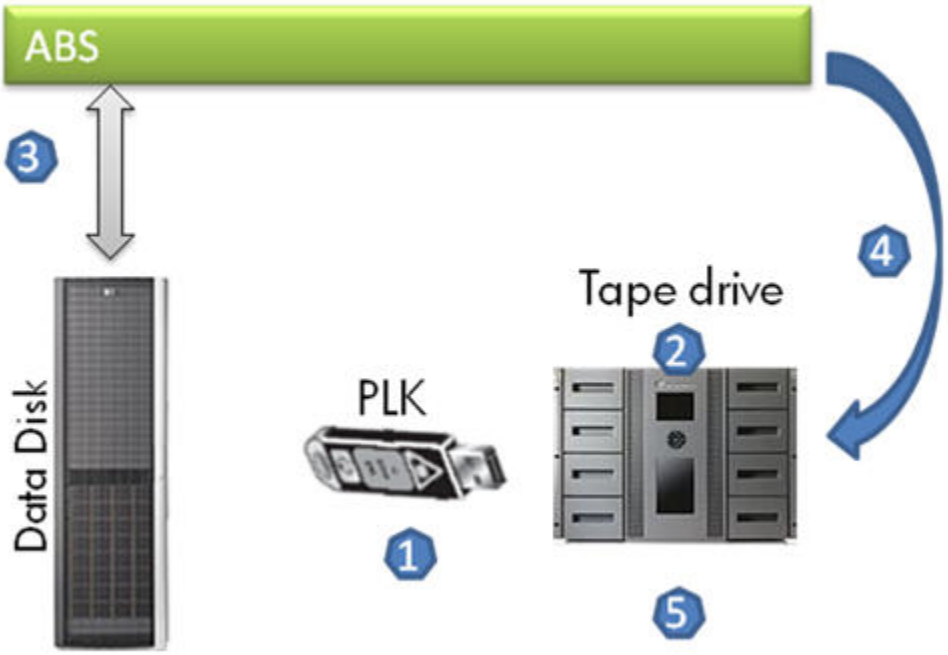
PLK Encryption Kit requirement

- A CTL or Autoloader with supported library/autoloader firmware.
- A Tape Library or Autoloader with at least one LTO-4 tape drive with supported firmware.
- An accessible USB port on the back of the Tape Library or Autoloader.

Hardware encryption using the PLK and LTO-4 tape drive

PLK is very similar to USB thumb drive, with key generation and storing capabilities.

The following figure demonstrates how hardware encryption happens with LTO-4 tape drive and PLK.



1. Insert the PLK to USB port of library/loader.
2. Specify the policy to create key in the library/loader console.
3. ABS reads data from the Data Disk.
4. ABS passes data to the Tape drive to store on tape.
5. Tape drive in library/loader encrypts the data received from ABS, using the generated key.

Note:
For more information on how to use PLK and supported hardware/firmware, see the *PLK User Manual*. These topics are outside the scope of this document.

Limitations

- In software encryption using ABS, the key management is not centralized; separate *key store file* is maintained per ABS/MDMS client, including server.
- You cannot decrypt the data if the key used for encryption is not available in the *key store file* where the data restore operation is taking place. In this case, the *key store file* must be copied to a node where you want to restore the encrypted data.
- ABS generates a unique key for each saveset. ABS does not support other type of key generation policy like key per cartridge or key per library and so on.
- Software encryption reduces the effectiveness of drive-based compression.

Support matrix

The following table provides the algorithms supported on an OpenVMS version.

OpenVMS version	Hardware platform	Supported algorithms
OpenVMS 7.3-2	Alpha	DESCBC, DESECB, DESCFB
OpenVMS 8.2/8.3/8.4	Alpha	DESCBC, DESECB, DESCFB, AESCBC128,AESCBC196,AESCBC256,AESECB128, AESECB196,AESECB256,AESCFB128,AESCFB196, AESCFB256,AESOFB128,AESOFB196, AESOFB256
OpenVMS 8.2-1 /8.3/8.3-1h1/8.4	Integrity servers	DESCBC, DESECB, DESCFB, AESCBC128,AESCBC196,AESCBC256,AESECB128, AESECB196,AESECB256,AESCFB128,AESCFB196, AESCFB256,AESOFB128,AESOFB196, AESOFB256

SSL version support

The OpenVMS System must have the compatible SSL version that is compiled with ABS. The ABS V4.5-1200 supports SSL V1.3 and ABS V4.5-1201 supports SSL V1.4. See the *ABS Release Notes* for supported SSL version with a particular ABS version.

Encryption support

Software encryption is supported from ABS V4.5-1200 onwards, and hardware encryption is supported from ABS V4.4A-1001 onwards. See corresponding version of ABS documents to get supported hardware list.

Best practices

Software versus hardware encryption

- Avoid using both software and hardware encryption for the same data. This increases the security, but it has performance overhead on software side.
- Drive-based hardware encryption is best in a situation where software and hardware encryption is available. Encryption of data happens on tape drive, which improves performance and requires no host CPU cycles to encrypt.

Securing *key store file*

For software encryption

Back up the *key store file* regularly to a secure place for successful restore operation. If a key is lost, the ABS and HP cannot help in restoring the encrypted data.

For hardware encryption

Automatic multi-site clustering, key replication, and failover feature of SKM are used to synchronize two SKM. At any time, if one SKM is not accessible, then the key can be accessed from the other SKM. For more information, see the *SKM Configuration Guide*.

PLK supports backing up of keys to a file that can be copied to another library for restore operation. HP recommends taking backup of keys to a file for security reason. For more information, see the *PLK User Manual*.

Handling backup and restore across multiple clients (hosts)

The key generated in each node/host for encrypt operation is placed in the same host in the *key store file*, for restoring encrypted data from different node/host. Hence the *key store file* has to be copied to a node/host, where the restore operation has to be performed; this will enable ABS to get the correct key for restoring encrypted data. The *key store file* must be placed in the ABS$SYSTEM directory.

Troubleshooting

The following table provides the list of errors while using the Encryption feature with key management.

Error message	Possible cause	Recommended steps
Error: Memory allocation failed!	There is no sufficient physical memory to reserve space for encryption key generation/store/retrieve operation	Stop some applications so that encryption process can get free memory for its operation
Error: Encryption Key define failed!	Some error occurred while defining key used for encryption operation	Make sure that the ENCRYPT utility is started and running
Error: Failed to open key file!	Encountered problem while trying to open the *key store file* for the read or write operation	Make sure that the *key store file* is not deleted or moved to different location or renamed
Error: Failed to connect key file!	Not able to open a channel for the read or write operation	Check the process quota and the number of channel it can open
Error: RMS Put operation failed!	Failed to write key into the *key store file*	Check if there is sufficient space in the disk where the *key store file* is placed
Error: Failed to close the key file!	Problem encountered while closing an open *key store file*	Check if any save/restore operation is hanging, if yes, stop jobs and start the save or restore operation
Error: Algorithm logical translation failed!	Encountered problem while translation of algorithm logical	Check that the logical ABS_MDMS_ENCRY_ALGO is defined properly
Error: Failed to translate ABS Object	Failed to get thread number	Some error for unknown problem. Report to Engineering after

number!	of the save or restore operation	collecting log file of the save or restore operation, and save, restore, environment, archive objects information
Error: Failed to translate saveset name!	Failed to get saveset number while running the save or restore operation	Some error for unknown problem. Report to Engineering after collecting the save or restore operation failed log file, save, restore, environment, archive object output
Error: Failed to translate saveset location!	Failed to recognize saveset location on a disk or tape name	Some error for unknown problem. Report to Engineering after collecting the save or restore operation failed log file, save, restore, environment, archive object output
Error: OpenSSL API Failed!	Problem while establishing communication with SSL	Check whether Open SSL is started and running
Error: SYS$DEQ Failed!	Not able to unlock the *key store file*	Report engineering with the save or restore log file. Try to take process dump of running save or restore job
Error: LIB$SET_SYMBOL Failed!	Encountered problem while defining symbol to pass encryption algorithm	Report Engineering with the save or restore log file. Try to take process dump of running save or restore job
Error: Failed to translate SYS$NODE logical!	Not able to get a node name	Make sure that SYS$NODE is defined in the system wide table in failed node
Error: SYS$PARSE Failed!	Encountered an error while accessing *key store files* while executing restore operation	Make sure that the required *key store file* is present in the ABS$SYSTEM directory. If the problem is still present, then take process dump of restore job and pass to Engineering

Error: SYS$ENQW Failed!	Encountered problem while getting lock on *key store file* for the read write operation	Report to Engineering with the save or restore log file. Collect process dump of the save or restore job

Frequently asked questions

Can I execute BACKUP/LIST to list content of saveset?
No, ABS encrypts header along with the data. Decryption key has to be supplied along with this command.

What are the advantages of using AES over DES?
AES is a cryptographic algorithm that protects sensitive and unclassified information. The National Security Agency (NSA) reviewed all the AES finalists, including Rijndael, and stated that all of them were secure enough for the U.S. Government non-classified data. In June 2003, the U.S. Government has announced that AES can be used for classified information by stating the following:

"The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use."

Is it possible to restore data if the *key store file* is deleted?
No. If the *key store file* is lost you cannot restore data, even HP cannot restore it.

For more information

Firmware links
http://www.hp.com/support

About Security
http://h71028.www7.hp.com/enterprise/us/en/solutions/storage-security.html

About Encryption
www.hp.com/go/dataencryption

Tape solution
www.hp.com/go/tape

ETLA Tape Libraries
www.hp.com/go/ETLA

Enterprise Backup Solutions
www.hp.com/go/EBS

Advanced Encryption Standard at the website
http://csrc.nist.gov/archive/aes/rijndael/

Linear Tape-Open at the website
http://www.lto.org/newsite/html/news_4_17_06.html

Encryption Technology for the HP StorageWorks Ultrium LTO4 tape drive white paper at the website
http://h71028.www7.hp.com/ERC/downloads/4AA1-4878ENW.pdf

ABS Support Matrix
http://h71000.www7.hp.com/openvms/storage/smstape_matrix.html

References

ABS Documents
http://h71000.www7.hp.com/doc/abs.html

Secure Key Manager
http://h18000.www1.hp.com/products/storageworks/secure_key/index.html

Secure Key Manager White Paper
http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA2-1403ENW.pdf

Federal Information Processing Standards Publications. 1996. Federal Information Processing
http://www.itl.nist.gov/fipspubs