Local Data Security

Mihai Ordean
Designing and Managing Secure Systems
University of Birmingham

Overview

- Device security
  - Is code on the device vulnerable to exploits ? (e.g. buffer overflows)
  - Is the code authenticated ? (i.e. has not been tampered with)
- **Local data security**
  - Is the stored data is accessible to everyone? (e.g. encrypted)
  - Is the stored data authenticated?
- Cloud data security
  - How is data stored in the cloud?
  - Who has access to data stored in the cloud?
- Metadata security
  - What does metadata reveal about stored data?
  - Can we tamper the metadata?

Overview

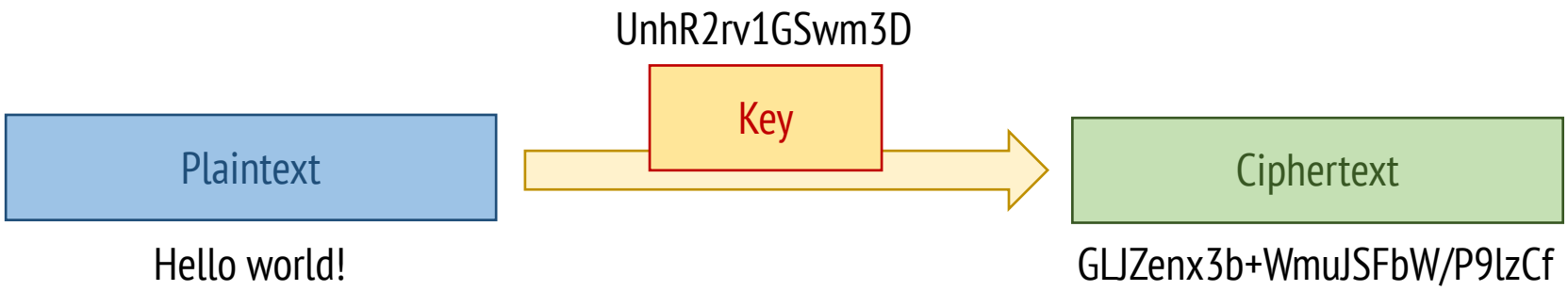
- Data security
  - **Protecting the operating system partition**
  - **Protecting user data**
  - Protecting user data in the cloud

Introduction

Symmetric Encryption

Key

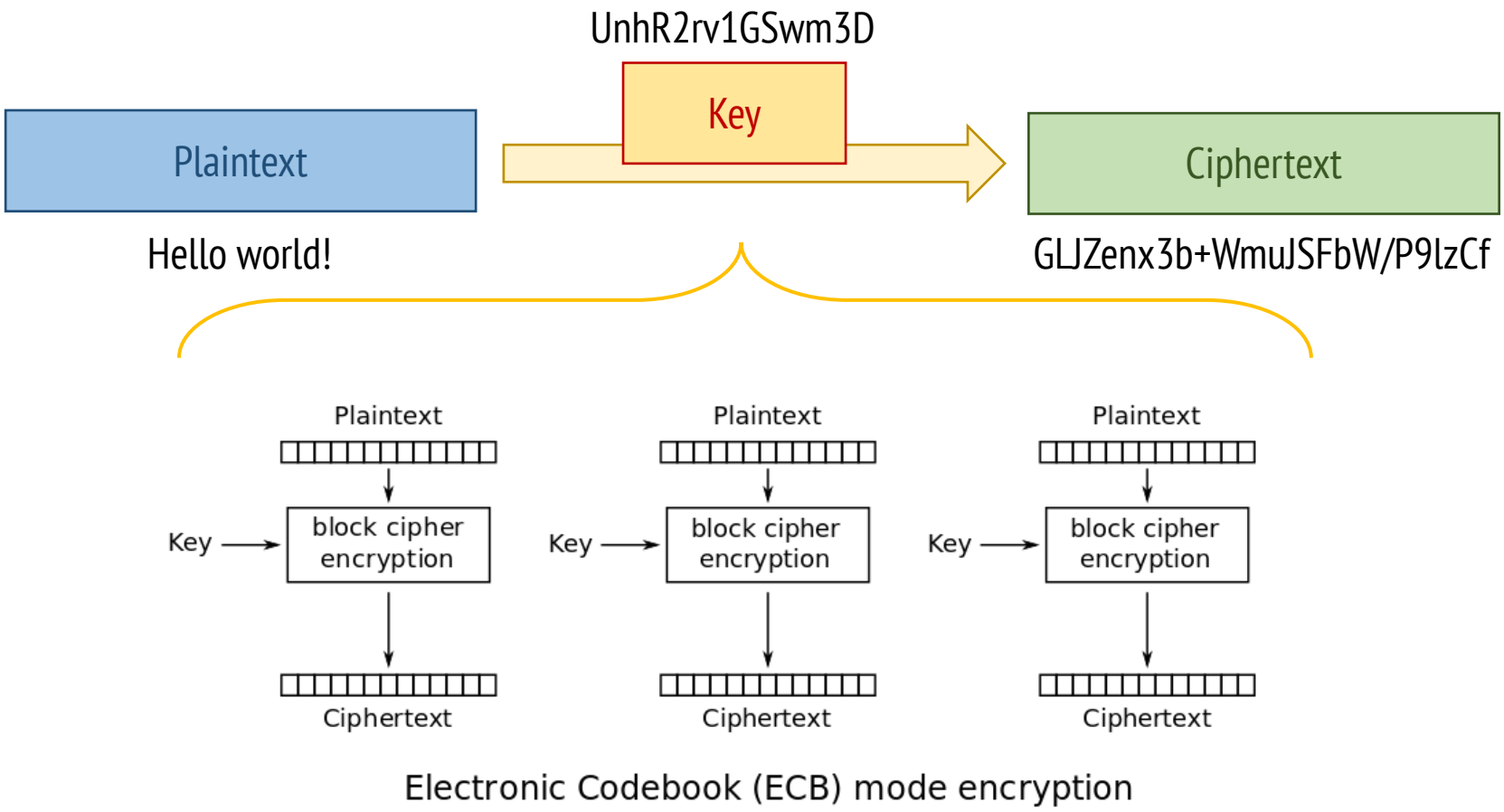
Encryption:



Symmetric Encryption

Key

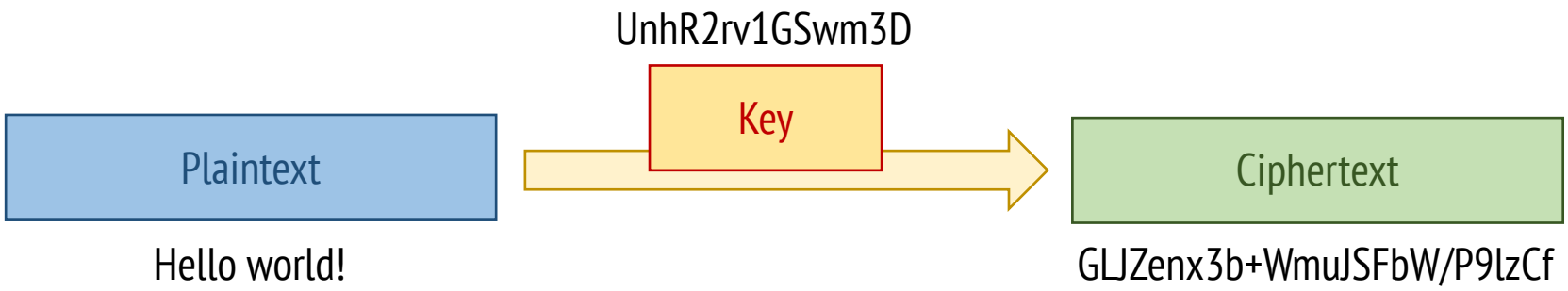
Encryption:



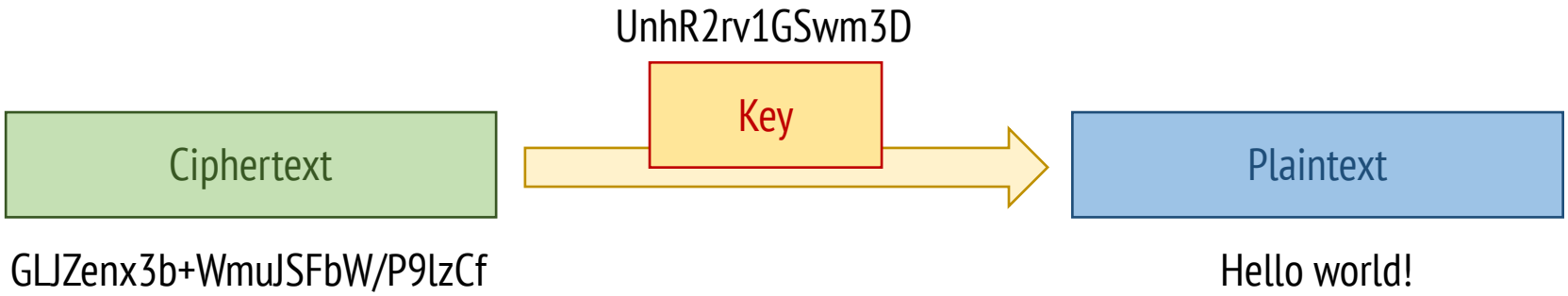
Symmetric Encryption

Key

Encryption:

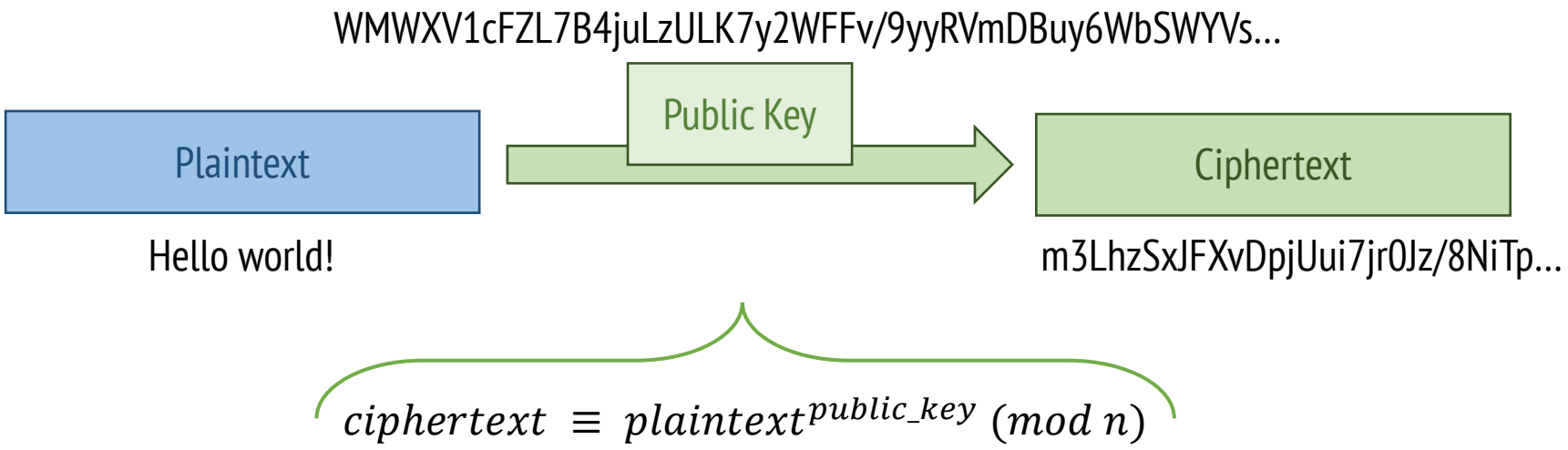


Decryption:



Public key encryption

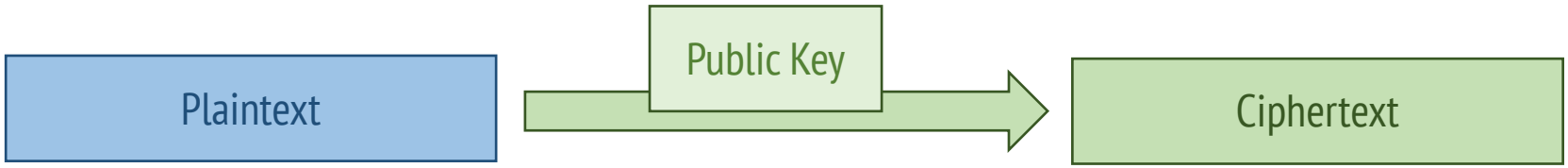
Encryption:



Public key encryption

Encryption:

WMWXV1cFZL7B4juLzULK7y2WFFv/9yyRVmDBuy6WbSWYVs...

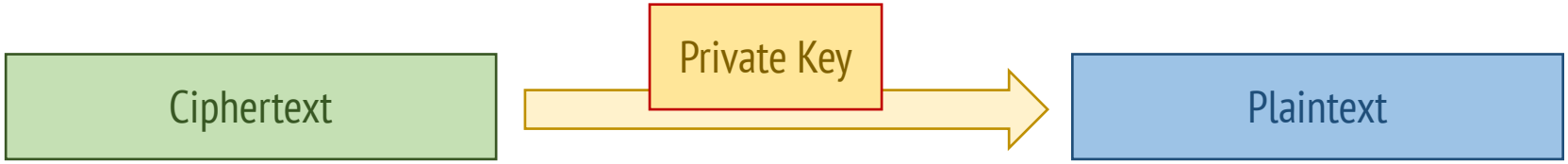


Hello world!

m3LhzSxJFXvDpjUui7jr0Jz/8NiTp...

Decryption:

VjurJb0ZlAkmQv8xDYyStiXnsm40vYEmGanwXMUVAN2xqYtb5YFb1aOLBDncMF...



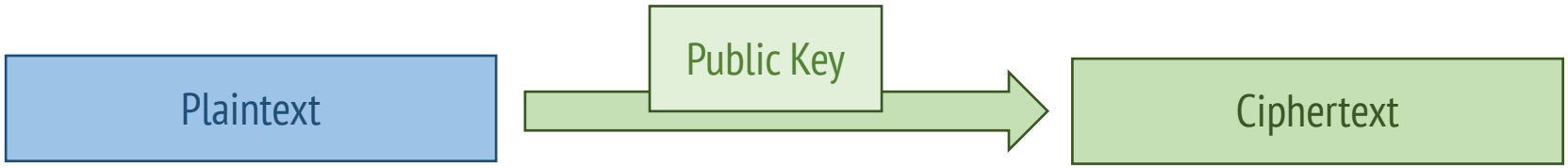
m3LhzSxJFXvDpjUui7jr0Jz/8NiTp....

Hello world!

Public key encryption

Encryption:

WMWXV1cFZL7B4juLzULK7y2WFFv/9yyRVmDBuy6WbSWYVs...

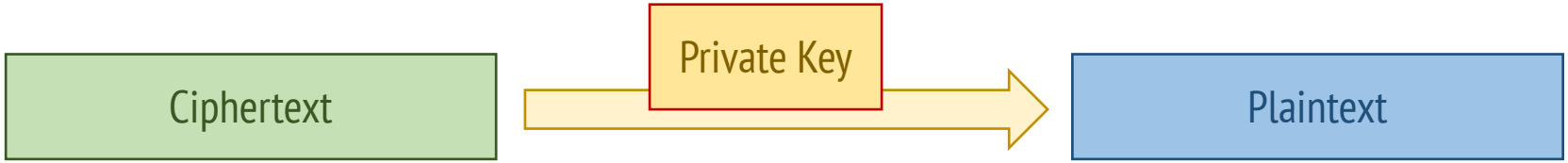


Hello world!

m3LhzSxJFXvDpjUui7jr0Jz/8NiTp...

Decryption:

VjurJb0ZlAkmQv8xDYyStiXnsm40vYEmGanwXMUVAN2xqYtb5YFb1aOLBDncMF...



m3LhzSxJFXvDpjUui7jr0Jz/8NiTp....

Hello world!

$$ciphertext^{private\_key} \equiv plaintext \ (mod \ n)$$

Public key vs. symmetric key

Public key cryptography

- Anyone can encrypt messages and only the key owner can decrypt the ciphertext
- Public key requires longer keys
- The resulting ciphertexts are larger than the plaintext
- …

Symmetric key cryptography

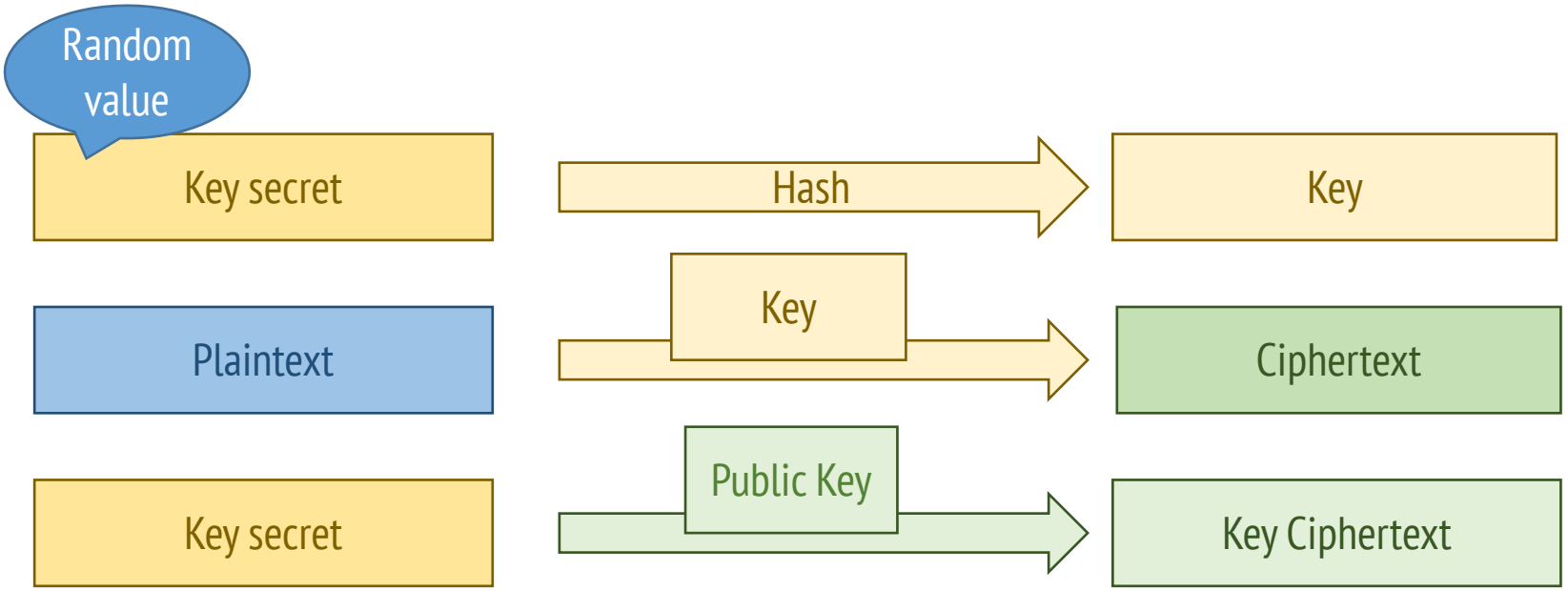
- The encryption/decryption key needs to be shared between parties
- Keys are relatively small
- Resulting ciphertext is about the same size as the plaintext
- …

Key encapsulation mechanisms (KEM)

KEMs are an efficient method to do public key encryption with the help of symmetric key cryptography.

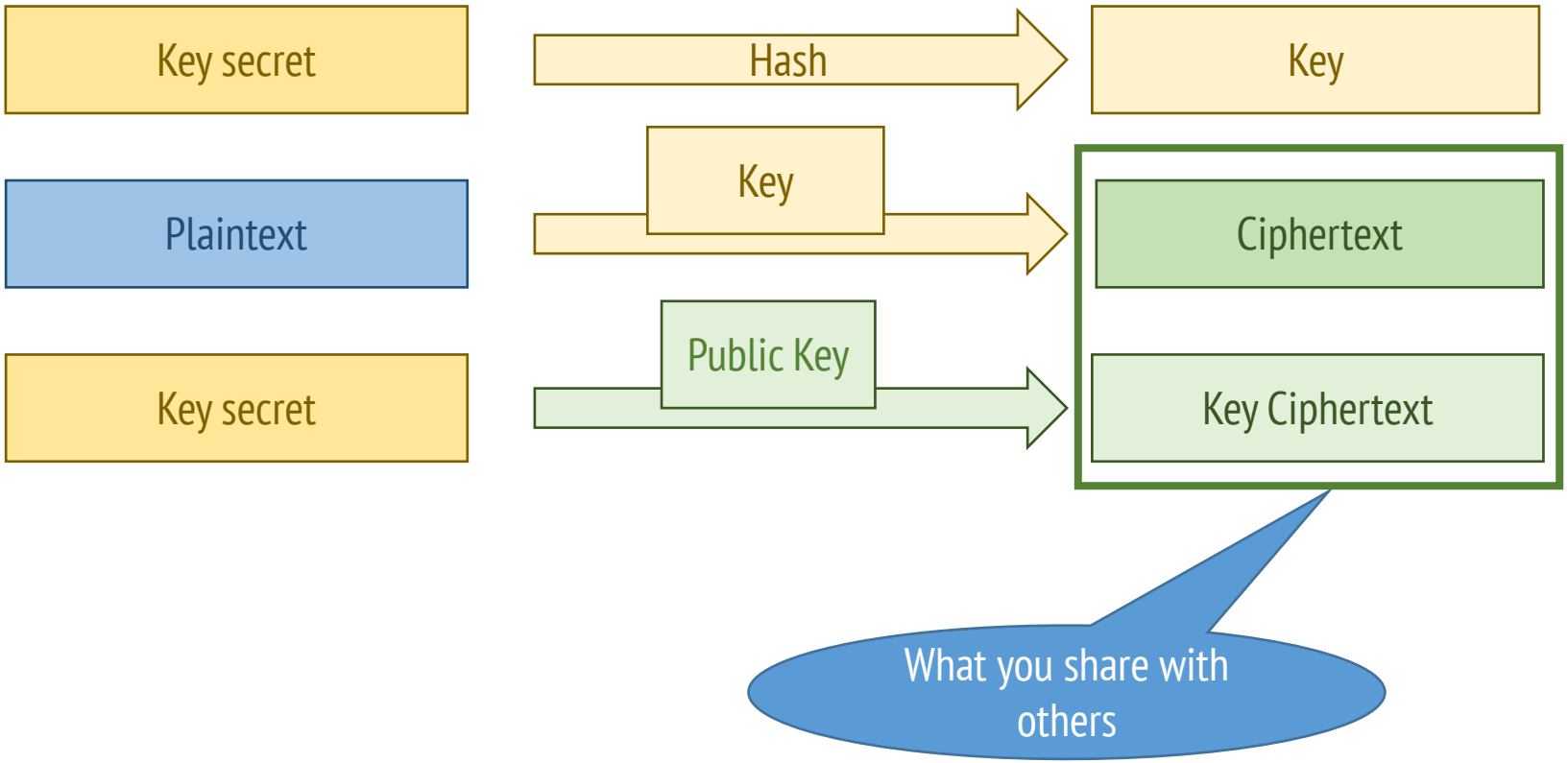
Key encapsulation mechanisms (KEM)

Encryption:



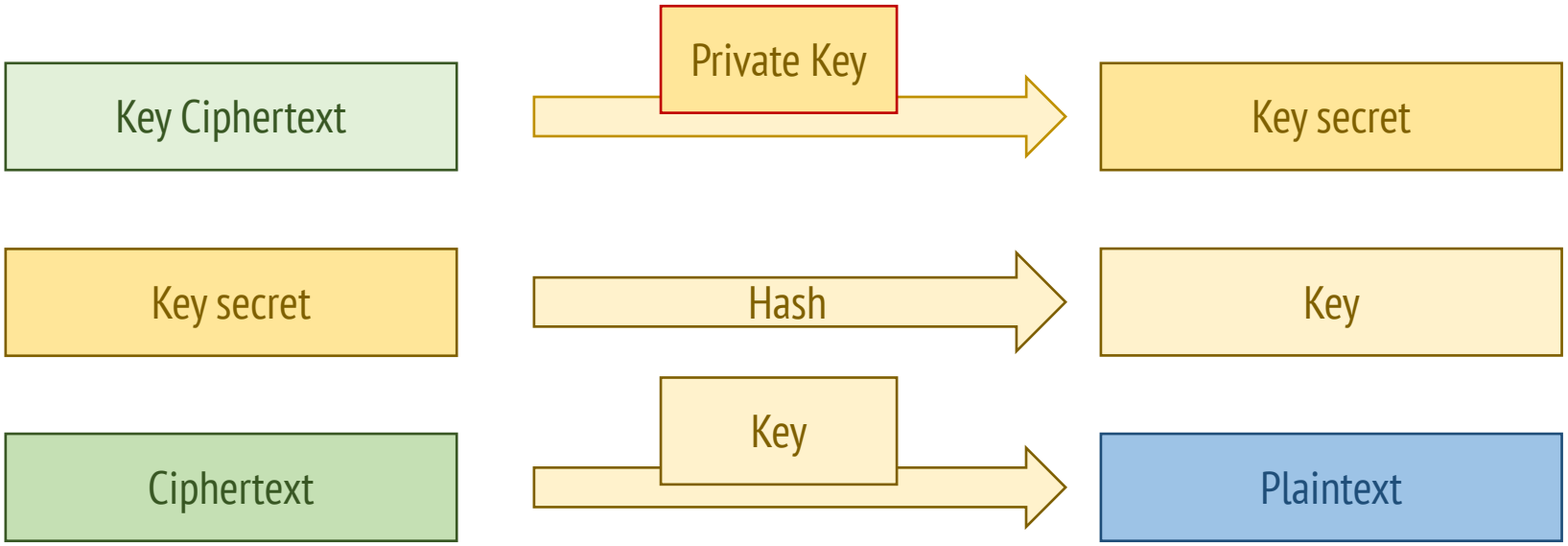
Key encapsulation mechanisms (KEM)

Encryption:



Key encapsulation mechanisms (KEM)

Decryption:



Key encapsulation mechanisms (KEM)

Advantages:

- Symmetric key has good entropy (output of hash function)
- Anybody can encrypt <plaintexts> for the private-key holder
- **Small overhead**

What can we protect?

- **Data at rest** is inactive data that is stored physically in any digital form e.g. files, databases, backups, but also swap

- **Data in use** is data being processed by a CPU or RAM.

What you get/don’t get from encryption?

Encryption does:

- Protect data while resting (i.e. your device is off)
- Protect data from apps who don’t have access to the keys (assuming sandboxing is used)
- Protect data from if un-authorised repairs are done (or device is stolen)

Encryption does not:

- Prevent data loss (it could actually make it easier).
- Make the system more resilient (quite the opposite: you will be more susceptible to DoS attacks).
- Data that has been decrypted in the volatile memory (RAM).

Challenges

Goal:

Complement the “trusted boot” with data confidentiality.

Challenges:

1a. How much information about data should be revealed?

2a. Who should be able access this information data?

…

1b. How to enable confidentiality for the **system-data** required to boot the system?

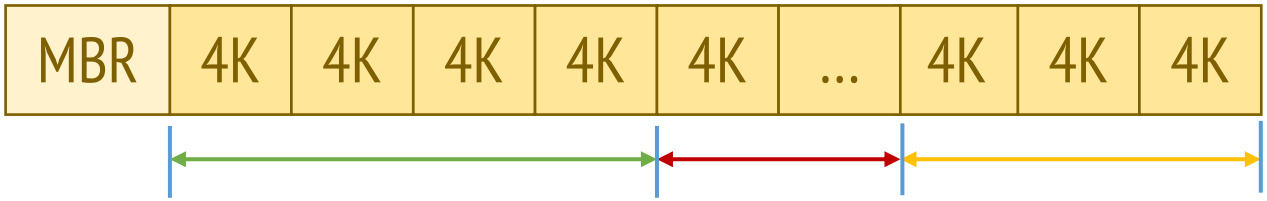
…

Types of data encryption

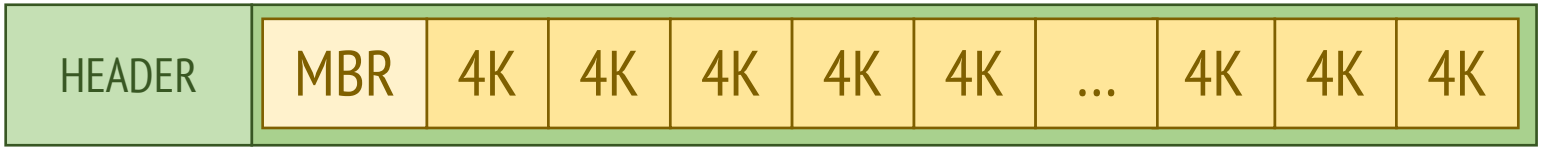
- Disk based
- File based

Disk based encryption

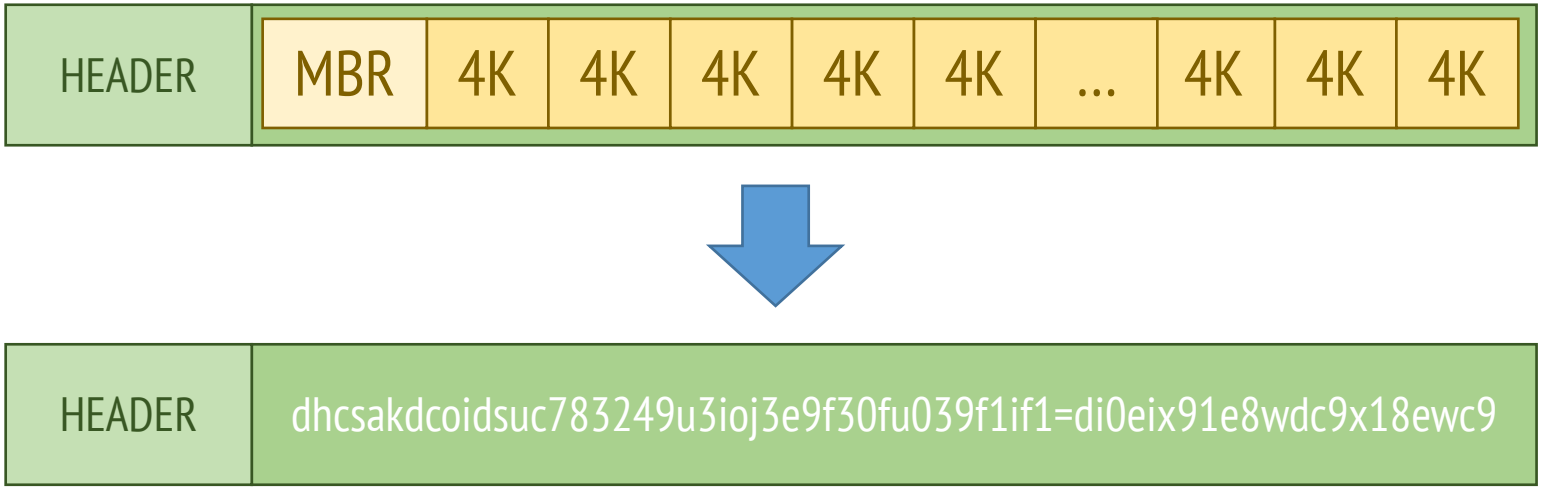
Partition:



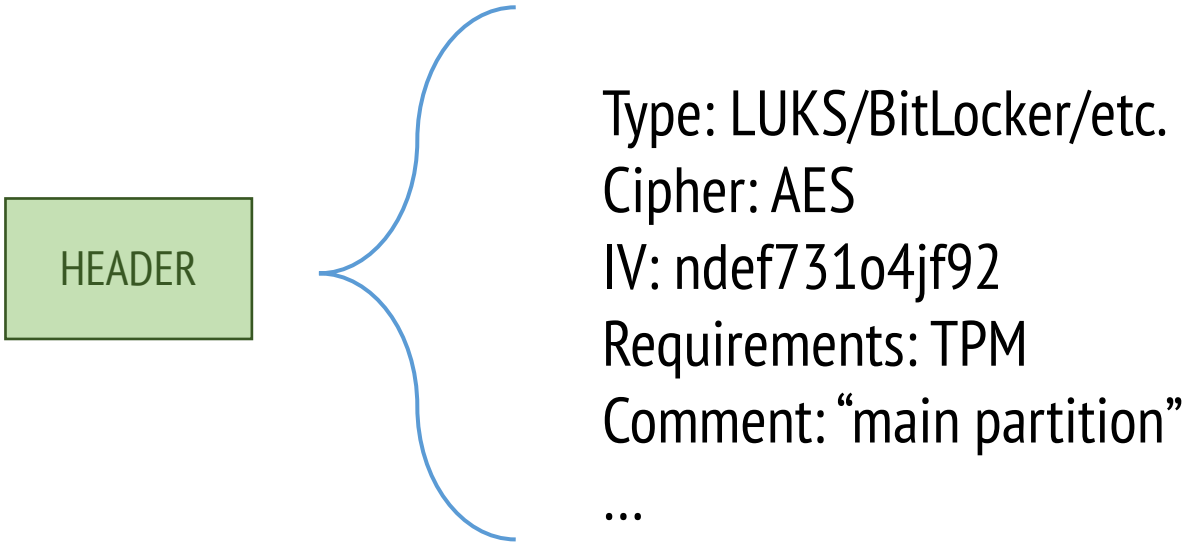
Disk based encryption



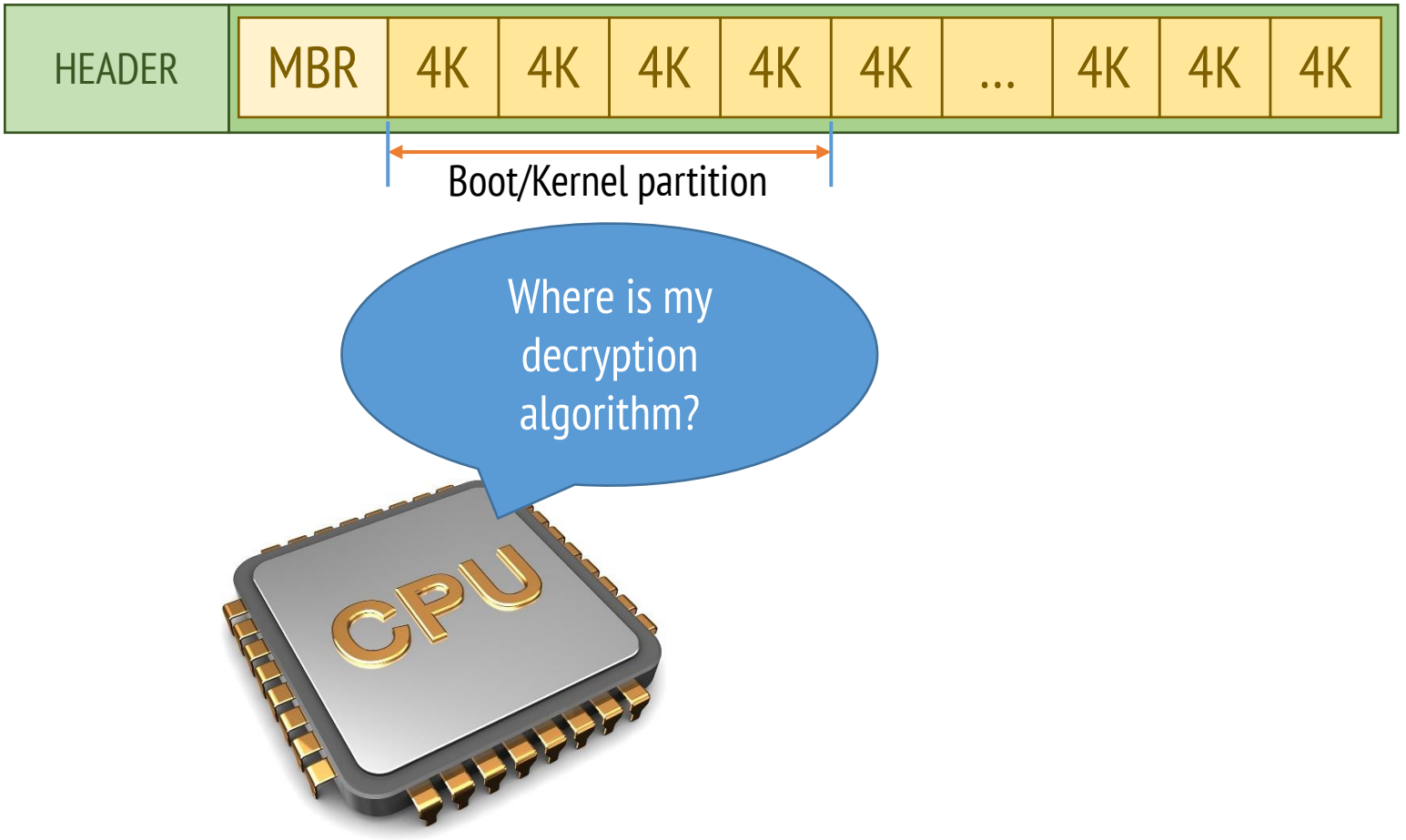
Disk based encryption



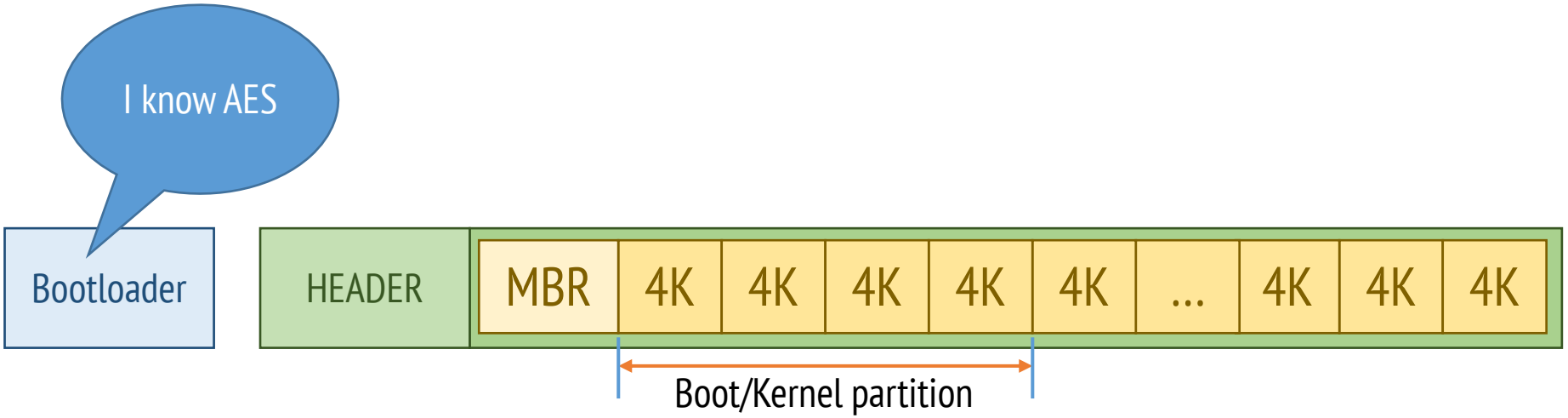
Disk based encryption

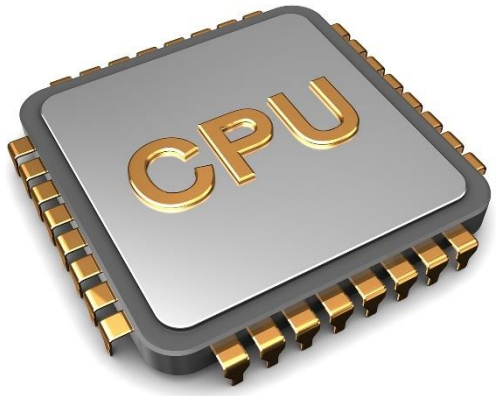


In practice challenges

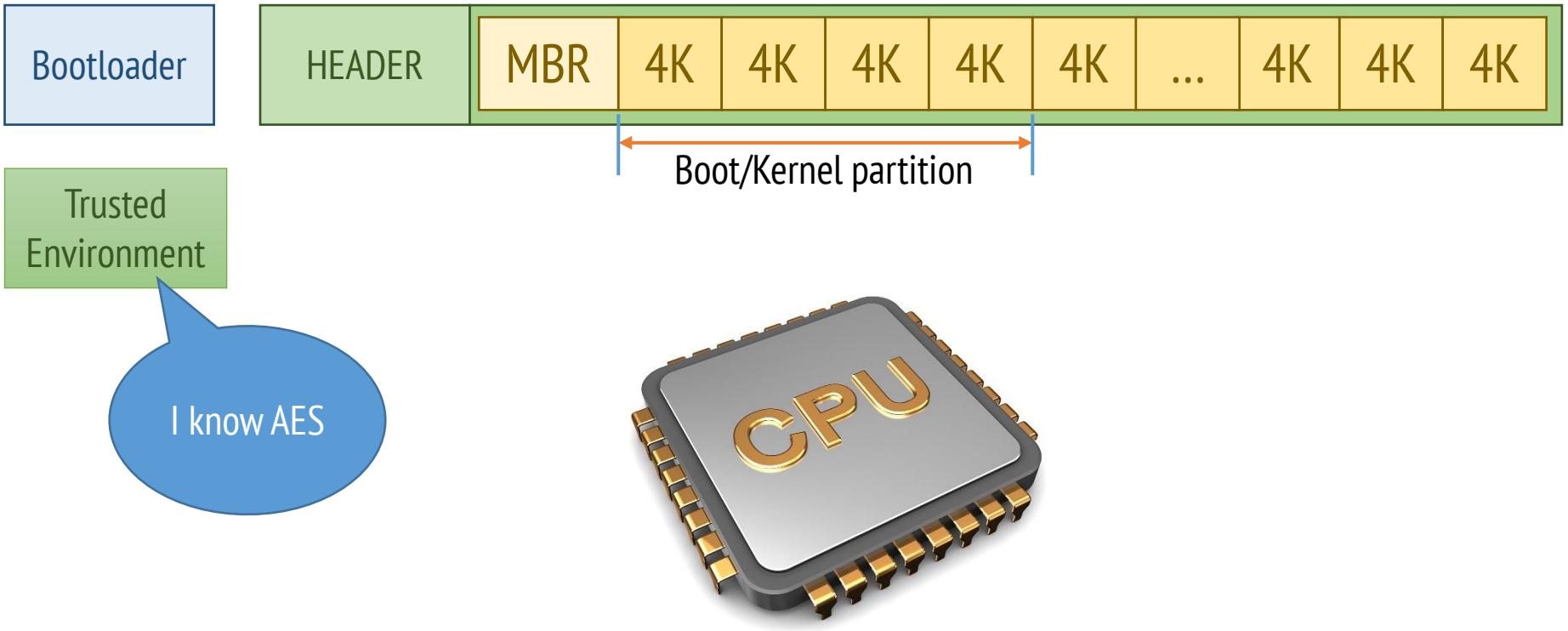


In practice challenges

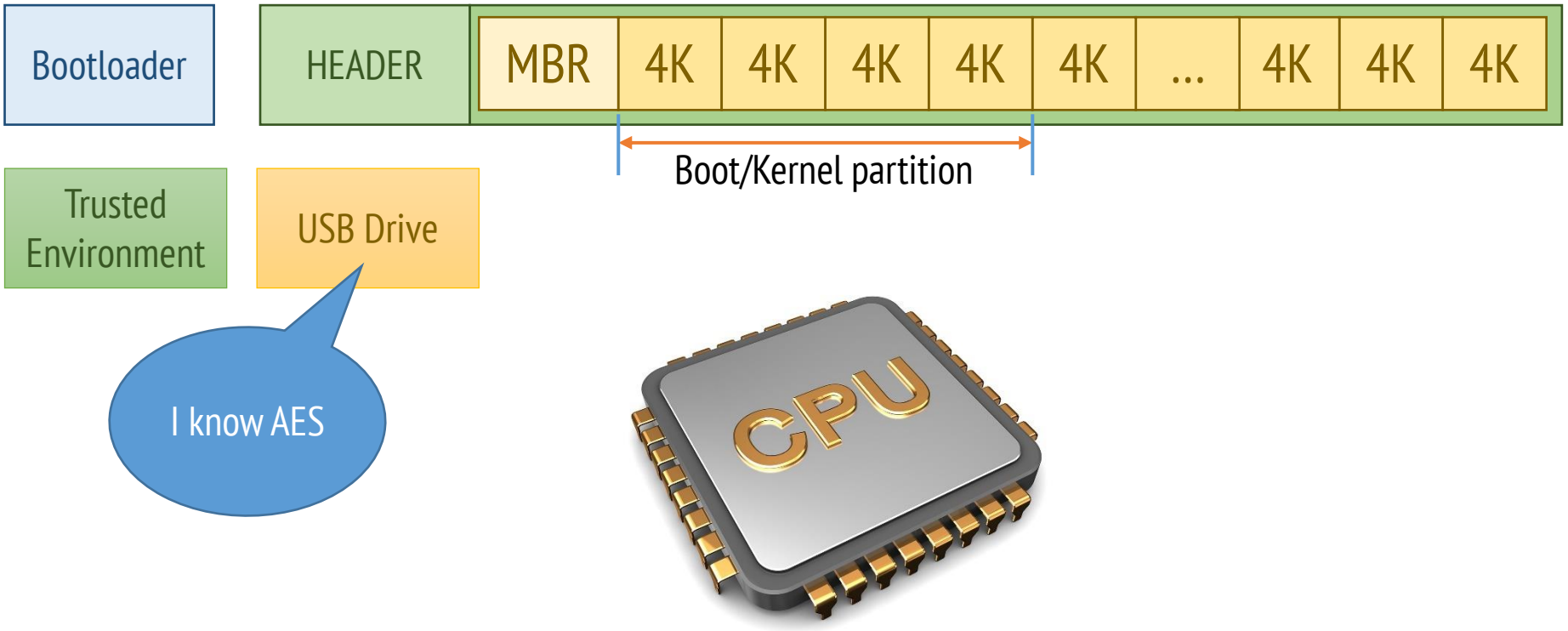




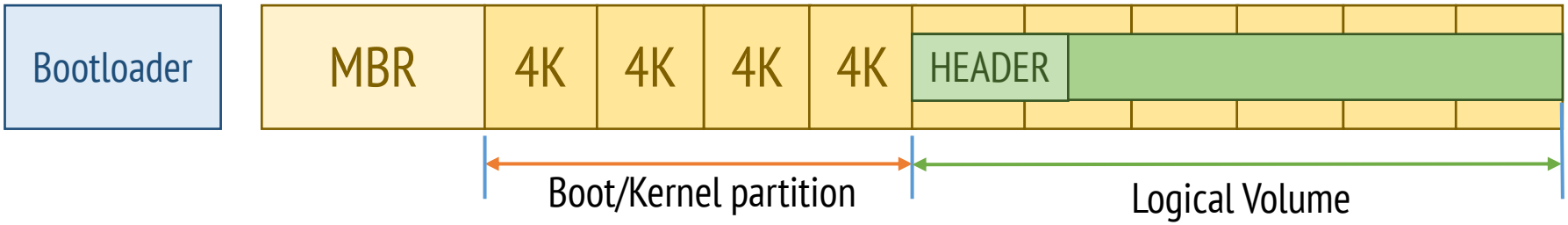
In practice challenges



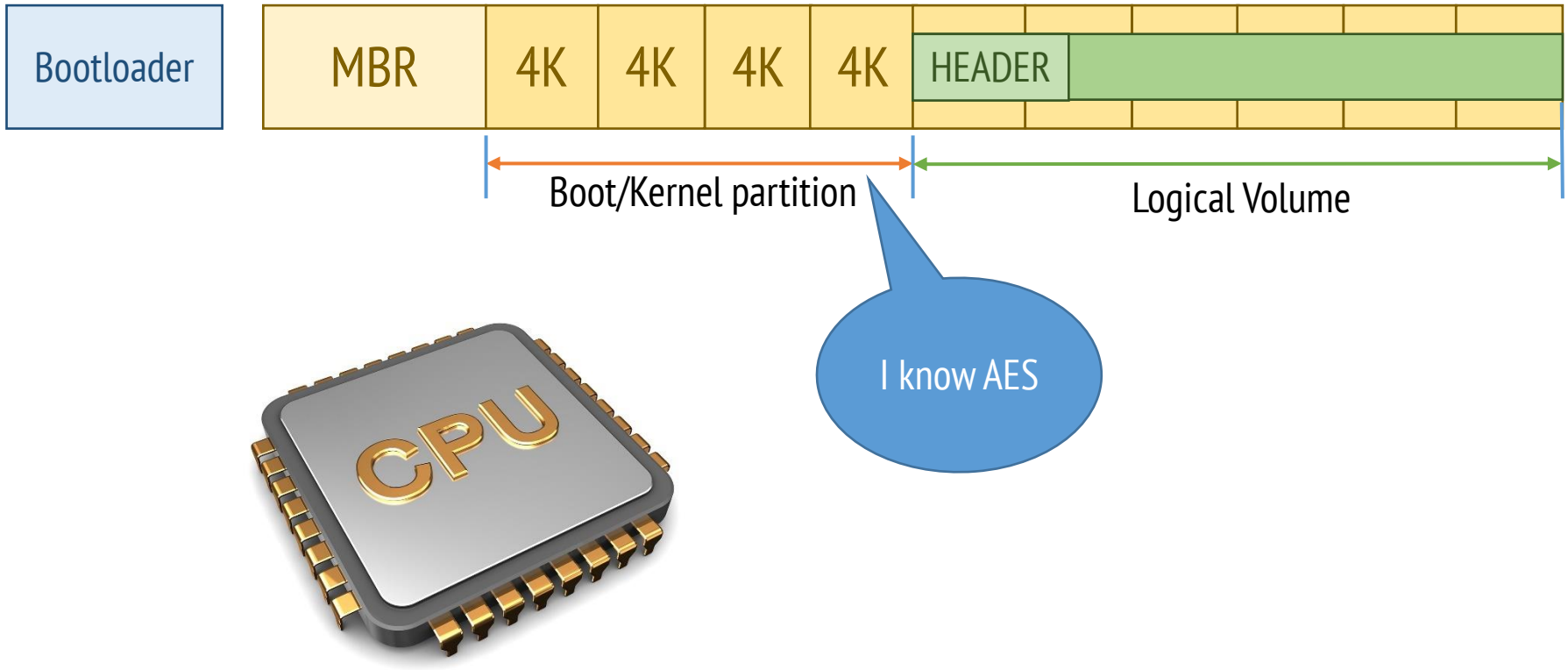
In practice challenges



In practice challenges



In practice challenges



Hold on, are we not missing something important?

Hold on, are we not missing something important?

Right… the **encryption key.**

Storing the key

Humans are **incapable** of securely **storing** high-quality cryptographic keys, and they have **unacceptable speed and accuracy** when performing cryptographic operations.

C. Kaufman, R. Perlman, M. Speciner

Storing the key

On a USB stick:

- Easy
- Requires USB to be accessible to the system
- Vulnerable to stealing

In the TPM (or TEE)

- More difficult to set up
- Transparent
- Protected from stealing

Storing the key

On a SmartCard:

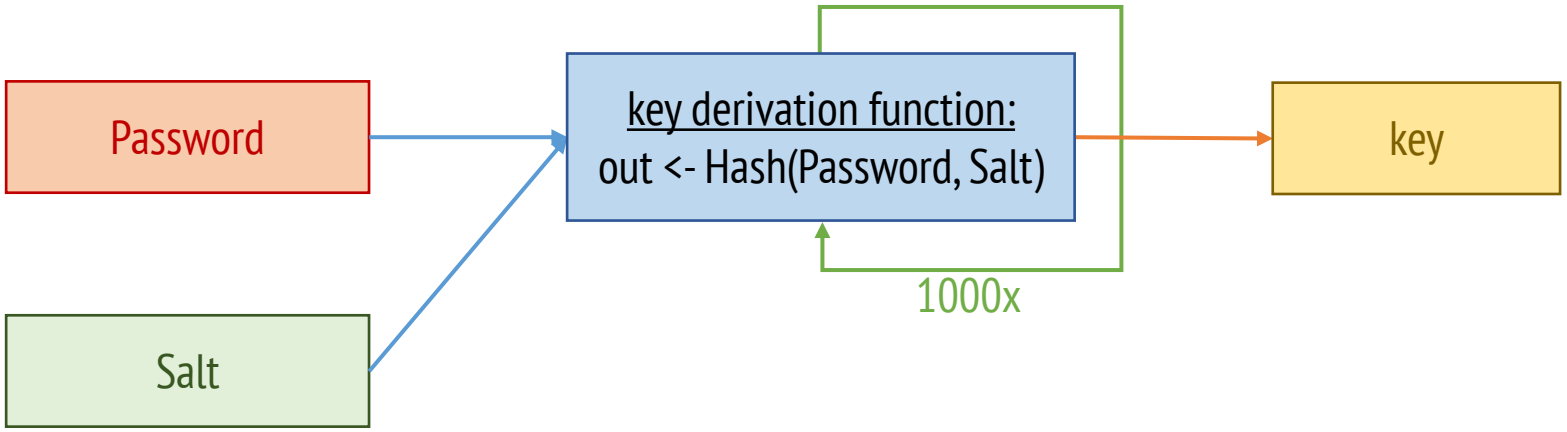
- Difficult to set up (e.g. requires special hardware)
- Requires presence of the card
- Protected from stealing

Deriving the key from a password

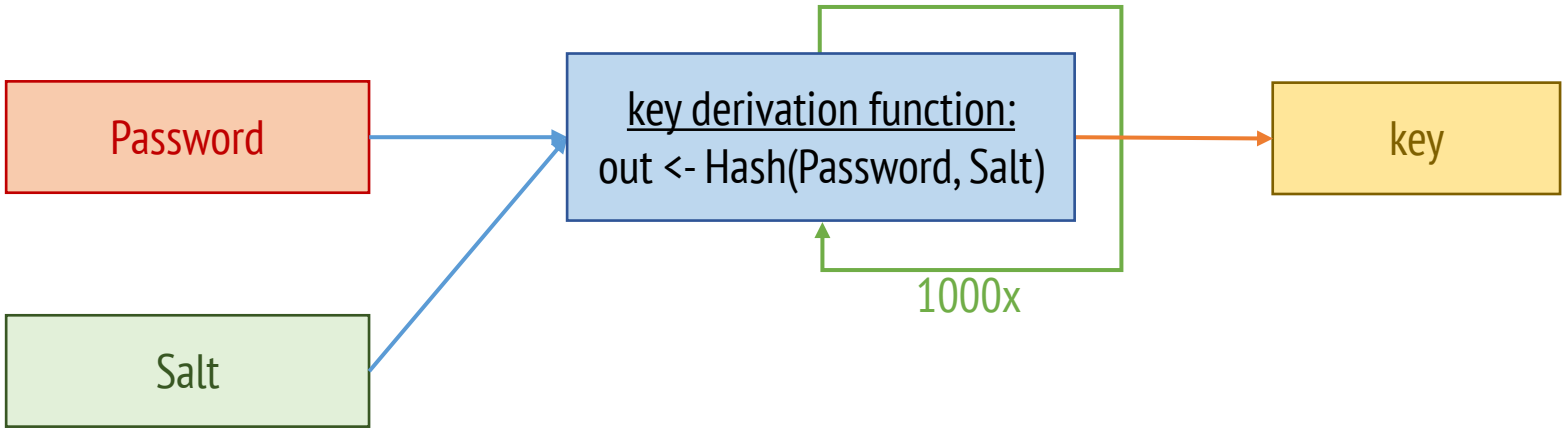
Challenge

- Human generated passwords have low entropy!

Key derivation functions



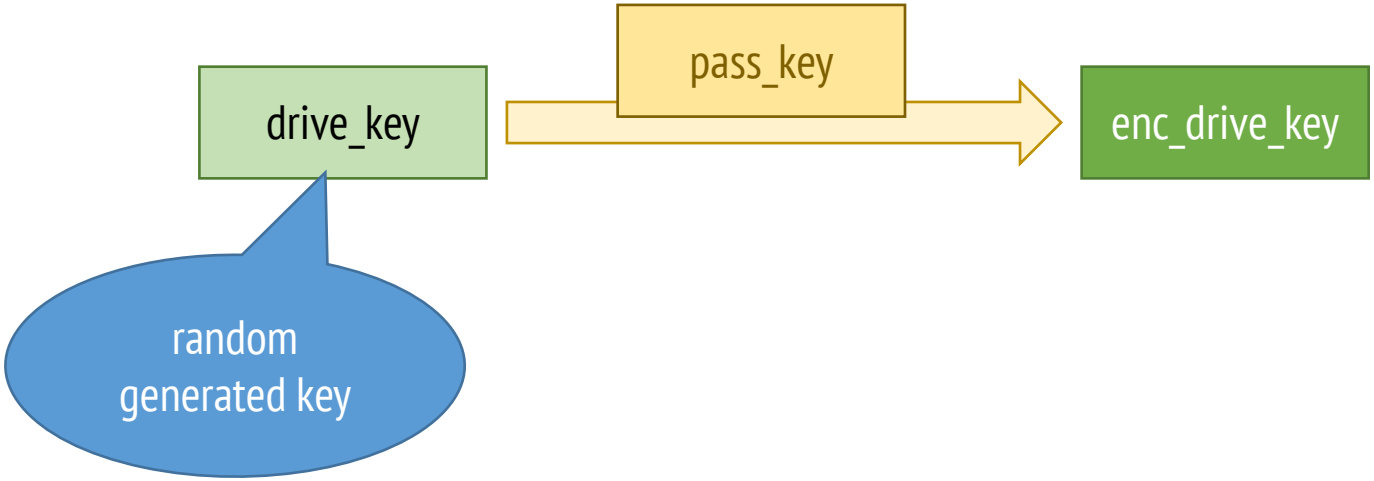
Key derivation functions



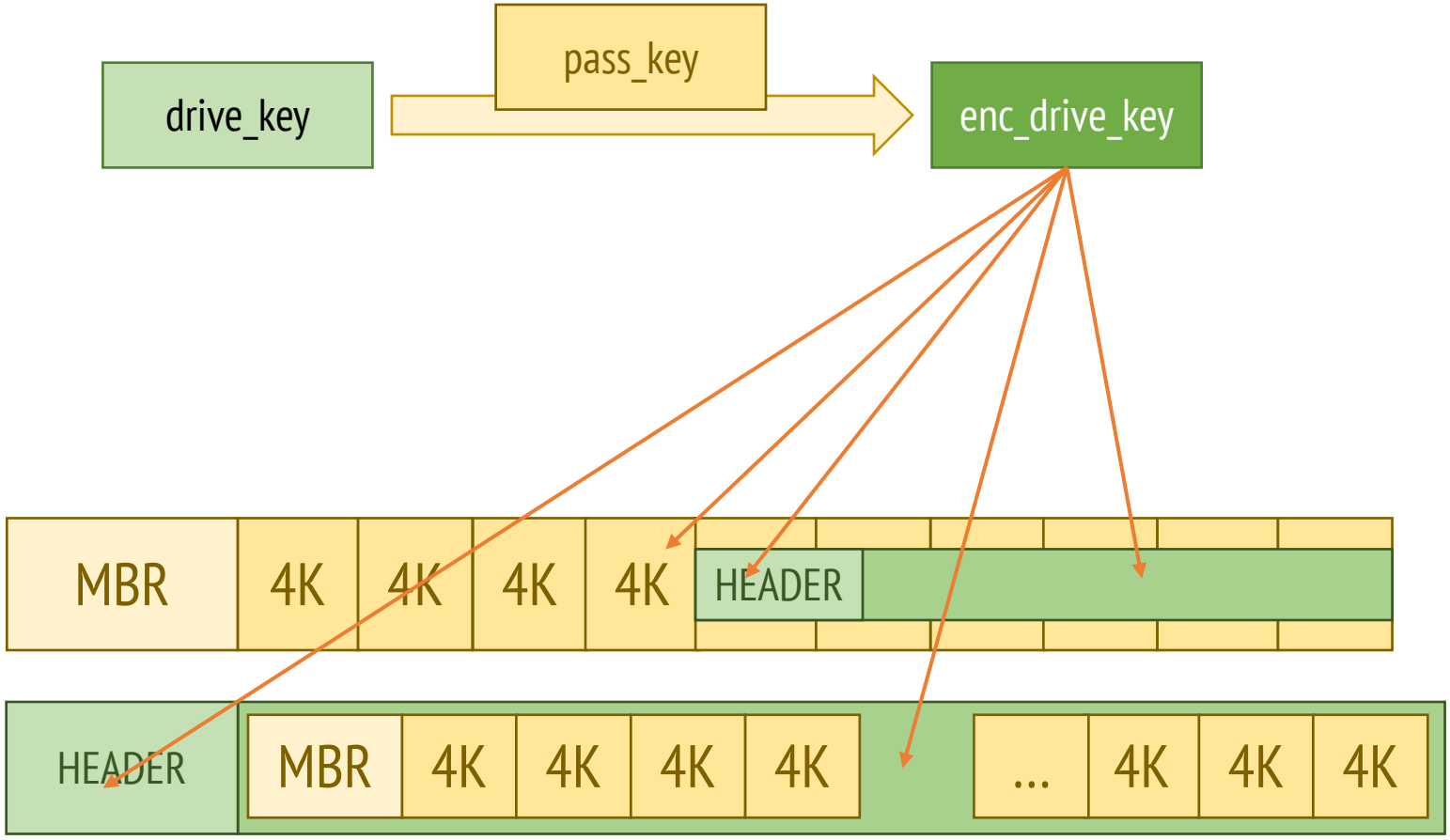
The key has sufficient entropy!

What if I want to change my password?
Do I have to re-encrypt the whole drive?

Key derivation functions



Key derivation functions



Decryption

Derived key decryption process:

- Load enc_drive_key from the drive
- Derive key from password
- Decrypt key
- Decrypt drive

Decryption

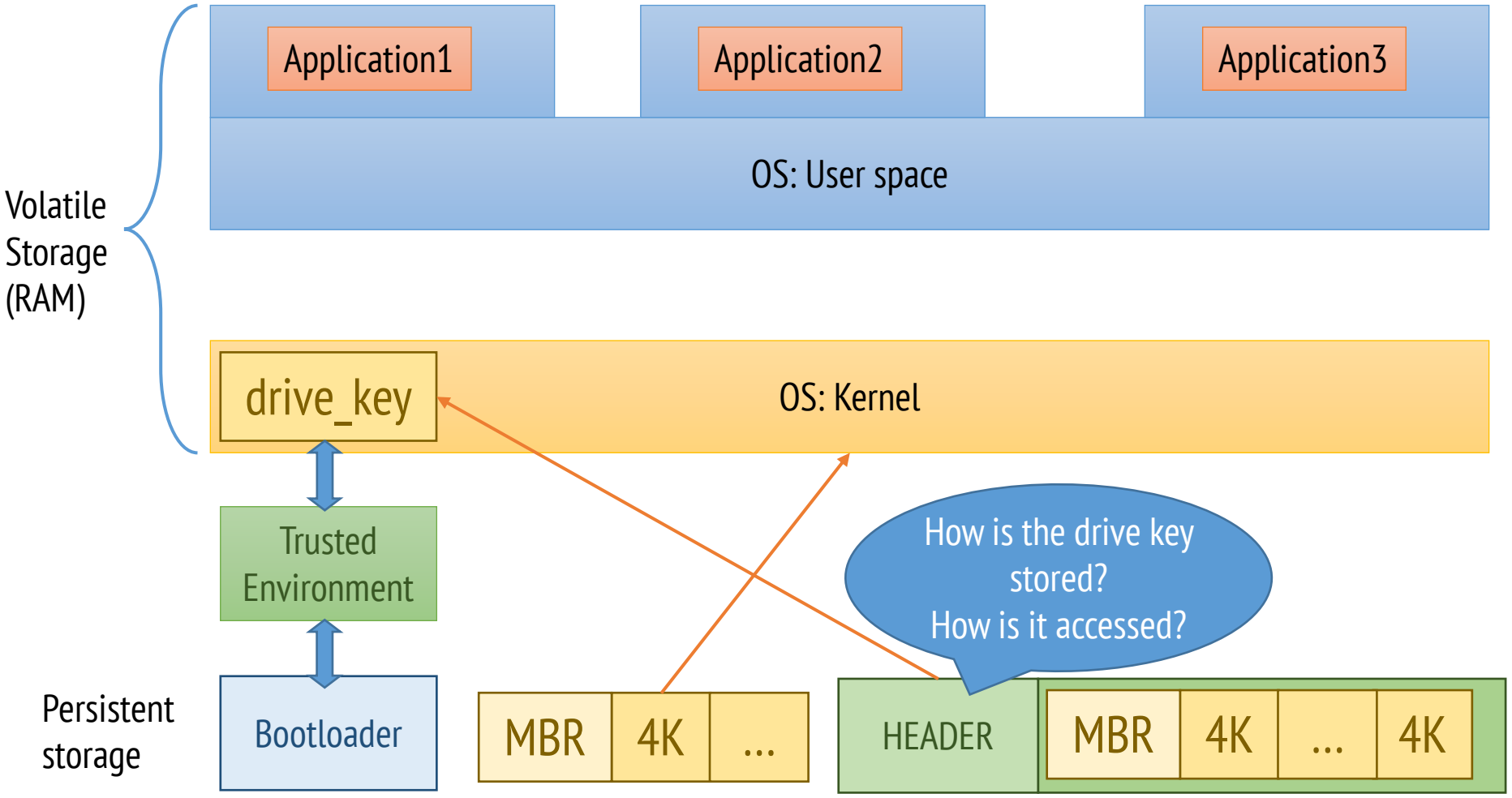
Stored key decryption:

1. Load key from USB
2. Decrypt drive

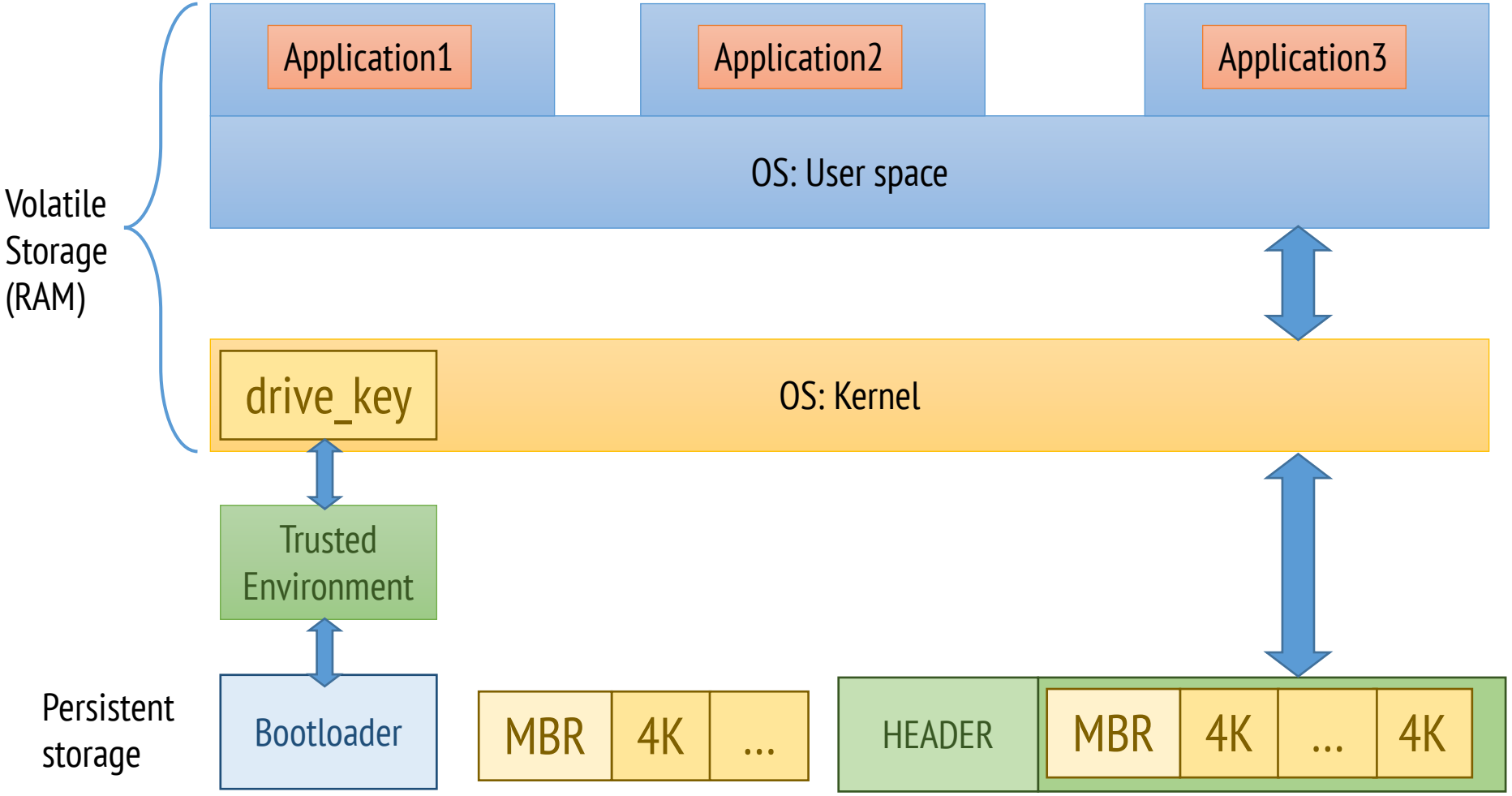
Or:

1. Load key from enc_drive_key
2. Load key from USB/SmartCard/TEE/TPM
3. Decrypt enc_drive_key
4. Decrypt drive

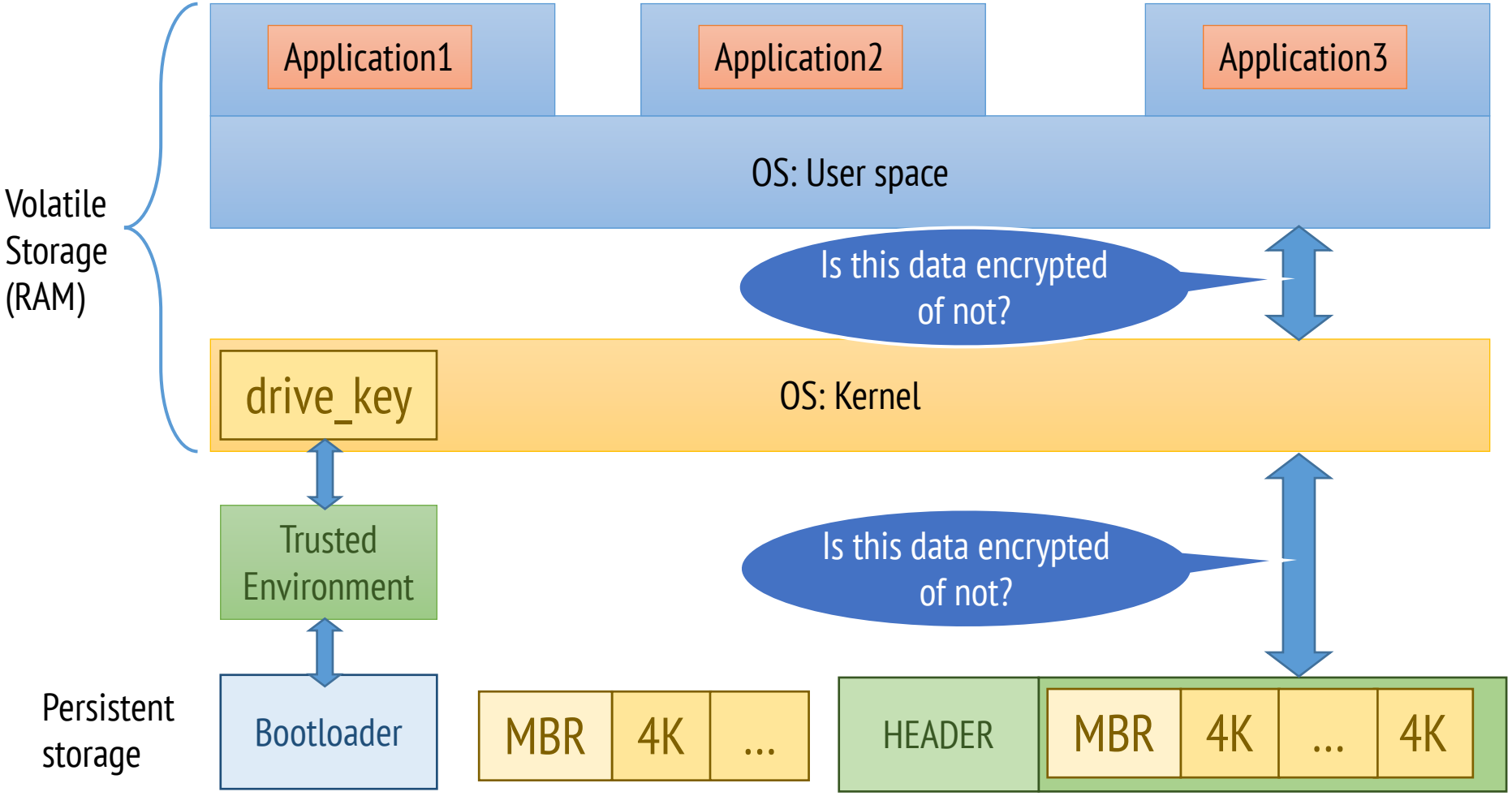
Usage


Application1
Application2
Application3
OS: User space
Volatile Storage (RAM)
drive_key
OS: Kernel
Trusted Environment
How is the drive key stored?
How is it accessed?
Persistent storage
Bootloader
MBR
4K
...
HEADER
MBR
4K
...
4K

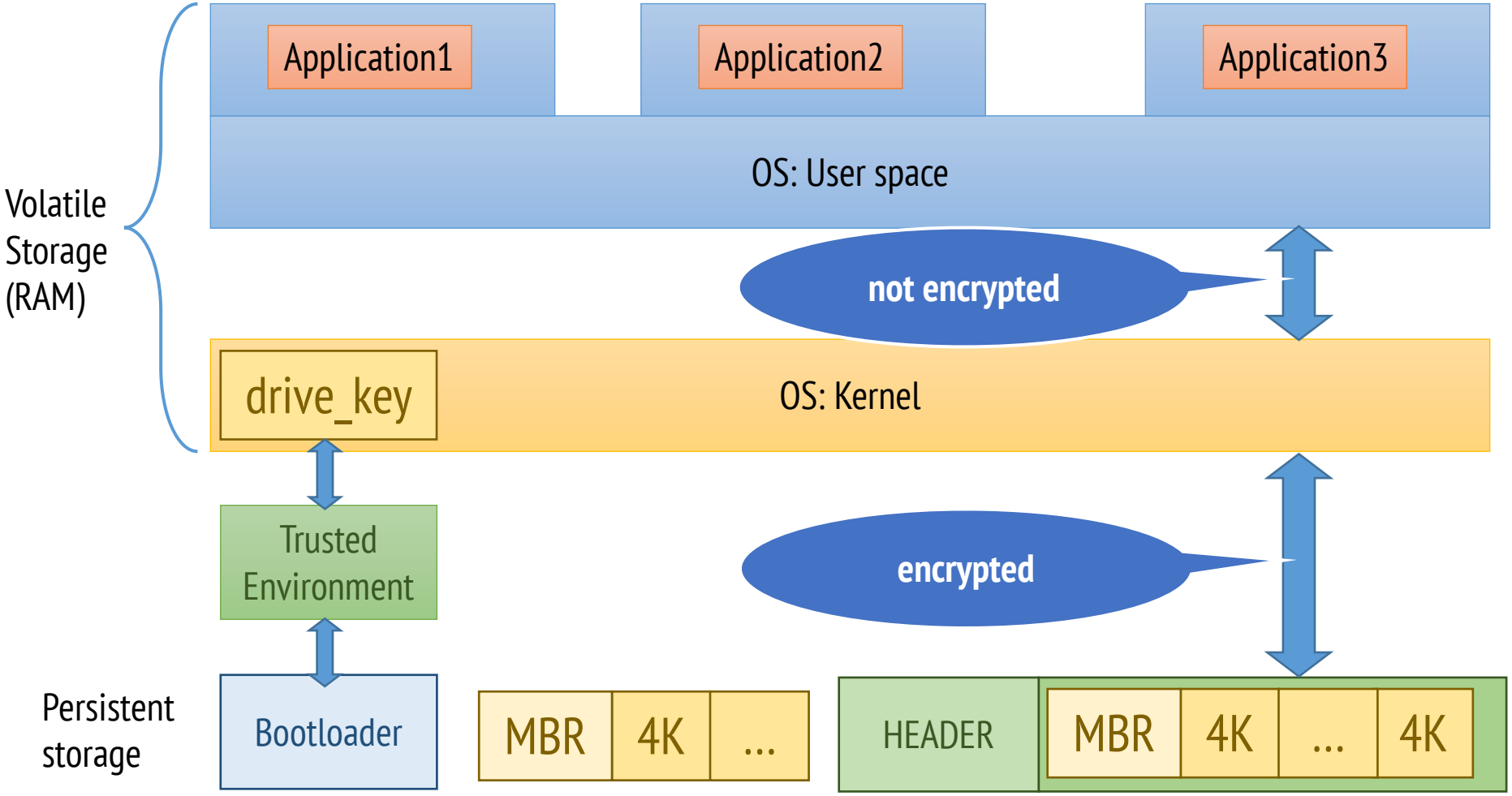
Usage


Application1
Application2
Application3
OS: User space
Volatile Storage (RAM)
drive_key
OS: Kernel
Trusted Environment
Persistent storage
Bootloader
MBR
4K
...
HEADER
MBR
4K
...
4K

Usage


Application1
Application2
Application3
OS: User space
Volatile Storage (RAM)
Is this data encrypted of not?
drive_key
OS: Kernel
Trusted Environment
Is this data encrypted of not?
Persistent storage
Bootloader
MBR
4K
...
HEADER
MBR
4K
...
4K

Usage


Application1
Application2
Application3
OS: User space
Volatile Storage (RAM)
not encrypted
drive_key
OS: Kernel
Trusted Environment
encrypted
Persistent storage
Bootloader
MBR
4K
...
HEADER
MBR
4K
...
4K

Real implementations

BitLocker

- **Transparent operation mode**: uses the capabilities of TPM hardware to provide a transparent user experience by sealing the drive keys on the TPM chip.

- **User authentication mode**: the user has to authenticate before decryption starts.

- **USB/ smartcard key mode**: the user must insert a USB device that contains a the key into the computer.

BitLocker keys

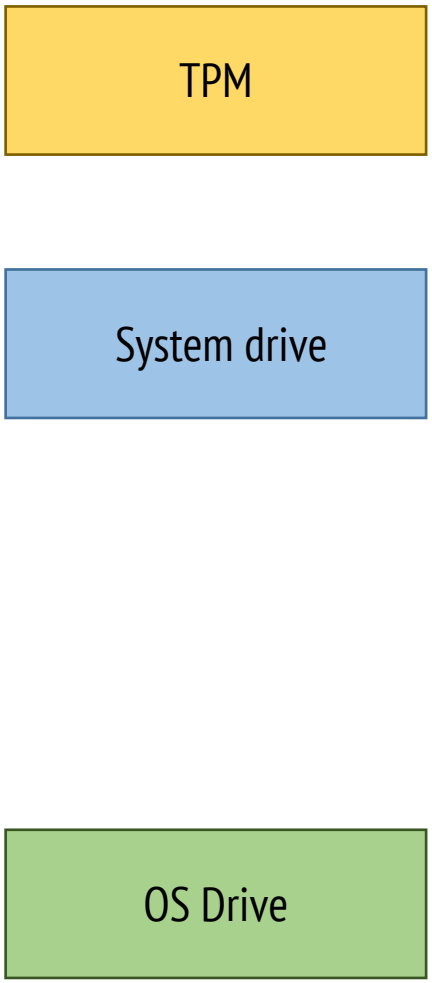
Keys:

- Data Encryption Key (DEK): the drive generates the DEK and it never leaves the device. It is stored in an encrypted format at a random location on the drive. If the DEK is changed or erased, data encrypted using the DEK is irrecoverable.

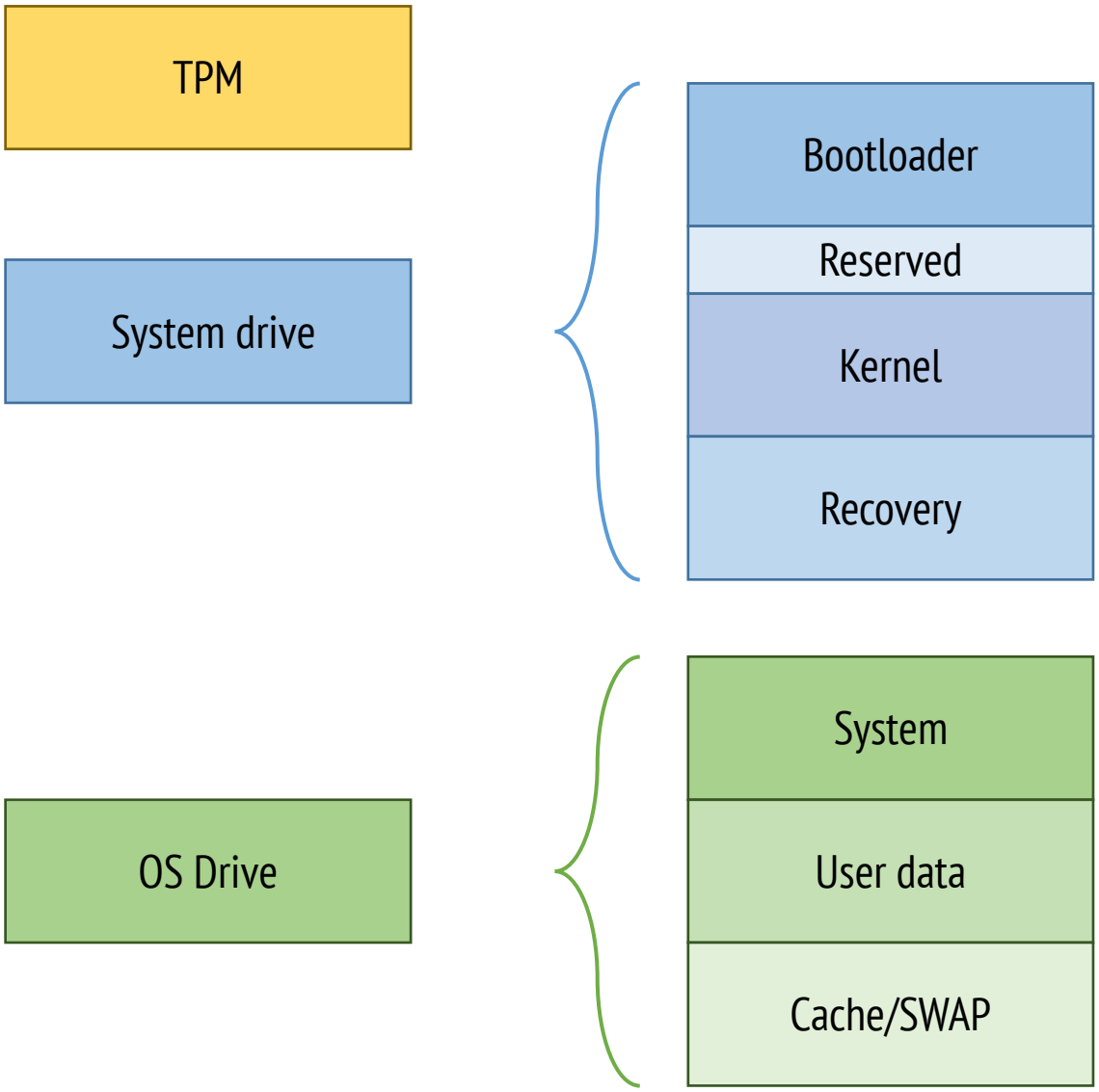
- Authentication Key (AK): the key used to unlock data on the drive. A **hash** of the key is stored on drive and requires confirmation to **decrypt** the DEK.

- Data Encryption Key is encrypted with the Authentication Key

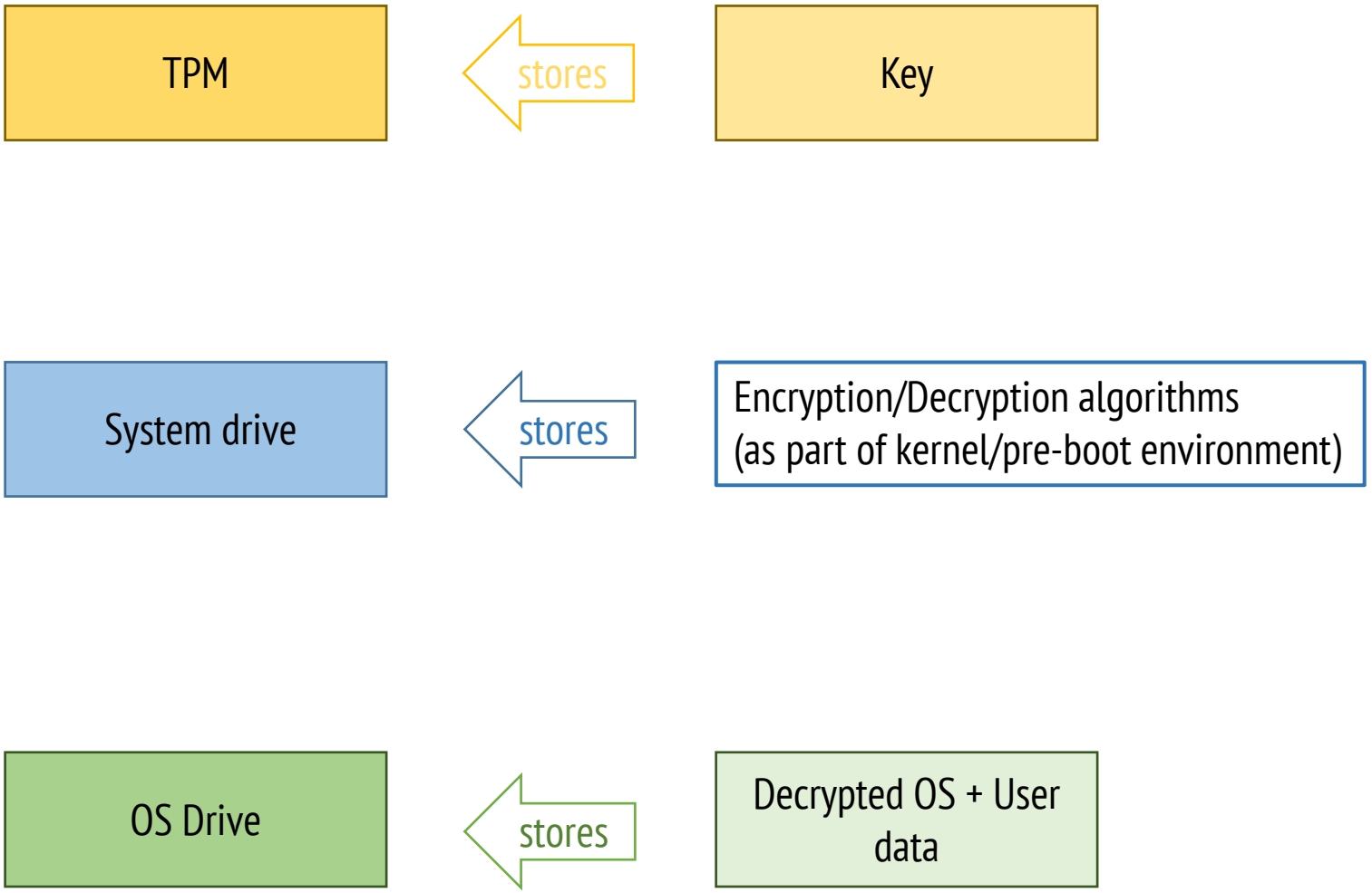
BitLocker



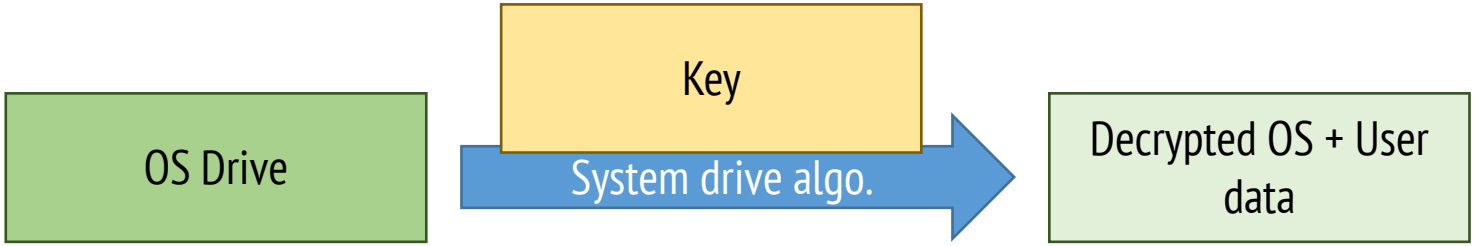
BitLocker


TPM
System drive
OS Drive
Bootloader
Reserved
Kernel
Recovery
System
User data
Cache/SWAP

BitLocker


TPM
stores
Key
System drive
stores
Encryption/Decryption algorithms
(as part of kernel/pre-boot environment)
OS Drive
stores
Decrypted OS + User
data

BitLocker



TPM extra protection

The **key** is sealed inside the TPM's memory
The **key** is only released if early boot files appear to be unmodified

Android 5.0 (with Linux kernel & dm-crypt)

dm-crypt:

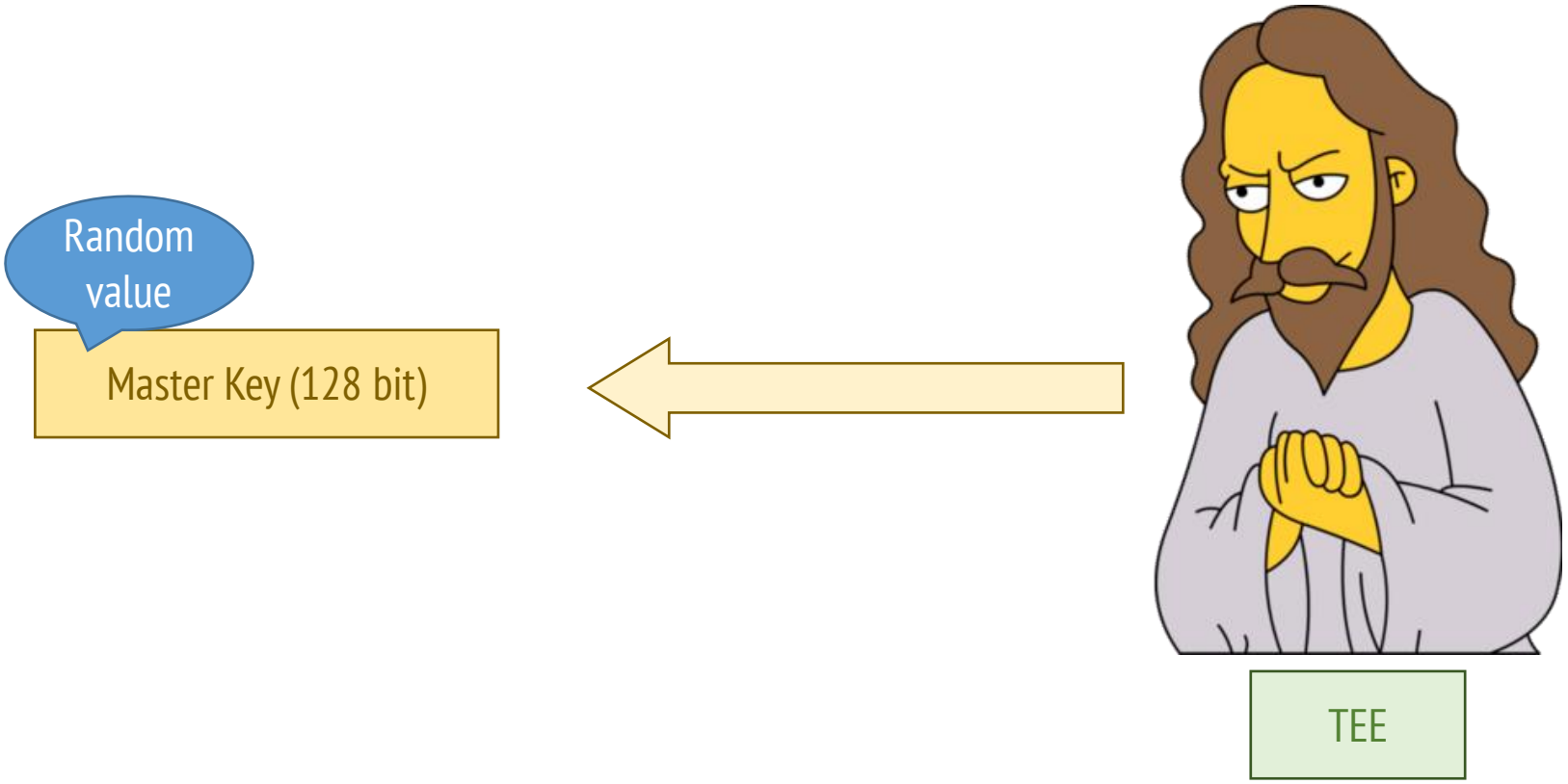
- Kernel module (runs in kernel space)
- Provides transparent disk encryption
- Supports the kernel only keys (i.e. logon keys)
- Uses cryptographic routines from the kernel's Crypto API

Android 5.0 (with Linux kernel & dm-crypt)

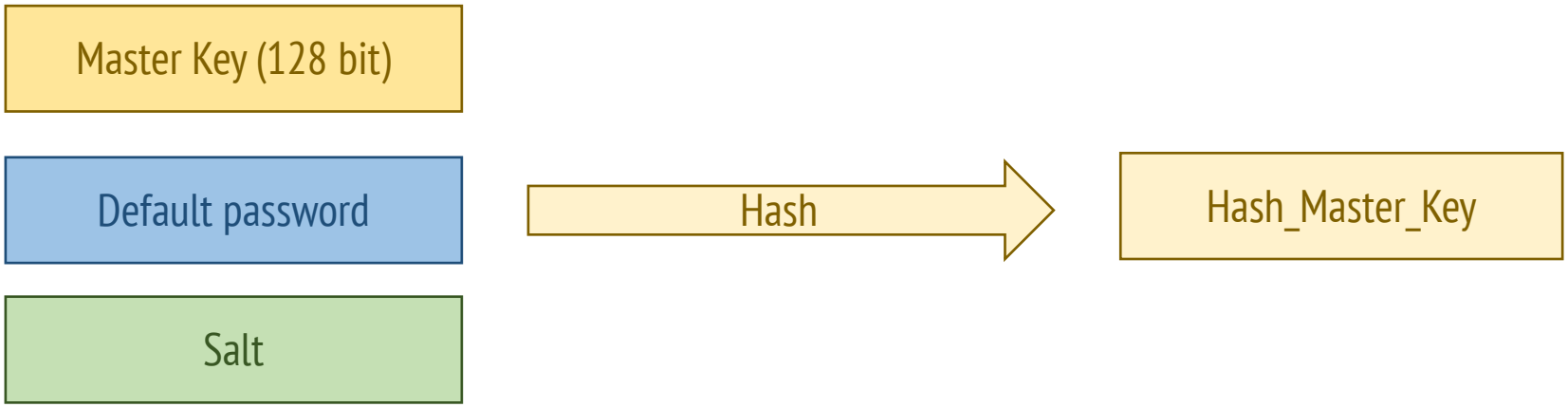
Android user authentication methods:

- default
- PIN
- password
- pattern

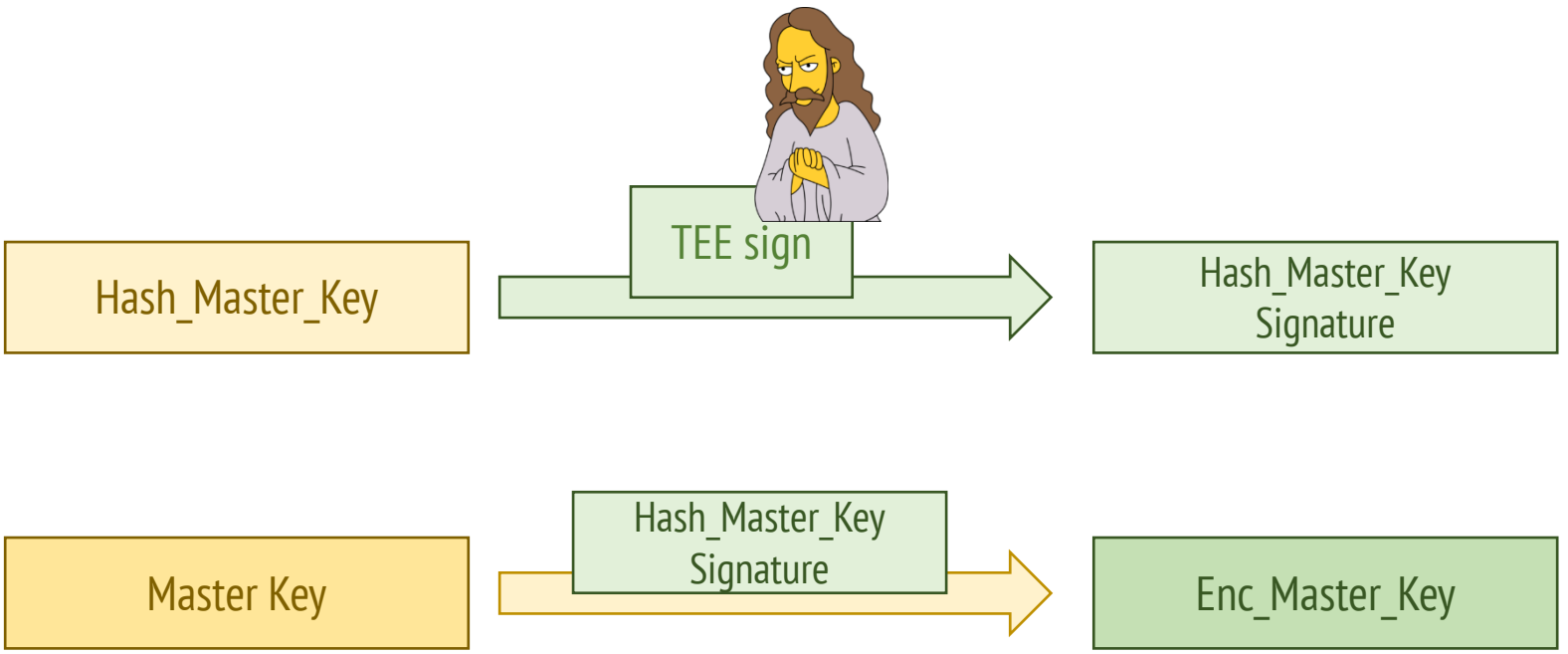
Android 5.0 (with Linux kernel & dm-crypt)



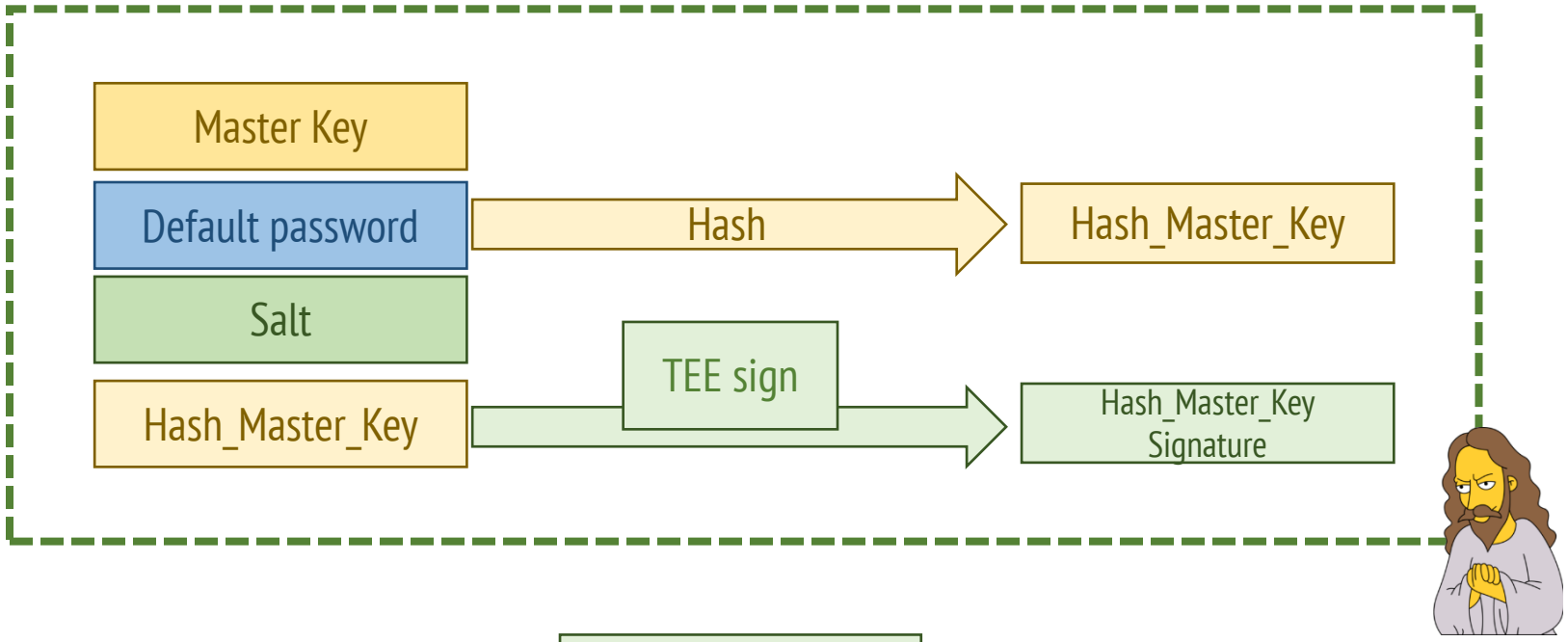
Android 5.0 (with Linux kernel & dm-crypt)



Android 5.0 (with Linux kernel & dm-crypt)



Android 5.0 – Decryption (dm-crypt)

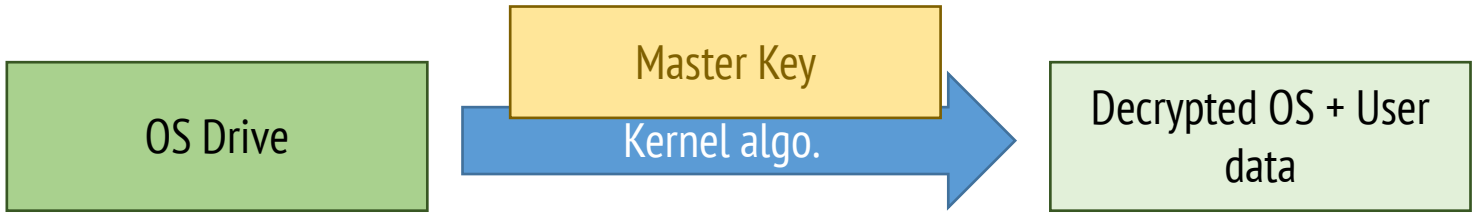


Android 5.0 (with Linux kernel & dm-crypt)

Which key am I using to decrypt my data?

Android 5.0 (with Linux kernel & dm-crypt)

Which key am I using to decrypt my data?

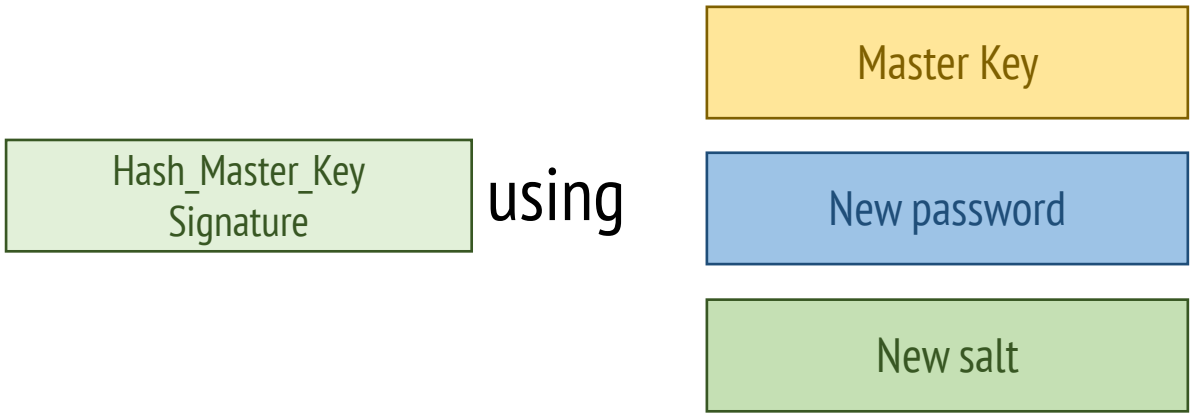


Android 5.0 (with Linux kernel & dm-crypt)

What happens to the Master Encryption key when I change my password, i.e. Default password?

Android 5.0 (with Linux kernel & dm-crypt)

What happens to the Master Encryption key when I change my password, i.e. Default password?



Android 5.0 (with Linux kernel & dm-crypt)

Design choices:

- **Static encryption key**
- Salt
- Credential support e.g. password
- **Access to key for credential update**
- **TEE bind/anchor**

Advantages/disadvantages of full disk encryption

Full disk encryption

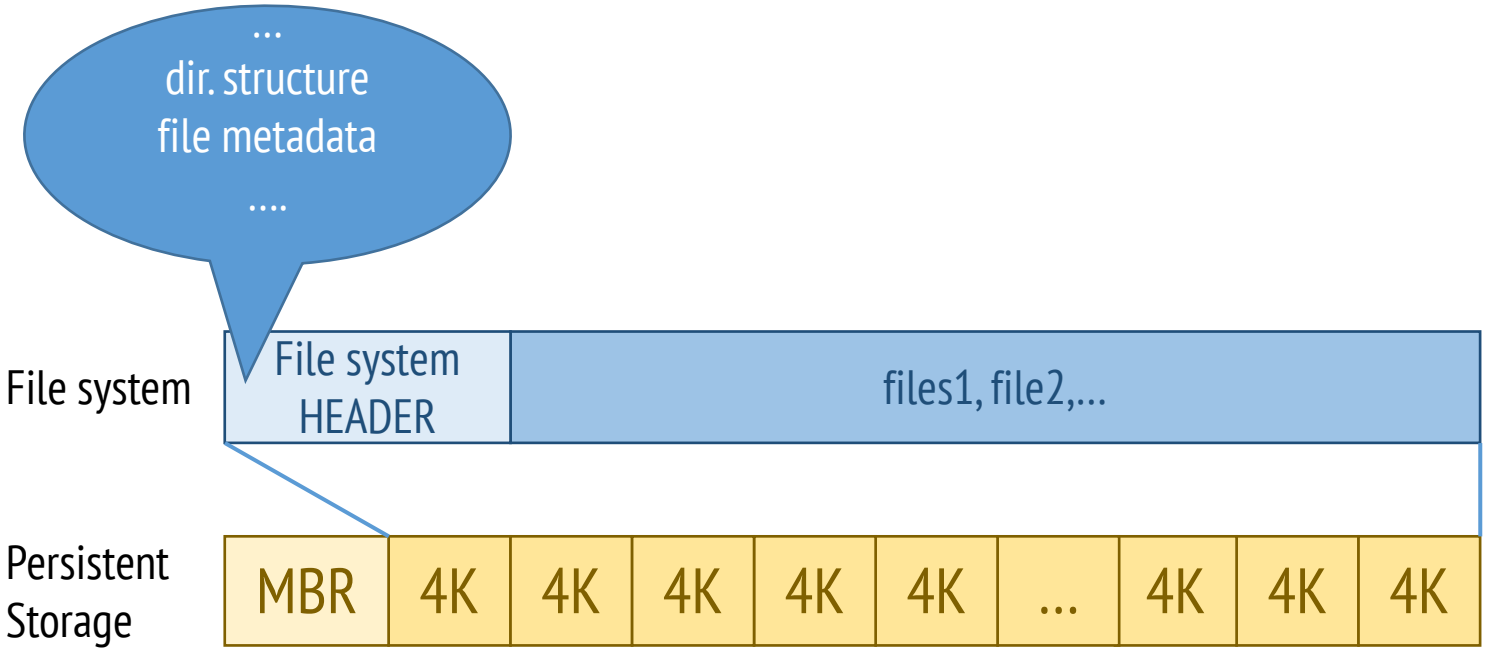
- Simple design: generally only one key is used.
- Protects filesystem meta data e.g. directory structure, file names, modification timestamps.
- If the key is compromised, the attacker has access to all files.

File based encryption

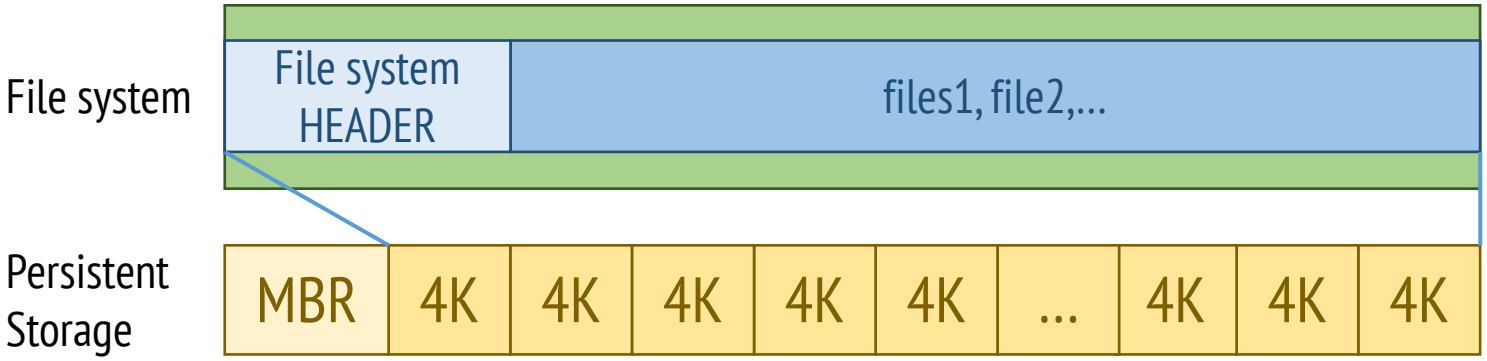
Components

- File contents
- File metadata
- Memory storage
- Disk storage
- Access control (type, user)

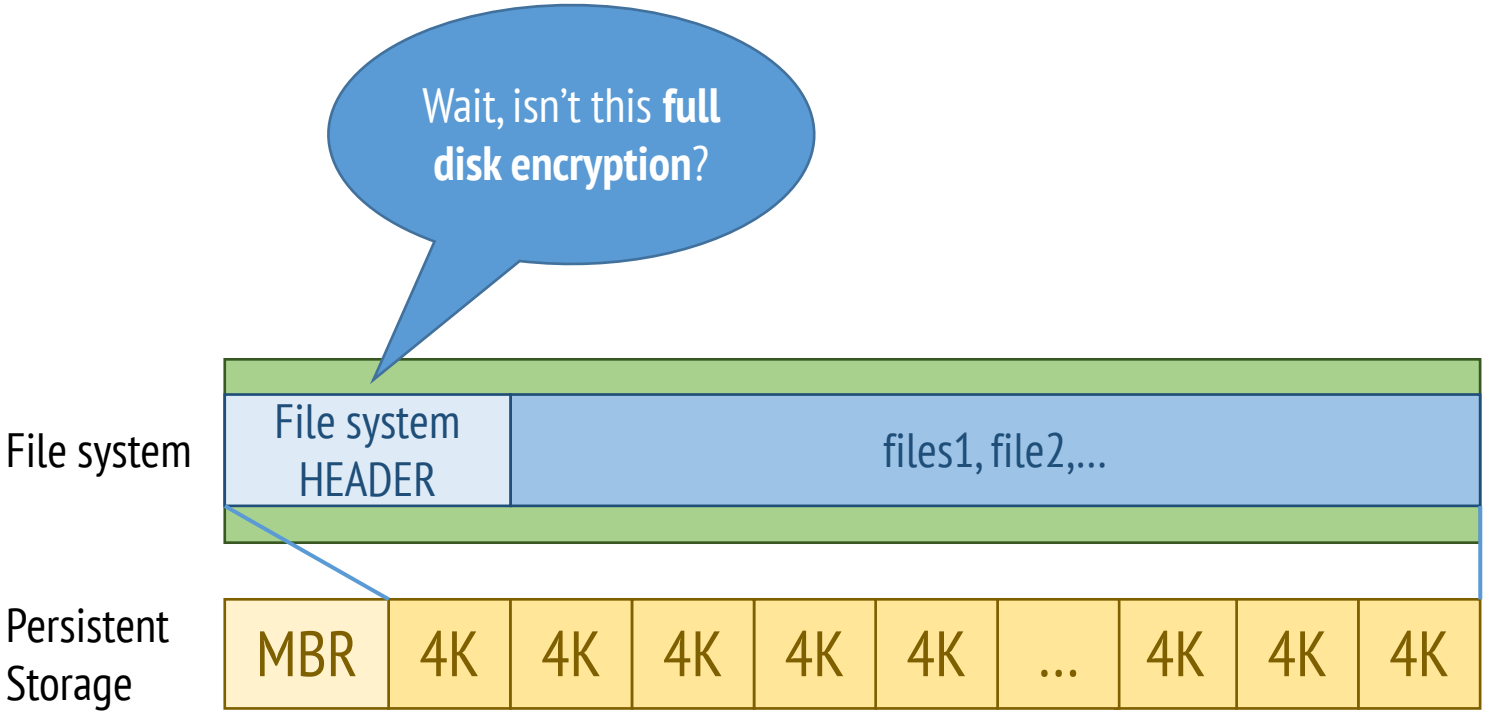
File based encryption



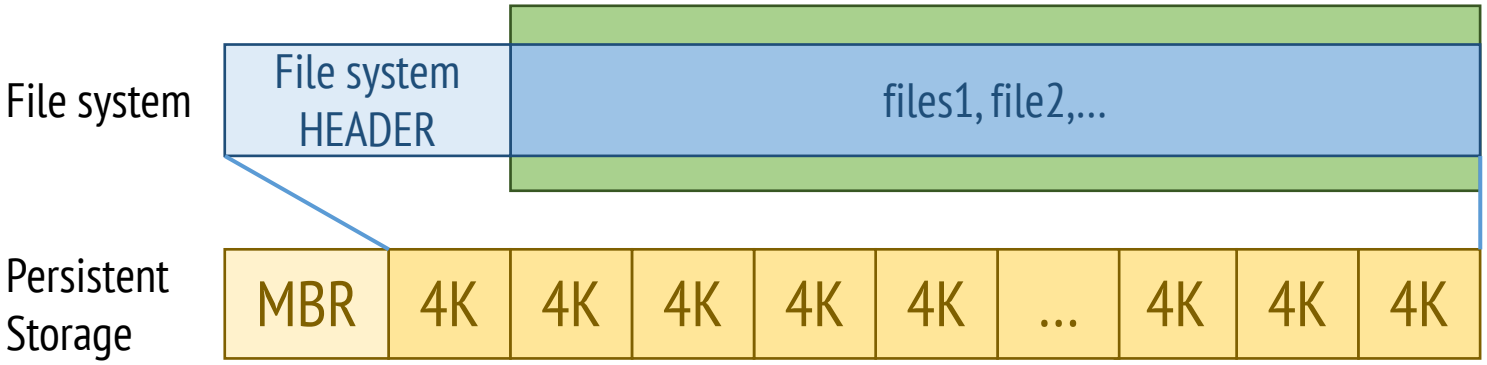
File based encryption



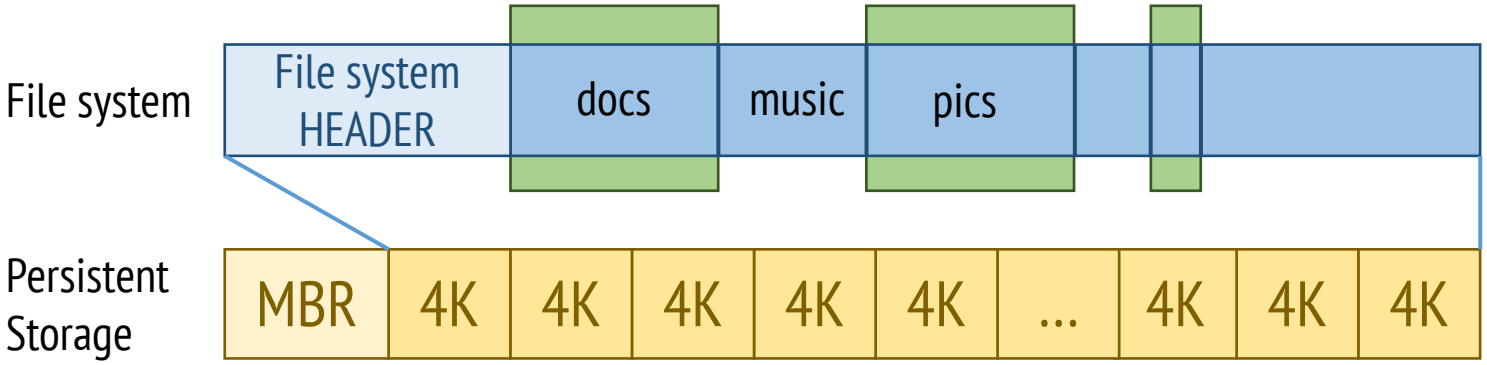
File based encryption



File based encryption



File based encryption

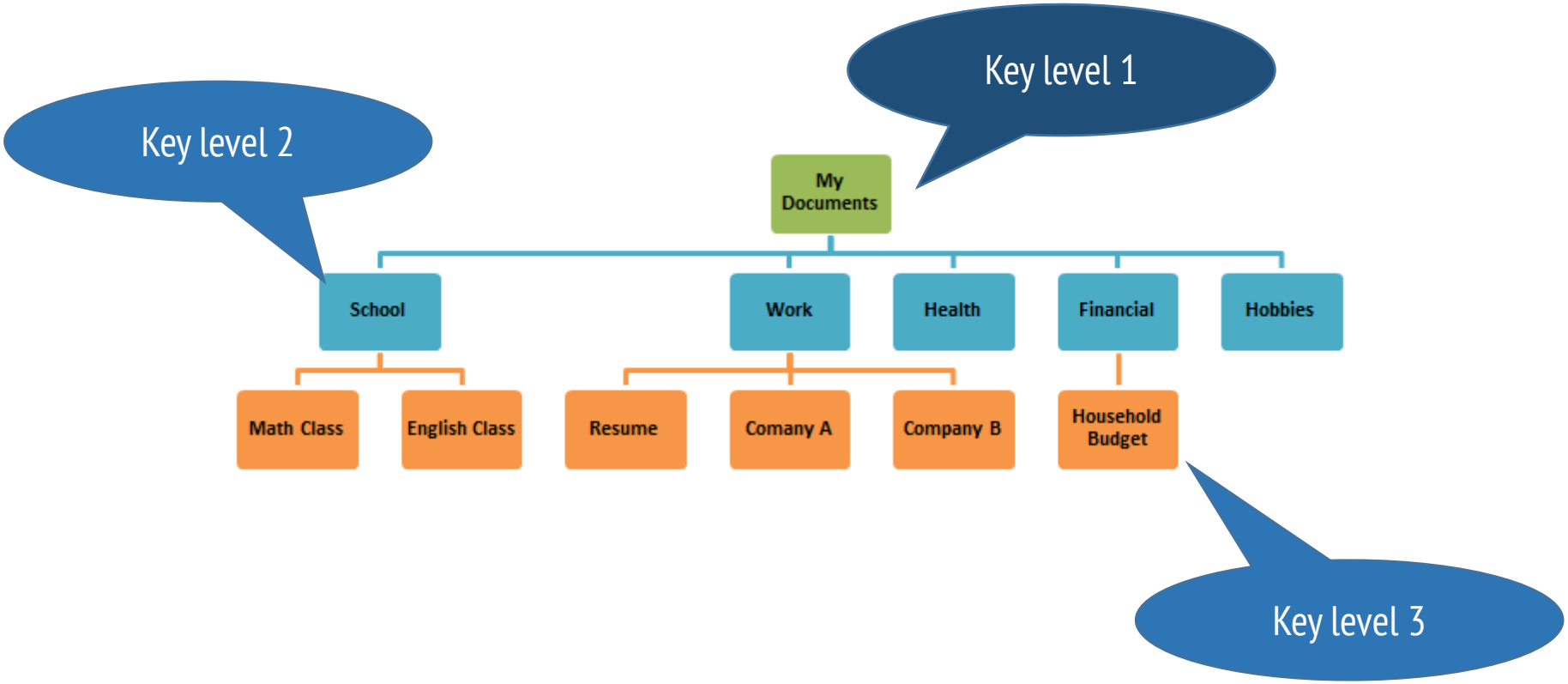


This is pretty cool. How do we do it?

Try to map a key structure to the file system structure.

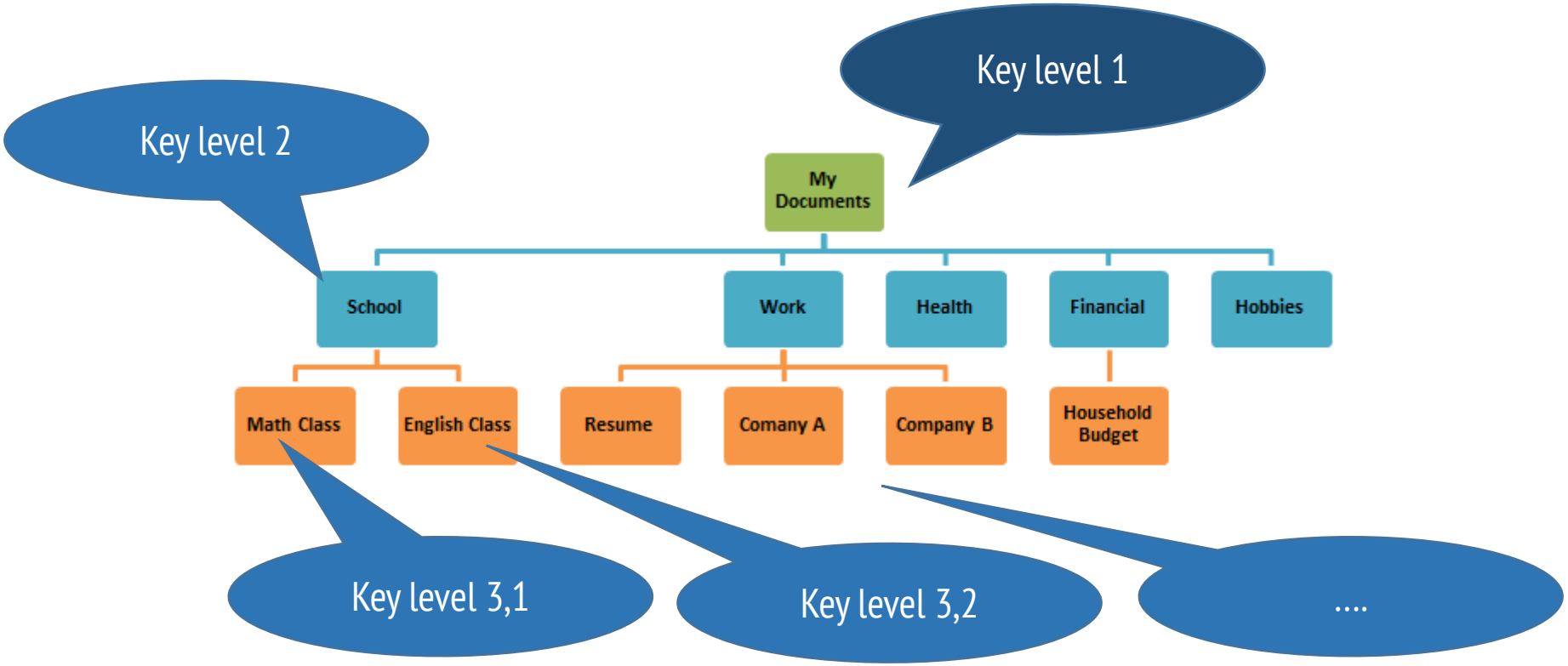
This is pretty cool. How do we do it?

Try to map a key structure to the file system structure.



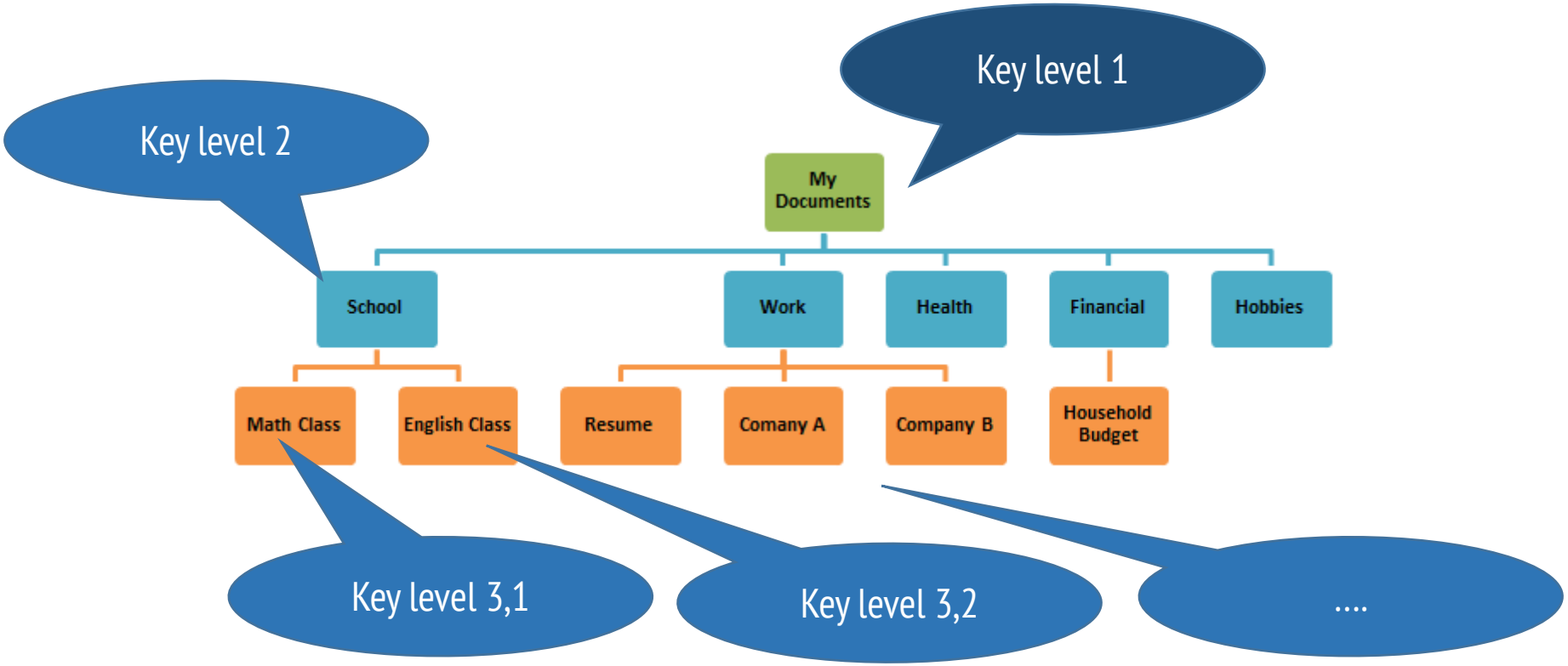
This is pretty cool. How do we do it?

Try to map a key structure to the file system structure.



This is pretty cool. How do we do it?

Try to map a key structure to the file system structure.

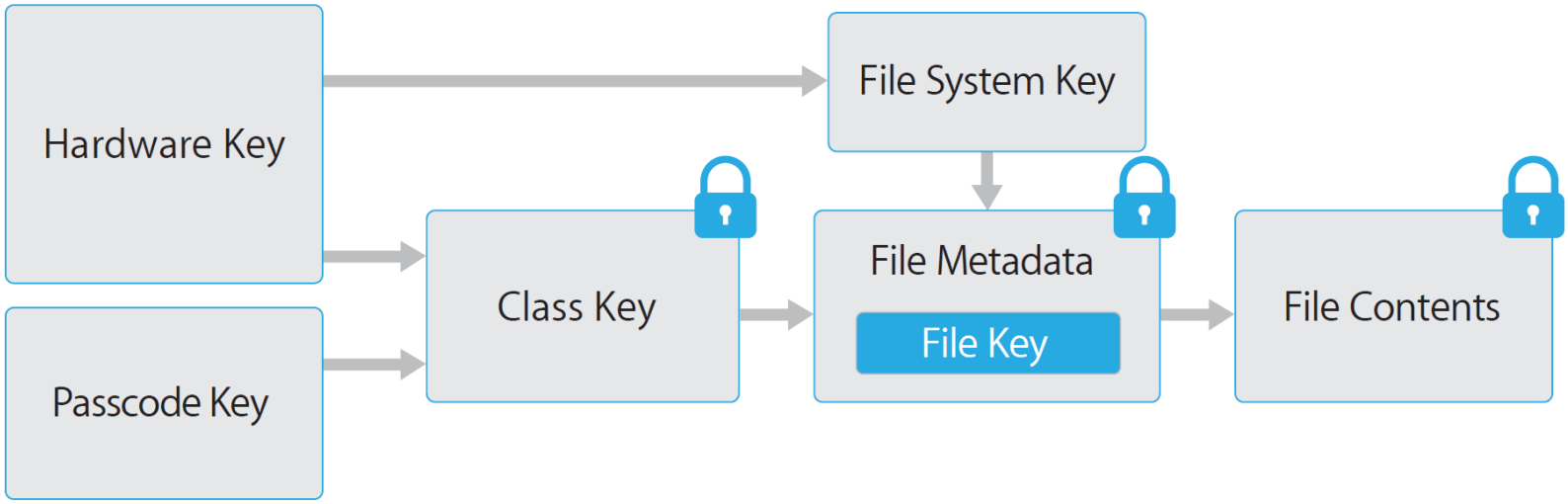


Real implementations

Challenges

- Follow the file structure (What does “directory” mean for ext4?)
- What level of access to allow if no key? (i.e. What is fail to safe?)
- How to do indexing/search?
- What protection can we afford? (i.e. Can we provide authentication?)

IOS file encrypt



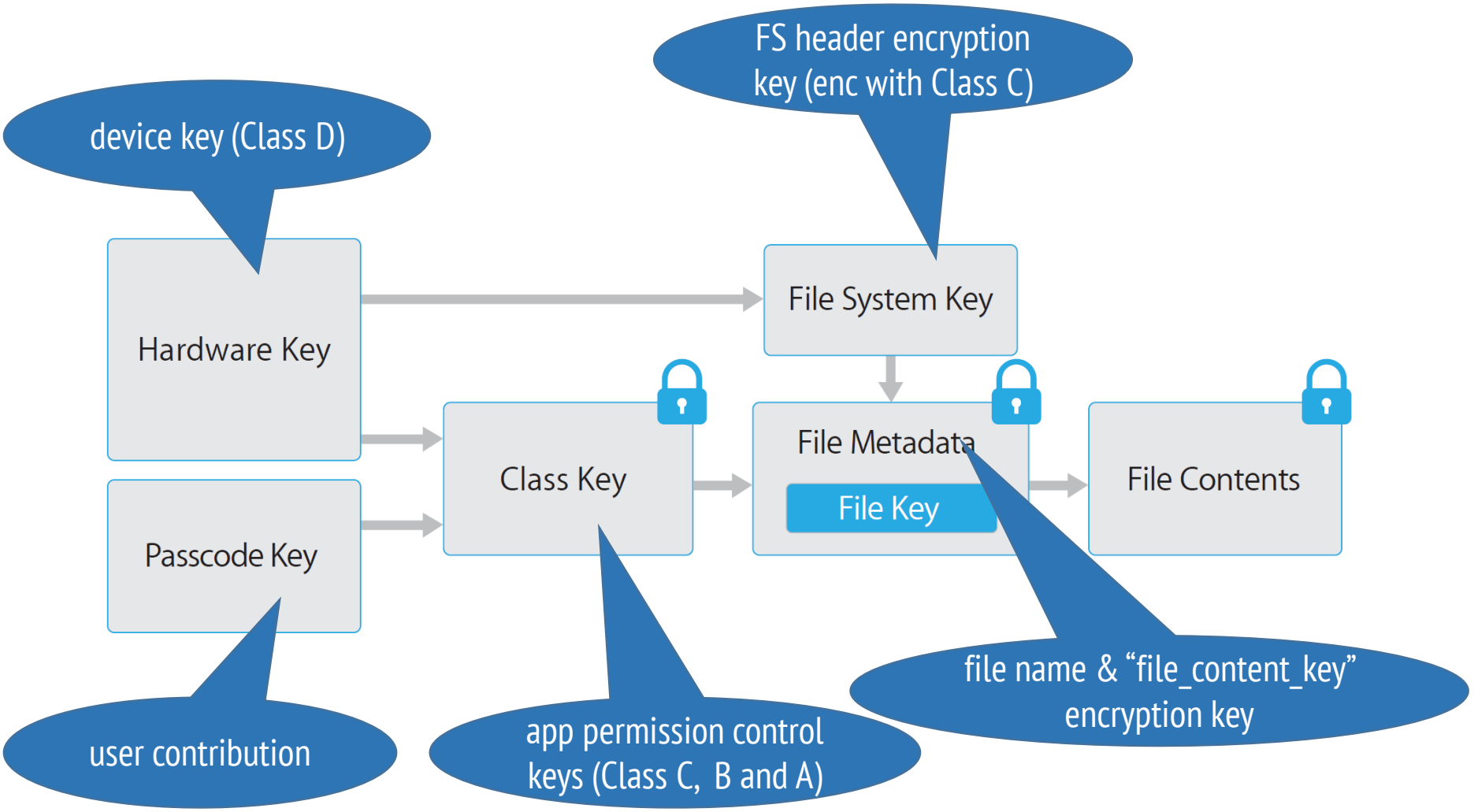
https://www.apple.com/business/docs/iOS_Security_Guide.pdf

IOS file encrypt

IOS has 4 protection classes (or policies)

- Class D – uses a symmetric key held by the SEP. In addition to protecting some files this key also protects the other keys.
- Class C - The default encryption policy for system and user apps and user data.
- Class B – This class facilitates the ability to write encrypted data when the device is locked, whilst at the same time prevent reading/access of the same data.
- Class A – Keys for this class are derived from a user credential. The keys are only stored in RAM and are wiped 10s after the device is locked.

IOS file encrypt



Algorithm 2: WrapKey$^{\sigma}$ in iOS

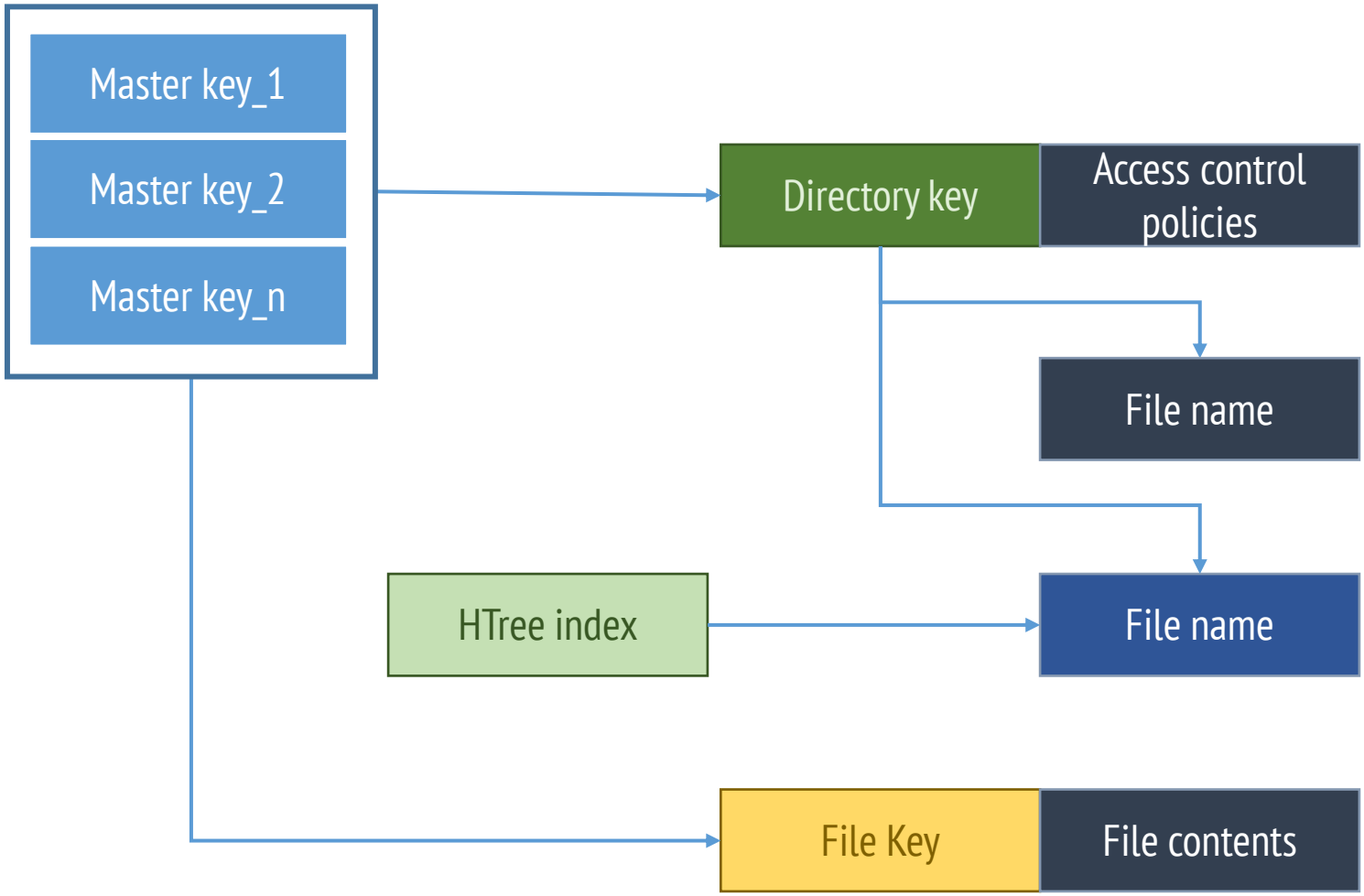
Input: k, w, pol
Output: $\phi, \overline{pol}$
Constants: $\sigma = \{\sigma\text{-}key, \sigma\text{-}ciph\}$

```
1  function WrapKey(k, w, pol)
2      ciph ← pol.cipher
3      pol̄ ← pol
4      if pol.usage = FileProtectionNone then
           /* ClassD i.e. FileProtectionNone */
5          φ′ ← k
6      else
7          if pol.authToken = null then
               /* Encrypt class B_pub key k with wrap key
                  w and policy cipher. */
8              φ′ ← ENC^ciph(k, w)
9          else
               /* Encrypt class A, B_prv and C keys k with
                  wrap key w, policy cipher and user
                  password derived key k_master_key. */
10             k_master_key ← pol.authToken
11             φ″ ← ENC^ciph(k, k_master_key)
12             φ′ ← ENC^ciph(φ″, w)
           /* Bind the policy with the TEE encrypted
              wrap key. */
13         pol̄.wrapkey ← ENC^σ-ciph(w, σ-key)
       /* Encrypt wrapped keys φ′ with the hardware
          key. */
14     φ ← ENC^σ-ciph(φ′, σ-key)
15     return {φ, pol̄}
```

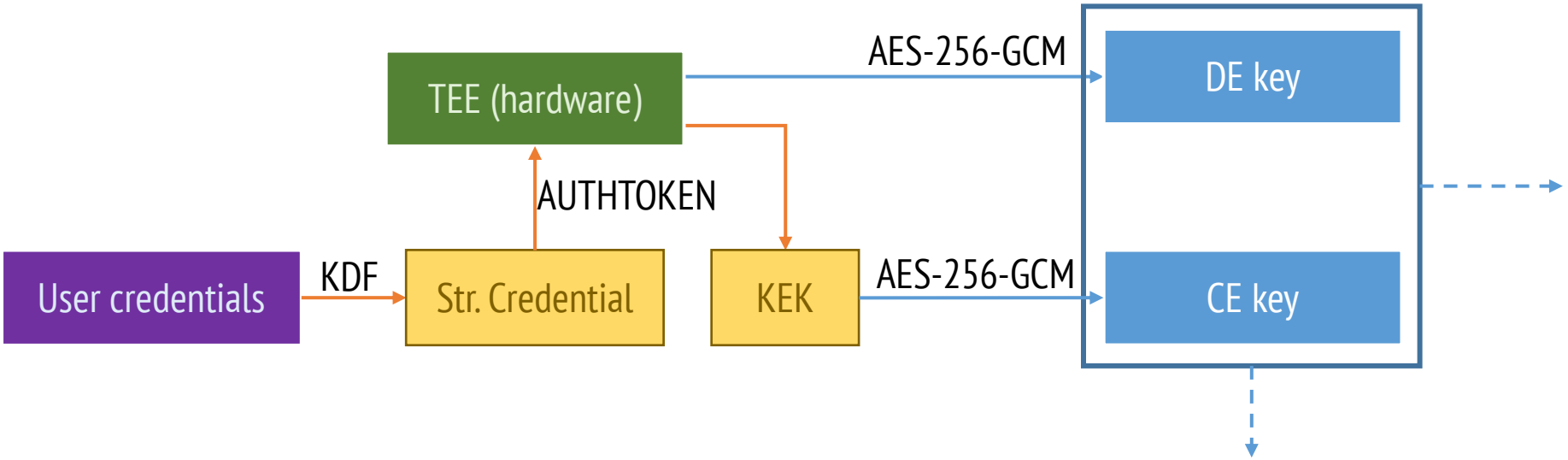
Actual wrapping !

David Galindo, Chris McMahon Stone, Mihai Ordean, Untangling File-based Encryption on Mobile Devices, 2018

Android 7.0+ (ext4 file encryption)



Android 7.0 (key management)



Releasing KEK requires:

1. Stretched Credential: The users' authentication credentials
2. AuthToken: A cryptographically authenticated token generated by gatekeeper.

http://www.delaat.net/rp/2016-2017/p45/report.pdf

Algorithm 1: WrapKey$^{\sigma}$ in Android

Input: k, w, pol
Output: $\phi, \overline{pol}$
Constants: $\sigma = \{\sigma\text{-}key, \sigma\text{-}ciph\}$

```
1  function WrapKey(k, w, pol)
2      ciph ← pol.cipher
3      pol̄ ← pol
       /* Encrypt class keys k with wrap key and
          policy cipher.                          */
4      if pol.usage = DeviceDataAfterBoot
5         or pol.usage = UserDataAfterBoot then
           /* DE class has no user token.         */
6          φ ← ENC^ciph(k, w)
           /* Encrypt wrap key w and bind it to the
              policy.                             */
7          pol̄.wrapkey ← ENC^(σ-ciph)(w, σ-key)
8          return {φ, pol̄}
9      else if pol.usage = UserDataAfterAuth
10        and VerifyToken(pol.authToken) then
           /* CE class verifies the user token.   */
11         φ ← ENC^ciph(k, w)
           /* Encrypt wrap key w and bind it to the
              policy.                             */
12         pol̄.wrapkey ← ENC^(σ-ciph)(w, σ-key)
13         return {φ, pol̄}
14     else
15         return ⊥
```

Actual wrapping !

David Galindo, Chris McMahon Stone, Mihai Ordean, Untangling File-based Encryption on Mobile Devices, 2018

Are there differences between IOS and Android?

- Key management and derivation seems similar
  - Both platforms perform key wrapping, unwrapping, file encryption, file decryption, …
  - IOS captures an extra scenario with Class B
- Both platforms use the same encryption primitive to encrypt files (AES-XTS)

- Are we missing anything?

Are there differences between IOS and Android?

	KGen	ProvisionKey	EvictKey	WrapKey	UnwrapKey	Encrypt	Decrypt
Android							
User space	✓						
Kernel space		✓	✓			✓	✓
Sec. Elem.				✓	✓		
iOS							
User space							
Kernel space							
Sec. Elem.	✓	✓	✓	✓	✓	✓	✓

David Galindo, Chris McMahon Stone, Mihai Ordean, Untangling File-based Encryption on Mobile Devices, 2018

Advantages/disadvantages of file based encryption

File based encryption

- Complex design: generally many keys are used
- Does not protect metadata as well as full disk encryption
- If a key is compromised attacker gets limited access.
- More flexible

Conclusions

- Encryption provides confidentiality to data

- Full disk encryption has a simpler structure
- Full disk encryption hides metadata
- Full disk encryption usually uses one key per disk

- File based encryption has a complex structure
- File based encryption uses many keys thus is more resilient to key compromise
- File based encryption does not hide metadata as well as FDE