Osama A. Khashan, Abdullah M. Zin, Elankovan A. Sundararajan, 2015. ImgFS: a transparent cryptography for stored images using a filesystem in userspace. *Frontiers of Information Technology & Electronic Engineering*, **16**(1):28-42. [doi:10.1631/FITEE.1400133]

ImgFS: a transparent cryptography for stored images using a filesystem in userspace

Key words: Storage image security, Cryptographic file system, Filesystem in userspace (FUSE), Transparent encryption

Contact: Osama Ahmed Khashan,
E-mail: o_khashan@yahoo.com
ORCID: http://orcid.org/0000-0003-1965-1869

Introduction

- Traditional software encryption applications generally suffered from the expense of user convenience, performance efficiency, and the level of security provided.
- Transparent encryption is the solution, and can be implemented most efficiently with the operating system's file systems.
- ImgFS is a cryptographic file system that uses the filesystem in userspace (FUSE) technology, designed on the principle that the trusted system components and all stored image files should be protected in a full transparent manner without a single user interaction.

ImgFS Design & Implementation

- **Auto-mounting the ImgFS:**
  - Automatically authenticating a user using PAM during Linux startup time.
  - ImgFS is mounted over the entire ‘./home/’ hierarchy tree.
  - The source ‘./home’ and all stored images take the suffix extension ‘.imgfs’.
- **User authentication:**
  - Generating the user authentication signature token.
  - Defining the user signature token on PAM and protecting it on user’s USB disk.

ImgFS Design & Implementation

- **Key management:**
  - Each image is encrypted using a unique FEK generated randomly.
  - Access-verifier is used to verify the authenticity of the image file owner.
  - FEK is encrypted using a user's PEK and then appended together with the access-verifier to the image file header.

- **Cryptographic operation:**
  - Unique IV of 16 bytes for each image is generated.
  - The OpenSSL library was integrated with the ImgFS daemon, and the Blowfish cipher was picked as a symmetric encryption algorithm of 128-bit key length.

Performance evaluation

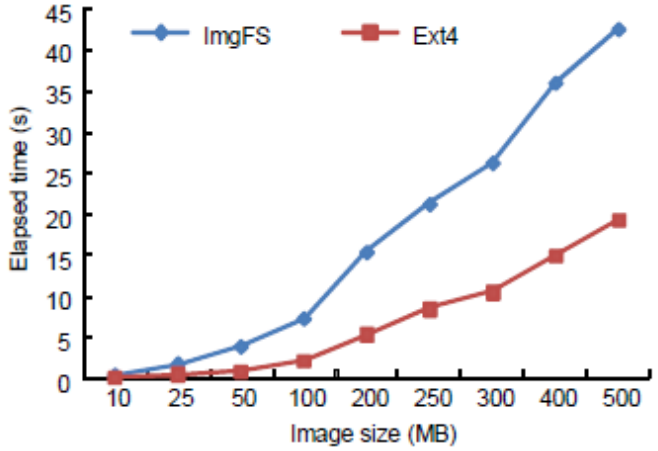

Fig. 6 Comparison of computational times for writing large BMP image files using ImgFS and Ext4

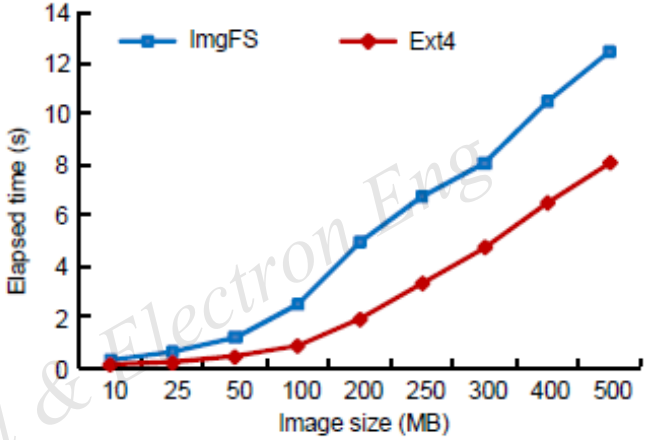

Fig. 8 Comparison of computational times for reading large BMP image files using ImgFS and Ext4

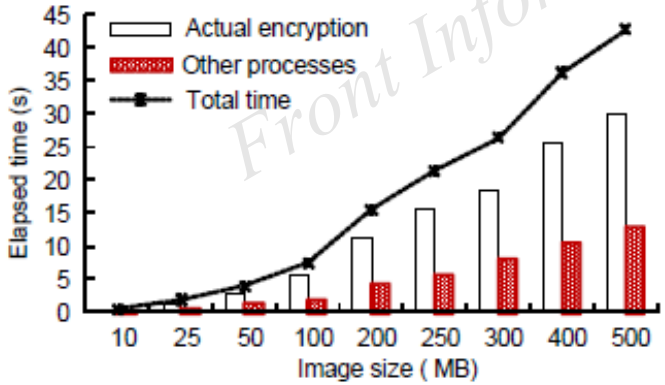

Fig. 9 Comparison of computational times for the actual encryption process compared to other write-related processes' time for writing large image files on ImgFS

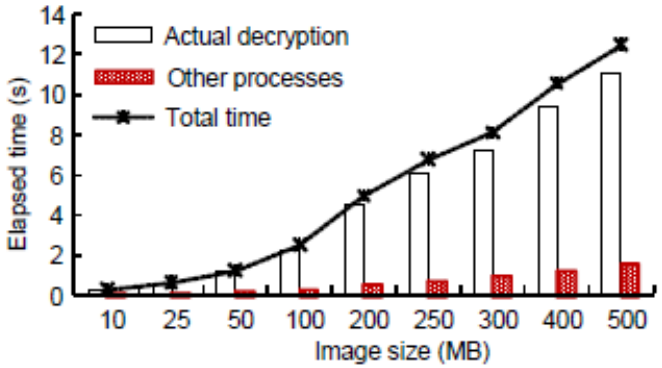

Fig. 10 Comparison of computational times for the actual decryption process compared to other read-related processes' time for reading large image files on ImgFS

Performance comparison

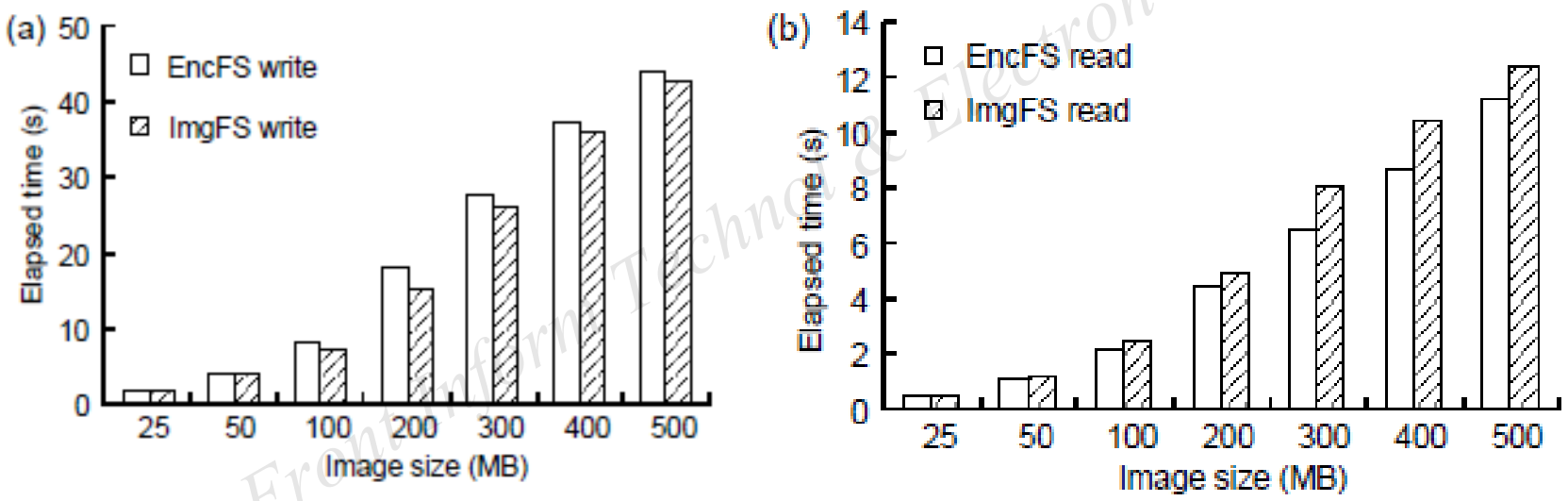

Fig. 12 Comparison of computational times for writing (a) and reading (b) images of large sizes on ImgFS and EncFS

Conclusions

- ImgFS has successfully achieved user convenience by supporting full transparent file system mount, user authentication, and cryptographic image files service.
- ImgFS can enforce strong access control to access a secure mount session as well as access protected image files by giving users the ability to perform the cryptographic task on per-image file basis, per-file keys, and per-user key-pairs.
- ImgFS is easily portable and applicable, and the image files can be easily shared in multi-user systems.