Insecure Until Proven Updated: Analyzing AMD SEV‘s Remote Attestation

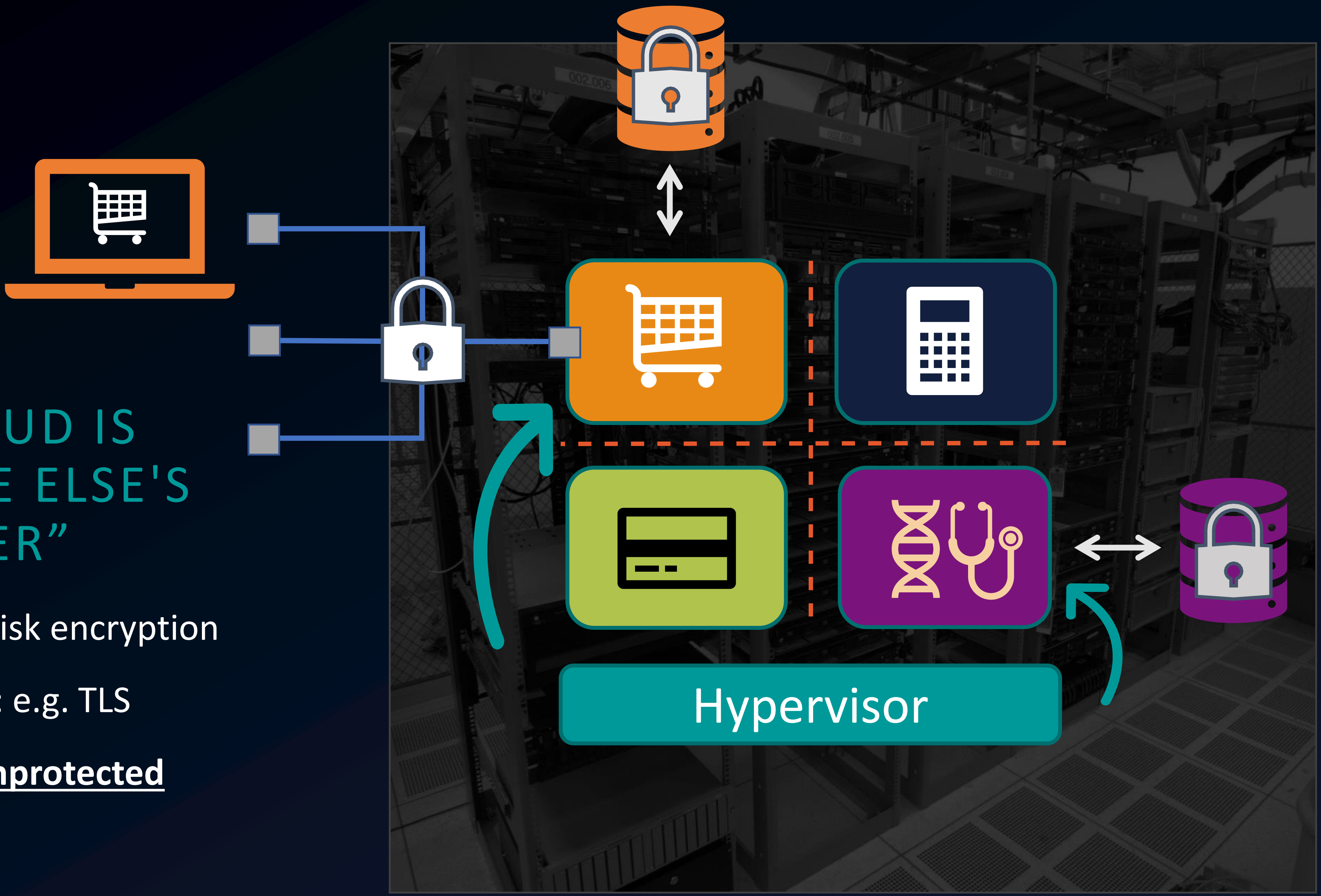
Robert Buhren Technische Universität Berlin
Christian Werling Hasso-Plattner Institute Potsdam
Jean-Pierre Seifert Technische Universität Berlin

2019 ACM SIGSAC Conference on Computer and Communications Security [CCS’19]

Technische Universität Berlin

“THE CLOUD IS SOMEONE ELSE'S COMPUTER”

“THE CLOUD IS SOMEONE ELSE'S COMPUTER”



Data-At-Rest: disk encryption

Data-In-Transit: e.g. TLS

Data-In-Use: **unprotected**

SECURE ENCRYPTED VIRTUALIZATION

AMD

SECURE ENCRYPTED VIRTUALIZATION

"… SEV technology is built around a threat model where an attacker … can potentially execute malware at the higher privileged hypervisor level as well"

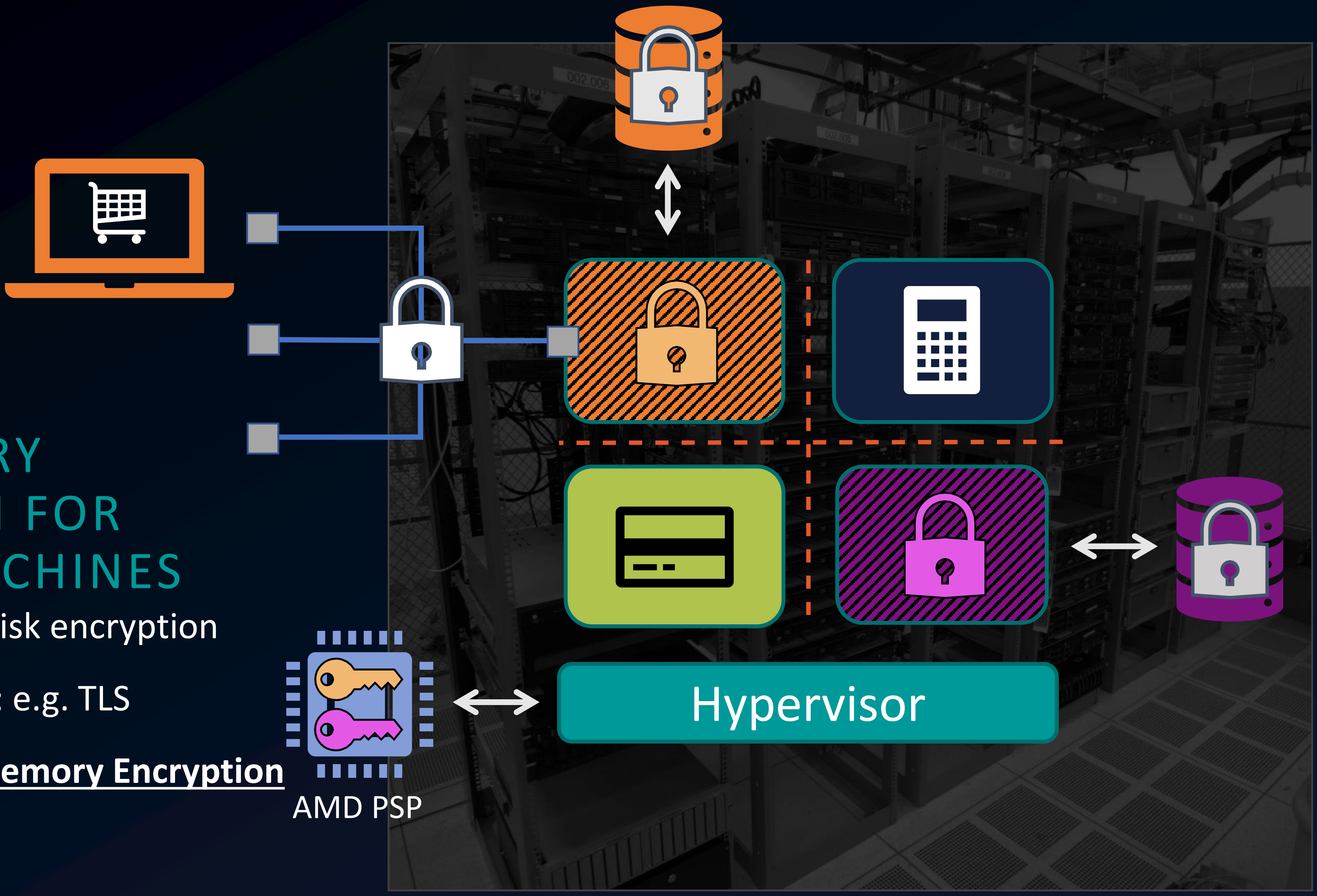
https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf

SEV: MEMORY ENCRYPTION FOR VIRTUAL MACHINES

Data-At-Rest: disk encryption

Data-In-Transit: e.g. TLS

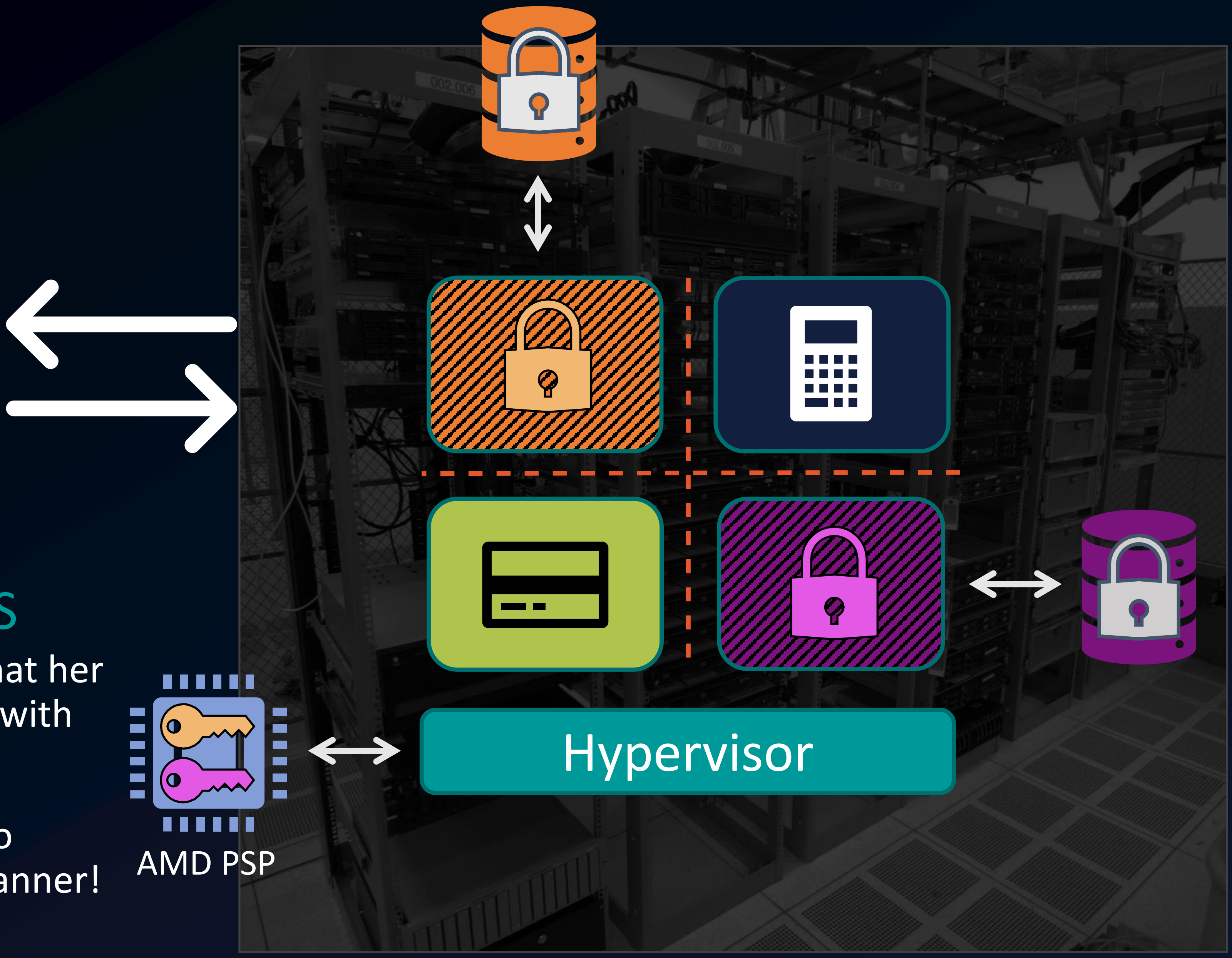
Data-In-Use: **Memory Encryption** (AES-128)



SEV: MEMORY ENCRYPTION FOR VIRTUAL MACHINES

A customer needs to ensure that her virtual machine was deployed with SEV protection in place!

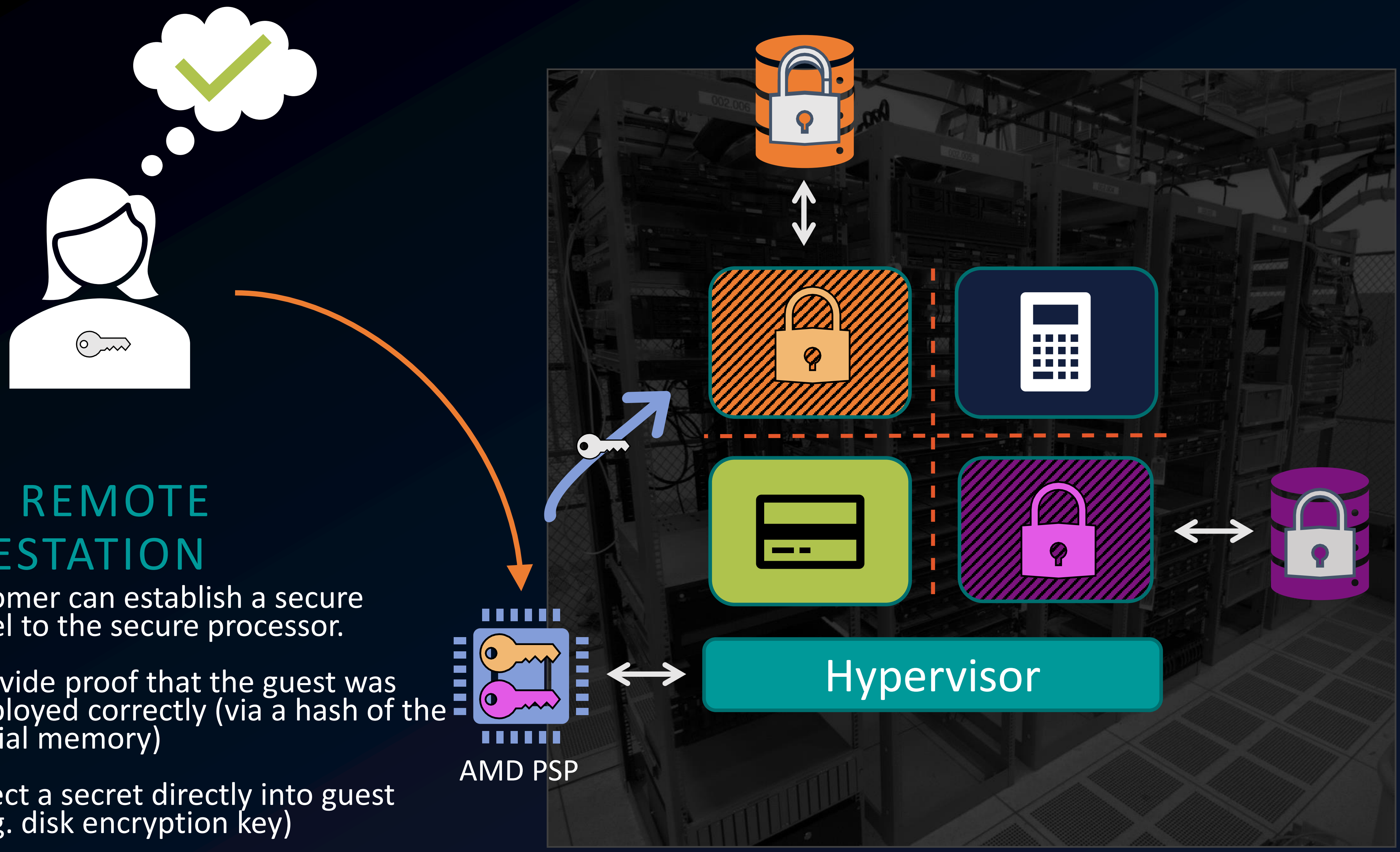
A customer needs to be able to provide a secret in a secure manner!



SEV: REMOTE ATTESTATION

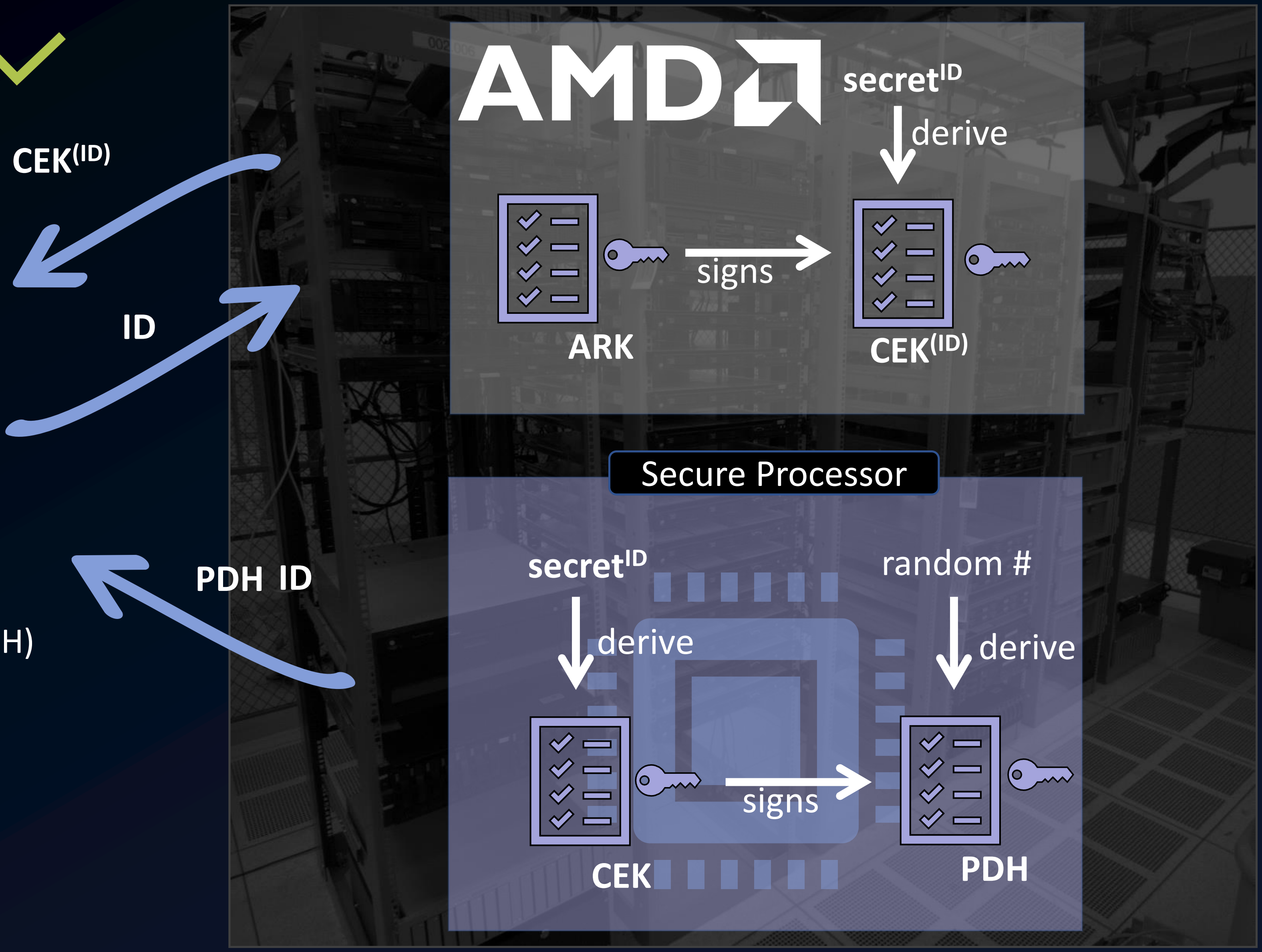
A customer can establish a secure channel to the secure processor.

- Provide proof that the guest was deployed correctly (via a hash of the initial memory)
- Inject a secret directly into guest (e.g. disk encryption key)



PDH -> CEK -> ARK

An authentic AMD system: ✓



SEV KEYS (simplified)

Platform Diffie Hellman Key (PDH)

Chip Endorsement Key (CEK)

AMD Root Key (ARK)

PDH -> CEK -> ARK

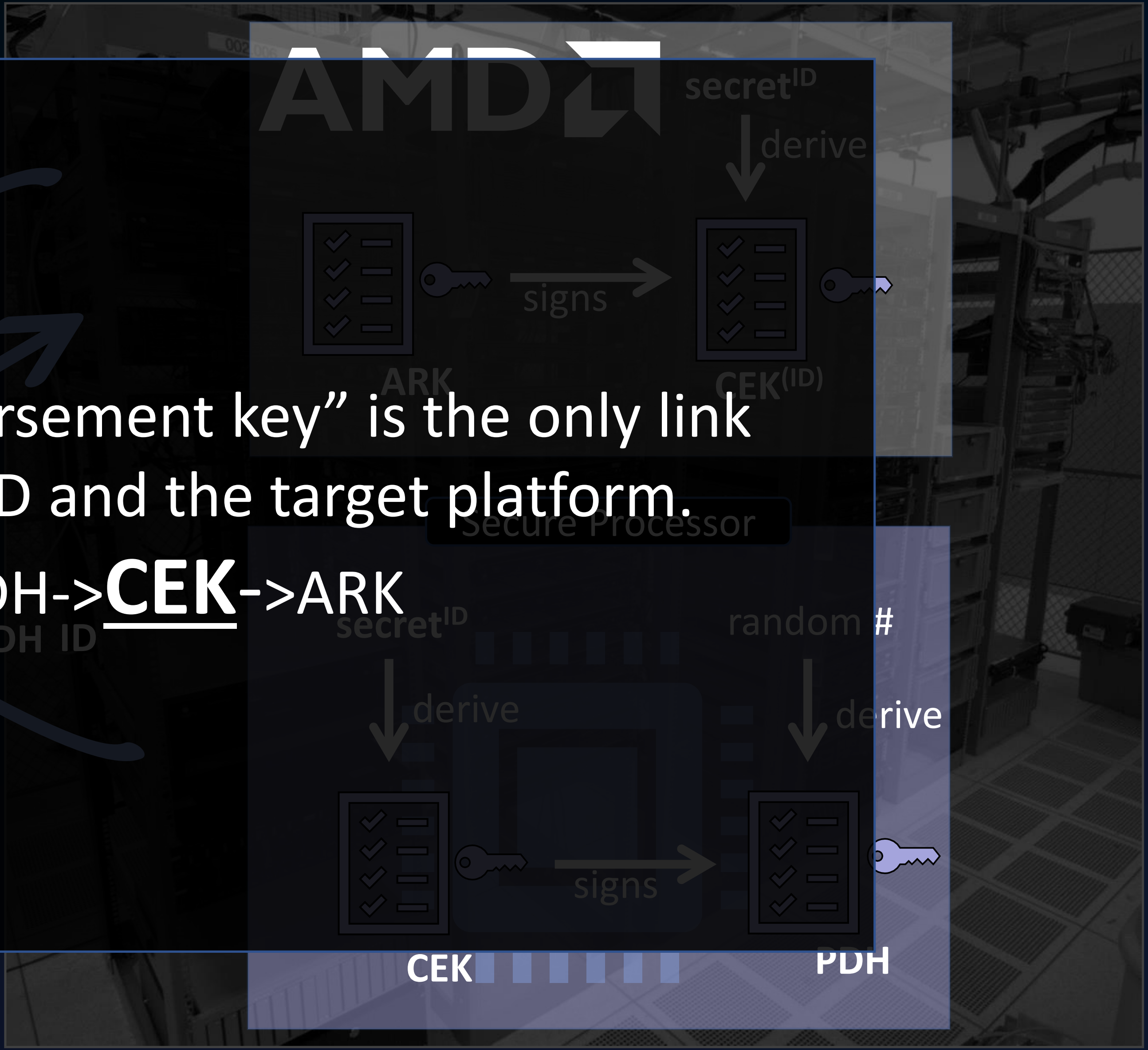
An authentic AMD system:

SEV KEYS (simplified)

Platform Diffie Hellman Key (PDH)

Chip Endorsement Key (CEK)

AMD Root Key (ARK)



The “chip endorsement key” is the only link between AMD and the target platform.

PDH->**CEK**->ARK

FIRMWARE ANALYSIS

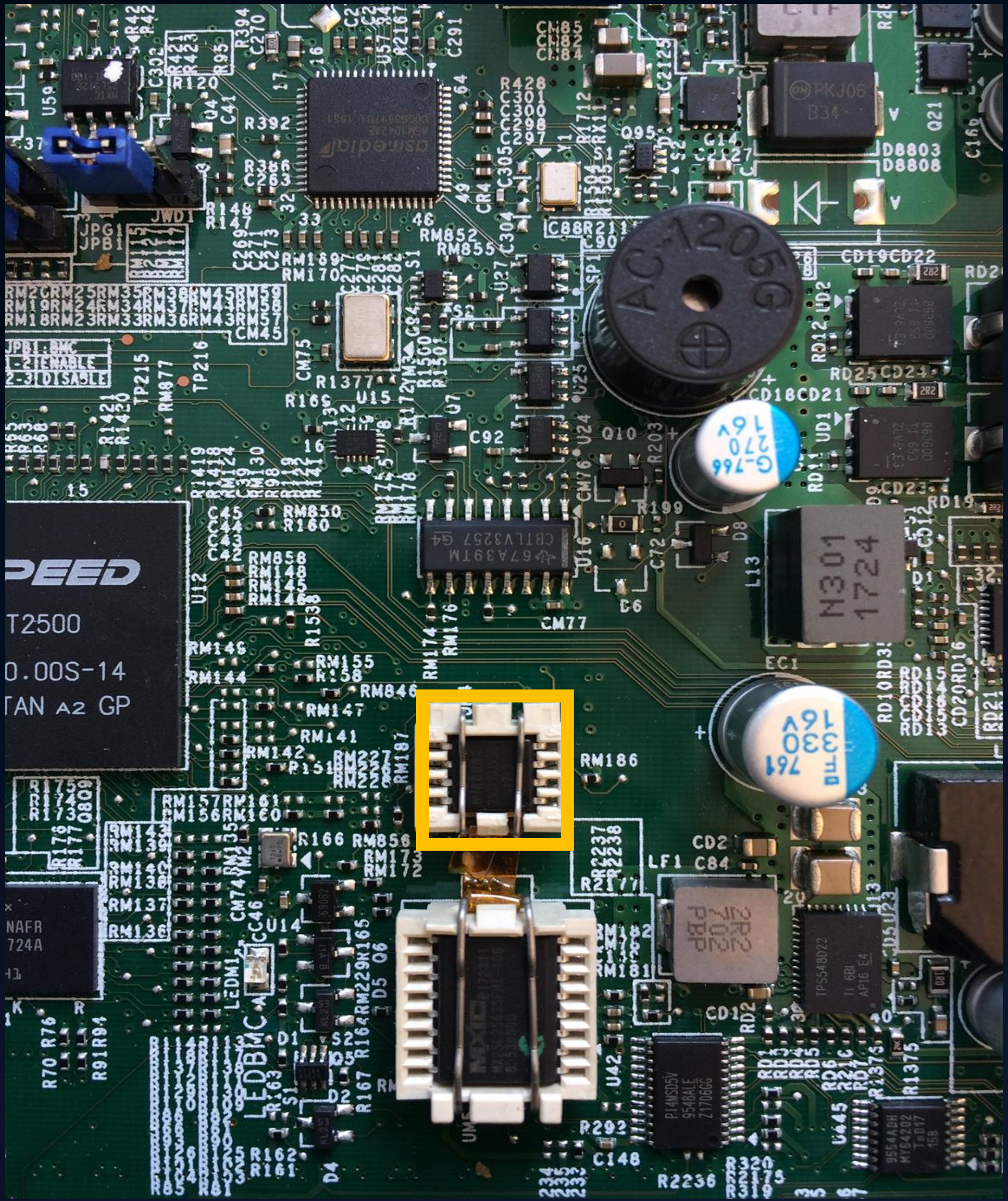
FIRMWARE ANALYSIS

Secure Processor is part of x86 die.

- ARM Cortex A5

Firmware is stored along UEFI FW!

Updatable through UEFI update.

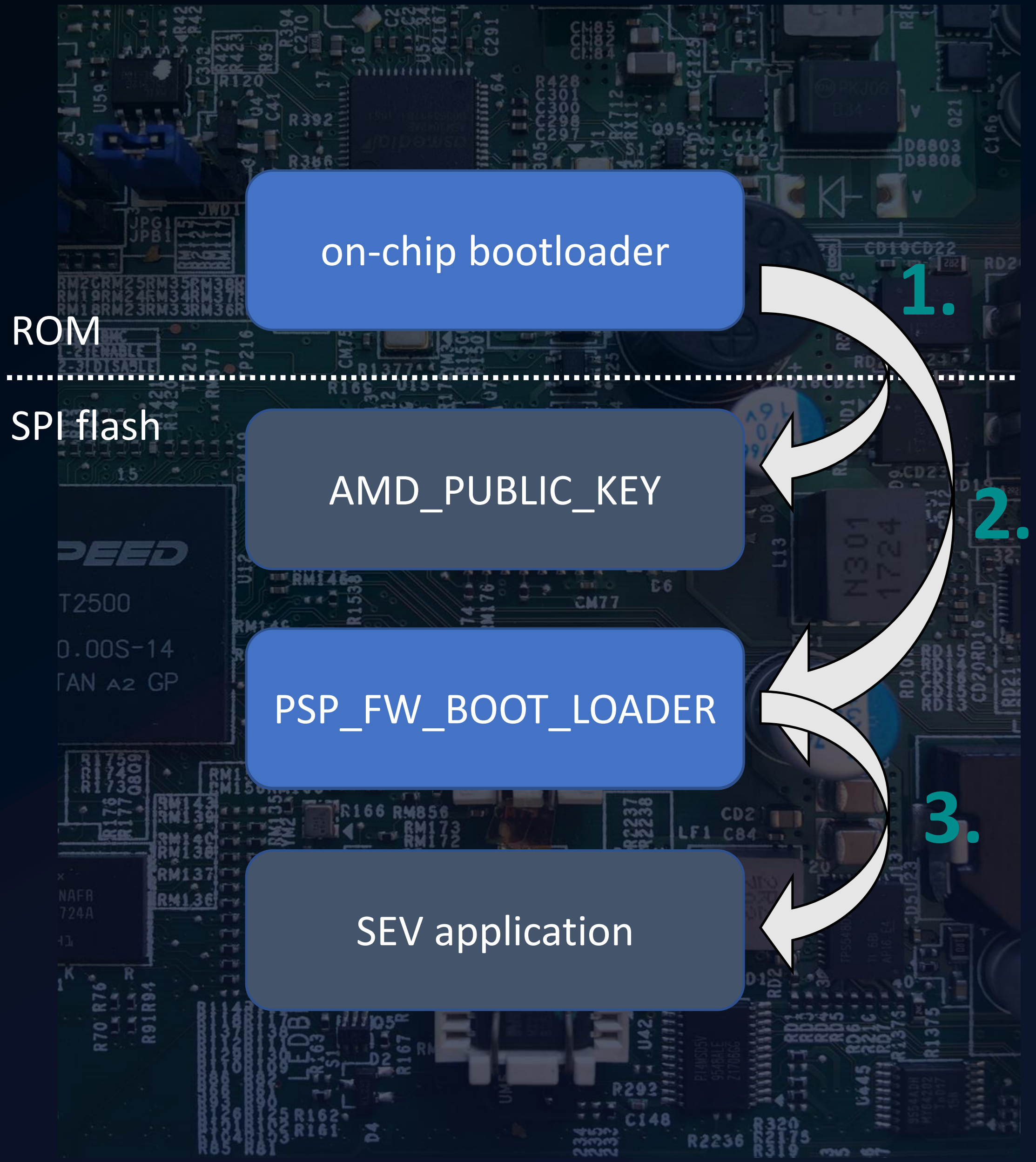


```
$ psptool uefi_image.bin
+---+-------+----------+------------+-------+------------------------------+----------+----------------+
|   | Entry |  Address |       Size |  Type |                    Type Name |  Version |      Signed by |
+---+-------+----------+------------+-------+------------------------------+----------+----------------+
|   |     0 |  0xc2000 |      0x240 |   0x0 |               AMD_PUBLIC_KEY |          |                |
|   |     1 | 0x281000 |     0x8000 |   0x1 |           PSP_FW_BOOT_LOADER | 0.5.0.3B | AMD_PUBLIC_KEY |
|   |     2 | 0x289000 |    0x14000 |   0x8 |               SMU_OFFCHIP_FW |  0.0.0.0 |           None |
|   |     3 |  0xc3000 |     0x6000 |   0x3 | PSP_FW_RECOVERY_BOOT_LOADER | 0.5.0.17 | AMD_PUBLIC_KEY |
|   |     4 |  0xc9000 |      0x340 |   0x5 |              BIOS_PUBLIC_KEY |          |                |
|   |     5 | 0xfff000 |     0x1000 |   0x6 |            BIOS_RTM_FIRMWARE |          |                |
|   |     6 | 0x29d000 |    0x1e000 |   0x2 |            PSP_FW_TRUSTED_OS | 0.5.0.3B | AMD_PUBLIC_KEY |
|   |     7 |  0xa0000 |    0x10000 |   0x4 |                  PSP_NV_DATA |          |                |
|   |     8 | 0x2bb000 |    0x14000 | 0x108 |          PSP_SMU_FN_FIRMWARE |  0.0.0.0 |           None |
|   |     9 |  0xca000 |      0x340 |   0x9 |       AMD_SEC_DBG_PUBLIC_KEY |          |                |
|   |    10 |      0x1 | 0xffffffff |   0xb |       AMD_SOFT_FUSE_CHAIN_01 | E9.0.0.0 |           None |
|   |    11 |  0xcb000 |      0x340 |   0xd | PSP_BOOT_TIME_TRUSTLETS_KEY |          |                |
+---+-------+----------+------------+-------+------------------------------+----------+----------------+
```

psptool: https://github.com/cwerling/psptool

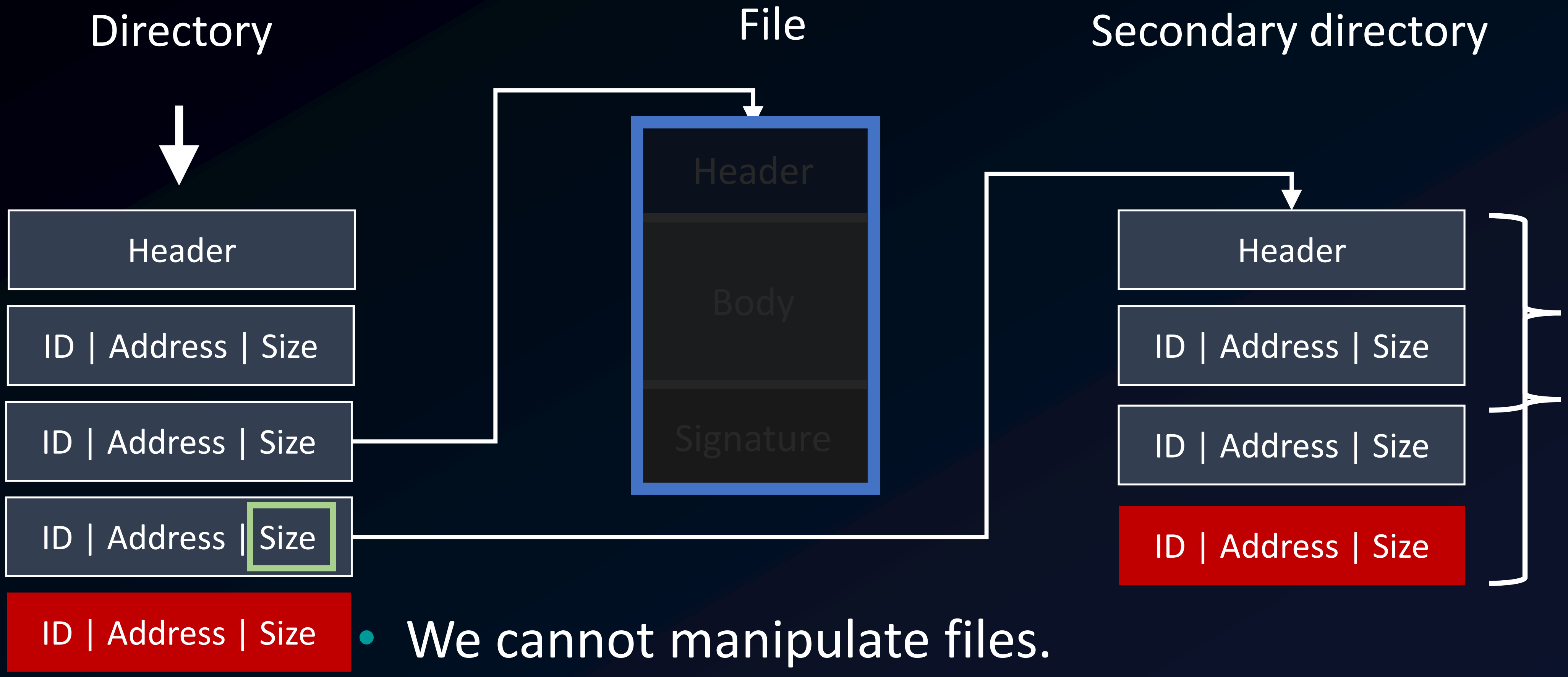
FIRMWARE ANALYSIS

1. Load & verify AMD_PUBLIC_KEY
   - verify using hash stored in fuses
2. Load & verify PSP_FW_BOOT_LOADER
   - verify using verified public key
3. Load & verify SEV application
   - verify using verified public key



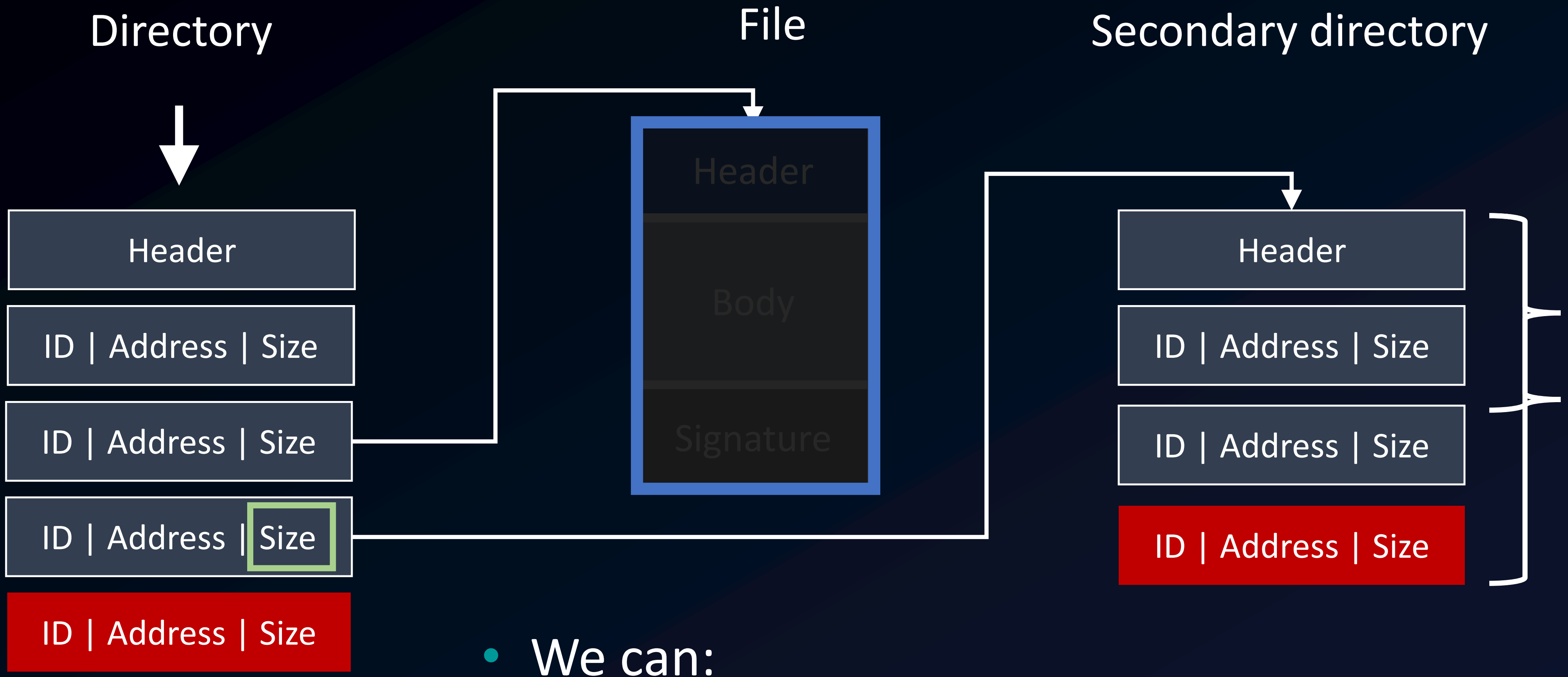
The Bug

Attacker Capabilities

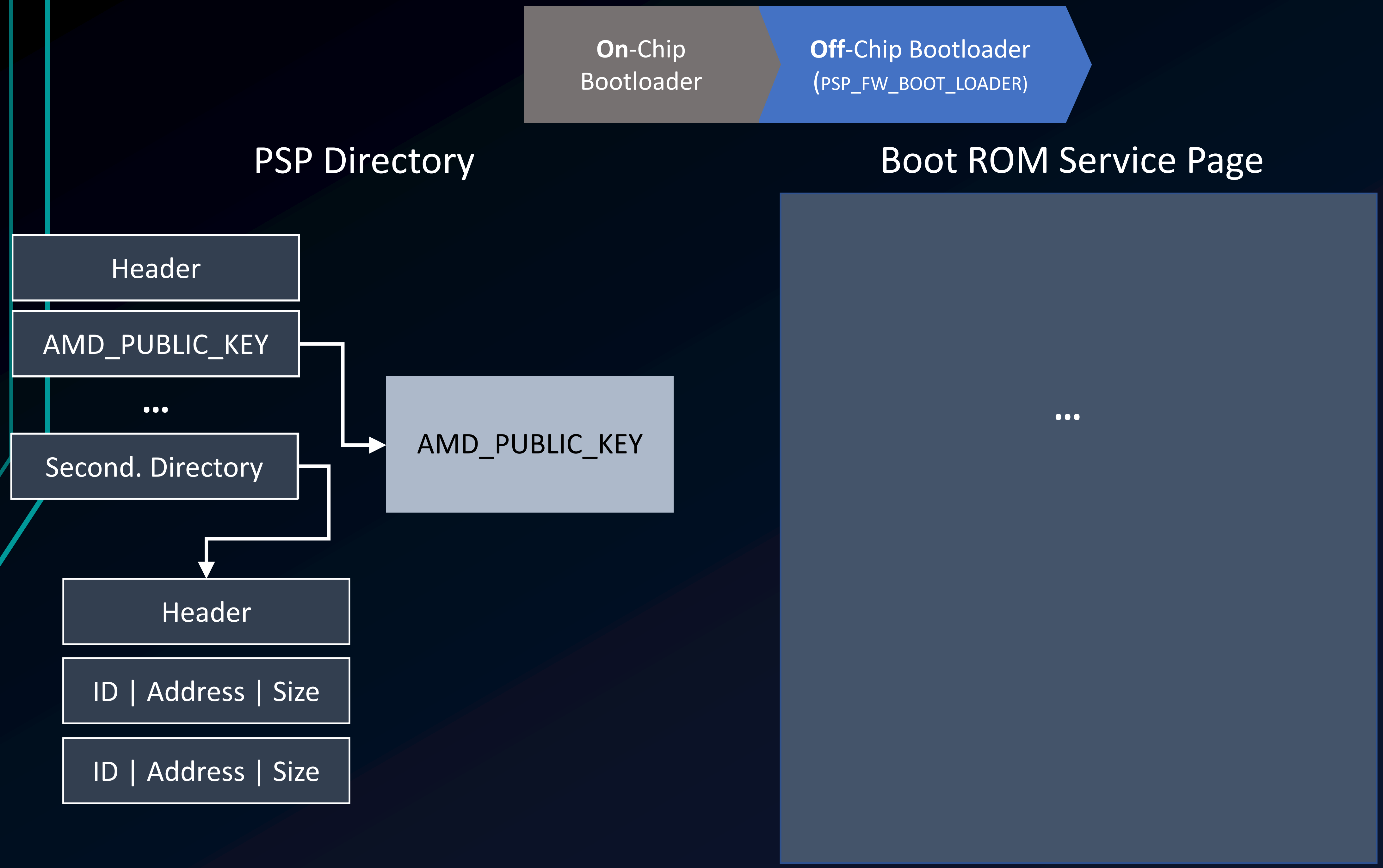


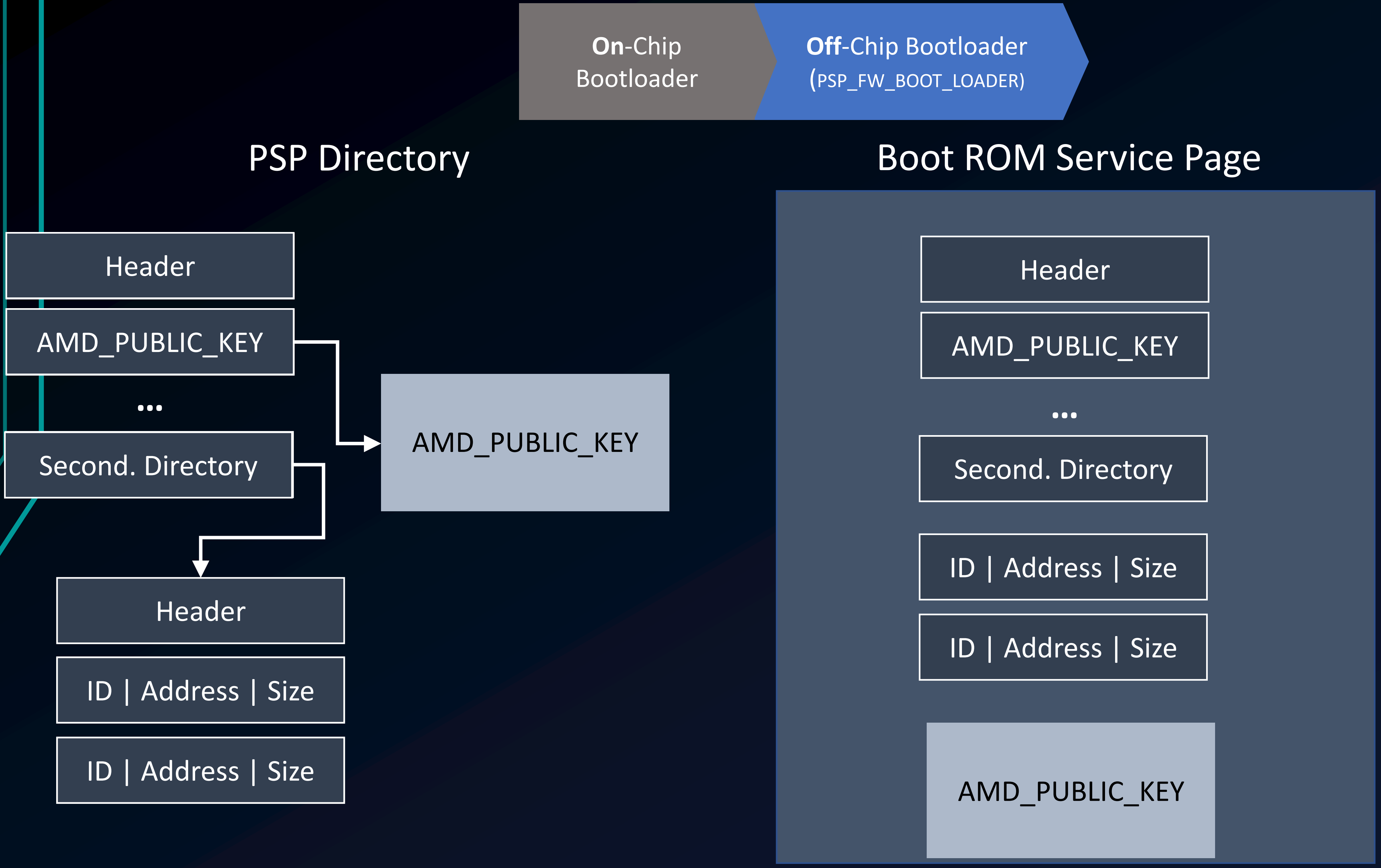
- We cannot manipulate files.
- We ***can*** manipulate the directories!

Attacker Capabilities



- We can:
  - Add Entries
  - Remove Entries
  - Change Entries


On-Chip Bootloader
Off-Chip Bootloader (PSP_FW_BOOT_LOADER)
PSP Directory
Boot ROM Service Page
Header
AMD_PUBLIC_KEY
...
Second. Directory
AMD_PUBLIC_KEY
Header
ID | Address | Size
ID | Address | Size
...


On-Chip Bootloader
Off-Chip Bootloader (PSP_FW_BOOT_LOADER)
PSP Directory
Header
AMD_PUBLIC_KEY
...
Second. Directory
AMD_PUBLIC_KEY
Header
ID | Address | Size
ID | Address | Size
Boot ROM Service Page
Header
AMD_PUBLIC_KEY
...
Second. Directory
ID | Address | Size
ID | Address | Size
AMD_PUBLIC_KEY


What could possibly
go wrong?

On-Chip Bootloader → **Off**-Chip Bootloader (PSP_FW_BOOT_LOADER)

PSP Directory

Header

AMD_PUBLIC_KEY

...

Second. Directory

Header

ID | Address | Size

ID | Address | Size

Max. 64

Boot ROM Service Page

Header

AMD_PUBLIC_KEY

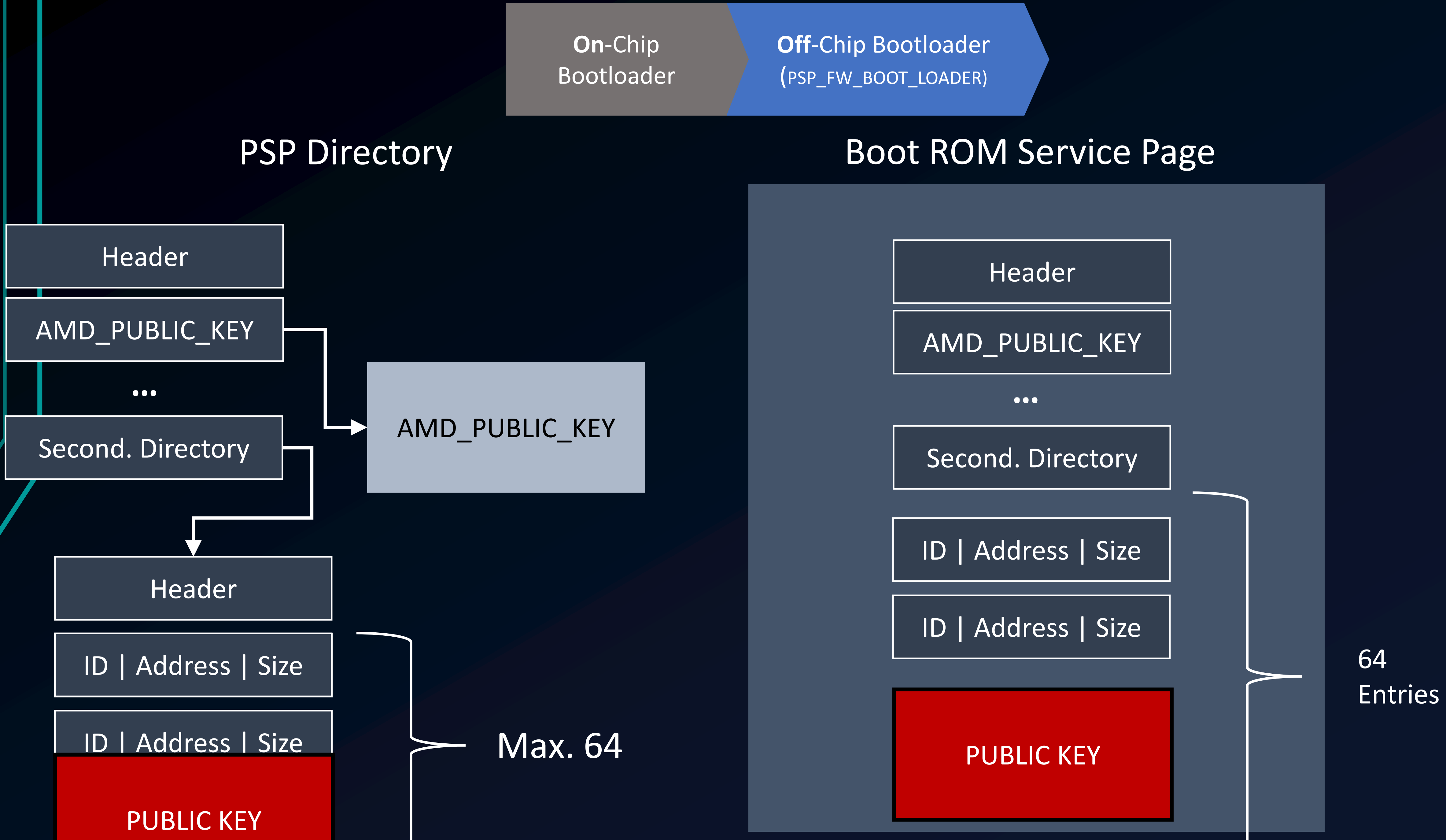
...

Second. Directory

64 Entries

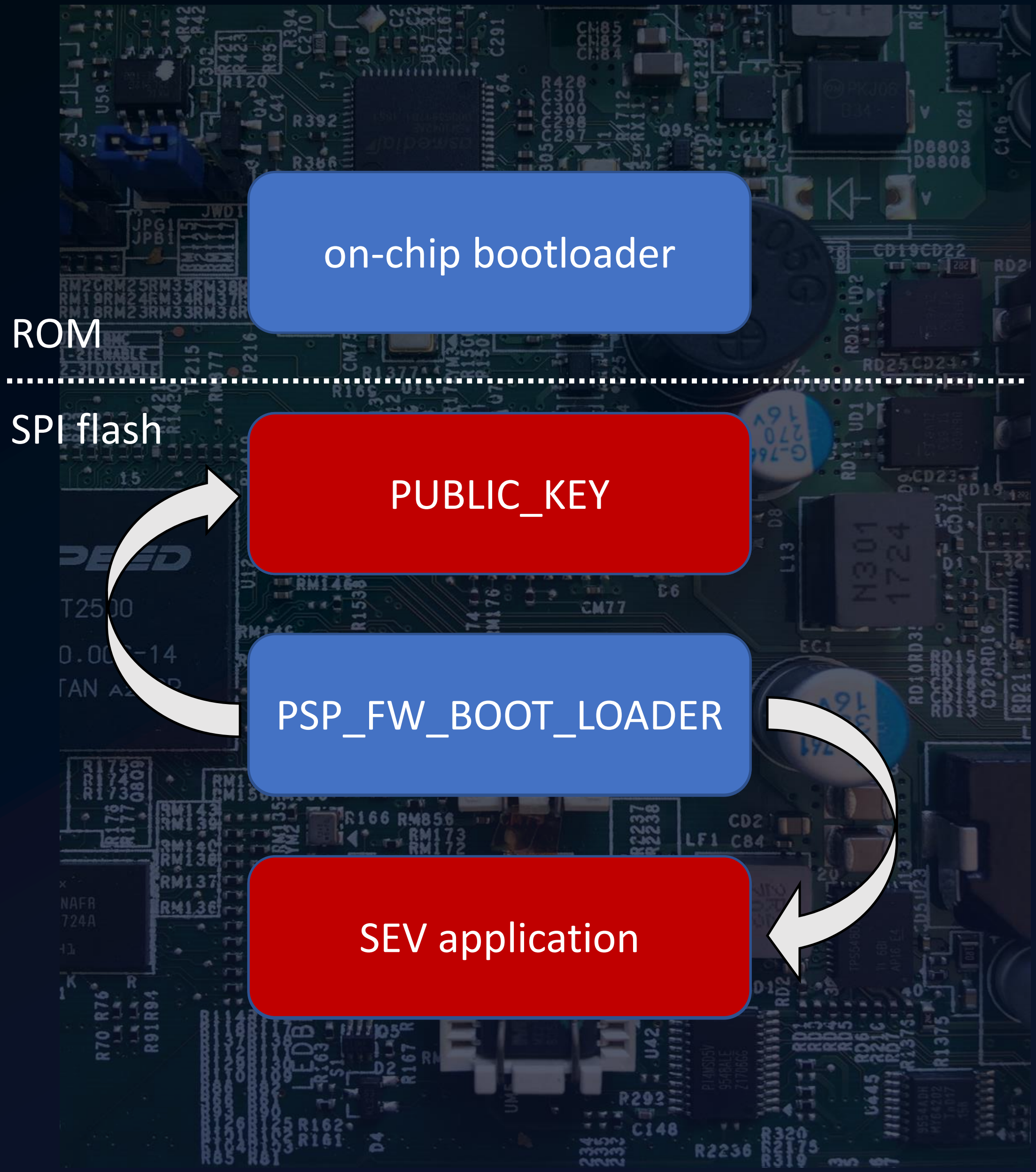
AMD_PUBLIC_KEY

```
int append_second(void) {
  ...
  if (nr_entries > 64u)
    return -1;
  ...
  return 0;
}
```


On-Chip Bootloader
Off-Chip Bootloader (PSP_FW_BOOT_LOADER)
PSP Directory
Header
AMD_PUBLIC_KEY
...
Second. Directory
AMD_PUBLIC_KEY
Header
ID | Address | Size
ID | Address | Size
PUBLIC KEY
Max. 64
Boot ROM Service Page
Header
AMD_PUBLIC_KEY
...
Second. Directory
ID | Address | Size
ID | Address | Size
PUBLIC KEY
64 Entries

ATTACKS

- The off-chip bootloader uses the public key to verify applications signatures.
- Firmware issues allow us to provide our own signing key for applications.

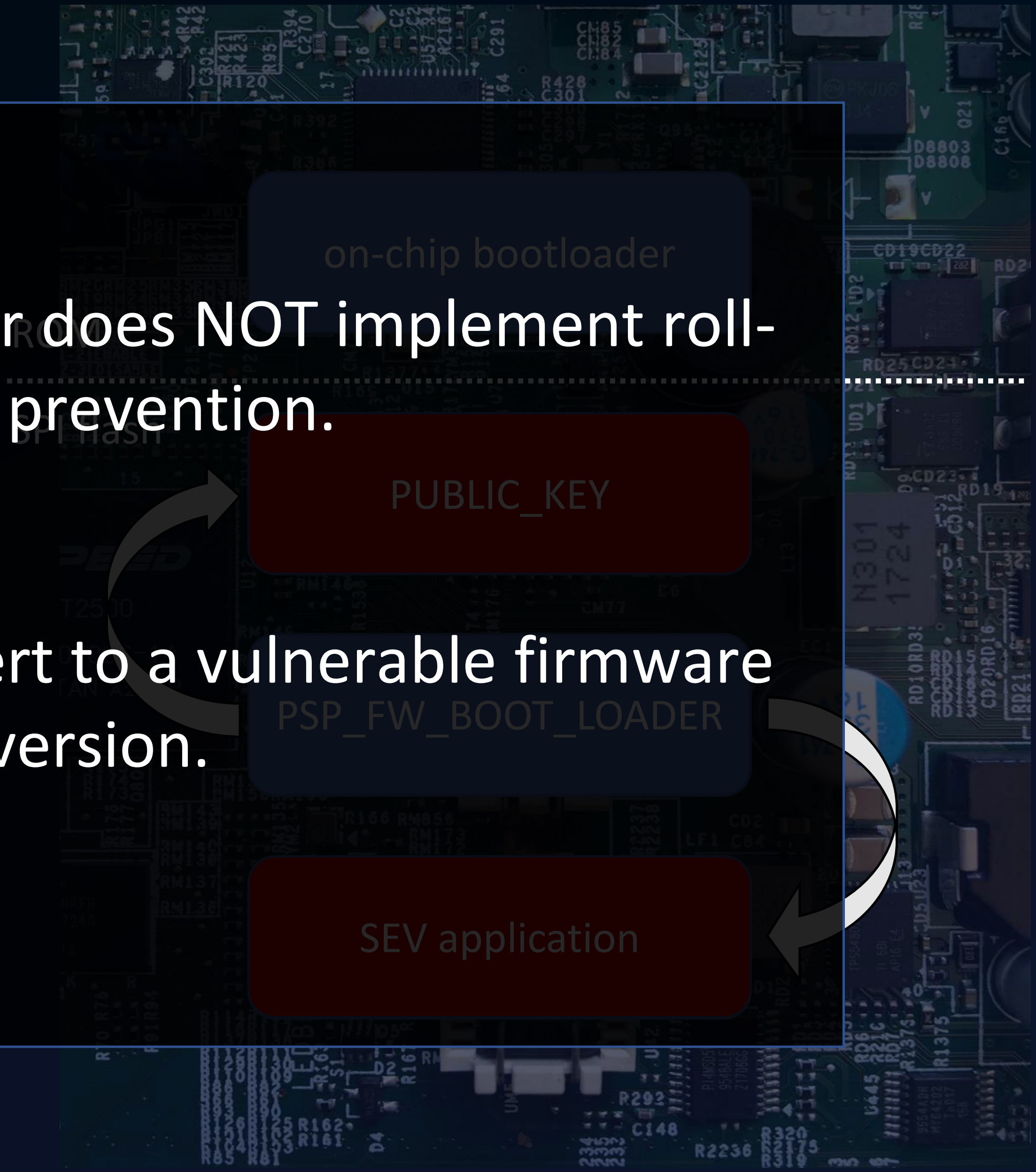


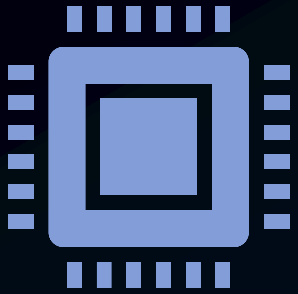
The secure processor does NOT implement roll-back prevention.

An attacker can revert to a vulnerable firmware version.

ATTACKS

- The off-chip bootloader uses the public key to verify applications signatures.
- Firmware issues allow us to provide our own signing key for applications.

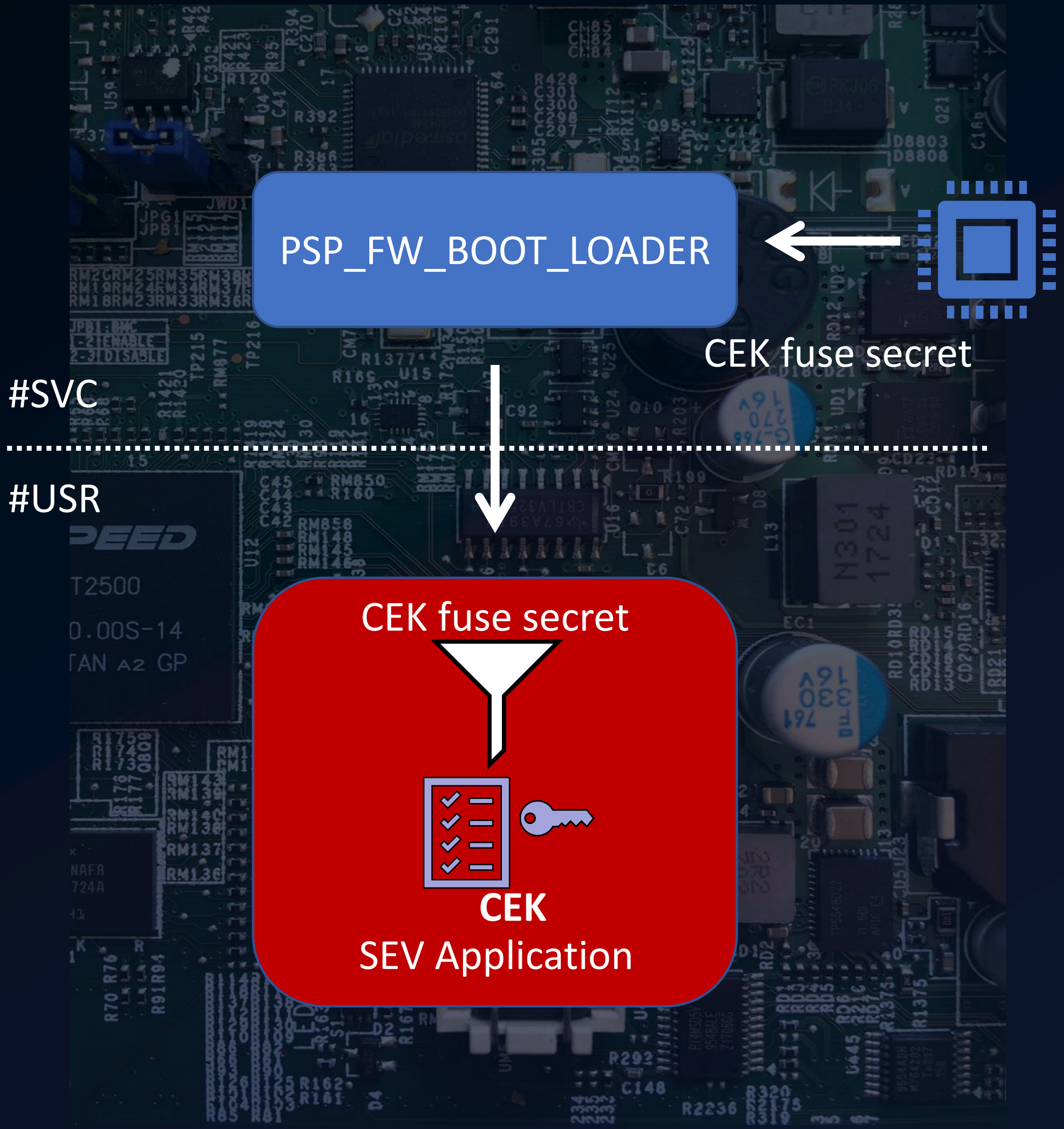




CEK DERIVATION

Chapter 2.1.3 AMD SEV API Specification:

“It exists for the lifetime of the platform and is stored within the hardware of the AMD Secure Processor”



CEK DERIVATION

Chapter 2.1.3 AMD SEV API Specification:

“It exists for the lifetime of the platform and is stored within the hardware of the AMD Secure Processor”

The CEK does not depend on firmware versions.

An extracted CEK is still valid after the firmware issues have been fixed.

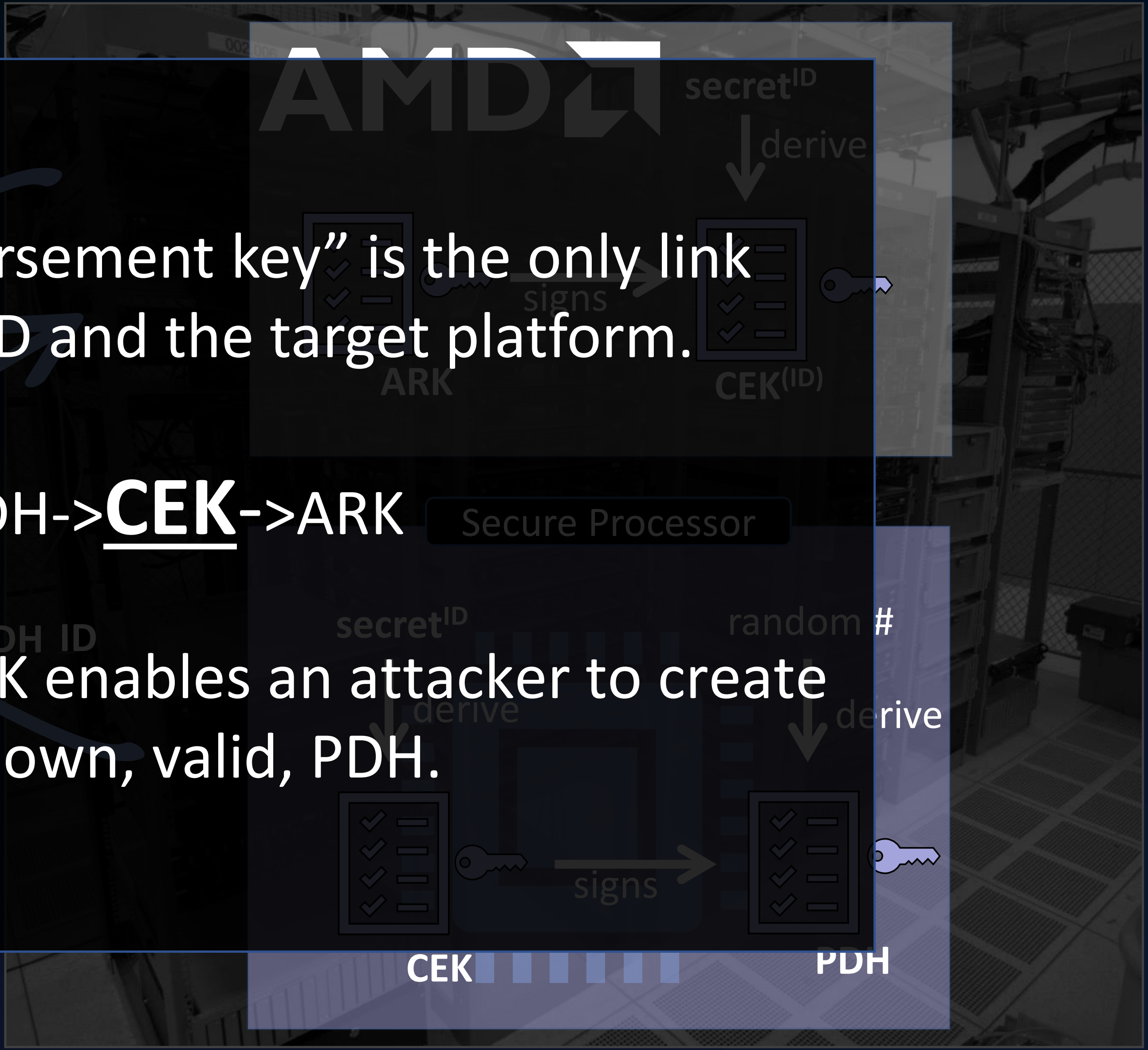
An authentic AMD system:

SEV KEYS (simplified)

Platform Diffie Hellman Key (PDH)

Chip Endorsement Key (CEK)

AMD Root Key (ARK)



The “chip endorsement key” is the only link between AMD and the target platform.

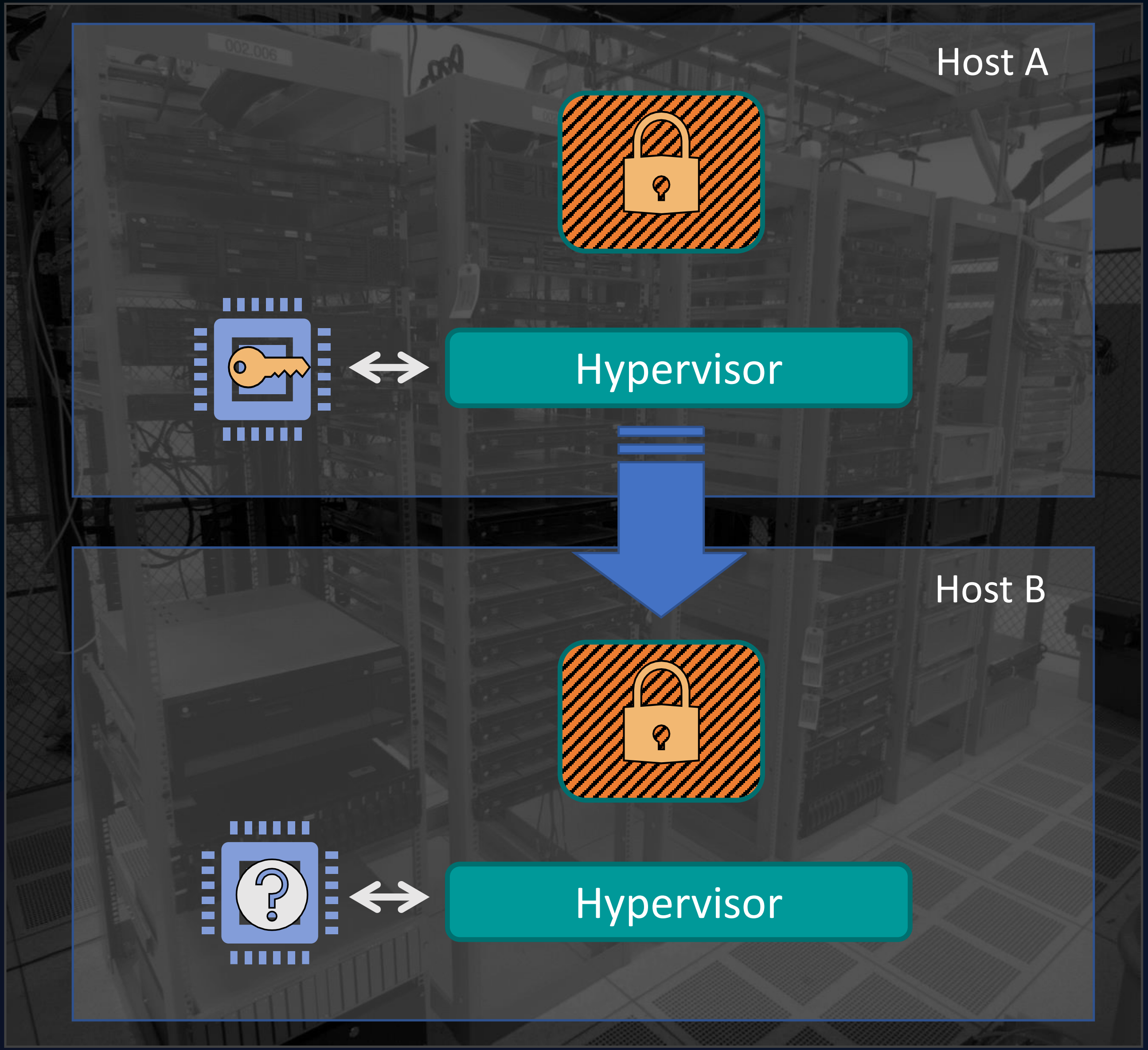
PDH->**CEK**->ARK

Controlling the CEK enables an attacker to create her own, valid, PDH.

MIGRATION ATTACK

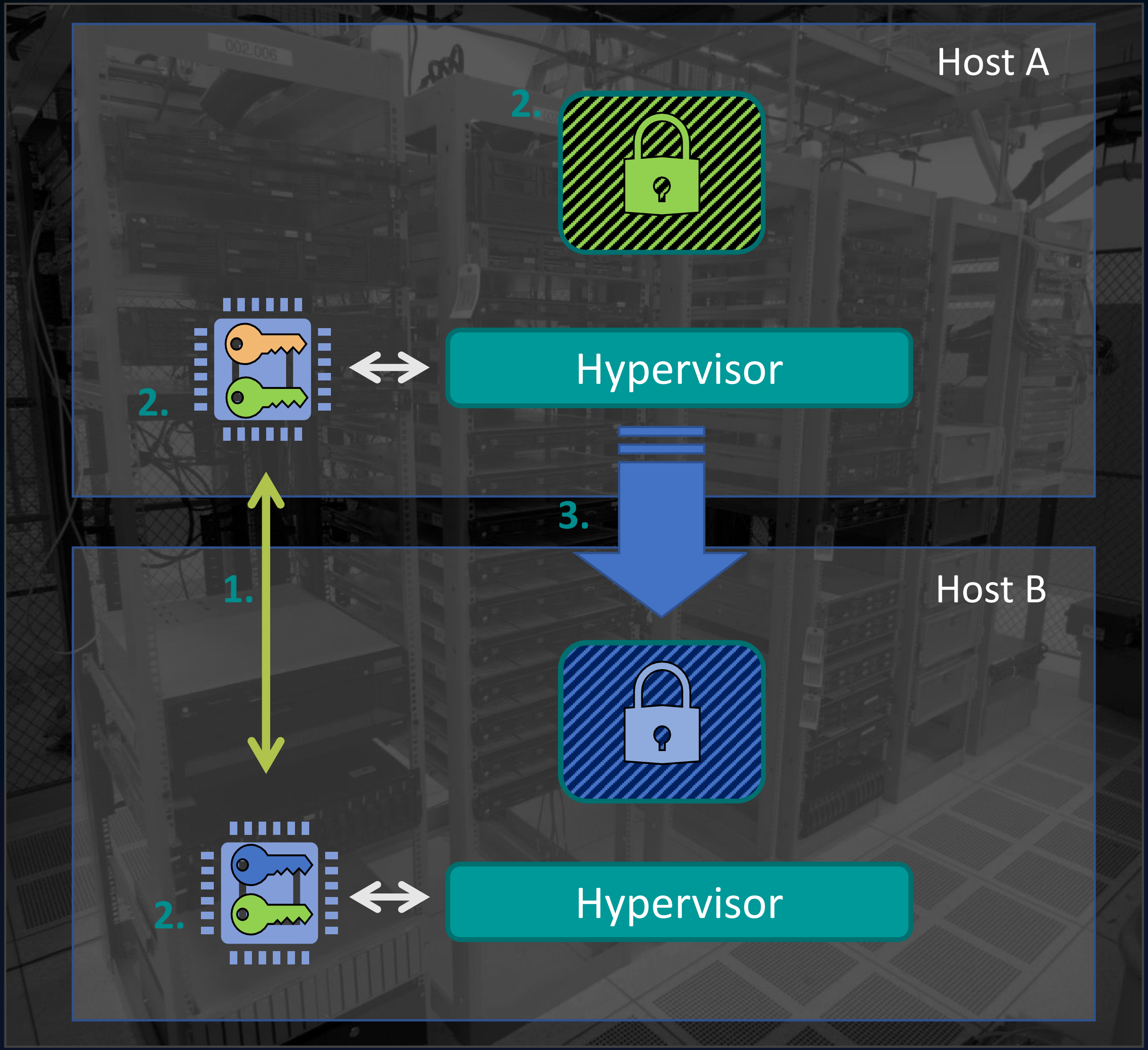
MIGRATION

- Load balancing in case of overload.
- High availability in case of host failure.



SEV MIGRATION

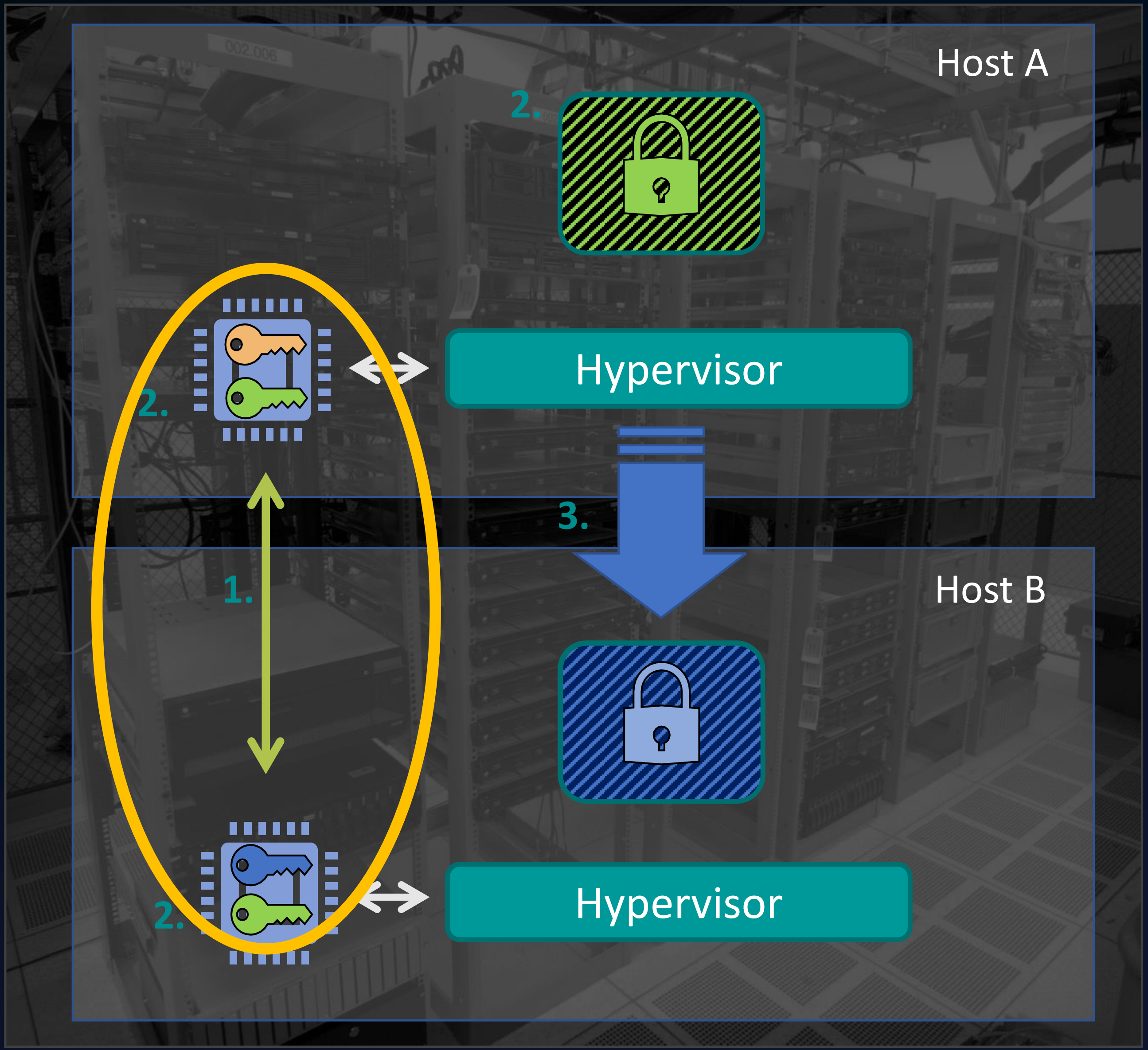
1. Establish secure channel to target secure processor.
2. Derive shared transport keys & re-encrypt VM using transport keys.
3. Transfer VM.
4. Re-encrypt VM using fresh key.



PDH->**CEK**->ARK

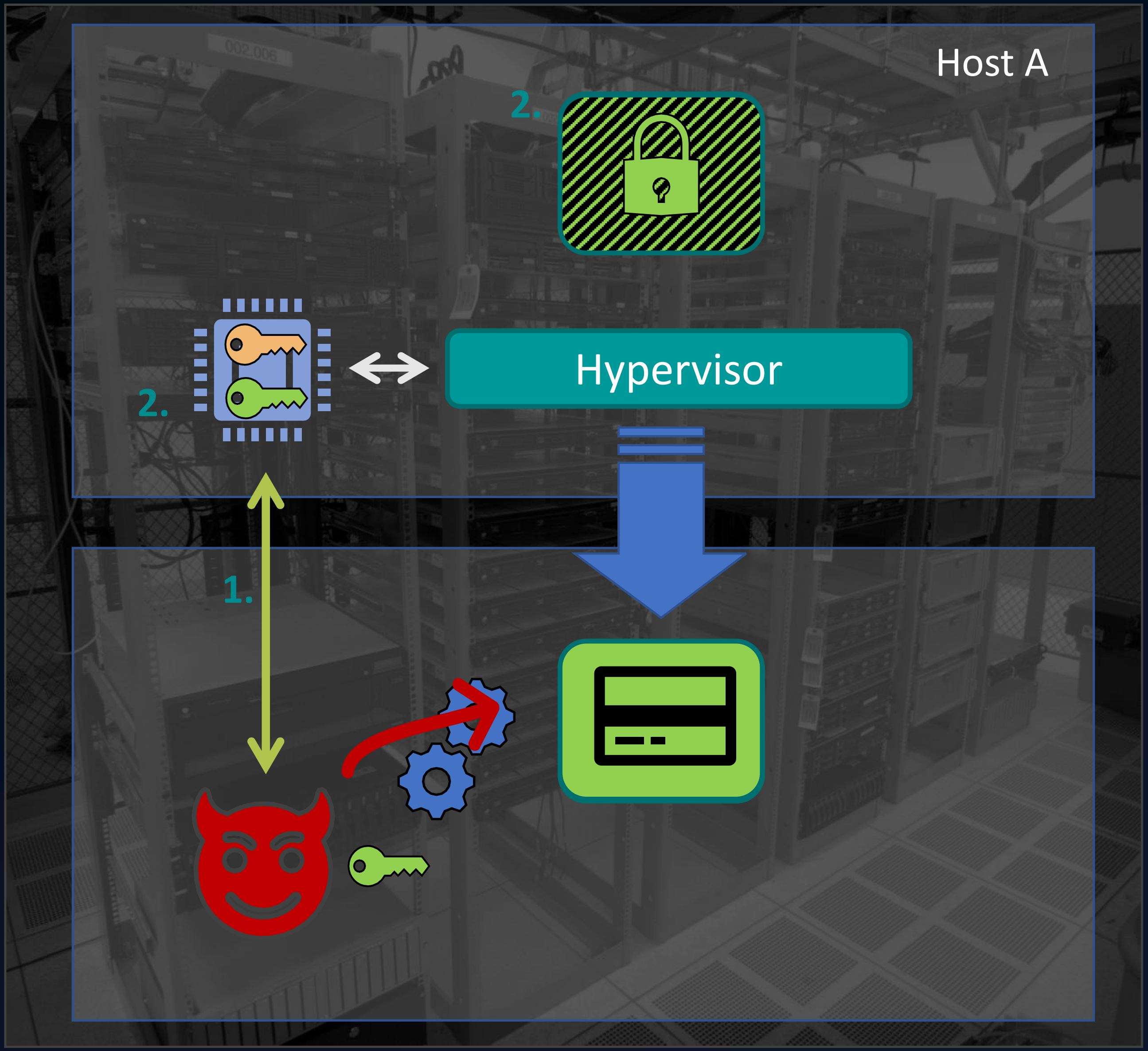
SEV MIGRATION

1. Establish secure channel to target secure processor.
2. **Derive shared transport keys** & re-encrypt VM using transport keys.
3. Transfer VM.
4. Re-encrypt VM using fresh key.



SEV MIGRATION

1. Establish secure channel to target secure processor.
2. **<u>Derive shared transport keys</u>** & re-encrypt VM using transport keys.
3. Transfer VM.
4. Re-encrypt VM using fresh key.



Any valid CEK is sufficient.

The target host does **not** need to be vulnerable.

The attacker does **not** need physical access.

A guest owner can configure a VM as “non-migratable”

PDH->CEK->ARK

Host A

Hypervisor

SEV MIGRATION

1. Establish secure channel ... secure processor.
2. **Derive shared transport keys** & re-encrypt VM using transport keys.
3. Transfer VM.
4. Re-encrypt VM using fresh key.

MITIGATIONS

No roll-back prevention!

-

A malicious cloud provider can **always** install a vulnerable firmware version.

A previously extracted CEK is still valid after a firmware update!

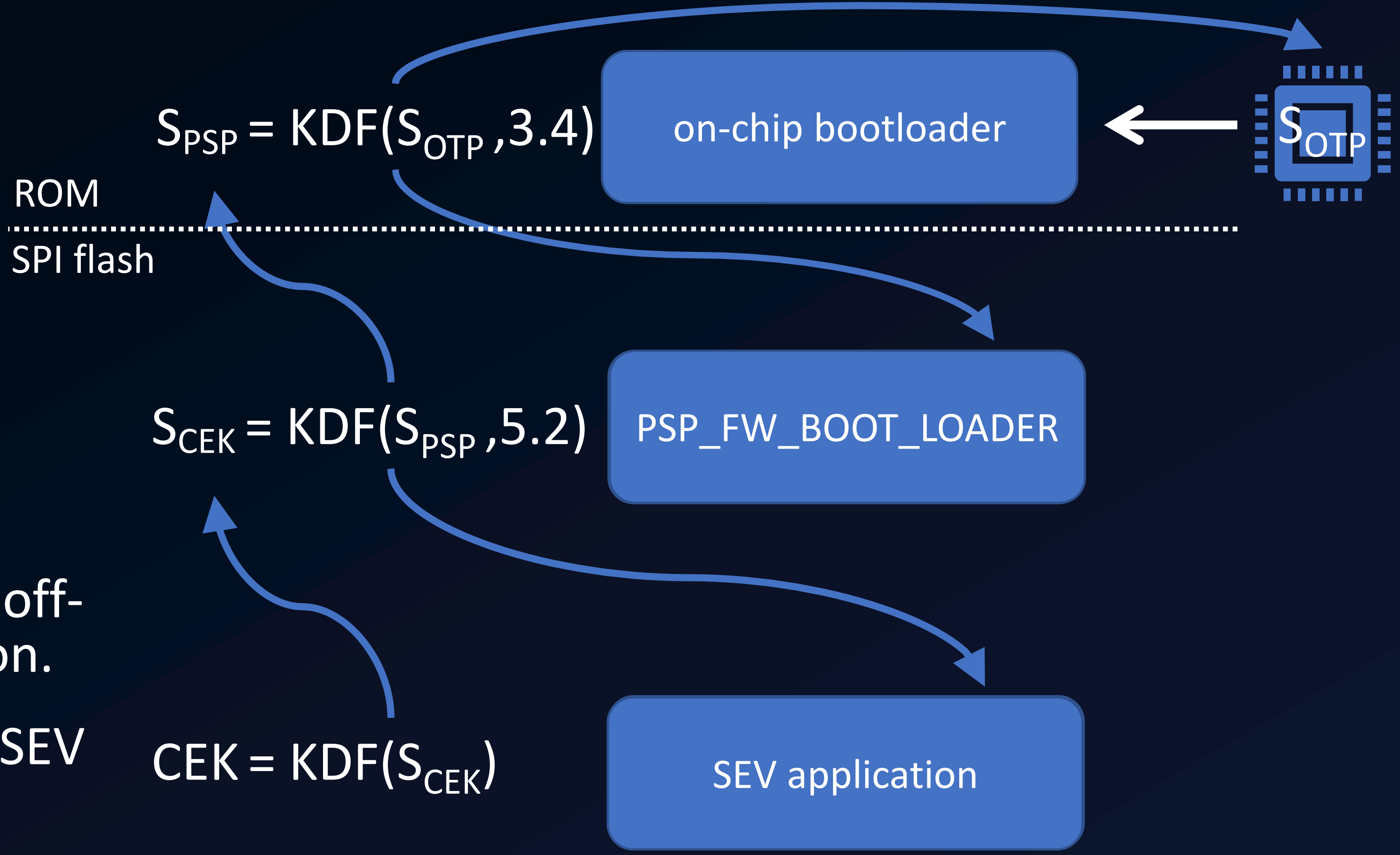
FIRMWARE ANALYSIS

1. The off-chip bootloader uses the ARK to verify applications signatures.
2. Firmware issues allow us to provide our own signing key for applications.

CEK lifetime:

Chapter 2.1.3 AMD SEV API Specification:

"**It exists for the lifetime of the platform** and is stored within the hardware of the AMD Secure Processor"



PROPOSED DESIGN

1. S_{PSP} based on S_{OTP} and off-chip bootloader version.
2. S_{CEK} based on S_{PSP} and SEV FW version.
3. CEK based on S_{CEK}

There exists a valid CEK for every firmware combination of a platform.

The lifetime of a CEK is limited to the lifetime of the firmware components.

A previously extracted CEK is <u>NOT</u> valid after a firmware update!

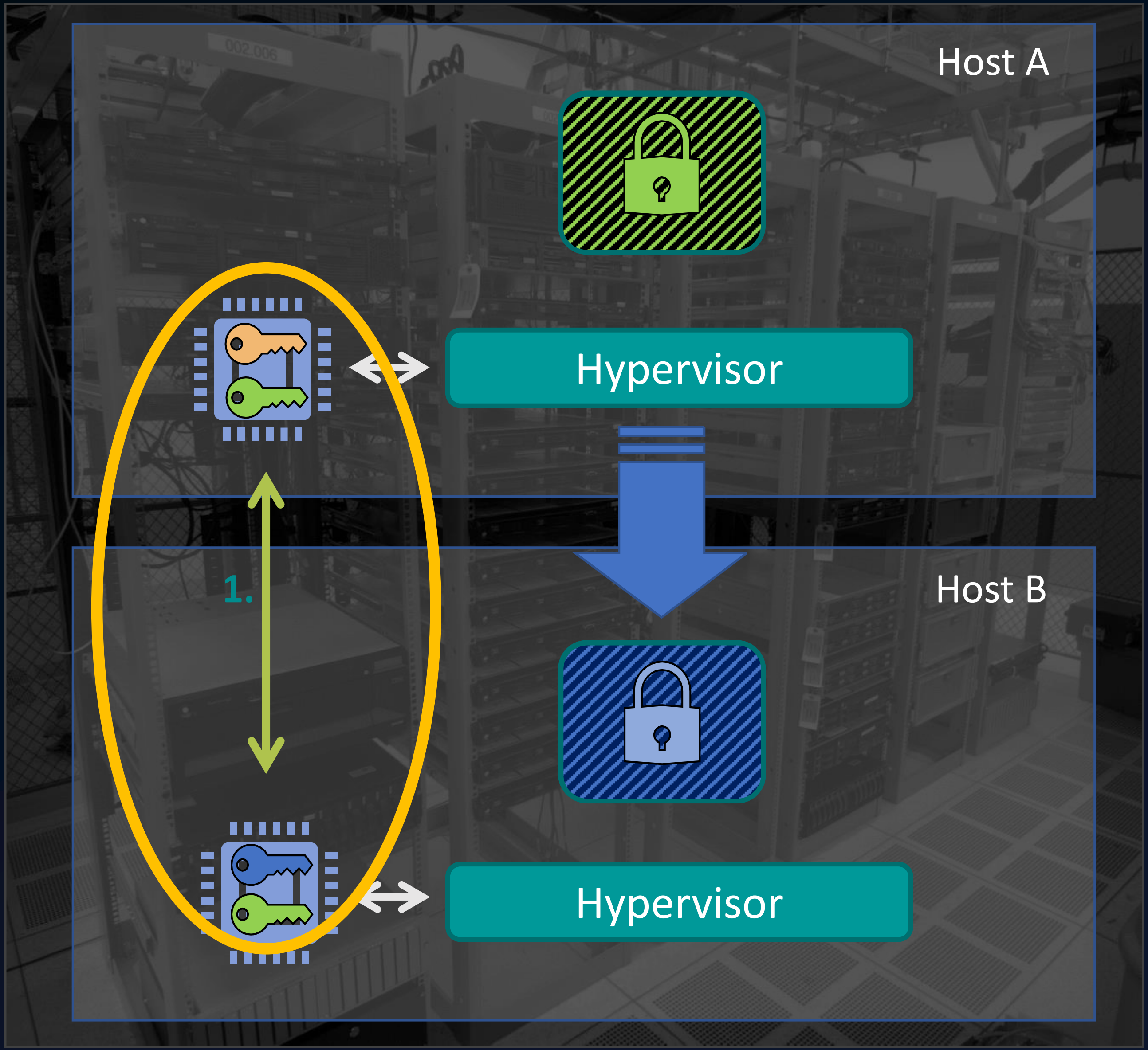
PROPOSED DESIGN

1. S_{PSP} based on S_{OTP} and off-chip bootloader version.
2. S_{CEK} based on S_{PSP} and SEV FW version.
3. CEK based on S_{CEK}

PDH->CEK(FW VER.)->ARK

SEV MIGRATION

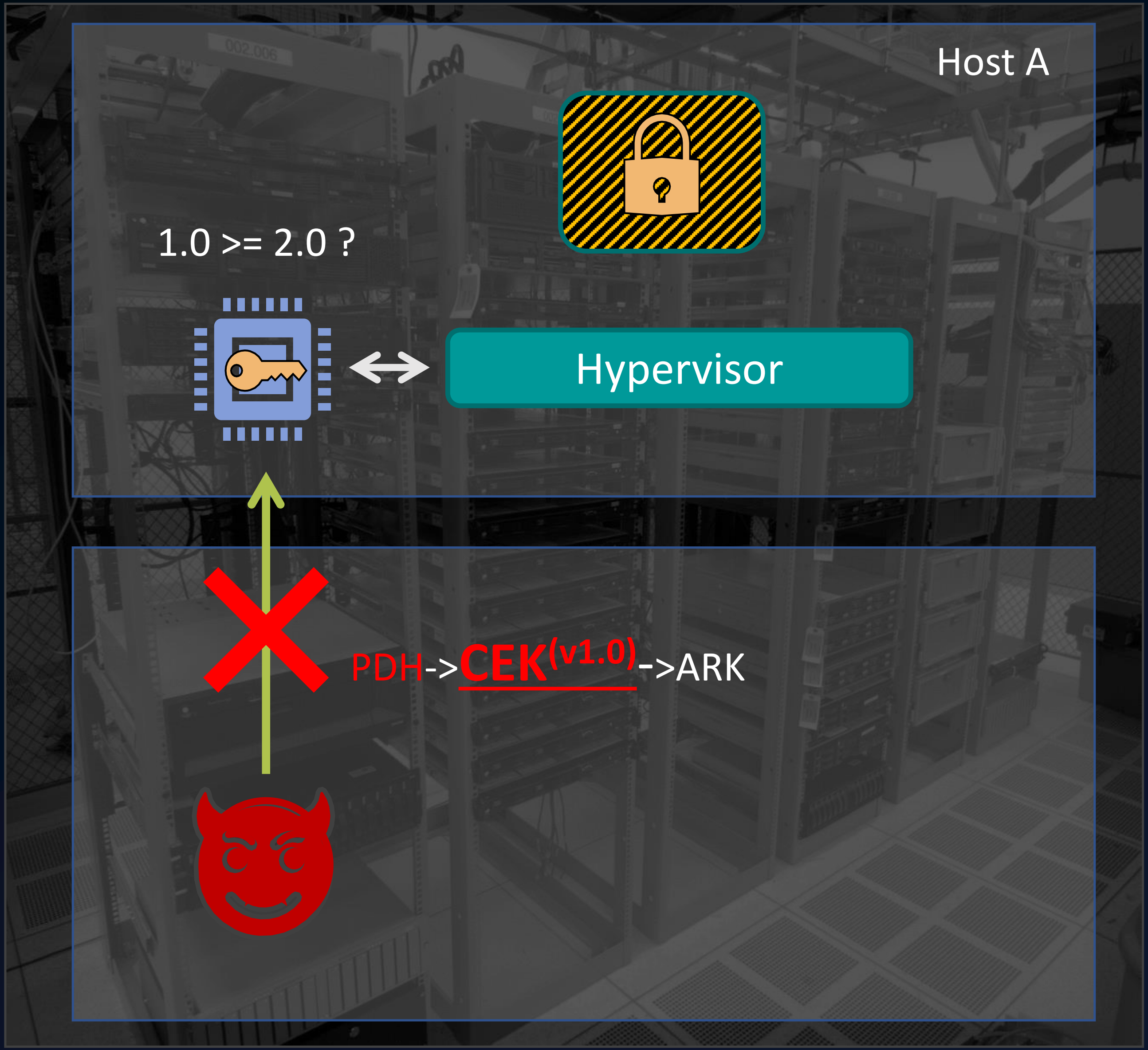
- The source secure processor can enforce minimum version requirements before accepting a provided CEK.

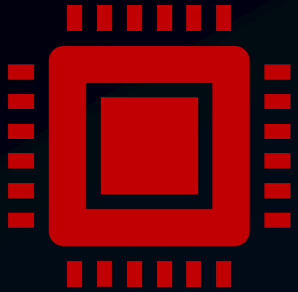


PDH->CEK(2.0)->ARK

SEV MIGRATION

- The source secure processor can enforce minimum version requirements before accepting a provided CEK.



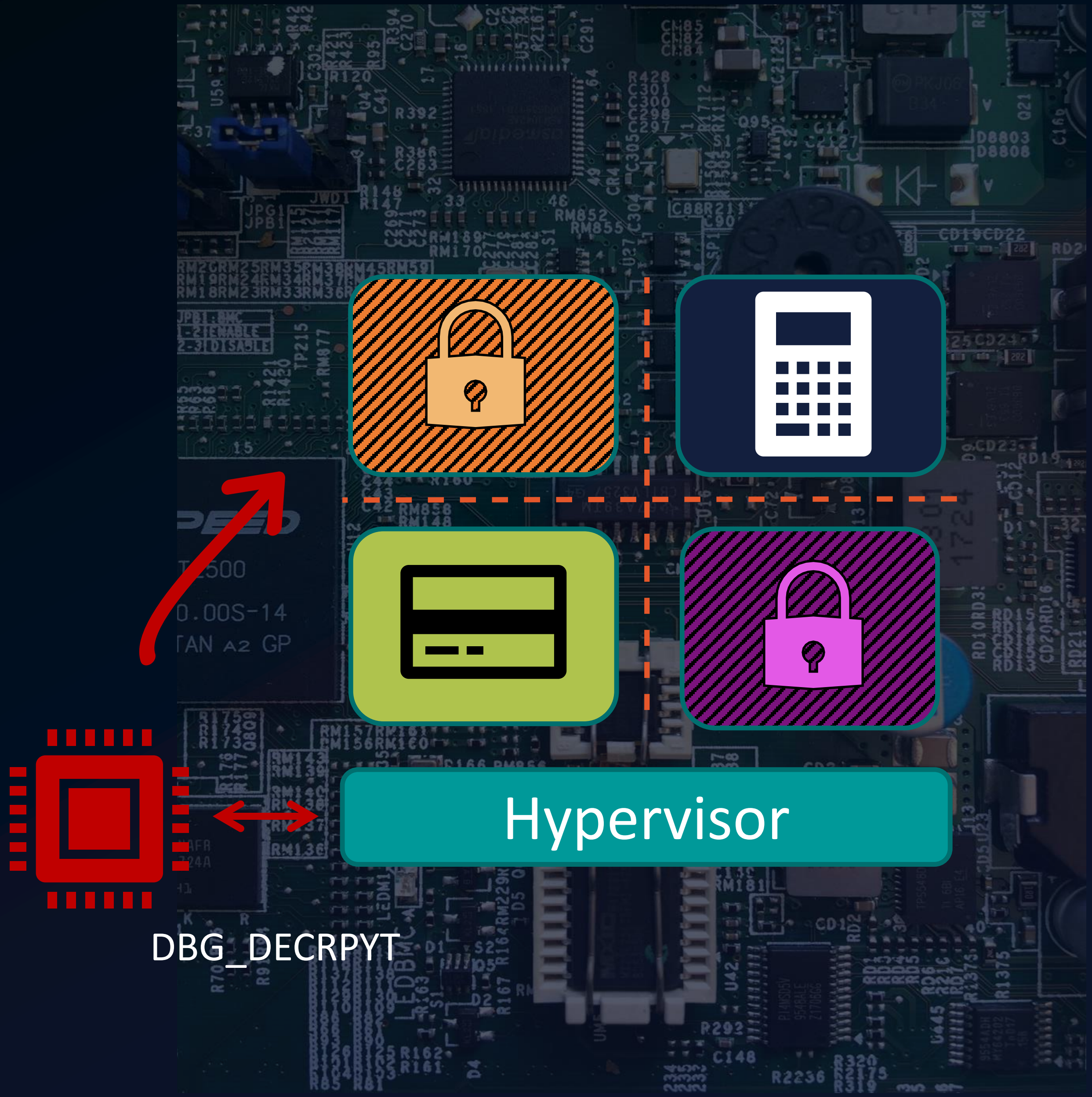


ATTACKS

Chapter 7 AMD SEV Specification “Debugging API”:

- DBG_DECRYPT
- DBG_ENCRYPT

An attacker-controlled FW can override guest security policies.

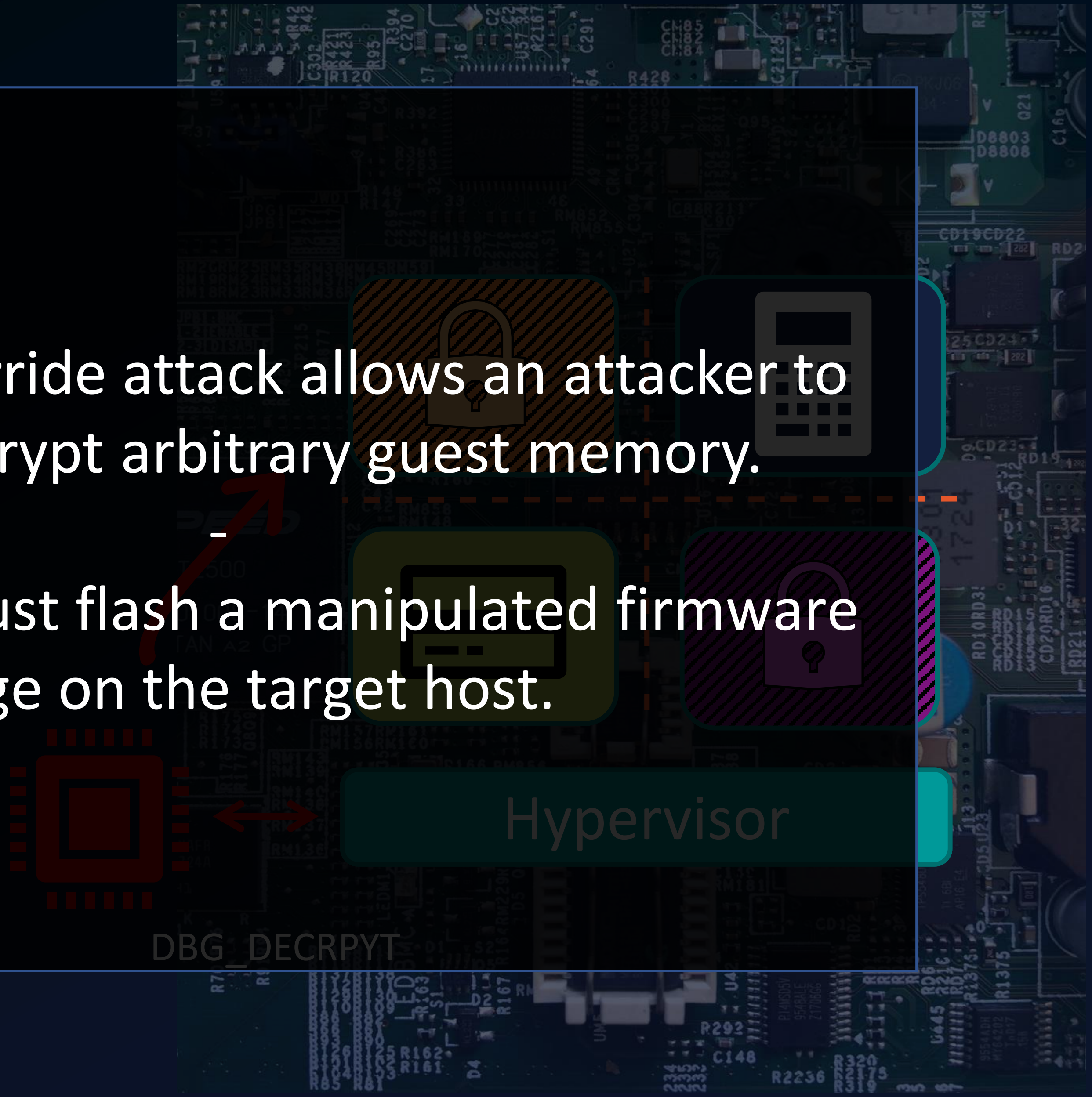


ATTACKS

Chapter 7 AMD SEV Specification “Debugging API”:

- DBG_DECRYPT
- DBG_ENCRYPT

An attacker-controlled FW can override guest security policies.



The debug override attack allows an attacker to decrypt/encrypt arbitrary guest memory.

-

The attacker must flash a manipulated firmware image on the target host.

SUMMARY

- Firmware issues allow us to extract the CEK.
  - Missing roll-back prevention and the longevity of the CEK thwart software-based fixes.
- Attacks are possible even if the target host is free of any vulnerability.

➡ The current SEV design cannot cope with firmware issues.

- We proposed design changes that bind the CEK to specific firmware versions.
  - The proposed changes allow to reassure trust in the SEV technology in case of KNOWN firmware issues.

RESOURCES

https://github.com/RobertBuhren/amd-sev-migration-attack

- Proof-of-concept implementation of the migration attack.

https://github.com/RobertBuhren/Insecure-Until-Proven-Updated-Analyzing-AMD-SEV-s-Remote-Attestation

- Proof-of-concept signature created with an extracted CEK.

https://github.com/PSPReverse

- psptool & psptrace & PSPEmulator etc…

https://lsseu2019.sched.com/event/TynP/upcoming-x86-technologies-for-malicious-hypervisor-protection-david-kaplan-amd

- AMD SEV-SNP Talk at the Linux Security Summit 2019.

THANK YOU

Robert Buhren

robert.buhren@sect.tu-berlin.de

https://sect.tu-berlin.de

Security in Telecommunications