Authenticated and Encrypted Storage on Embedded Linux

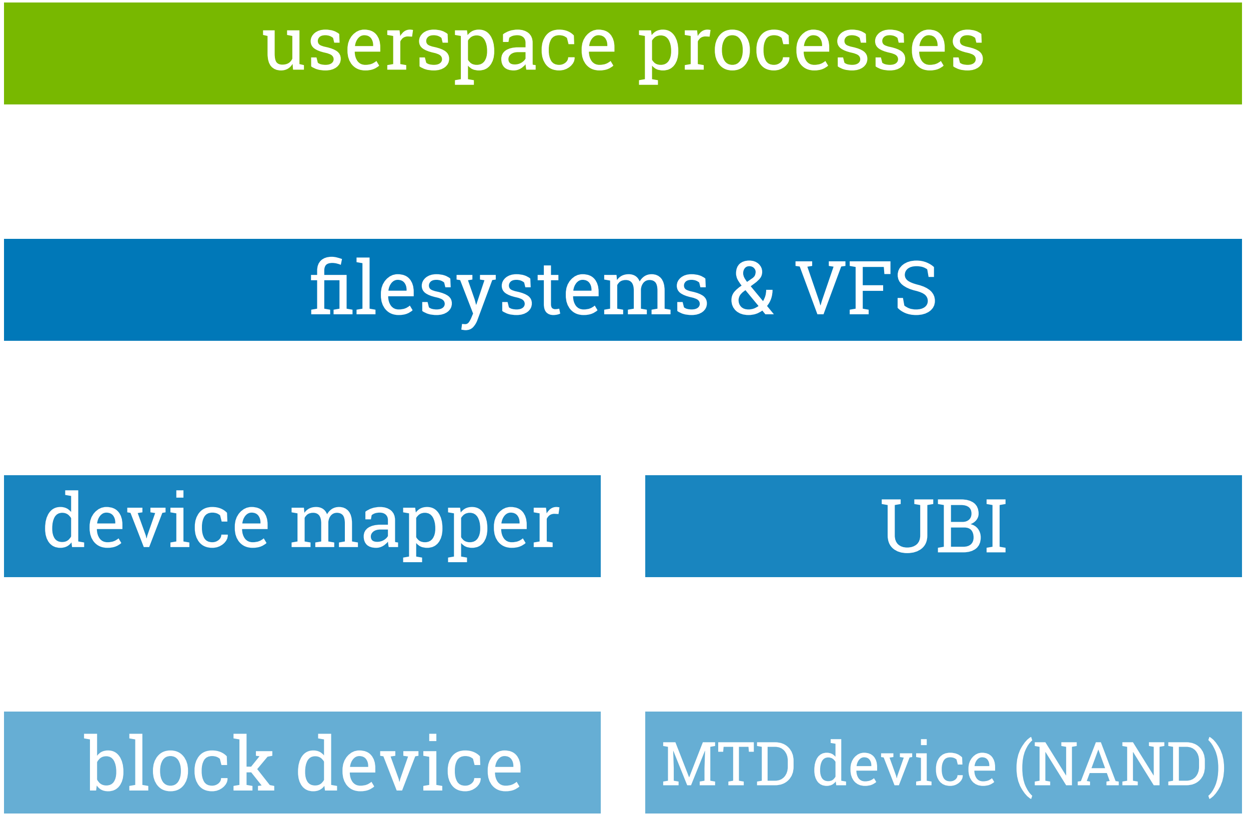
ELC Europe 2019

Jan Lübbe – jlu@pengutronix.de

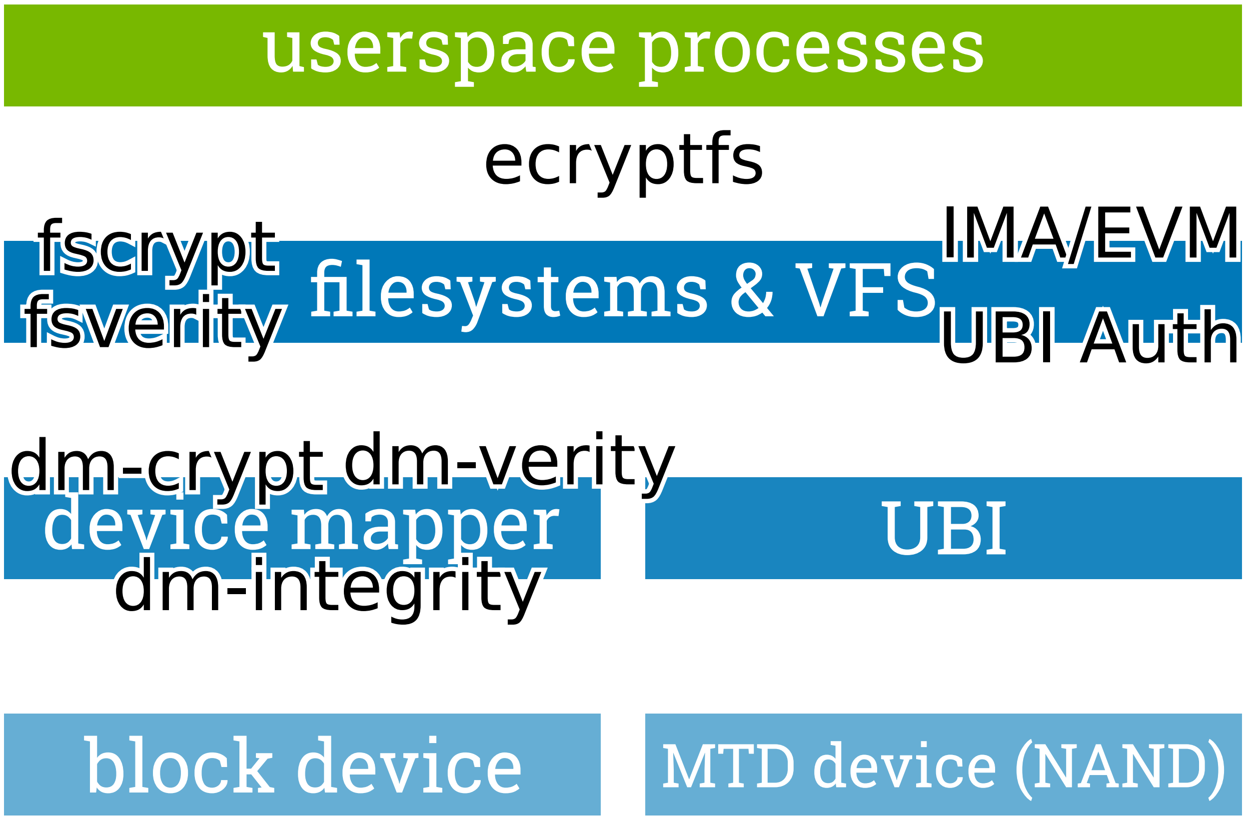
Pengutronix.

https://www.pengutronix.de

Linux Storage Stack



Transparent Authentication and Encryption



Crypto?

https://www.instructables.com/id/Laser-Cut-Cryptex/

Quick Crypto Refresher

Hash: one-way function, fixed output size (SHA*)

HMAC: data authentication using hash and shared secret

Signature: data authentication using public key cryptography (keys & certificates, RSA & ECDSA)

Unauthenticated encryption: attacker can‘t read private data, but could modify it (AES-CBC, AES-XTS, ...)

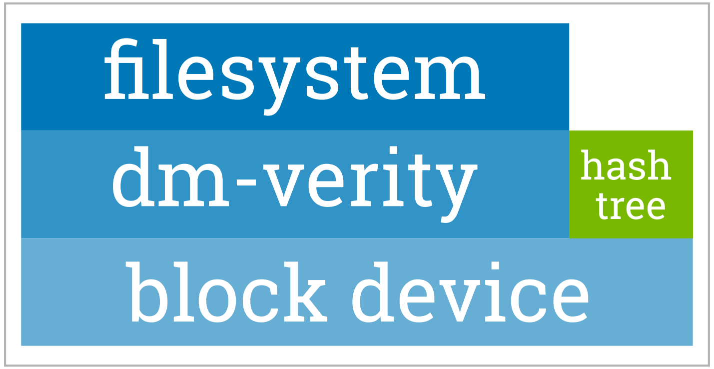
Authenticated encryption: attacker can‘t read private data and modification is detected (AEAD: AES GCM, AEGIS)

Overview

- Building Blocks
  - authentication
  - encryption
  - authenticated encryption
- General Considerations

dm-verity (since 2012, v3.4)

- authentication via hash tree: **read-only**
- used by Chrome OS & Android for rootfs
- root hash provided via out-of-band (kernel cmdline) or via signature in super block (since 5.4)
- can be created and configured via `veritysetup` (LUKS2)
- combine with ext4, SquashFS or EROFS



hash-tree image

⇒ best choice for RO data

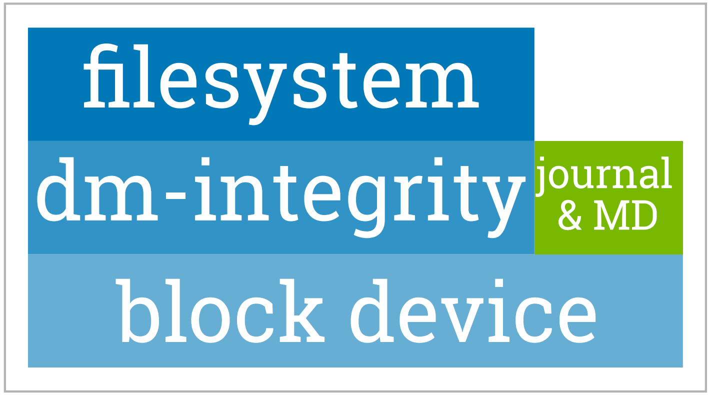
fsverity (since 2019, v5.4)

- “dm-verity for files”: efficient authentication of (large) read-only files via a hash tree
- root hash provided out-of-band
- integrated into ext4
- could be integrated with IMA/EVM to improve performance

⇒ Android will likely be the main user (for .apk authentication)

dm-integrity (since 2017, v4.12)

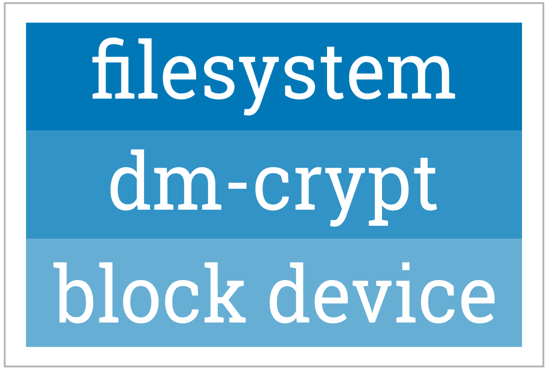
- emulates integrity data on normal block devices
- performance overhead (data written twice due to journaling)



- one meta-data block per n data blocks, interleaved
- can provide simple check-sums without encryption (CRC32/SHA256/-HMAC)
- usually configured via `integritysetup` (LUKS2)

dm-crypt

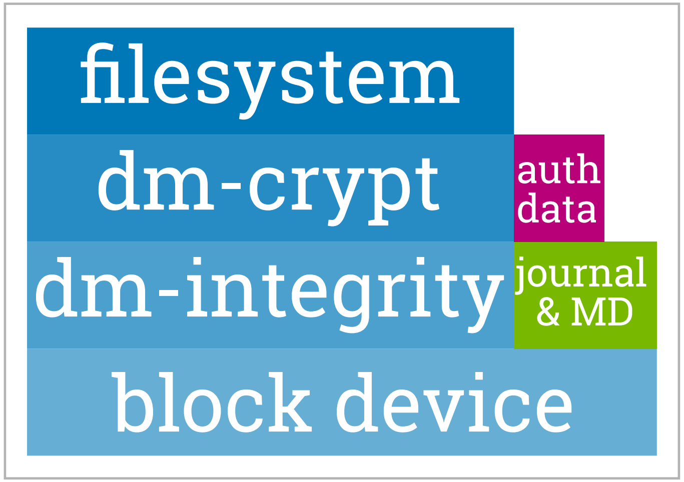
- sector-based encryption of block devices
- supports multiple algorithms and modes
- usually configured using `cryptsetup` (LUKS2)
  - experimental online reencryption
- does **not** authenticate, because that would need additional space (uses “length-preserving encryption”)



⇒ best choice on RW block devices (if auth is not critical)

dm-crypt with authentication

- needs dm-integrity or block device with T10/DIF
- can also use a random initialization vector (IV)
- uses AEAD cipher modes:
  - AES256-GCM-random, AEAD (12B IV, 16B auth tag)
  - **AEGIS128-random**, AEAD (16B IV, 16B auth tag)
  - ChaCha20-random, integrity Poly1305 (16B IV, 32B auth tag)
- only authenticates individual sectors, replay is possible



⇒ best choice on RW block devices for authenticated encryption

fscrypt

- initially ext4-only (2015), then F2FS, generalized in (2016, v4.6), UBIFS support (2017, v4.10)
- file-based encryption, supports different keys for multiple users
- files can be removed without key
- no authentication

⇒ alternative to dm-crypt for multi-user systems (like Android)

ecryptfs (since 2006, v2.6.19)

- stacked file system (problems)
- default home directory encryption method for Ubuntu beginning with 9.04, now deprecated, maintenance unclear
- no authentication, GCM patches posted, but not merged
- encrypts data and filenames

⇒ superseded by fscrypt

IMA/EVM (since 2009/2011, v2.6.30/v3.2)

- initially developed for usage with TPMs, Verified Boot and Remote Attestation
- uses extended attributes
- EVM appraisal can protect against file data modification, but currently not against directory modification (`cp /bin/sh /sbin/init`)

⇒ IMA for remote attestation, EVM is problematic for local auth.

UBIFS Authentication (since 2018, v4.20)

- UBIFS is copy-on-write (because flash): a “wandering tree”
- Hashes added to tree nodes
- root hash (in superblock) authenticated via HMAC or signature for image deployment (since v5.3)
- is the only FS which authenticates full data and metadata

⇒ best choice for raw NAND/MTD devices

Master Key Storage

How can we protect the key that protects the data?

- embedded: no user to enter a password
- Many SoCs have HW that can “wrap” (encrypt) keys with a fixed per-device key (only useful with secure boot)
- Other options: OP-TEE or TPM

See Gilad Ben Yossefs talk on hardware protected keys (earlier today): https://sched.co/TLJE

Recovery: Split RO and RW?

Authenticated, writable storage can only detect offline attacks!

- no difference between intentional and malicious modification (possibly caused by root-level intrusion)

⇒ signed root file system allows recovery via reboot

⇒ read-only recovery system allows factory reset

Field Return Mode

How can we analyze problems on returned hardware?

⇒ implement authenticated method to:

- erase keys for private data
- disable verified boot

Recommendations

- dm-crypt (maybe with dm-integrity) for RW block device
- dm-veritiy for RO data
- UBIFS authentication for NAND
- secure boot and key wrapping for master key protection
- HW acceleration for ciphers

⇒ avoid complexity, select only the necessary components

Thanks!

Questions?

Further Reading

dm-verity: https://gitlab.com/cryptsetup/cryptsetup/wikis/DMVerity

dm-integrity: https://gitlab.com/cryptsetup/cryptsetup/wikis/DMIntegrity

dm-crypt+dm-integrity: https://arxiv.org/abs/1807.00309

fscrypt: https://www.kernel.org/doc/html/latest/filesystems/fscrypt.html

fsverity: https://www.kernel.org/doc/html/latest/filesystems/fsverity.html

ubifs auth: https://www.kernel.org/doc/html/latest/filesystems/ubifs-authentication.html

Authenticated and Encrypted Storage on Embedded Linux

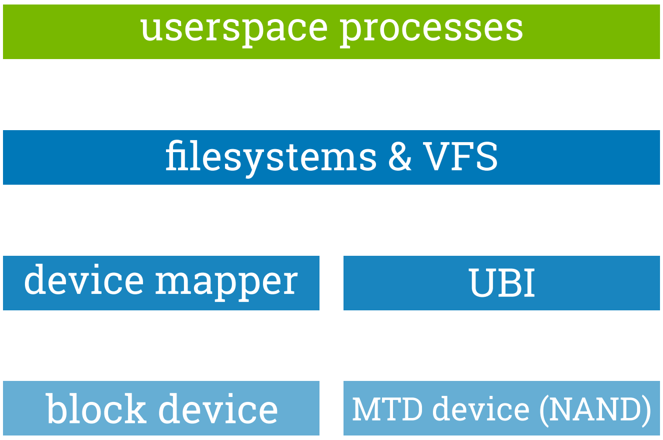
ELC Europe 2019

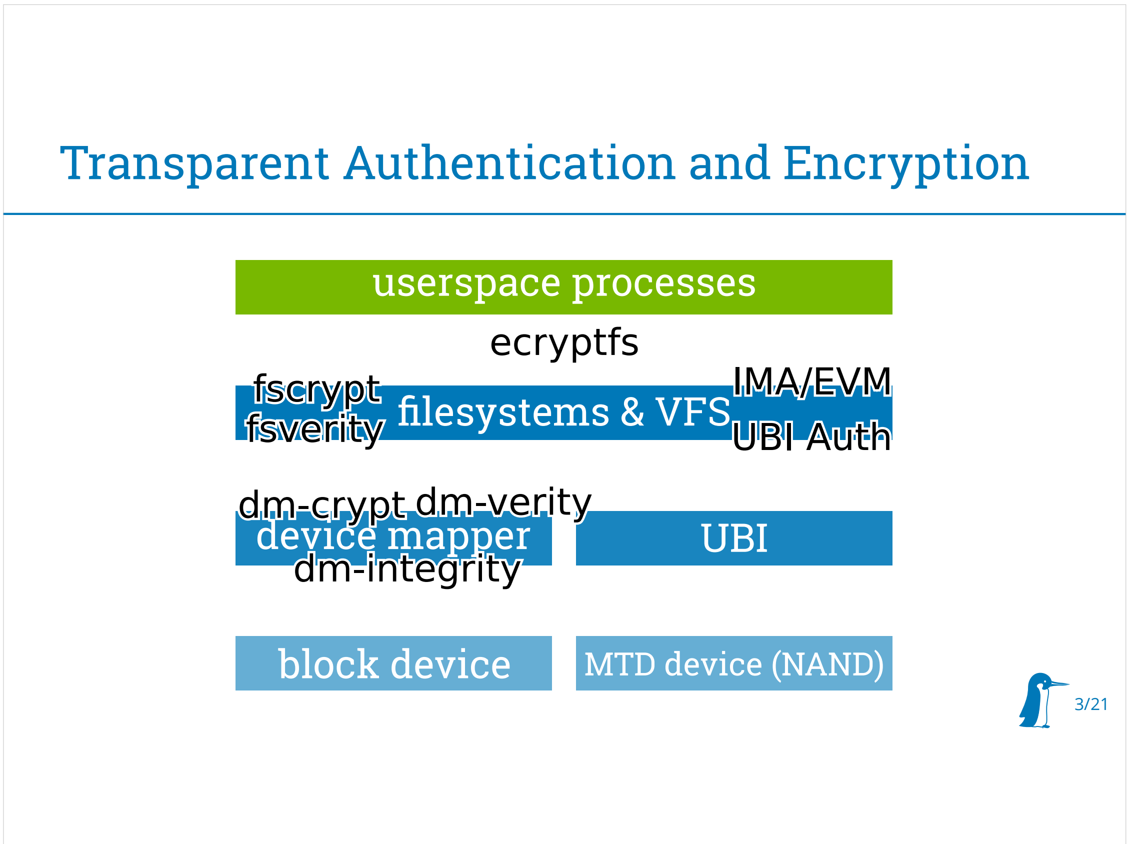
Jan Lübbe – jlu@pengutronix.de

Pengutronix

https://www.pengutronix.de

Linux Storage Stack





we only look at kernel infra, transparent for applications

audience: developers, need to decide between tools

often only one correct choice for a given project

Crypto?

https://www.instructables.com/id/Laser-Cut-Cryptex/

Quick Crypto Refresher

Hash: one-way function, fixed output size (SHA*)

HMAC: data authentication using hash and shared secret

Signature: data authentication using public key cryptography (keys & certificates, RSA & ECDSA)

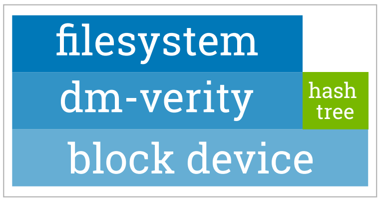
Unauthenticated encryption: attacker can‘t read private data, but could modify it (AES-CBC, AES-XTS, ...)

Authenticated encryption: attacker can‘t read private data and modification is detected (AEAD: AES GCM, AEGIS)

Overview

- Building Blocks
  - authentication
  - encryption
  - authenticated encryption
- General Considerations

dm-verity (since 2012, v3.4)



- authentication via hash tree: **read-only**
- used by Chrome OS & Android for rootfs
- root hash provided via out-of-band (kernel cmdline) or via signature in super block (since 5.4)
- can be created and configured via `veritysetup` (LUKS2)
- combine with ext4, SquashFS or EROFS

hash-tree image

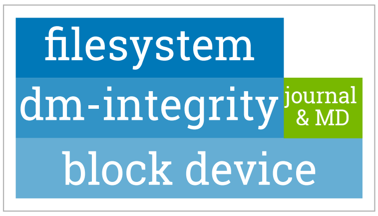
⇒ best choice for RO data

fsverity (since 2019, v5.4)

- “dm-verity for files”: efficient authentication of (large) read-only files via a hash tree
- root hash provided out-of-band
- integrated into ext4
- could be integrated with IMA/EVM to improve performance

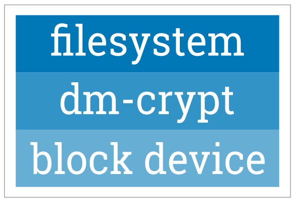
⇒ Android will likely be the main user (for .apk authentication)

dm-integrity (since 2017, v4.12)



- emulates integrity data on normal block devices
- performance overhead (data written twice due to journaling)
- one meta-data block per n data blocks, interleaved
- can provide simple check-sums without encryption (CRC32/SHA256/-HMAC)
- usually configured via `integritysetup` (LUKS2)

dm-crypt

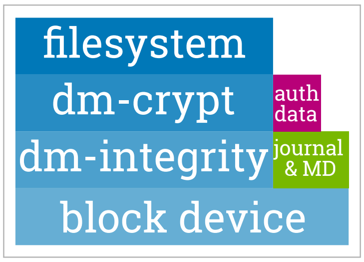


- sector-based encryption of block devices
- supports multiple algorithms and modes
- usually configured using `cryptsetup` (LUKS2)
  - experimental online reencryption
- does **not** authenticate, because that would need additional space (uses "length-preserving encryption")

⇒ best choice on RW block devices (if auth is not critical)

dm-crypt with authentication

- needs dm-integrity or block device with T10/DIF
- can also use a random initialization vector (IV)
- uses AEAD cipher modes:
  - AES256-GCM-random, AEAD (12B IV, 16B auth tag)
  - **AEGIS128-random**, AEAD (16B IV, 16B auth tag)
  - ChaCha20-random, integrity Poly1305 (16B IV, 32B auth tag)
- only authenticates individual sectors, replay is possible



⇒ best choice on RW block devices for authenticated encryption

fscrypt

- initially ext4-only (2015), then F2FS, generalized in (2016, v4.6), UBIFS support (2017, v4.10)
- file-based encryption, supports different keys for multiple users
- files can be removed without key
- no authentication

⇒ alternative to dm-crypt for multi-user systems (like Android)

ecryptfs (since 2006, v2.6.19)

- stacked file system (problems)
- default home directory encryption method for Ubuntu beginning with 9.04, now deprecated, maintenance unclear
- no authentication, GCM patches posted, but not merged
- encrypts data and filenames

⇒ superseded by fscrypt

IMA/EVM (since 2009/2011, v2.6.30/v3.2)

- initially developed for usage with TPMs, Verified Boot and Remote Attestation
- uses extended attributes
- EVM appraisal can protect against file data modification, but currently not against directory modification (`cp /bin/sh /sbin/init`)

⇒ IMA for remote attestation, EVM is problematic for local auth.

UBIFS Authentication (since 2018, v4.20)

- UBIFS is copy-on-write (because flash): a “wandering tree”
- Hashes added to tree nodes
- root hash (in superblock) authenticated via HMAC or signature for image deployment (since v5.3)
- is the only FS which authenticates full data and metadata

⇒ best choice for raw NAND/MTD devices

Master Key Storage

How can we protect the key that protects the data?

- embedded: no user to enter a password
- Many SoCs have HW that can “wrap” (encrypt) keys with a fixed per-device key (only useful with secure boot)
- Other options: OP-TEE or TPM

See Gilad Ben Yossefs talk on hardware protected keys (earlier today): https://sched.co/TLJE

Recovery: Split RO and RW?

Authenticated, writable storage can only detect offline attacks!

- no difference between intentional and malicious modification (possibly caused by root-level intrusion)

⇒ signed root file system allows recovery via reboot

⇒ read-only recovery system allows factory reset

Field Return Mode

How can we analyze problems on returned hardware?

⇒ implement authenticated method to:

- erase keys for private data
- disable verified boot

Recommendations

- dm-crypt (maybe with dm-integrity) for RW block device
- dm-veritiy for RO data
- UBIFS authentication for NAND
- secure boot and key wrapping for master key protection
- HW acceleration for ciphers

⇒ avoid complexity, select only the necessary components

Thanks!

Questions?

Further Reading

dm-verity: https://gitlab.com/cryptsetup/cryptsetup/wikis/DMVerity

dm-integrity: https://gitlab.com/cryptsetup/cryptsetup/wikis/DMIntegrity

dm-crypt+dm-integrity: https://arxiv.org/abs/1807.00309

fscrypt: https://www.kernel.org/doc/html/latest/filesystems/fscrypt.html

fsverity: https://www.kernel.org/doc/html/latest/filesystems/fsverity.html

ubifs auth: https://www.kernel.org/doc/html/latest/filesystems/ubifs-authentication.html