CSC 580
Cryptography and Computer Security

Tweakable Block Ciphers and Disk Encryption

(Sections 7.7)

February 15, 2018

Goal: Encrypt a Block Storage Device

Block storage devices

- Used for “bulk storage”
- Hard drives, solid-state drives, thumb drives, …
- Devices often portable and can’t be physically protected

What encryption is out there?

Software FDE (Full Disk Encryption)

VeraCrypt

VeraCrypt

HOME | SOURCE CODE | DOWNLOADS | DOCUMENTATION | DISCUSSIONS

Page Info | Change History (all pages)

Project Description

VeraCrypt is a free disk encryption software brought to you by **IDRIX** (https://www.idrix.fr) and that is based on TrueCrypt 7.1a.

Latest Stable Release - 1.19 (Mon Oct 17, 2016)

VeraCrypt is a successor to TrueCrypt

TrueCrypt was used for years as a cross-platform disk encryption tool - development discontinued in 2014 (interesting story…)

Microsoft FDE for Windows

BitLocker combines software FDE with hardware key protection

- Uses the Trusted Platform Module (TPM)
- Can be tightly integrated with UEFI Secure Boot
- Can also require a USB drive as a key
- Can encrypt USB drives...

Disk Encryption in the Disk Itself

TECHSPOT

LOGIN

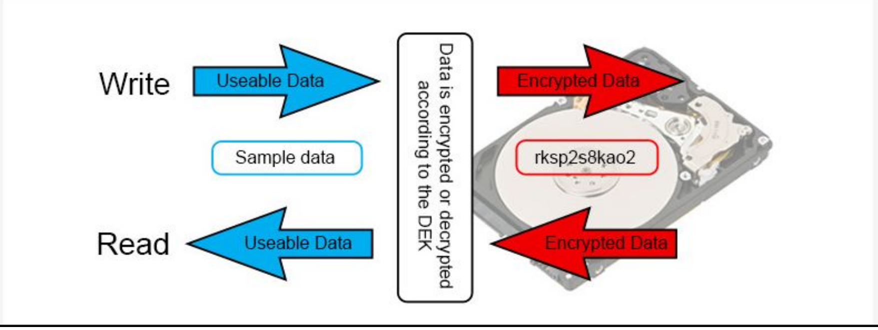
FEATURES HARDWARE

Self-Encrypting Drives: A Brief Introduction and Step by Step Guide

By Matt Bach on August 22, 2014

A SED, or self-encrypting drive, is a type of hard drive that automatically and continuously encrypts the data in it without any user interaction. What may surprise many is that a decent potion of the drives currently in the market, including the popular Samsung 840 Pro SSD series are in fact SEDs. But since manufactures do not tout this as a major feature, it often gets lost in the large number of typically more important specifications.

Even once you purchase, install, and start using one of these SED drives, the encryption is so transparent to the user that it is unlikely they would ever realize the drive is a SED.



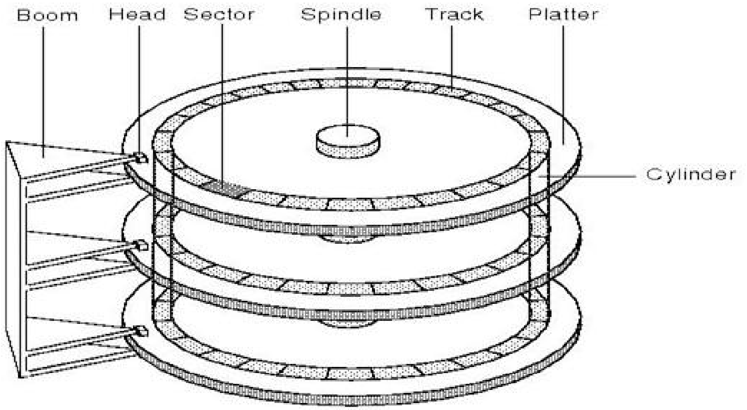
Properties of Block Storage

Data in fixed-size blocks/sectors

Only full blocks can be read/written

Data structures optimized for layout

- Filesystems
- B-trees (databases)



Some desirable properties (more in textbook)

- Data size must remain the same (think about CBC)
- Data layout must remain the same (blocks map to blocks)
- Same data in different locations has different ciphertext
- Vital for this to be fast!

Tweakable Block Ciphers

Tweakable Encryption: E(K, T, P) = C

Key Tweak Plaintext

Goal: “Tweak” adds variability without IV or CT length increase

Efficiency goal: More efficient than changing key

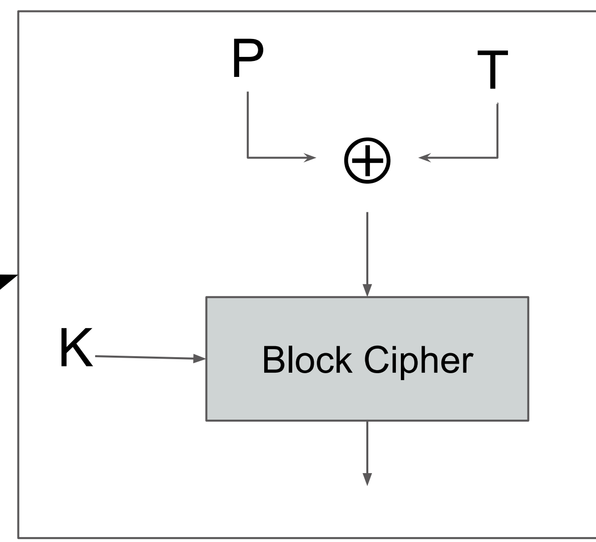
- Remember: Can precompute key schedule

Attempt 1:

- CTR mode with T as CTR?
- Bad: Malleable

Attempt 2:

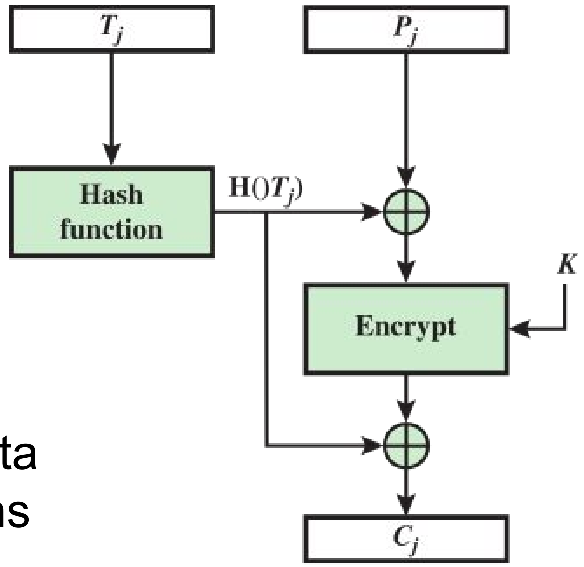
- XOR plaintext blocks with counter
- Good: Mixes up ciphertext
- Bad: What if plaintext blocks are counters?



Tweakable Block Ciphers

Tweakable Encryption: E(K,T, P) = C

Key Tweak Plaintext



Attempt 3:

- XOR before and after with “random looking” data
- Good: Unlikely to accidentally have bad patterns
- Bad: Can an attacker create bad patterns?
  - Is this a danger? Unclear...

One that works: XTS-AES

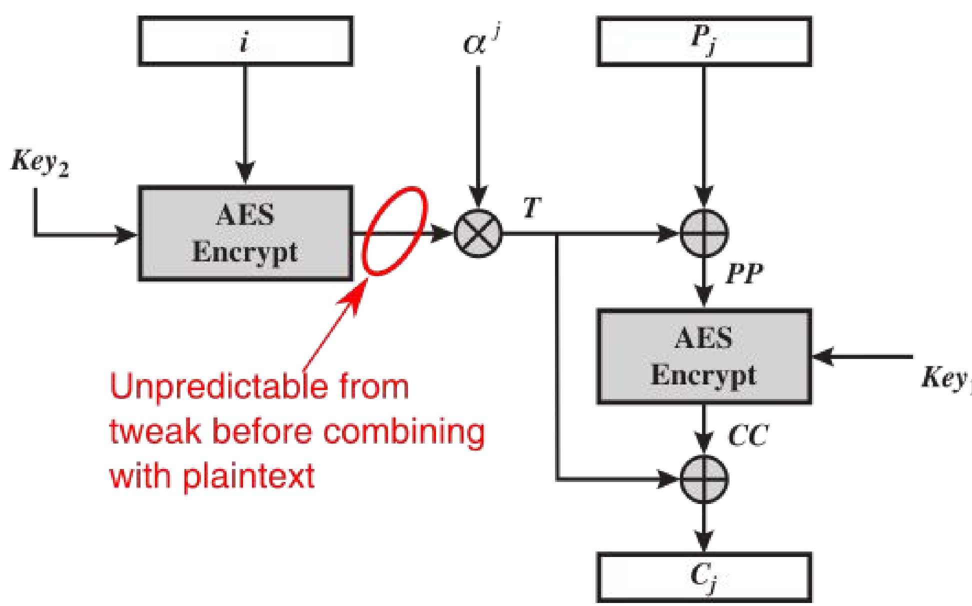
Idea: Encrypt sector number for unpredictable plaintext adjustment.

Efficiency:

- Circled part is the same for all blocks in sector - compute once!
- Block adjustments (α^j) doesn't depend on i - precompute!
- Combination ($\otimes$) sped up in AES-NI instructions

i: Sector number
j: Cipher block number within sector

Key is really two keys...



Test your understanding...

How many block cipher encryptions are needed to encrypt a 512-byte sector?

[redacted]

Programming with Crypto

Discussion on board and looking at JCA documentation:

Using block cipher modes

- Handling the IV
    - Importance of randomness
    - Sending with the ciphertext
    - Extracting and using to decrypt
- Binary, text, and Base64