File and disk encryption

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

Centre for Research on Cryptography and Security

File and disk encryption

Lecture

- File and disk encryption (data-at-rest)
- Distributed storage encryption
- Confidentiality and integrity protection
- Encryption modes
- Key management
- Attacks and common issues

- We will focus on low-level building blocks so you can understand storage security in general

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)

Terminology

(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, database, ...
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)

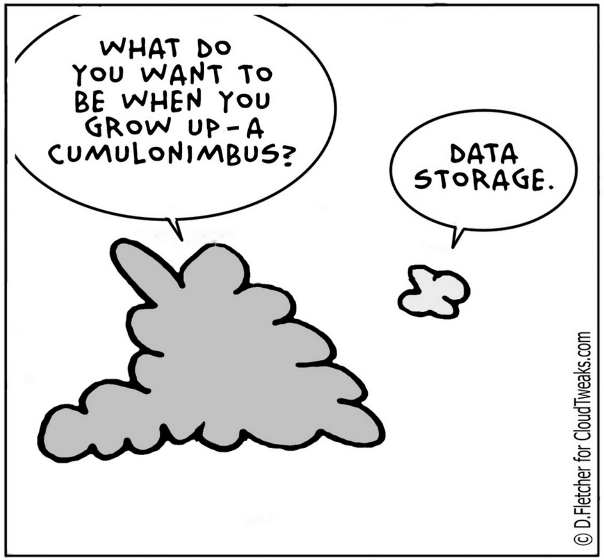
- **Distributed storage – storage + network layer**
  - **Must** use also network layer encryption
  - Note differences in network and storage encryption (replay attack resistance, integrity protection, …)

Distributed Storage, Cloud & Encryption

Distributed storage – add network layer

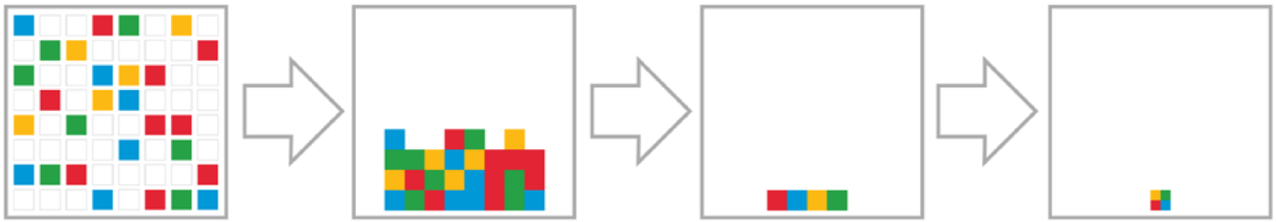
- **Shared volumes** (disk encryption below)
- **Clustered file-system** (fs encryption)
- **Distributed object store** (object encryption)

- **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 (container)
- 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

+/- 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)
- more complicated sw, usually more bugs
- unusable for swap partitions

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

File vs. disk encryption - data integrity

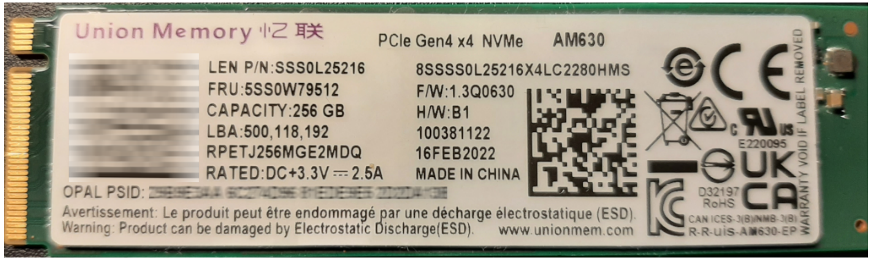
- **confidentiality**, but usually **no data integrity protection**
- often **non-cryptographic parity**/checksum only
    - fs checksums (CRC, xxhash)
- **HW support** (DIF - data integrity field)
    - usually not large enough
- Linux kernel authenticated encryption
    - bcachefs (filesystem)
    - dm-integrity + dm-crypt (LUKS2 FDE)
- performance problems

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)
- new **KPIO** (key-per-io) - multiple keys implanted from OS



File and disk encryption

DATA ENCRYPTION

Disk encryption algorithms primitives

Symmetric encryption

- Block ciphers
- Cipher block mode + initial vector / tweaks
- Hash, HMAC
- Authenticated encryption (AEAD)

Key management and key storage

- Random Number Generators (RNG)
- Key Derivation Functions (KDF)
- Key wrapping

Data confidentiality, integrity, resilience

Confidentiality

Data are available only to authorized users

Integrity

Data consistency

Data cannot be 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.

Resilience

Data integrity can be securely recovered

(Backup, redundancy / replication, error correction, ...)

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

Combination of features...

Storage performance, reliability and easy to use

- is often enemy to storage cryptographic security :-)
- weak (but fast) algorithms
- non-cryptographic hashes
- redundancy (RAID, FEC - forward error correction)
- deduplication, compression, acess recovery

The goal is to understand threat model and design and implement system without introducing too many weak points.
There is always a trade-off in storage security for commodity HW.

File and disk encryption

DATA ENCRYPTION MODES

Symmetric encryption (examples)

AES, Cammelia, Adiantum, Serpent, Twofish, ...

Confidentiality-only modes

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

Authenticated modes (encryption + integrity)

- AES-GCM, (X)ChaCha20-Poly1305, AEGIS
- Integrity protection often on higher layer

Standards

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

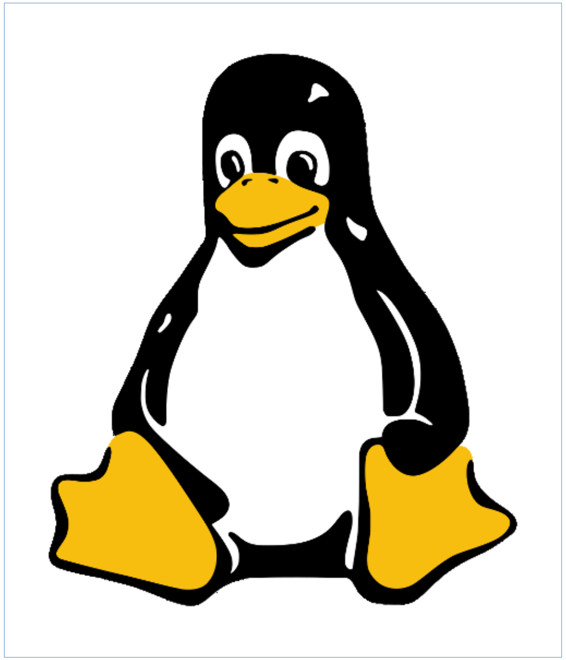
Many other as IETF **RFC** documents.

Propagation of plaintext changes

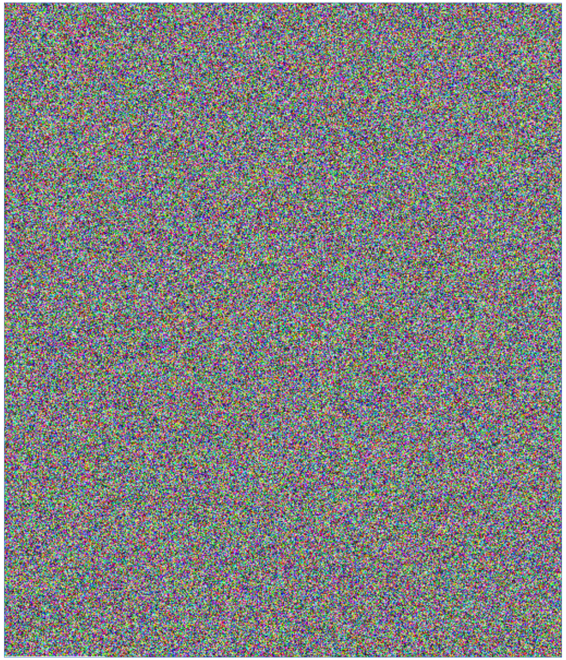
A change in plaintext should transform to randomly-looking change in the whole ciphertext sector. Solutions?

- **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

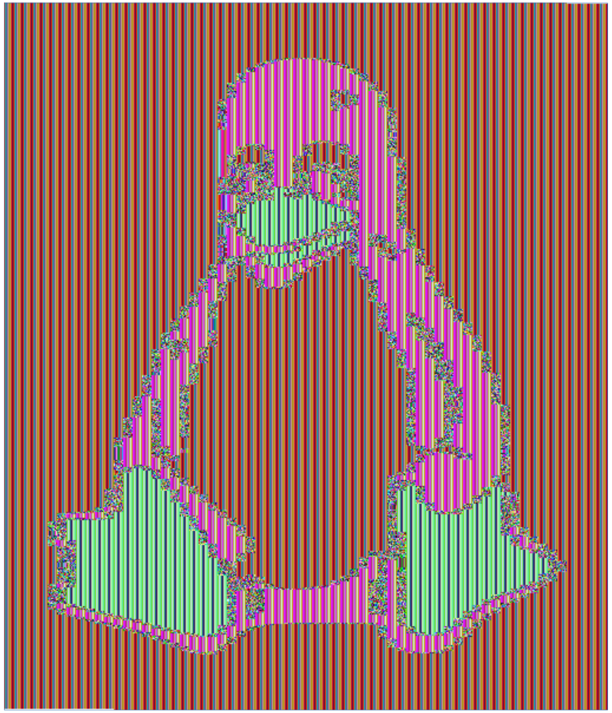


plaintext

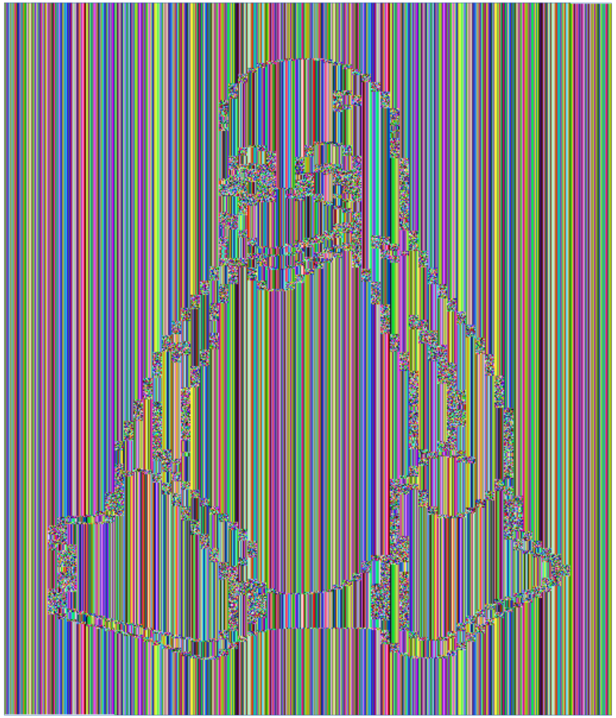


ciphertext

Wrongly used encryption – patterns, leaks



ECB mode

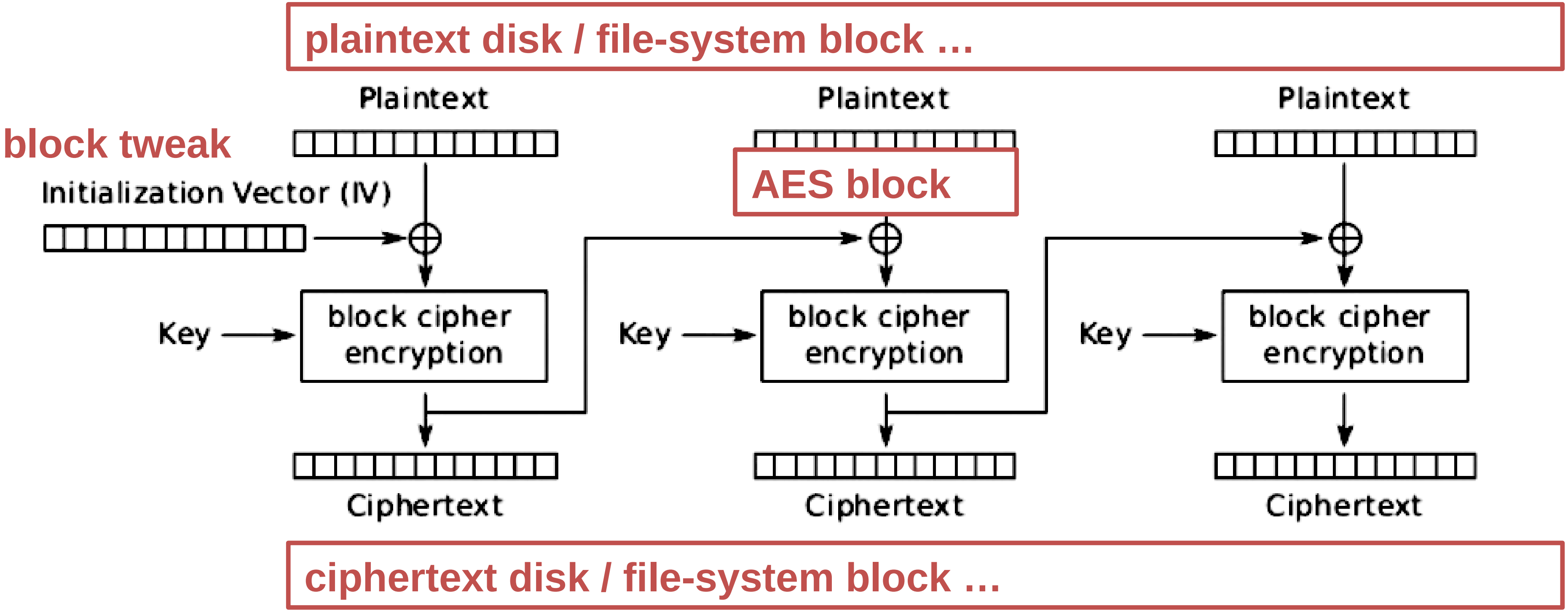


AES-XTS & constant IV

Cipher-Block-Chaining (CBC) mode

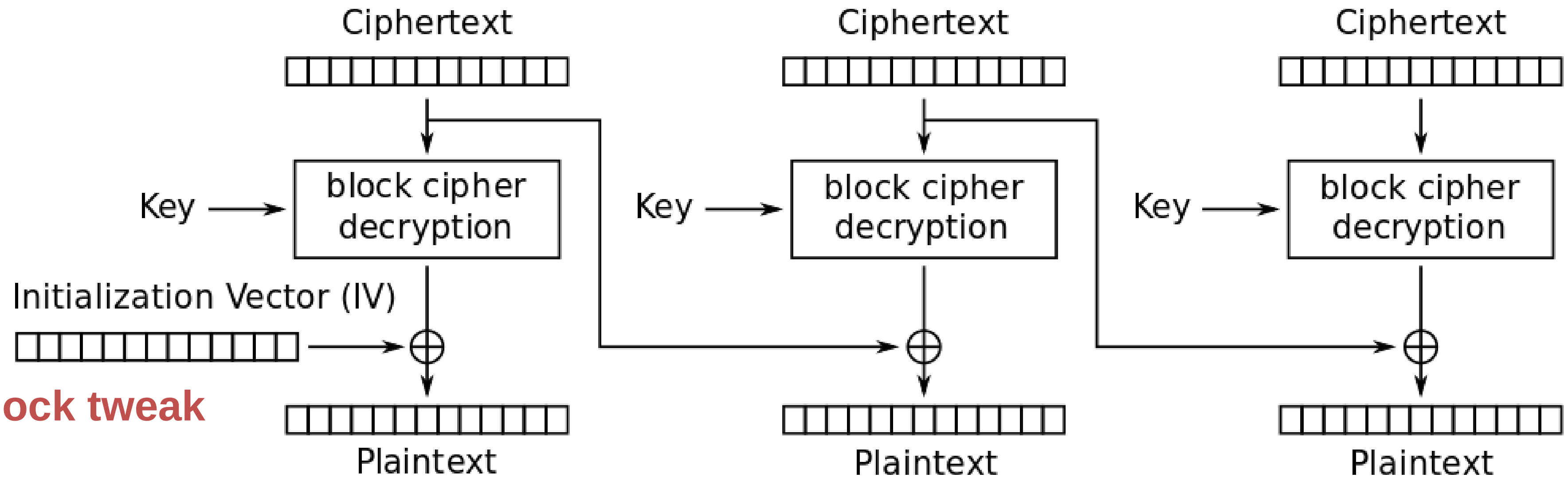
- Blocks cannot be encrypted in parallel
- Blocks can be decrypted in parallel
- Tweak must be non-predictable (watermarking!)

CBC encryption



CBC decryption

ciphertext disk / file-system block …



block tweak

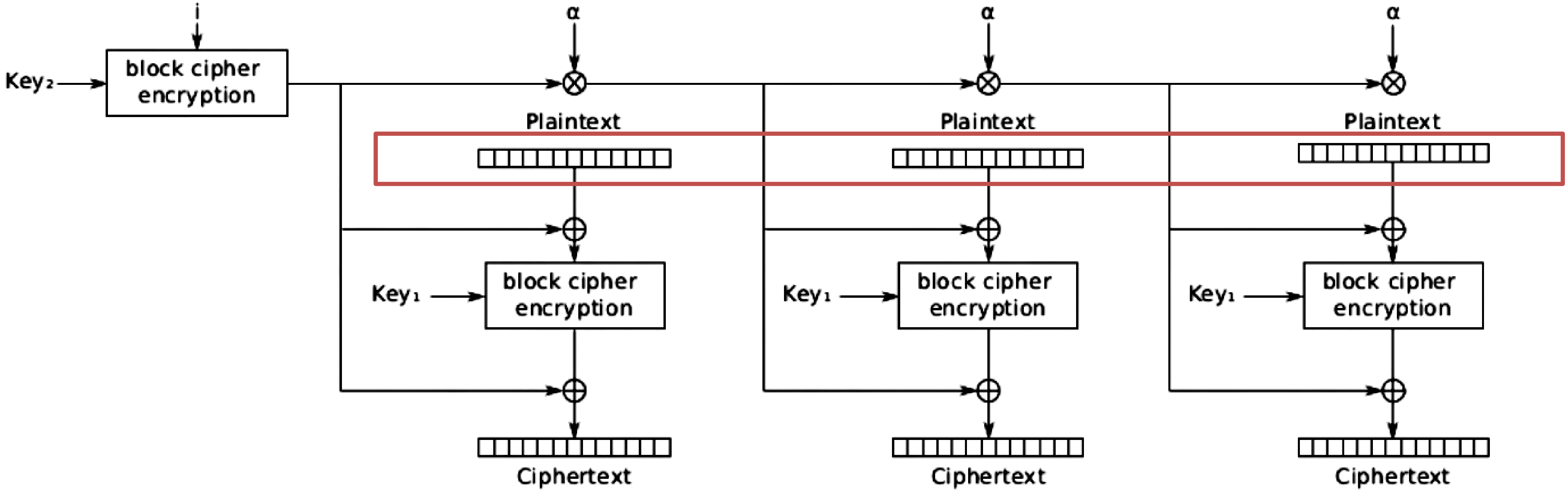
plaintext disk / file-system block …

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 (2024)
- It is not a wide mode!
- Trade-off for performance

XTS mode encryption/decryption

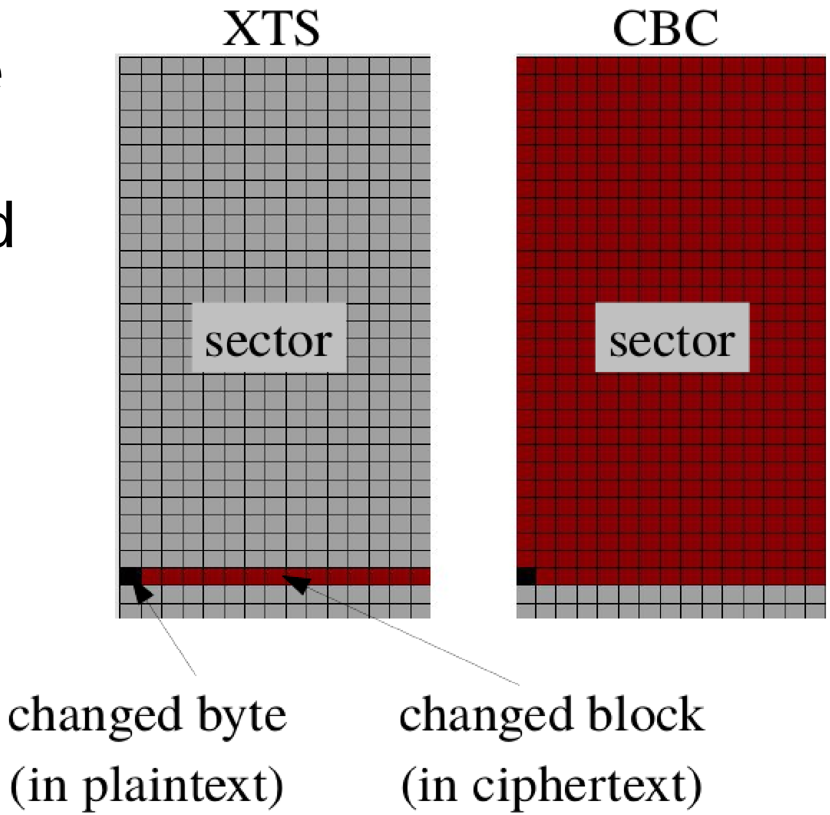
block tweak



ciphertext disk / file-system block …

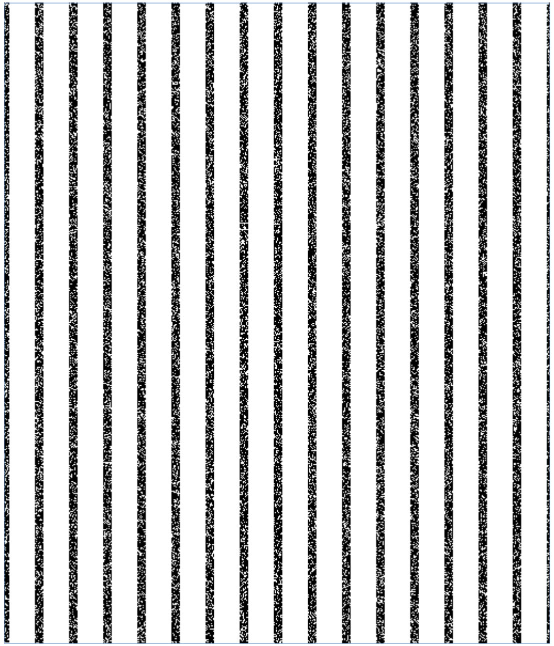
CBC vs XTS change propagation

- 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

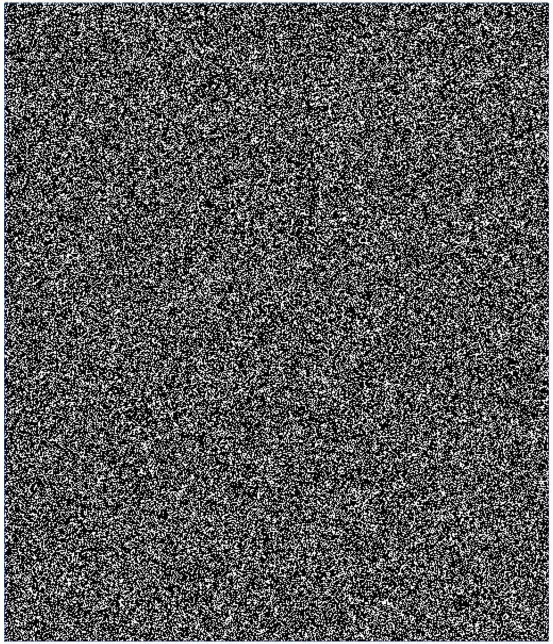


AES-XTS IV mode – sector# vs random

Every 64 byte changed (ciphertext differences)



IV is sector number

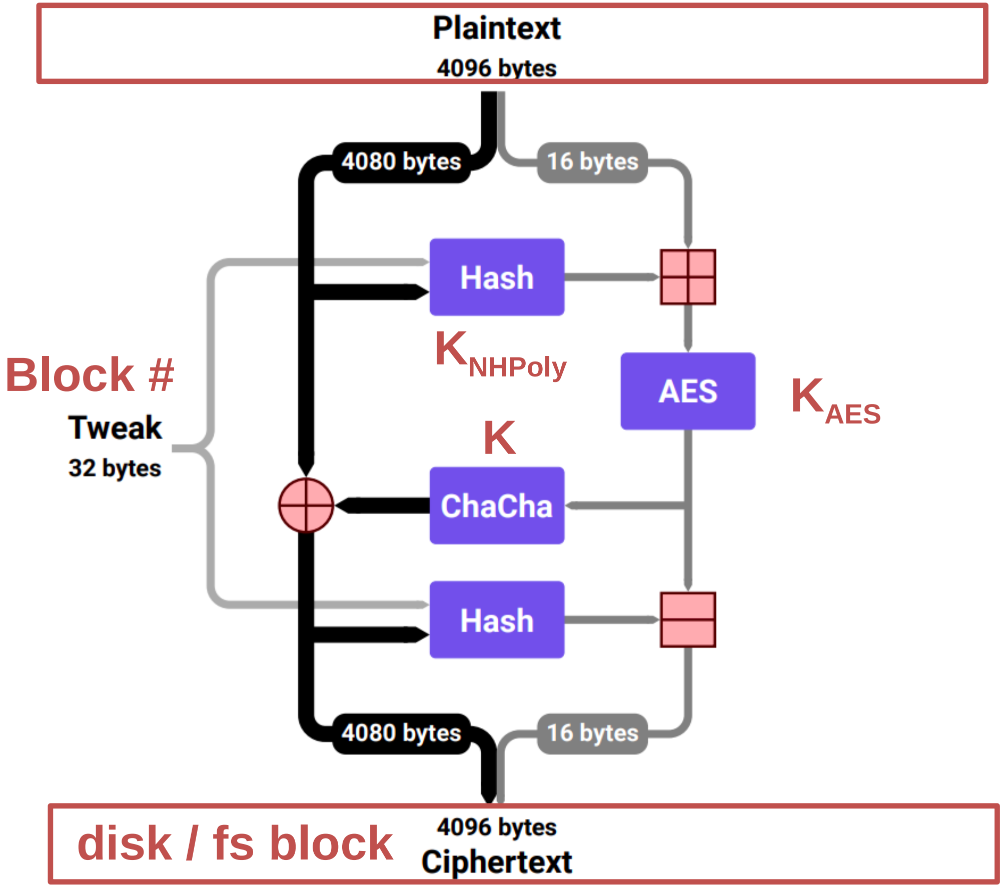


randomized IV

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

  $K_{AES} || K_{NHPoly}$ = 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

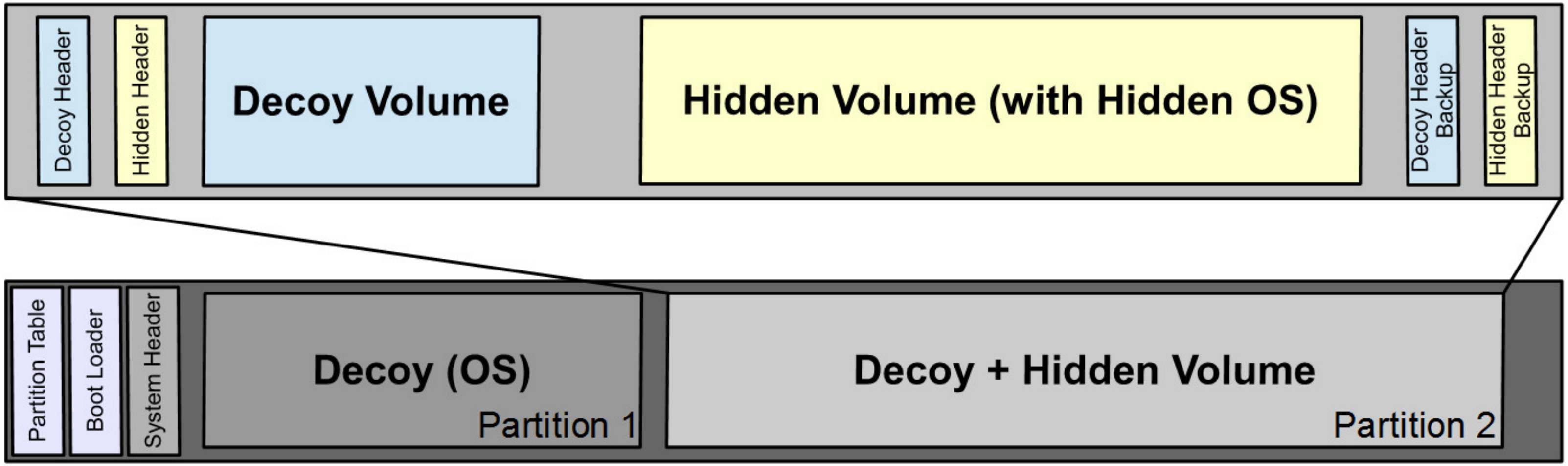
Hiding data in another data object

Some more recent examples:

- TrueCrypt / VeraCrypt - hidden disk
- Shufflecake - multiple hidden filesystems

Trivial example: VeraCrypt hidden disk

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



Deniable encryption problems

Side-channels

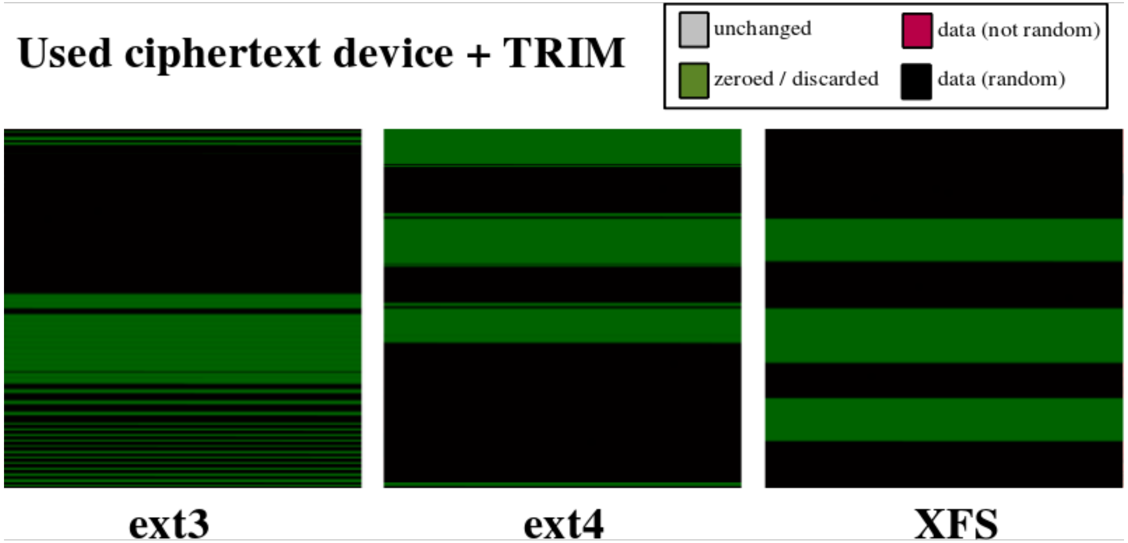
- Tracking activity that cannot be explained for decoy system
- Software: link to recently open documents, …
  Suspicious parameters (FAT), disabled TRIM, …
- Hardware: internal SSD block allocations
  (access to “unused” areas)

Incompatibility with new drives (TRIM)

Note: flash storage is more complicated (NAND chips management, wear-leveling, ...)
With low-level HW access you could detect suspicious patterns.

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

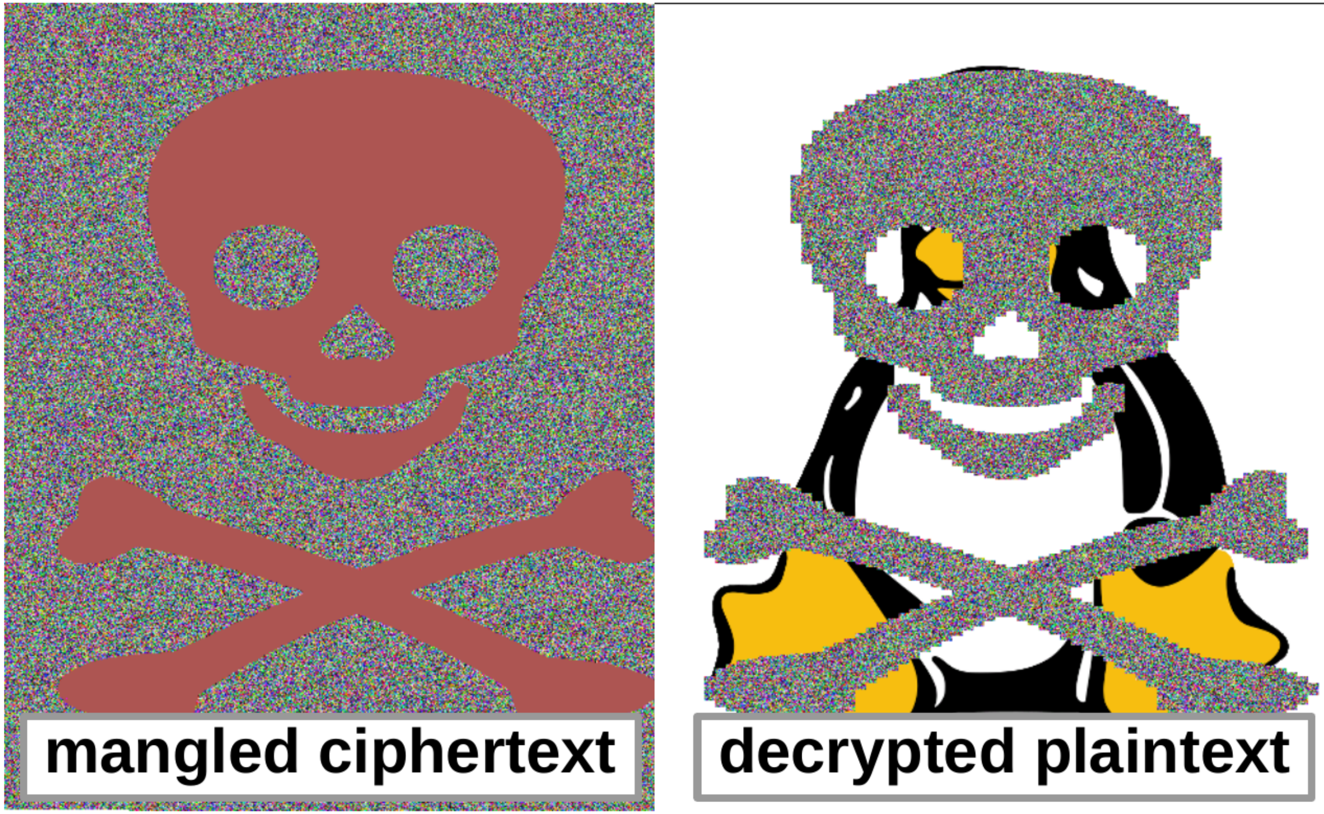
No integrity protection

- Inserted random block
  => undetected data corruption
- Inserted block from other part of disk
- Undetected random error (like **bit flip**) or **erasure** (like hw-replaced unreadable sector)
  => “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)

Scanning memory image for encryption key
- ColdBoot attack principle

HW key-logger attack
- Why you have to trust your HW

Sector data integrity, error correction
- basic principles demonstrated with cryptsetup tools

Optional: flawed algorithm and watermarking
- Revealing legacy TrueCrypt hidden disk existence (CBC)