# File and disk encryption

**Milan Brož, xbroz@fi.muni.cz**
Faculty of Informatics, Masaryk University

CRoCS
Centre for Research on
Cryptography and Security

# Storage encryption

**Lecture**

- File and disk encryption
- Distributed storage encryption
- Abstraction layers
- Confidentiality and integrity protection
- Encryption modes
- Key management
- Attacks and common issues
- Laboratory

- We will focus on low-level architecture in this lesson

File and disk encryption

# MOTIVATION & STORAGE LAYERS OVERVIEW

# Motivation

**Offline, "Data at Rest" protection**

notebook, server or external drives, data in cloud, backups

**Key removal = easy data disposal**

**Confidentiality protection**

- often enforced **policy** to encrypt portable devices
- prevents data leaks (stolen device)

**Data integrity protection?** (not often yet)

# Overview

**(Distributed) Storage Stack**

layers accessing storage through blocks (sectors)
distributed => storage + network layer

**Full Disk Encryption (FDE)**

- self-encrypted drives, (software) sector-level encryption

**Filesystem-level encryption**

- general-purpose filesystem with encryption
- cryptographic file systems

# Storage stack & encryption layers

| | | |
|---|---|---|
| **Userspace** | **Application** | Application specific cloud API, ... |
| **OS kernel** or drivers in userspace | **Virtual file-system** (directories, files, …) | **File-system encryption** |
| | **Specific file-system** (NTFS, ext4, XFS, APFS…) | |
| | **Volume Management** (partitions, on-demand allocation, snapshots, deduplication, …) | **Disk (sector) encryption** |
| | **Block layer** (sectors I/O) | |
| | **Storage transport** (USB, SCSI, SAS, SATA, FC, NVMe…) | **HW-based encryption** self-encrypted drives, inline (slot) encryption, chipset-based encryption |
| | **Device drivers** | |
| **“Hardware”** | **Hardware** (I/O controllers, disks, NAND chips, …) | |

# Software Defined Storage (SDS)

- commodity hardware with abstracted storage/network logic
- **encryption is "just" one logic function**
- usually combination with classic storage (and encryption)

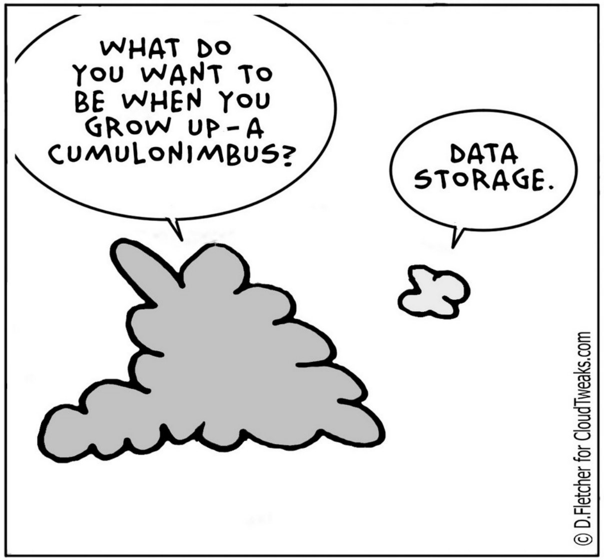
- **Distributed storage – storage + network layer**
  - **must** use also network layer encryption
  - note differences in network and storage encryption (replay attack resistance, integrity protection, …)

# Distributed Storage, Cloud & Encryption

Distributed storage – add network layer

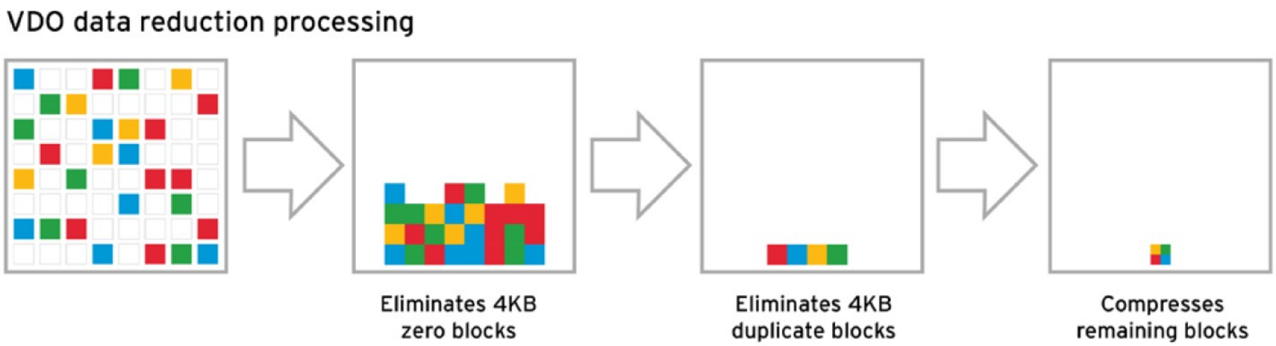
- **Shared volumes** (disk encryption below)
- **Clustered file-system** (fs encryption)
- **Distributed object store** (object encryption)

- **Cloud data storage - REST API**
  (not part of this lecture)
  - DropBox, Microsoft OneDrive, Google Drive
    Amazon S3, ...



# Cloud storage – common features

**Deduplication** – avoid to store repeated data




**Compression**

special case: zeroed blocks

**Data snapshots** (in time)

COW (copy on write)

# Cloud storage & encryption

Encryption with storage backend, network access and compression & deduplication & snapshots …

**Encryption on client side** (end-to-end)

- inefficiency for deduplication/compression

~ in future homomorphic encryption?

**Encryption on server side**

- confidentiality for clients is lost
- server can access decrypted data

# Full Disk Encryption (FDE)

## Block device – disk sector level

- disk, partition, disk image
- ciphertext device / virtual plaintext device
- atomic unit is sector (512 bytes, 4k, 64k)
- consecutive sector numbers
- sectors encrypted independently

## One key decrypts the whole device

- media (volume) key – one per device
- unlocking passphrases / keys / tokens

# Filesystem-level Encryption

## File / Directory

- atomic unit is filesystem block (~ compare sector in FDE)
- blocks are encrypted independently
- **Generic filesystems with encryption**
  - some metadata can be kept in plaintext (name, size, …)
- **Cryptographic filesystems**
  - metadata encrypted
  - ~ stacked layer over generic filesystem

## Multiple keys / multiple users

# File vs. disk encryption

## Full disk encryption

+ for notebook, external drives (offline protection)
+ no user decision later what to encrypt, transparency
+ hibernation partition and swap encryption
- more users – whole disk accessible
- key disclosure – complete data leak
- usually no integrity protection

+/- self-encrypted drives – you have to trust hw

Examples: Opal2 (SED), LUKS, VeraCrypt, BitLocker, FileVault

# File vs. disk encryption

## Filesystem based encryption

+ multiple users
+/- user can decide what to encrypt
+ copied files keeps encryption in-place
+ more effective (encrypts used blocks only)
+ should provide integrity protection (not always!)
- more complicated sw, usually more bugs
- unusable for swap partitions

Examples: Linux fscrypt API, ZFS, APFS (Apple fs)

# Examples of HW-based encryption

- **Self-encrypting drives (SED), Opal2 standard**
  - Encryption on the same chip providing media access
- **Inline encryption**
  - Slots for keys (through OS context)
- **Chipset-based encryption**
  - Encryption on controller chip (e.g. USB bridge)
- **Hardware acceleration**
  - AES-NI, accelerators, ASICs, GPUs, …
- **Secure hardware / tokens**
  - HSM, TPM, SmartCards, ...

# Opal2 - self-encrypting drive

- **Trusted Computing Group (TCG) standard**
  - many optional features, usually implemented only mandatory
  - single user mode or multiple users
  - locking ranges
  - shadow boot record (MBR)
  - PSID reset
- **Used for SSD or NVMe drives**
- **Opal** - full media encryption
- **Pyrite** - only authentication, no data encryption
- (other variants - Opalite, enterprise Ruby)

File and disk encryption

# DATA ENCRYPTION

# Disk encryption algorithms primitives

- **Symmetric encryption**
  - block ciphers
  - cipher block mode + initial vector / tweaks
- **Key management and key storage**
  - Random Number Generators (RNG)
  - Key Derivation Functions (KDF)
- **Deniable encryption / Steganography**

# Data confidentiality & integrity

**Confidentiality**

Data are available only to authorized users

**Integrity**

Data are consistent

Data has not been modified by unauthorized user

=> all modifications must be detected

*Note: replay attack (revert to old valid data)*
*detection cannot be provided without separate trusted store*

# Data integrity / authenticated encryption

**Poor man's authentication** (= no authentication)
- User is able to detect unexpected change
- Very limited, cannot prevent old content replacement

**Integrity – additional overhead**
- Where to store integrity data?
- Encryption + separate integrity data
- Authenticated modes (combines both)
- Tamper Evident Counter (TEC)
- Merkle tree

File and disk encryption

# DATA ENCRYPTION MODES

# Symmetric encryption (examples)

**AES, Cammelia, Adiantum,** Serpent, Twofish, (Specks, Kuznyechik, …)

## Encryption-only modes

- Storage encryption mostly CBC, XTS
- Length-preserving encryption, block tweak

## Authenticated modes (encryption + integrity)

- Integrity protection often on higher layer.

# Standards

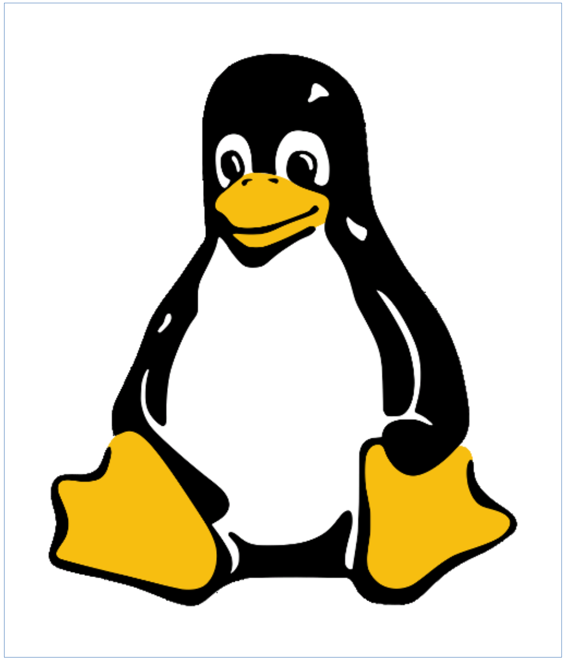
**IEEE 1619** – encryption modes for storage
**NIST Special Publications (SP)** –
    ciphers,modes, KDF, password handling, ...
**TCG Opal2** – self-encrypted drives
**IEEE 1667** – authentication
**FIPS 140-2, 140-3, Common Criteria (CC)**

# Propagation of plaintext changes

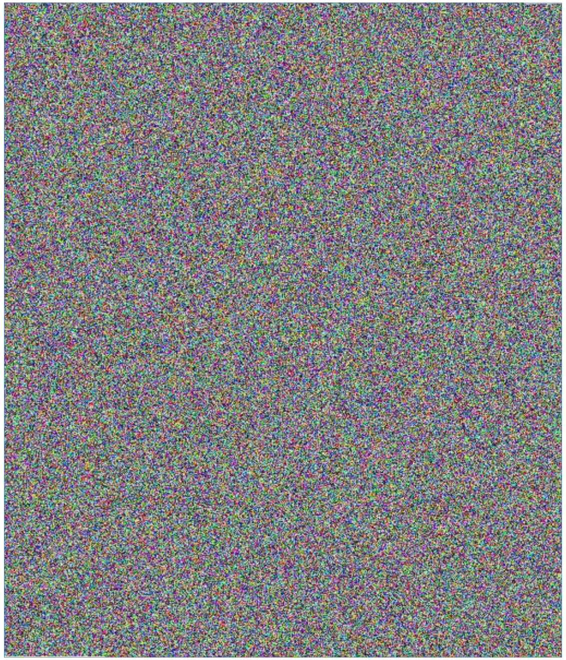
**A change in plaintext should transform to randomly-looking change in the whole ciphertext sector. Solutions?**

- **Ignore it**, and decrease granularity of change
  => change location inside ciphertext sector
- **Use wide mode** (encryption block size = sector size)
  - requires at least 2x encryption loop
  - modes are patent encumbered
- **Use additional operations**
  - Elephant diffuser in Windows Bitlocker
  - Google Adiantum (cipher composition)

# Encryption example output

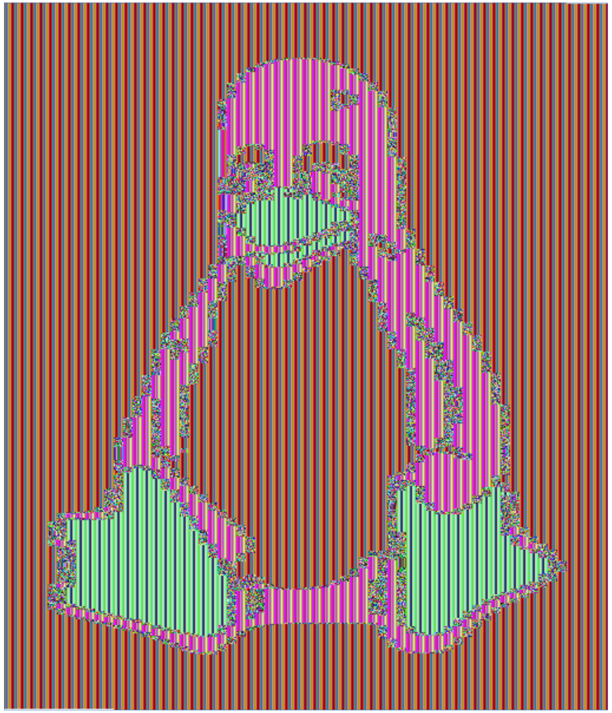


**plaintext**

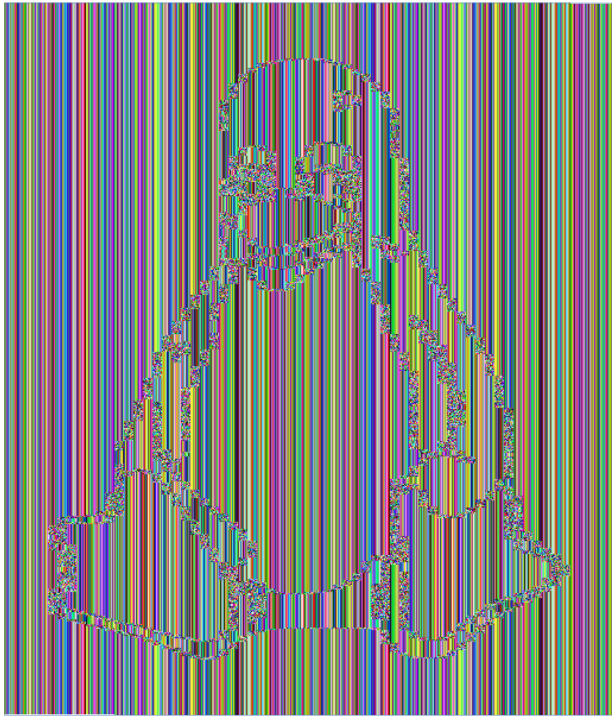


**ciphertext**

# Wrongly used encryption – patterns, leaks



**ECB mode**



**AES-XTS & constant IV**

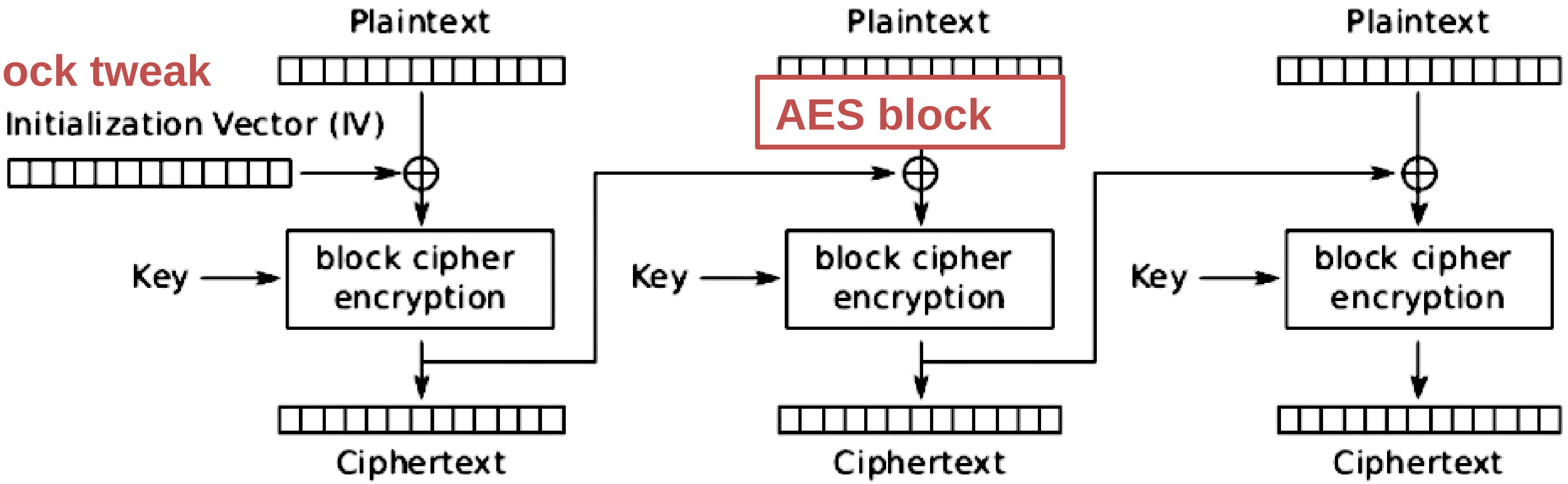
# Cipher-Block-Chaining (CBC) mode

- Blocks cannot be encrypted in parallel
- Blocks can be decrypted in parallel
- Tweak must be non-predictable (watermarking!)

# CBC encryption

plaintext disk / file-system block …

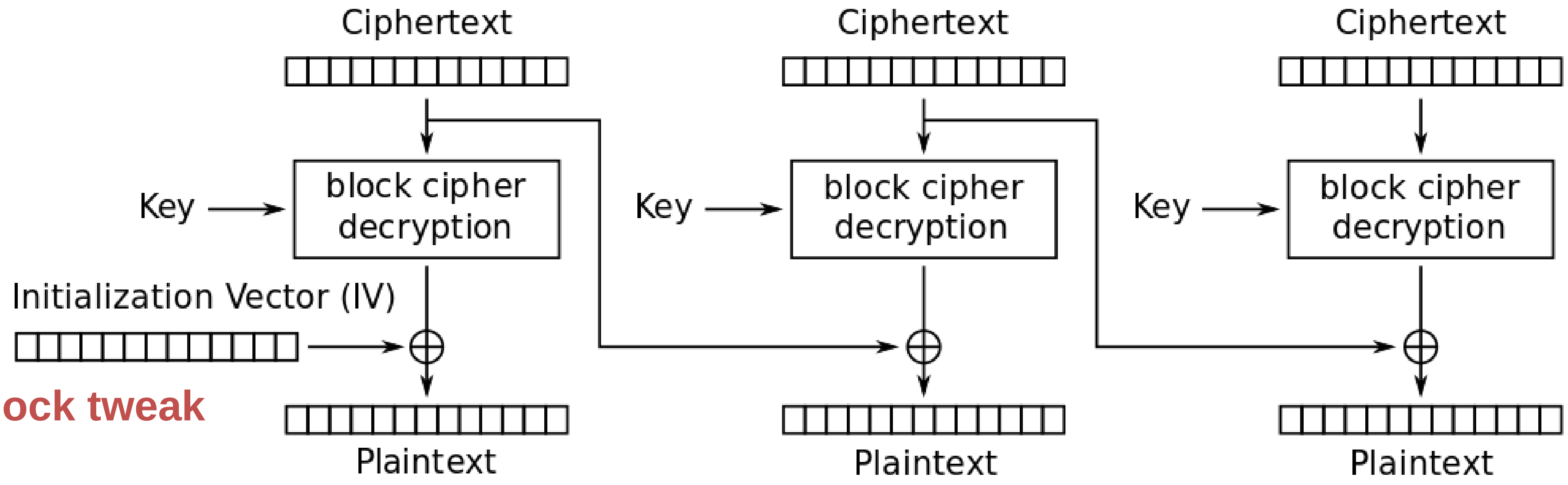
block tweak




ciphertext disk / file-system block …

# CBC decryption

ciphertext disk / file-system block …




block tweak

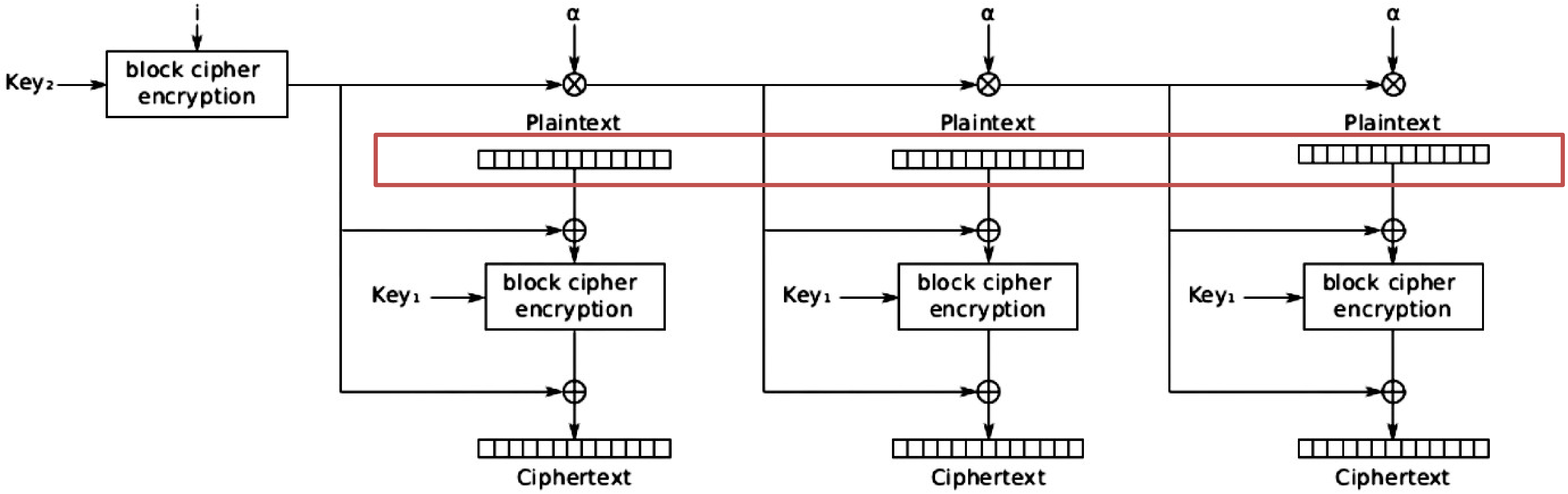
plaintext disk / file-system block …

# XOR-Encrypt-XOR (XEX / XTS) mode

- Encryption / decryption can run in parallel
- Two keys – 512-bit key means AES-256
- Tweak can be predictable nonce – sector number (offset)
- Ciphertext stealing not needed for common sector sizes
- Used in most of FDE systems today (2023)
- It is not a wide mode!
- Trade-off for performance

# XTS mode encryption/decryption

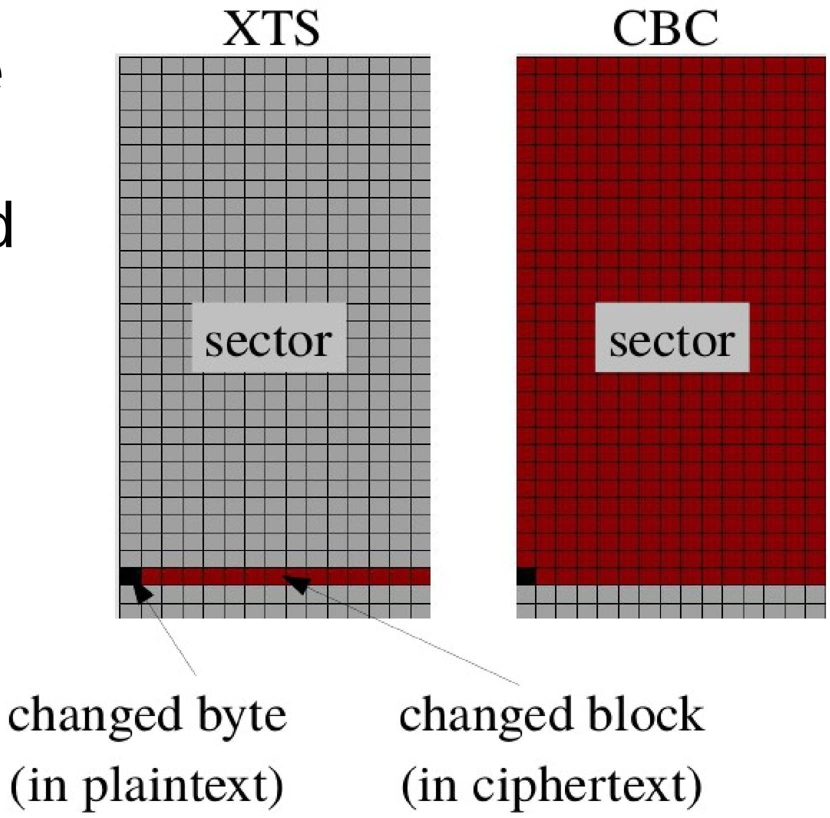
block tweak




ciphertext disk / file-system block …

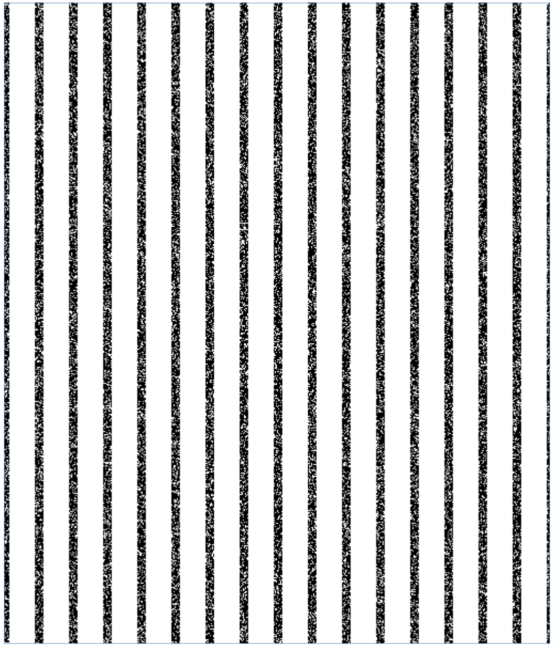
# CBC vs XTS change propagation

- XTS is trade-off for performance
- For storage, data always aligned to encryption blocks
  XTS: no ciphertext stealing
- Initial vector/tweak is important
- CBC is phased out today

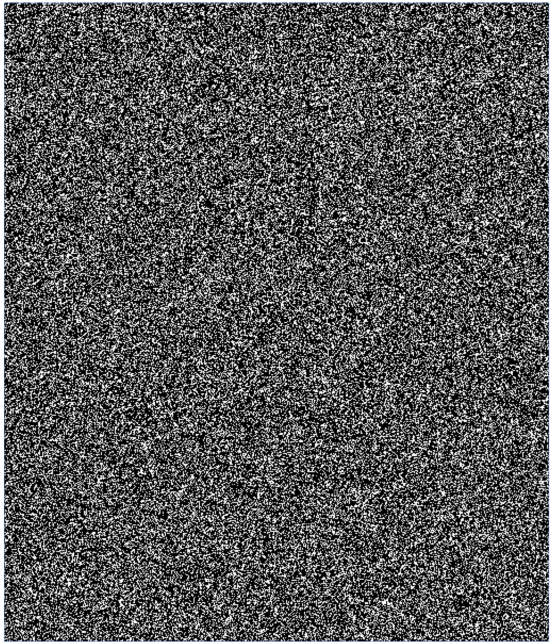


# AES-XTS IV mode – sector# vs random

**Every 64 byte changed (ciphertext differences)**



**IV is sector number**

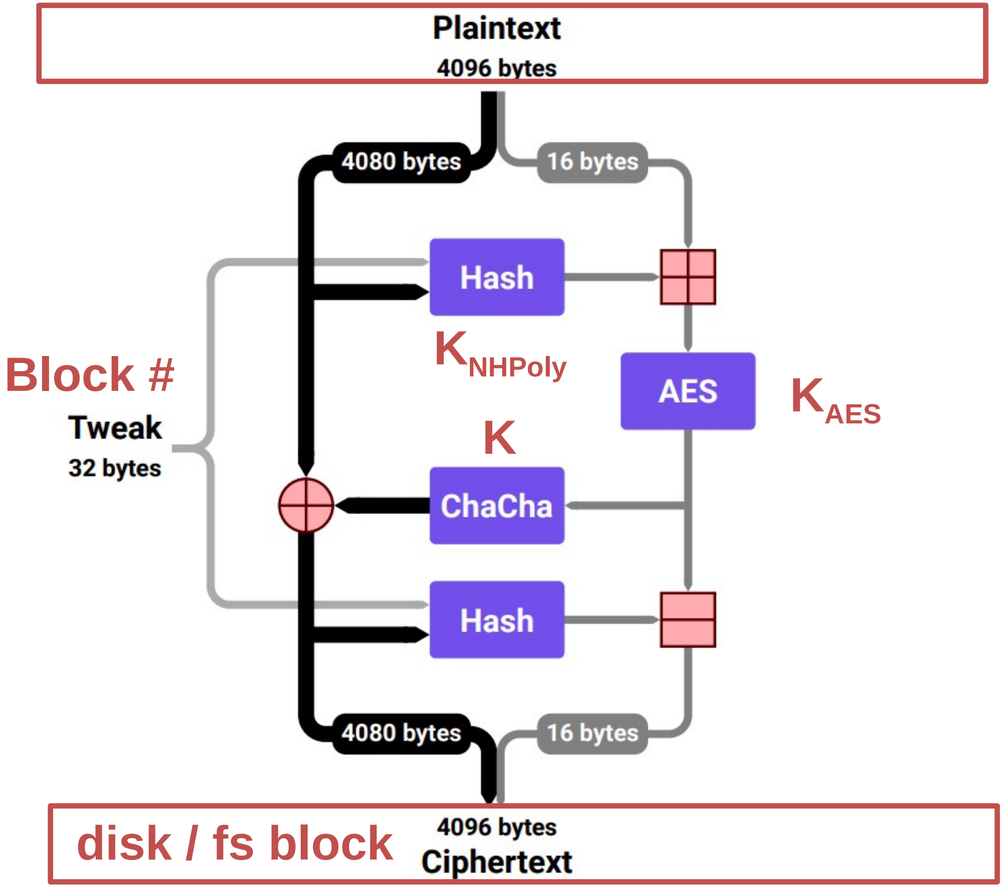


**randomized IV**

# Adiantum

- Low-end mobile device disk / file encryption
- Wide "mode"
- HBSB composition:
  - Hash – NHPoly1305)
  - Block Cipher – AES
  - Stream Cipher – XChaCha12,20
  - Hash – NHPoly1305
- Key derivation

  $K_{AES} || K_{NHPoly} = \mathrm{XChaCha}(K, 1|0..0)$




https://eprint.iacr.org/2018/720
https://security.googleblog.com/2019/02/introducing-adiantum-encryption-for.html

# Steganography / deniable encryption

**Plausible deniability:**

- existence of encrypted data is deniable
- if adversary cannot prove that it exists

**Steganography**

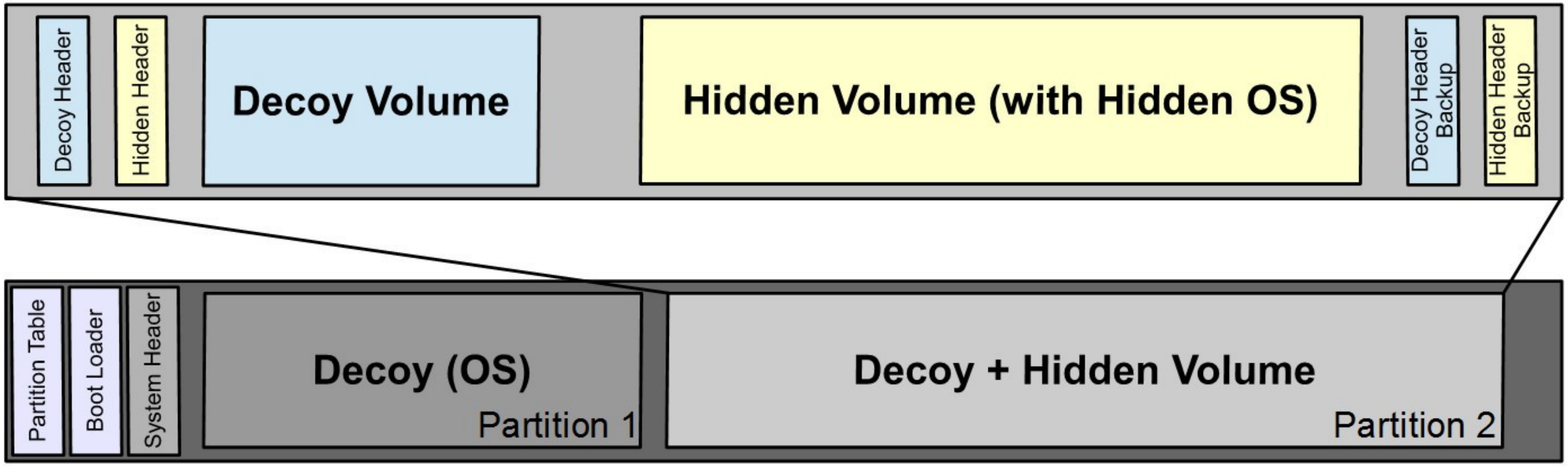
- hiding data in another data object

**Steganographic file-systems**

**Deniable disk encryption**

# Trivial example: VeraCrypt hidden disk

- FAT linear allocation (other fs are very problematic)
- Hide another disk in unallocated space



# Deniable encryption problems

**Side-channels**

tracking activity that cannot be explained for decoy system

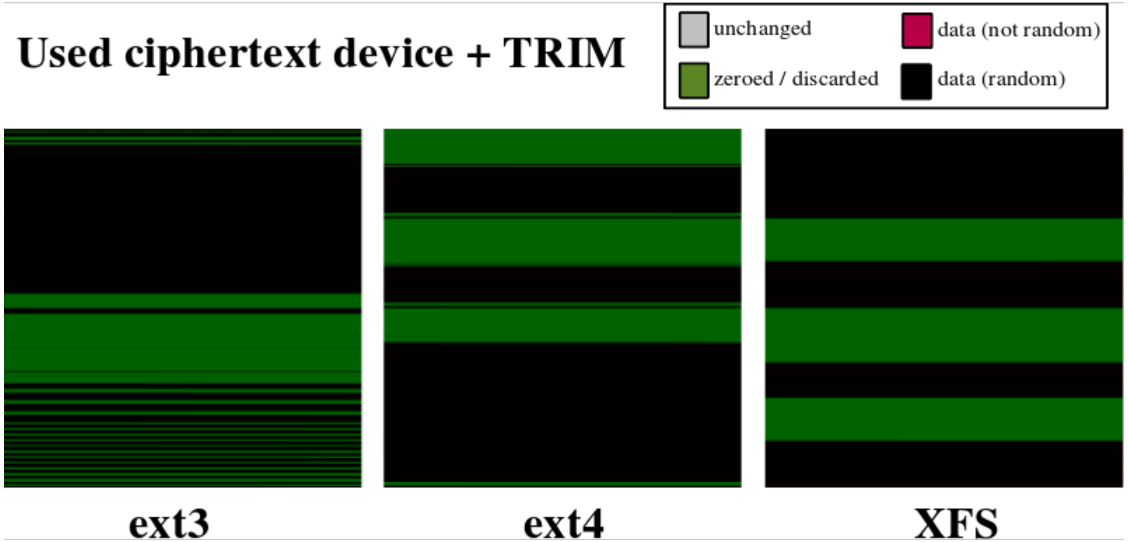
- Software: link to recently open documents, …
  Suspicious parameters (FAT), disabled TRIM, …
- Hardware: internal SSD block allocations
  (access to “unused” areas)

**Incompatibility with new drives (TRIM)**

*Note: flash storage HW is much more complicated (NAND chips management, wear-leveling, ...).*
*With low-level access you can detect suspicious patterns.*
*Note: Also see shufflecake.net another try for plausible deniability.*

# TRIM / discard and encryption

- TRIM informs SSD drive about unused space
- Unused space is detectable
- Pattern recognition (fs type) example



File and disk encryption

# KEY MANAGEMENT

# Long-term key generation and key store

## Encryption key (~ Media Encryption Key – MEK)

- Used to encrypt device
  - change means complete reencryption
- Usually generated by a secure RNG

## Unlocking key (~ Key Encryption Key – KEK)

- Key wrap (MEK remains the same)
- Can be derived from passphrase
  - PBKDF2 (Password Based Key Derivation)
  - scrypt, Argon2 (memory-hard KDFs)
    dictionary and brute-force resistance

# Key storage

**Outside of encrypted device / filesystem**

- Another device, file, token, SmartCard, TPM, HSM
- On a key server (network)
- Protected by another key – key wrap, key encapsulation

**On the same disk (with encrypted data)**

- Metadata on-disk – key slots

**Integration with key management tools**

- LDAP, Active Directory, ...

**Combination of above**

# Key removal and recovery

## Key removal (wipe of key) = data disposal

- intended (secure disk disposal)
- unintended (error) => complete lost of data

## Key recovery

- trade-off between security and user-friendly approach
- metadata backups
- multiple metadata copies
- Key Escrow (key backup to different system)
- recovery key to regenerate encryption key

File and disk encryption

# ATTACK EXAMPLES

# **Attacks** always get better, they never get worse.

- **Against algorithm design**
  - wrongly used encryption mode, IV
- **To implementation**
  - insufficient entropy (broken RNG)
  - weak derivation from weak passwords
  - side channels
- **Obtaining key or passphrase in open form**
  - Cold Boot
  - “Black bag analysis” - Malware, key-logger
  - social engineering, “Rubber-hose cryptoanalysis”

# Integrity attacks

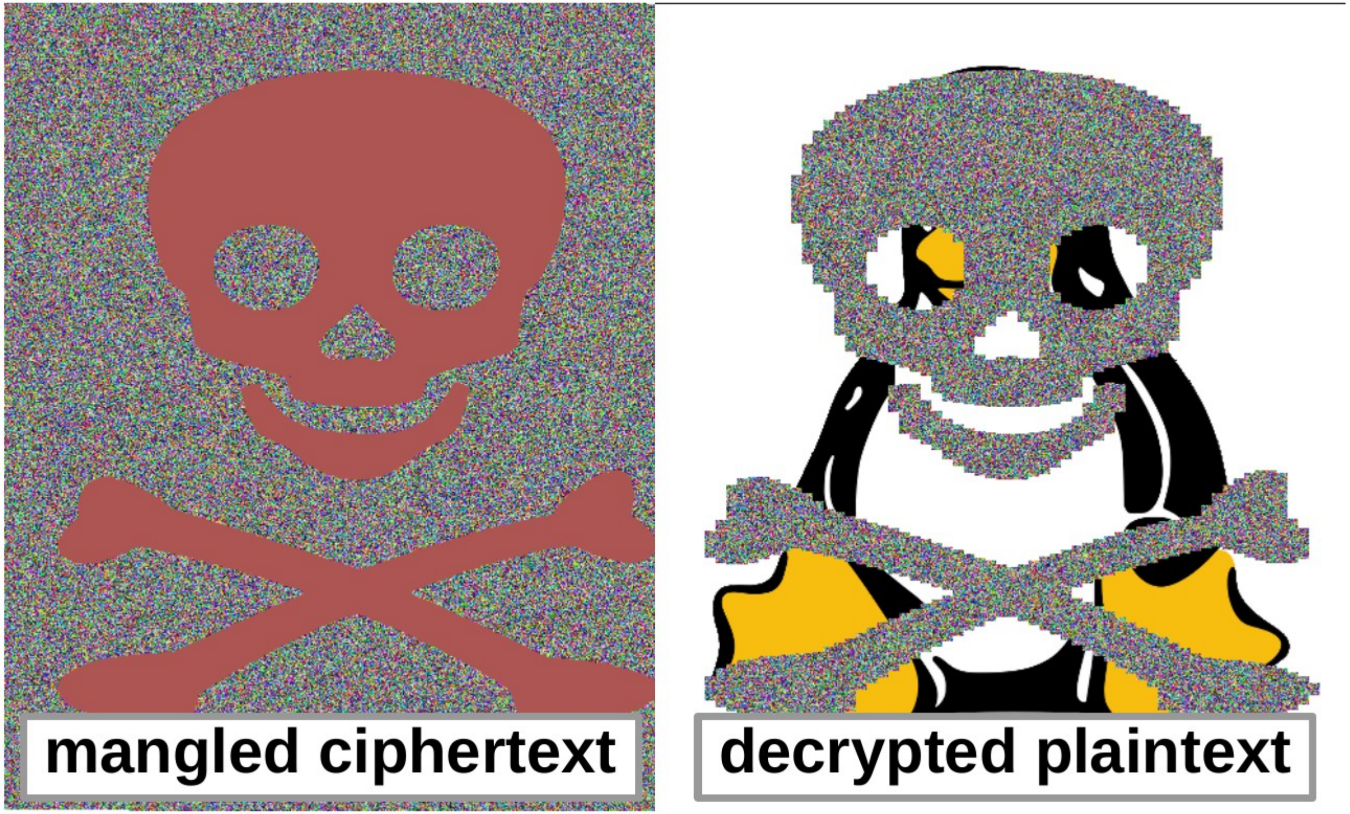
## No integrity protection

- Inserted random block
  => undetected data corruption
- Inserted block from other part of disk
- Random error (RAM bit flip)
  => “silent data corruption”

## Weak integrity protection

- Inserted previous content of (ciphertext) block
  => replay attack

# Integrity attacks



# FDE attacks – real-world examples

- Some chipsets use ECB mode
- Weak key derivation (brute-force possible)
- Trivial unlocking mode (1-bit password is ok/bad)
- Weak key-escrow (backup key in EEPROM)
- SED – switch power attacks
- SED – ransomware and unconfigured passphrase
- Cold boot – key in memory
- Key loggers
- Weak RNG (key is not random)
- LUKS2 reencryption (forced decryption)

# LAB

# Laboratory – FDE attack examples

**Basic understanding of FDE**

- VeraCrypt, LUKS, (BitLocker, FileVault2)

**Scanning memory image for encryption key**

- ColdBoot attack principle

**HW key-logger attack**

- Why you have to trust your HW

**Optional: flawed algorithm and watermarking**

Revealing legacy TrueCrypt hidden disk existence (CBC)