AD Image Encryption

Format Version 1.2

17 May 2010

Table of Contents

Introduction

AccessData delivers forensic tools that allow the creation and analysis of disk images and AD1 (custom content) images for use in criminal and internal investigations. One of the concerns during the investigation process is protecting sensitive information that is collected inside these images. Security protocols must be set up so that only authorized users can access the data.

Many techniques can be used to control access to sensitive information. One technique is to store sensitive data on a secured remote drive and use protocols like Active Directory to restrict access to certain individuals. Another technique is to encrypt image files so they can only be accessed by those who have the necessary credentials (passwords, etc.). They may be encrypted directly using tools such as WinZip, or they may be encrypted transparently using techniques such as EFS (Windows filesystem-level encryption) or by creating a TrueCrypt volume to store them on.

Unfortunately, these techniques can be error-prone, costly, or hard to work with. For example, network shares can be difficult to set up and maintain, and the data gets transmitted and stored in unencrypted form. Direct encryption has a similar problem in that the file must exist on disk in unencrypted form first, and for analysis most forensic tools require the file to be decrypted, so it is stored on disk in unencrypted form before it can be analyzed. Using transparent encryption like EFS solves this problem, but EFS files cannot easily be set up for allowing multi-user access, nor can they be transmitted in encrypted form. Virtual disks such as TrueCrypt are a great solution for many people, but setting them up can be difficult, especially when enabling multi-user access over a network. The network authentication must be handled separately and uses different credentials than the virtual disk.

To address these concerns AccessData has implemented an easy-to-use, built-in solution for encryption called **AD Encryption**. Encrypted images can be created and analyzed directly using AccessData forensic tools. At the current time, AccessData ships two applications with the capability of creating and viewing AD encrypted images: Imager and FTK-3. This document provides a description of the AD encryption algorithm and file format.

Overview

AD Encryption is whole image encryption. Not only the raw data (e.g., sector data) is encrypted, but also any metadata inside the image. Each segment (file) of the image is encrypted with a randomly-generated **image key** using strong encryption (AES-256). The image key is protected by desired credentials, a password or a cert public key. The encrypted image key is prepended to the front of the first image segment.

Image Formats

AD Encryption is currently enabled for E01, S01, and raw/dd disk images, and for AD1 images. More image formats may be supported in the future, as the design is not tied to any specific format.

Keys

The following key algorithms are supported by AD encryption:

- AES-128
- AES-192
- AES-256 (default)

The following hash algorithms are supported by AD encryption:

- SHA-256
- SHA-512 (default)

The user interface in FTK and Imager currently do not allow the user to specify the key and hash algorithms, so the defaults of AES-256 and SHA-512 are always used.

Credentials

AD encryption supports either a password or cert to protect the image key.

Password:

A password is any word or phrase and can contain spaces and any Unicode characters. The password is converted to UTF-8 for hashing.

Certificate (public/private keys):

A certificate is a standard X.509 cert inside a container like PKCS#12 or PEM. Many tools can be used to create certificates (e.g., Certmgr.exe). A certificate often contains both a public key and a private key, but they can also be stored separately. The public key from a cert is used for encryption, and the private key is used for decryption.

Certificates

The following X.509 certificate formats are supported by AD encryption:

- PKCS#12 / PFX (*.p12, *.pfx)
- PEM (*.pem)
- Stand-alone public key only (*.cer, *.crt, *.der)

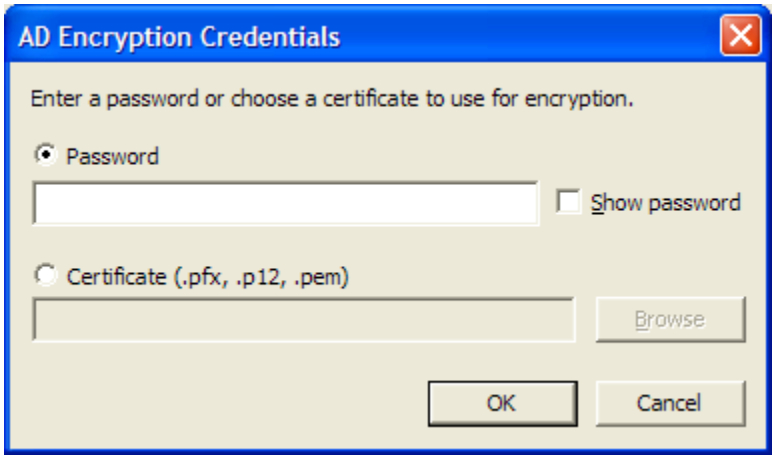
Only certificates containing RSA keys are supported. Encrypted certs are supported.

Image Key encryption

The image key is encrypted using a password or a cert. A random salt is generated and then encrypted with the cert public key if provided. A key is then generated using the standard PBKDF2 algorithm (PKCS#5 compliant) from the passwords (if using a password) and the salt. This key is used to encrypt the image key and to compute an HMAC of the encrypted key for verification.

Appendix A – Credentials Screenshot

Here is a screenshot illustrating the dialog used to enter credentials for AD encryption. The same dialog is used for encryption and decryption.



Appendix B - ADEncrypt Header File Format

The ADEncrypt header contains the encrypted image key and is prepended to the first image segment. The ADEncrypt header can be decrypted when the correct credentials are provided.

Offset	Length	Name	Description
00h	8	Signature	"ADCRYPT\0"
08h	4	Version	1
0Ch	4	Total size	Header size + salt length + encrypted key length + HMAC length + padding
10h	2	Pass count	# of password; -1 if unknown (default)
12h	2	Raw key count	# of raw keys; -1 if unknown (default)
14h	2	Cert count	# of certs; -1 if unknown (default)
16h	2	Reserved	Reserved; must be 0
18h	4	Key algorithm	1 = AES-128, 2 = AES-192, 3 = AES-256
1Ch	4	Hash algorithm	1 = SHA-256, 2 = SHA-512
20h	4	Iteration count	# of iterations for hashing (default is 4000)
24h	4	Salt length	Length of salt
28h	4	Encrypted key length	Length of encrypted key
2Ch	4	HMAC length	Length of HMAC

The salt, encrypted key, and HMAC immediately follow the header. Then there is padding with zeroes up to the next 512-byte boundary.