Strongbox: Using Stream Ciphers for Full-Drive Encryption

Bernard Dickens III, Ariel J. Feldman, Haryadi S. Gunawi, Henry Hoffmann

University of Chicago

Abstract

Full Drive Encryption (FDE) is especially important for mobile devices because they both contain large amounts of sensitive data and are easily lost or stolen. Yet, the conventional approach to FDE, AES in XTS mode, is 3–5x slower than unencrypted storage. Authenticated encryption based on stream ciphers like ChaCha20 is already used as a faster alternative to AES in other contexts, such as HTTPS, but the conventional wisdom is that stream ciphers are a unsuitable for FDE. Used naively in drive encryption, stream ciphers are vulnerable to many-time pad attacks and rollback attacks, and mitigating these attacks with on-disk metadata is generally believed to ruin performance.

In this paper, we argue that recent developments in mobile devices invalidate this assumption and make it possible to use fast stream ciphers for drive encryption. Modern mobile devices rely on NAND-flash storage with a Flash Translation Layer (FTL), which functions very similarly to a Log-structured File System (LFS), and include trusted hardware such as Trusted Execution Environments (TEEs) and secure storage areas. Leveraging these two trends, we propose StrongBox, a stream cipher-based FDE layer that is a drop-in replacement for dm-crypt, the standard Linux disk/drive encryption module based on AES-XTS. StrongBox introduces a system design and on-disk data structures that exploit LFS's lack of overwrites to avoid costly rekeying and a counter stored in trusted hardware to implement rollback protection.

We implement StrongBox on an ARM big.LITTLE mobile processor and test its performance under the F2FS below (among others).

Motivation

Full Drive Encryption (FDE) is an essential technique for protecting the privacy of data at rest. Considering the state of the art, the conventional wisdom for securing this data is to use the AES block cipher in XTS mode [5]. Potentially more performant steam ciphers are not typically considered.

However, technological shifts in mobile devices overturn this conventional wisdom and make it possible to use more performant stream ciphers for drive encryption.

Figure 1: Mobile devices commonly use Flash Translation Layers (FTL) and/or Log-structured File Systems (LFS) [2, 3, 6] to increase the lifetime of their solid-state drives (SSD).

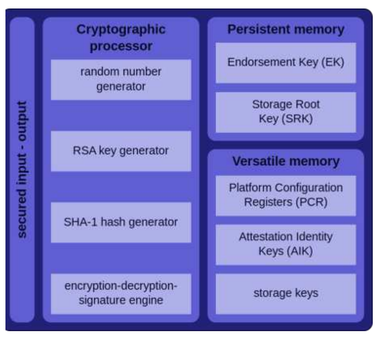

Figure 2: Modern mobile devices like smartphones now come equipped with trusted hardware [4, 7], such as Trusted Execution Environments (TEE) and secure storage areas [1].

We demonstrate the FDE performance win from switching to a stream cipher by comparing AES-XTS to ChaCha20+Poly1305. Fig. 3 shows the distinct advantage of the stream cipher over AES: a consistent $2.7\times$ reduction in run time.

We believe stream ciphers are best suited for encrypting block devices backing LFSes, as these filesystems are designed to append data to the end of a log rather than over-write data. In practice, some overwrites occur; *e.g.,* in metadata, but they are small in number during normal execution (Fig. 1). This motivates our approach of using a stream cipher to perform FDE under LFSes.

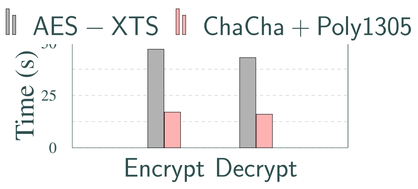

Figure 3: AES-XTS and ChaCha20+Poly1305 Comparison.

File System	Total Write Ops	Overwrites
ext4	16,756	10,787
LogFS	4,244	32
NILFS	4,199	24
F2FS	2.107	2

Table 1: File System Overwrite Behavior

StrongBox Design

StrongBox's design is illustrated in Fig. 4. StrongBox's metadata— the key to its operation—is encapsulated in three primary components: an in- memory *Merkle Tree* and two disk-backed byte arrays, the *Keycount Store* and the *Transaction Journal*. These components are integrated into the *Cryptographic Driver*, which is responsible for handling data encryption, verification, and decryption. These interactions take place while fulfilling high-level I/O requests received from the overlying LFS.

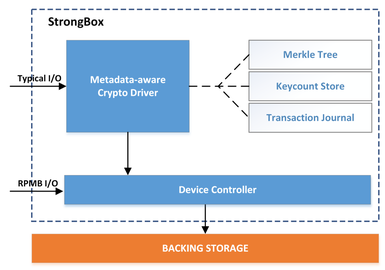

Figure 4: Overview of the StrongBox construction.

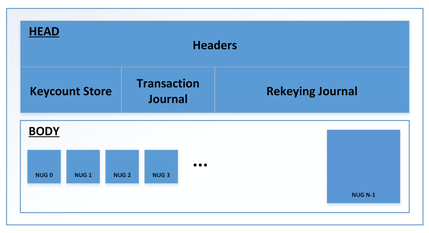

Figure 5: Layout of StrongBox's backing storage.

StrongBox vs Dm-crypt under F2FS

To evaluate the performance of StrongBox, we measure the latency (seconds/milliseconds per operation) of both sequential and random read and write I/O operations across four different standard Linux filesystems: NILFS2, F2FS (shown below), Ext4 in ordered journaling mode, and Ext4 in full journaling mode.

We include results of the F2FS LFS mounted atop both dm-crypt and StrongBox; median latency of different sized whole file read and write operations were normalized to unencrypted access. By harmonic mean, StrongBox is $1.6\times$ faster than dm-crypt for reads and $1.3\times$ faster for writes.

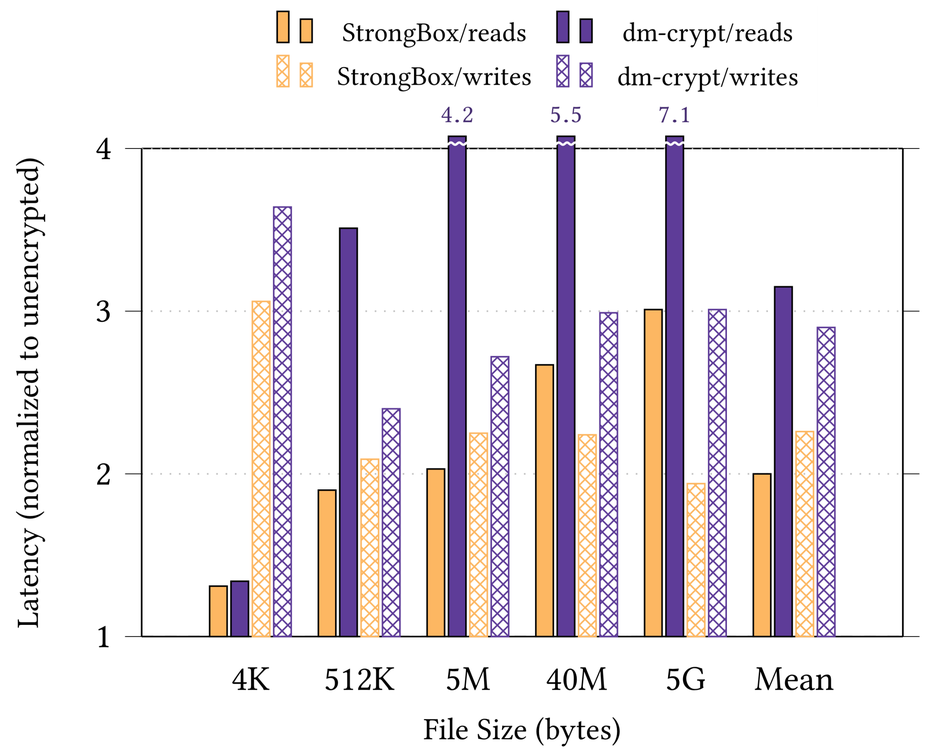

Figure 6: Sequential I/O F2FS result set.

Conclusion

- The conventional wisdom: securing data at rest requires one must pay the large performance overhead of encryption with the AES-XTS block cipher instead of using a stream cipher.
- The proliferation of NAND-flash FTL/LFS and secure hardware on modern/mobile devices overturns the conventional wisdom, making it practical to use stream ciphers to secure data at rest.
- We propose *StrongBox*: a stream cipher-based FDE layer and drop-in replacement for dm-crypt. StrongBox exploits LFS's lack of overwrites and the availability of trusted hardware to overcome the limitations of stream ciphers.
- Our results show that, under F2FS, StrongBox provides upwards of $2\times$ improvement on read performance and $1.3\times$ improvement on write performance over a standard dm-crypt configuration.

*StrongBox source is available on GitHub @ `https://github.com/research/buselfs-public`

References

[1] *EMBEDDED MULTI-MEDIA CARD (e•MMC), ELECTRICAL STANDARD (5.1)*. 2015. URL: `https://www.jedec.org/standards-documents/results/jesd84-b51` (visited on 04/26/2017).

[2] R. Konishi et al. "The Linux Implementation of a Log-structured File System". In: *SIGOPS Oper. Syst. Rev.* 40.3 (July 2006), pp. 102–107.

[3] C. Lee et al. "F2FS: A New File System for Flash Storage". In: *13th USENIX Conference on File and Storage Technologies (FAST 15)*. Santa Clara, CA: USENIX Association, 2015, pp. 273–286.

[4] A. Limited. *ARM security technology: Building a secure system using TrustZone technology*. PRD29-GENC-009492C. 2009.

[5] *Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*. NIST Special Publication 800-38E. 2010.

[6] M. Rosenblum and J. K. Ousterhout. "The Design and Implementation of a Log-structured File System". In: *ACM Trans. Comput. Syst.* 10.1 (Feb. 1992), pp. 26–52.

[7] G. P. D. Technology. *TEE client API specification version 1.0*. GPD_SPE_007. 2010.