mi2b2 User Documentation - Encryption

Encryption

Overview

When downloading studies, users may choose to encrypt the downloaded study, and encryption is on by default. This page explains how encryption in mi2b2 is accomplished. This document serves as a reference for those who want to determine whether the strength and techniques used in mi2b2 encryption is sufficient and suitable for their particular use case. This discussion, however, necessitates some technical details.

If you are looking for information on how to set up encryption key and configure other download settings in mi2b2, please refer to this page instead.

In this document, when we say "random" or "randomly", we mean "pseudo-random" and "pseudo-randomly", that is, the randomness is produced by a deterministic algorithm to simulate true randomness, which can only be reliably obtained via observing natural phenomena.

Encryption Standard

The mi2b2 client is written in Java 6 Standard Edition, and uses its **javax.crypto** package. In this package, a number of encryption algorithms are available. We have chosen to use the AES algorithm with 128-bit long keys.

The AES algorithm is used in Cipher Block Chaining (CBC) mode to prevent potential situations where images may still be partially readable even after encryption. CBC mode requires an initialization vector (a list of random numbers to bootstrap the encryption process), and the initialization vector is randomly generated before each application of the AES encryption algorithm.

A key (randomly generated via the key-generation mechanism or user-provided) and the initialization vector are then used in the encryption process.

Obtaining an AES 128-bit Key

Users have two ways of obtaining a key for encryption (see Key Dialog). The first is to generate a random key automatically. mi2b2 will use the build-in Java cryptographic methods to generate a key and write that key to a file. The file then can be shared among different users to open all images encrypted with that key.

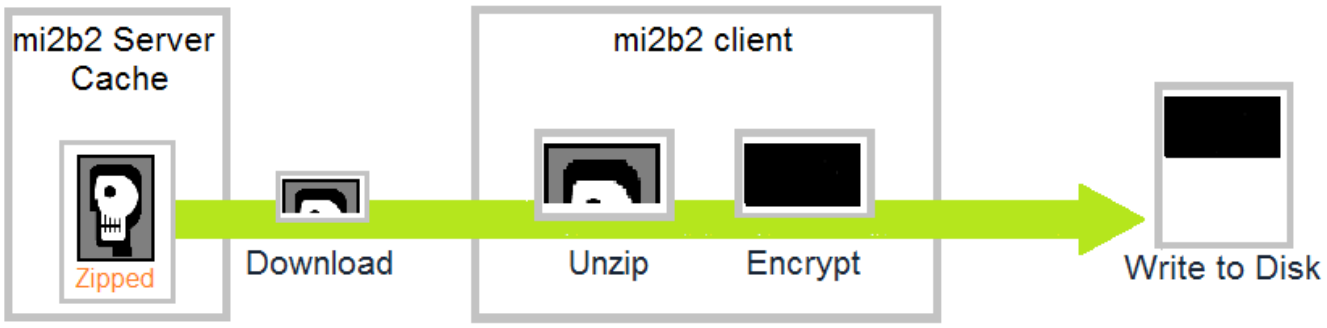
The second way is to use a password that is at least 8 characters long. The mi2b2 client will first generate a 16-byte long salt. The salt is randomly generated using the SHA1PRNG (SHA1 Pseudo Random Number Generator) algorithm. The random number generator is seeded with the user's username. (This is the reason why two users using the same password will generate two different keys.) If SHA1PRNG does not exist on the particular platform, the platform default will be used instead. After the salt is generated, the following process is repeated:

1. The password and the salt are concatenated and saved in a temporary array of bytes.
2. A 256-bit long cryptographic hash (using SHA-256) of the temporary array is computed.
3. The hash is saved as the new password .
4. Restart in the first step.

These steps repeat for 1024 times. The final password is then used as the AES encryption key. The use of salt and the 1024 repetition is to discourage and slow down dictionary and brute force attacks at guessing the password.

Encrypting a Study

When users download a study from the mi2b2 server's cache, the study is copied from the cache, and written to the user's download location. The study resides on the cache as a zip file. As the zip file is being streamed to user's mi2b2 client, the client performs unzipping and encryption as image data streams in. That is, the client does not wait for the entire zip file to complete download before unzipping and encrypting. As soon a sufficient data is available to unzip, the client will unzip, encrypt, and write that chunk of data to disk.



The diagram above depicts the stream process of a chunk of the streamed study from being downloaded, unzipped, encrypted, and finally written to the users' download location.

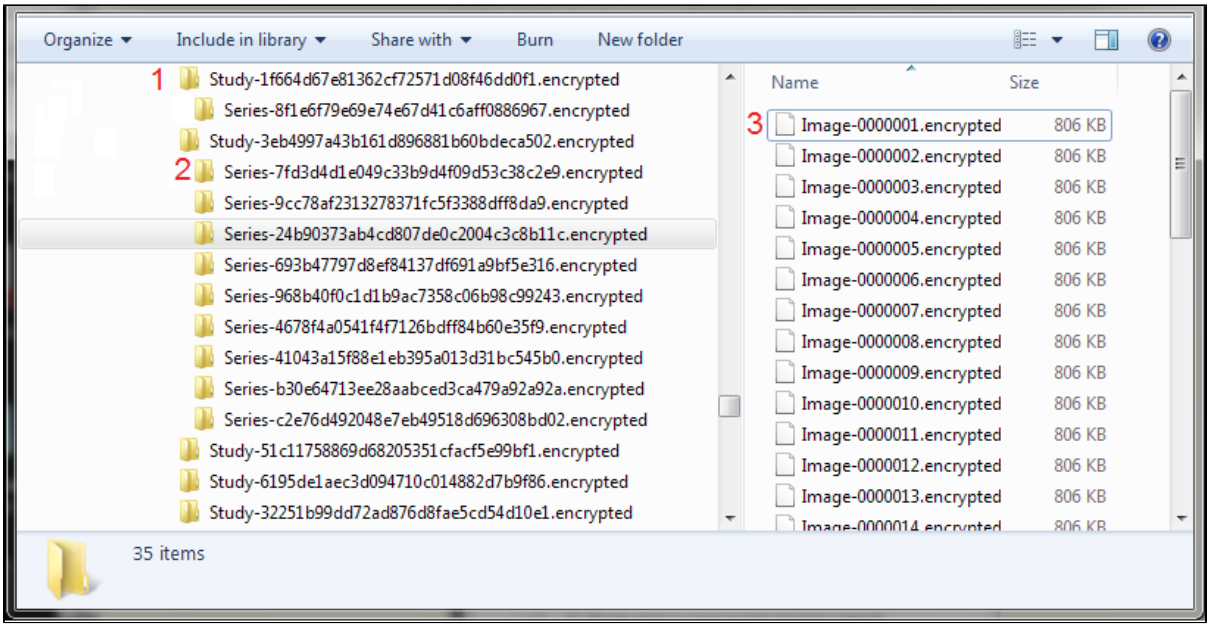
This streaming process ensures that there is not an unencrypted zip file sitting on the user's hard disk at any time. This prevents attackers, for example, interrupting the power supply after downloading completes but before encryption process takes place to obtain unencrypted images.

Writing Encrypted Files

When writing to an encrypted file, a cryptographic hash (secure one-way function) of the key and the initialization vector used in encryption are pre-pended to the output file. This allows mi2b2 to verify if the encrypted key is the same as the key users have chosen when opening an encrypted image without divulging the real key. The initialization vector is used in the decryption process to reverse the encryption and obtain a unencrypted image.

mi2b2 uses SHA-256 to hash the AES key, and SHA-256 produces a hash that is 256 bits (32 bytes) long. The initialization vector is the length of key (128 bits, or 16 bytes) plus 2 more bytes. This means every encrypted image file is pre-pended with 32 + 18 = 50 bytes of information. Otherwise the encrypted image files are exactly the same size as the unencrypted ones.

The zipped study file is organized hierarchically by Study UID (top level), Series ID (mid level), and Image Instance ID. The first two levels are directories, named by their IDs. These IDs are produced using DICOM standards, and those who know how to read the IDs can identify what institution, department, machine, and date/time the particular study started, and use the assembled information to narrow down or identify the patient. To prevent this kind of attack, when encryption is chosen, mi2b2 will rename these studies, series, and images. Studie UIDs and Series IDs will be hashed (using the MD5 cryptographic hash function). The hashed value (16-bytes long) is then written out to be a string of 16 hexadecimal numbers (32 characters, 2 characters per 1 hexadecimal number) and pre-pended with "Study-" or "Series-" appropriately and appended with ".encrypted" (**1**, **2**). The image files, on the other hand, are named "Study-*num*", where *num* stands for the instance number of that image in its series. The image files are also appended with ".encrypted" (**3**). This helps the users identify which files appear before which.



Decrypting Images

The mi2b2 client first reads the first 50 bytes of the encrypted file. It applies the same SHA-256 algorithm to the key users supplied, and compare the result to see if the hash matches with the hash of the key in the file (first 32 bytes). If true, the mi2b2 client then uses the last 18 bytes of the 50 bytes to recreate the initialization vector. The key (supplied by user and verified to match the encrypted key) and the initialization vector are then used to initialize the decrypting algorithm.

The decryption occurs when users select a series from the Image Browser or when users move the image slider in the Image Viewer. For most images, the decryption process is fairly fast. However, users may experience lack of responsiveness for some larger images, some modalities are worse than the others. No decrypted data is written to hard disk during decryption.

Last Page **(Managing Remote Cache)**

1. Quick Start
2. Searching for and Requesting DICOM Studies
3. Downloading and Viewing Studies
4. Managing Repository
5. Sorting and Filtering
6. Encryption
7. Frequently Asked Questions (FAQs)