Hycrypt: XTS Full Disk Encryption Alternative with Chacha and Poly1305

Bernard Dickens
University of Chicago
bd3@uchicago.edu

Ariel Feldman
University of Chicago
arielfeldman@uchicago.edu

Henry Hoffmann
University of Chicago
hankhoffmann@uchicago.edu

Abstract

Block-level full-disk encryption (FDE) is a non-trivial engineering problem rife with competing interests and messy tradeoffs. An ideal FDE solution must ensure data confidentiality, data integrity, and efficient operation within a potentially highly-constrained total energy budget, such as with a mobile or embedded system. All of this must be accomplished without adding significant overhead to system performance. The current de-facto FDE standard, AES in XTS mode, achieves only data confidentiality and does so by sacrificing energy and system performance while abandoning data integrity guarantees entirely. However, recent work on faster and more energy-efficient alternatives to the AES cipher allow us to revisit some of the design decisions that underpin AES-XTS. Specifically, we propose Hycatan FDE alternative to AES-XTS utilizing the Chacha20 stream cipher and Poly1305 MAC algorithm.

1 Introduction

Somewhat concise introduction like in the Chacha paper. Is this worth it? Briefly address project meaningfulness. Status quo disk encryption has a non-trivial cost. Enumerate any potential tradeoffs. Describe extra benefits of a Chacha-LFS construction (i.e. integrity checking, simpler design) over XTS.

Caveats, major limitations, and how they're handled.

- Here we summarize the main contributions (as with the brief introduction) with a focus on the justification for the project's existence. Tease apart and enumerate any other contributions at the end (see MEANTIME paper for format).
- Argue that this is a meaningful project by showing (via chart) that encryption has a non-trivial cost. We would use the initial FDE vs NFDE on RD experiment charts.
- Do we address limitations here or in the conclusion at the end?
- Argue that making a stream cipher work for FDE in this instance is non-trivial and comes with costs. Characterize. "We save X at the cost of Y". DE is the problem. The solution is Z. It costs Y. If willing to pay.
- Need to tell what the above costs are, if and how they can be minimized, and what trade-offs they constitute.

2 Motivational Example

(mobile phone battery life example, android K/L/M FDE stats)

3 Related Works

(broken off from the introduction)

3.1 AES-XTS

AES-XTS background information

3.2 Salsa20/Chacha20

Stream Cipher and Salsa20/Chacha20 background information

Show Chacha as a stream cipher to be faster than AES in a stream cipher-ey mode (CTR or GSM to compare with integrity checking). Need to show that these modes of AES are always? faster than the very slow double-keyed XTS construction. This would establish that there is some slack to be played with.

4 Hycrypt Full Disk Encryption

(maybe a short blurb here; see MEANTIME paper for direction)

4.1 Threat Model

(this will be very similar to the threat model established by the XTS project and writeups)

4.2 Design and Implementation

Describe LFS construction in detail, explain design decisions, argue security and preservation of crypto guarantees via assumed cryptographic primitives, describe perceived extra benefits (i.e. integrity checking, simpler design) over XTS

5 Experimental Setup

- RD+Fuse-Ext4 is the baseline for all experiments
- Crafting the story by gathering meaningful data using the performance of so-called "I/O bound applications" as metrics. Currently under consideration: git, mobile bench (as a suite), Chrome/FF (mobile and non-mobile; talk to Connor about FF), VLC x264 playback, graphg/social graph, Ferret (read paper), SQLite/Access/No-SQL local DB, Filezilla FTP, Google Maps offline map caching, mobile SMS history
- The above would be evaluated across NFDE (RDFx4), dmcrypt (RDFx4D), then LFS-Chacha (RDLC) and LFS-AESGCM (RDLA) [and also RDLFS for good measure]
- Describe how I/O bound applications were classified as such and why they were selected (explain I/O profiling tool usage and setup)
- Any official standard benchmarking suites?

6 Experimental Evaluation

Describe evaluation of Chacha-LFS versus AES-XTS

- Charts showing actual benchmarks of i/o bound and/or i/o heavy applications evaluated across testbed of filesystems
- Show a chart depicting energy use on mobile or device with typical pure NAND flash + dmcrypt FDE versus NAND flash + LFS Chacha versus NAND flash + LFS Chacha in "energy saving mode" (saving garbage collection etc for later). We would expect to see a decrease in energy use from dmcrypt down to LFS in energy saving mode but little to no degredation i performance from dmcrypt to LFS!
- Chart showing how the LFS performs with Chacha20+Poly1305 as stream cipher versus AES-GCM as stream cipher. This would demonstrated that the performance gains are not just from using the LFS over Ext4. We should expect to see the LFS under C20+P1305 outperform LFS under AES-GCM.

7 Conclusion

- Address (defend?) the use of RD+Fuse-Ext4 as a baseline. It would be best if we could relate any RD+Fuse-Ext4 based performance metrics with performance on a standard SSD (in Experimental Evaluation) so that it does not look like we're trying to hide anything.
- Address limitation: why would this project remain important even if SSDs sped up dramatically? (b/c energy savings)
- Address limitation: would this project remain relevant if hardware accelerated AES become prevalent? (yes b/c embedded devices and energy savings)
- Address choice of benchmarking suite

8 Acknowledgments

A polite author such as myself always includes acknowledgments. Thank everyone, especially those who funded the work.

9 Availability

Release LFS code, LFS+Chacha20, LFS+AESGCM, I/O profiling tool(s)/setup, etc. Do we released entire repo/all experimental code as well?

`ftp.site.dom/pub/myname/Wonderful`

Will also throw up some sort of homepage or webpage at some point:

`http://www.site.dom/~myname/SWIG`

References