A Report on the ScramFS File System

Jintai Ding, Algo Consulting, Inc.
jintai.ding@gmail.com
March 22, 2018

This report is a high level review of the ScramFS secure file system and it is is primarily based on 2 documents;

1. ScramFS file system symmetric encryption layer: security analysis summary (Document Version 1.0)

2. ScramFS file system symmetric encryption layer: Design requirements and specifications (Document Version 1.0)

Ron Steinfeld, a professor from Monash University, who is a well-known cryptographer with very solid reputation, authors both documents. We carefully went through the documents to check and verify the technical details.

Overview

ScramFS is a new cryptographic file storage and sharing system, which can be used on cloud services like Google drive, such that a user can be sure that the only thing the cloud server can see are encrypted file/directory names, and contents to ensure the complete secrecy and integrity of all the data. Such a system can be of great value in the current of state of prevalent cyber attacks, in particular, ransom-ware attacks.

The fundamental idea behind ScramFS is to provide a robust cryptographic file system using symmetric cryptography such that users can have an easy and smooth experience in protecting their data without needing to know about what is going on behind the scenes.

Here we study carefully the systems based on the description in the two documents, and the pseudocode inside. Although we examined both the

documents very carefully, our review concentrates on the cryptographic protocols used in the systems and their security analysis.

Encryption key management and design

A key point of the ScramFS system design is to separate key management and distribution completely from the file system itself so that a business or any organization can utilize its existing key management infrastructure, or infrastructure that Scram may provide in the future.

The system is designed so that sharing of subsets of the file system in a multi-user environment is simple and easy.

Another goal of such a system is to ensure minimum possible data leakage in case of key compromise.

The ScramFS design gives a very clear description of the solution: how the protocol is designed, how the parameters of the encryptions systems are designed, and what are the principles being used to explicitly decide those parameters. The selection of parameters is well supported by solid and updated references. It provides clear details on how different level of keys are derived and how those keys or family of keys are used such that directory names, file headers, and content of files are encrypted and decrypted, and how the authentication is achieved in the systems.

Security model and post-quantum resistance

The security model is very strong and clear.

We think the system is very well planed and designed and all the key details, in particular, key derivation are clearly presented, and we did not find any bug or defect in the design. We find the security analysis to be very solid and the choice of parameters to be very well justified, even in terms of future quantum computer attacks. The application of the authenticated encryption by Philip Rogaway in this case is an astute idea and it is very well used. We also like very much the fact that AES 256 is used, not AES 128. For a client who wants long-term security, this is a very sensible choice.

Recommendations regarding key management and implementation

We provided recommendations to the designers of ScramFS following our review. These recommendations and Scram's responses are provided below. We are satisfied with Scram's responses.

- How to stop file sharing in a file sharing system if someone is not supposed to access files after certain time? This is difficult since the key is already shared?

    - Scram's response: We recognize this. This was not covered in the design documents, but in practice it can be achieved at three levels. The first is for the user to revoke shared access to the underlying file system, which is an easy process on local and cloud file systems. Secondly, from an implementation viewpoint, such shared keys will never be remembered or stored by the software or exposed to the user, so there is no easy way for the user to manually access or remember the key. Thirdly, if the user requires an absolute guarantee of access revocation, we recommend the user re-encrypt their file system (by creating a new one and copying their data across). This new file system will have a new master key.

- In ScramFS, the assumption in ScramFS is that key distribution and management are controlled by users separately and will be done via some other means. But still how this is done is very important. We recommend that Scram provides guidelines or tools on where are the keys stored, and how are they managed.

    - Scram's response: We agree with this and will follow your recommendation when designing our key distribution systems.

- It appears keys are the sole responsibility of the user to protect. What if the keys are lost, how the key recovery is done? We recommend that Scram provides guidelines or tools on assisting with this. Is it the sole responsibility of the user only to protect the master key or ScramFS will provide some service? If so, does that mean the client's key is also controlled by Scram?

- Scram's response: as you have noted, we designed the system such that the master key is known only to the user. This was important to establish trust with users and satisfy data sovereignty regulations. In our initial implementation, we have provided a tool to allow the user to back up their master key and secure that backup with a password. Unlimited backups of keys can be made. We will provide guidelines to the user as you suggest. We are also considering ways to provide distributed (and possibly automatic) backups of master encryption keys via secret sharing. Further work will require careful design and peer-review.

- A Cryptographic Pseudorandom Bit Generator (PRG) is used throughout the system. We recommend that care be taken with the selection and implementation of the PRG. Many vulnerabilities have happened when PRG is not properly implemented.

    - Scram response: we agree about the importance of this. Our initial implementation of ScramFS uses the Operating System's CSPRNG, the source of entropy as used by other implementations of cryptography such as TLS. However, we aim to improve upon this over time, given there have been past failures in implementations and standards (e.g. Debian, Java SecureRandom, Dual_EC_DRBG). We are currently researching further methods of securing against compromised PRGs but of course these require careful design and peer-review.

- On page 32, the meaning of the statement "Key can be shared with another user to give non-recursive access." If the key is shared, how can you stop recursive access?

    - Scram's response: non-recursive in this sense means allowing access to a particular directory's files, but disallowing access to any subdirectories under the shared directory. In contrast, recursive access means allowing access to a shared directory plus its subdirectories. Recursive and non-recursive sharing use two different keys, so depending on which level of access the user grants, one or the other key can be shared.

Comparision with other systems

It is clear the success of the ScramFS depends on how the key management and distribution is done in the end. Currently Microsoft has EFS which claims similar functionality as the Scram File System.
https://patents.google.com/patent/US6249866B1

With Microsoft's EFS, we assume that each user who has access to a file already has public key. For each file that's encrypted, a unique symmetric File Encryption Key is generated randomly, and that FEK has to be encrypted with the public key of everyone who has access, and is stored along with the file. Therefore, under this scheme, each file has the FEK encrypted for each user who has access. In this case, file encryption and key distribution are no longer separated unlike in ScramFS.

The assumption in ScramFS is that key distribution and management are controlled by users separately and can be done via some other means (e.g. post-quantum public key cryptography, or direct personal KE). Under such a systems sharing files becomes super simple, but this doesn't handle revocations easily. From this perspective, we think ScramFS is especially suitable for small enterprises or individual users, where key distribution is very easy to manage. The MS EFS will be hard for small enterprises to manage since it requires them to have a PKI like tools. In addition, the current PKI used by MS systems are not quantum resistant, and therefore susceptible to quantum computer attacks, and therefore are not considered long terms secure if a post-quantum PKI is not used.

Conclusions

My overall view is that the ScramFS system is a very solid cryptographic solution to the problem it intends to address. The documents are very well written.

However, since this review is only based on some abstract documents with limited time of 40 hours, it should always be used very carefully. This report is not at all be interpreted as a promise that the ScramFS is absolutely secure, but rather the ScramFS is a system very carefully designed by a top cryptographer with highest professional standard.

We also believe for ScramFS to be widely and fully used around the world, it should at certain stage to make the designs, and the security analysis, and maybe even the implementations public so it can be really checked by the whole crypto community to ensure its long term security.

The opinions expressed in this report are solely our personal opinions.

About the author

Jintai Ding is a Professor of Mathematics at the University of Cincinnati, and a world expert at post-quantum cryptography. He holds multiple patents for his cryptographic algorithms, a PhD from Yale and has guest lectured extensively throughout the world at universities such as the University of Oxford, University of Tokyo and Technical University of Darmstadt. He is the inventor of Rainbow signature scheme and the LWE-based key exchange scheme, and a co-inventor of the Ding-Iohara algebra.