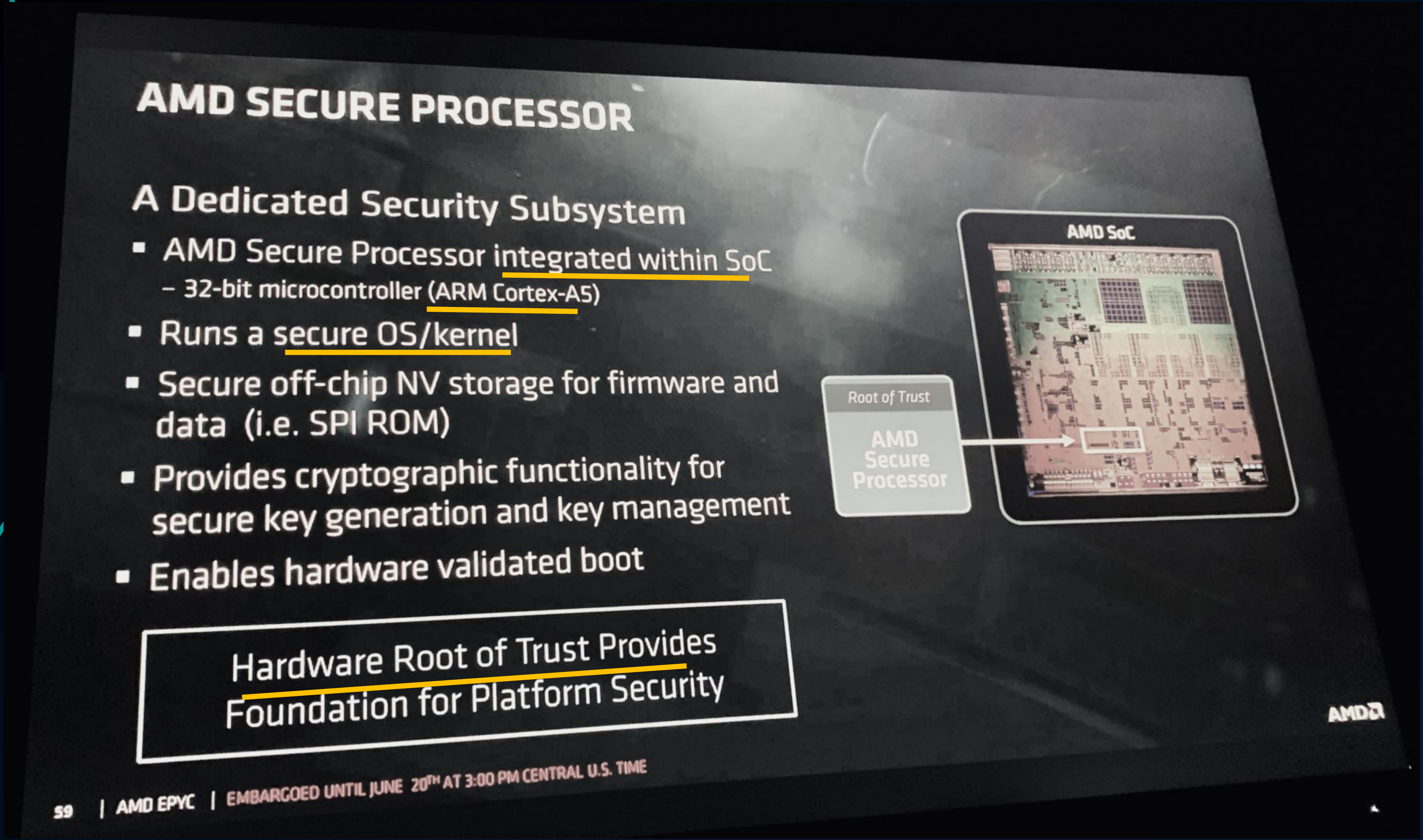
One Glitch to Rule Them All: Fault Injection Attacks against AMD's Secure Processor

Robert Buhren
Hans Niklas Jacob

Technische Universität Berlin

BLACKHAT EUROPE 2021

Technische Universität Berlin



[1] Formerly known as *Platform Security Processor (i.e.* ***PSP****)*

Applications

SECURE ENCRYPTED VIRTUALIZATION (EPYC)

- **SEV** protects virtual machines in **untrusted** environments by encrypting VM memory
- The AMD SP is responsible for key management
- Paper: “Insecure Until Proven Updated: Analyzing AMD SEV's Remote Attestation”

SECURE OS (RYZEN / TR)

- Firmware **TPM**
- …

TRUSTED EXECUTION ENVIRONMENT

- AMD SP Trusted Execution Environment
- Linux to support **AMD SP TEE API**

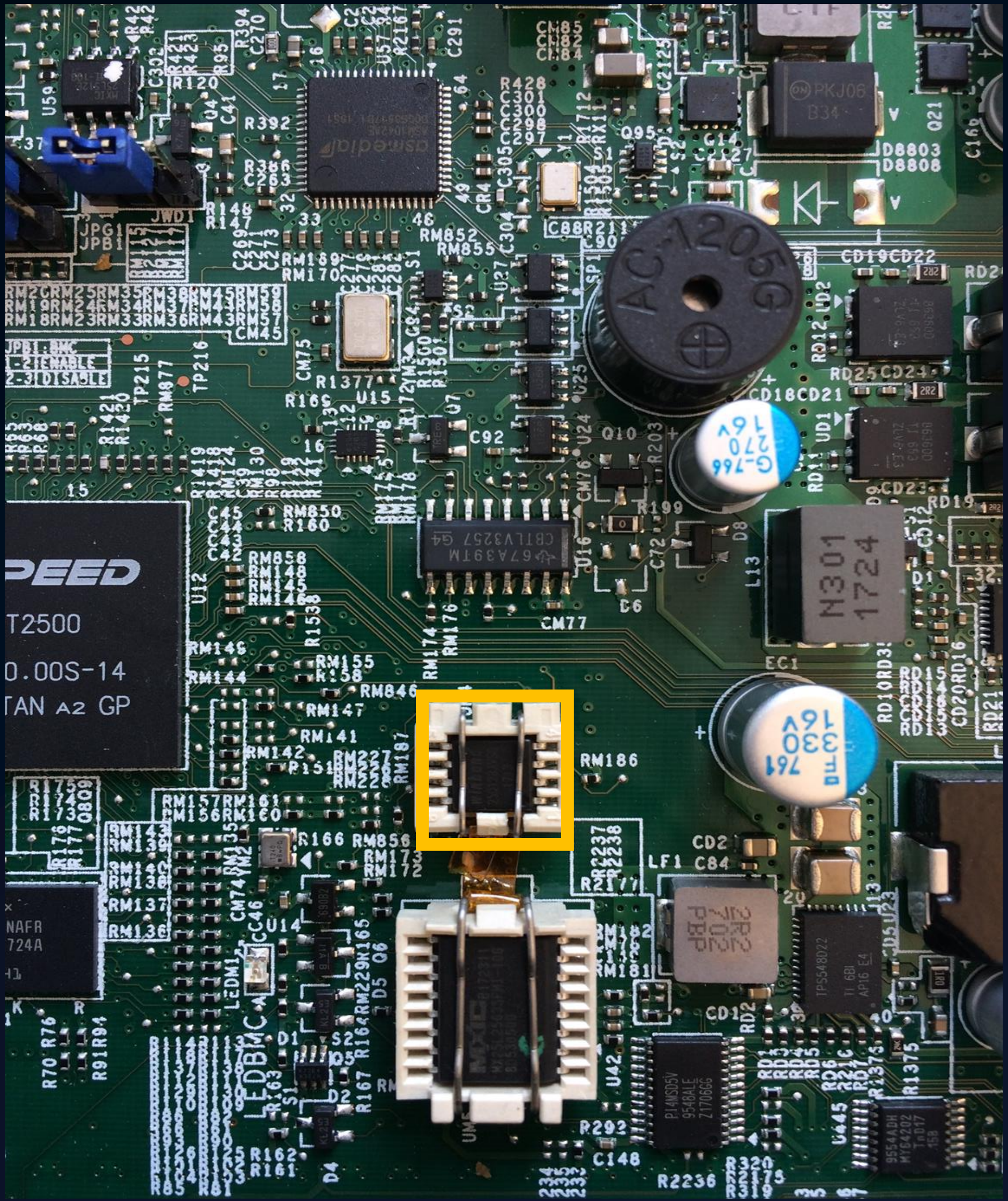
FIRMWARE ANALYSIS

Secure Processor is part of AMD CPU.

- ARMv7-A

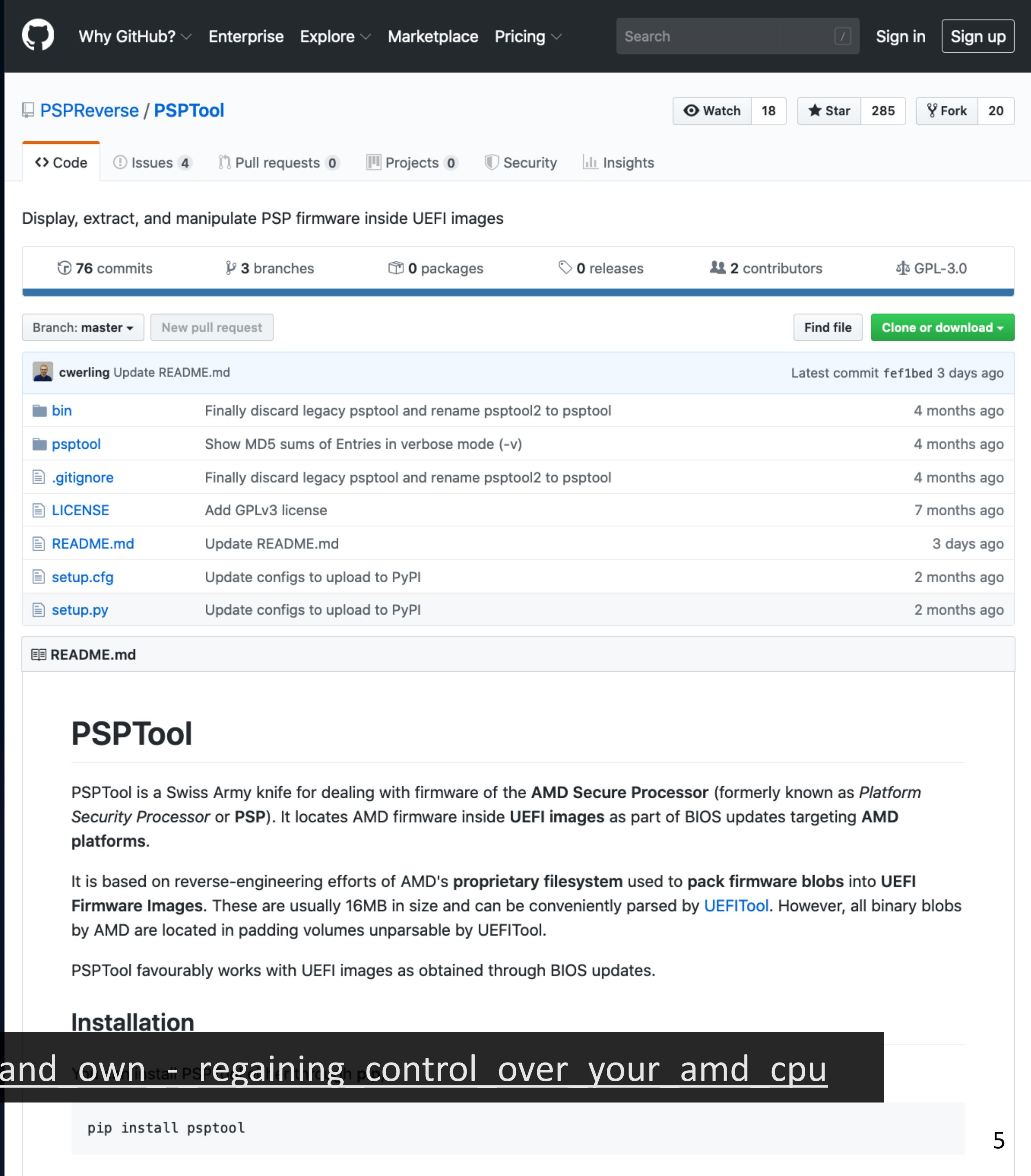
Firmware is stored along UEFI FW!

Updatable through UEFI update.



PSPTOOL

- Python-based
- Command-line interface
- Parsing
- Extraction
- Manipulation
- Decompression
- Signature verification
- PEM export of keys
- Duplicate detection
- Signature update
- Python API
- GPLv3



https://media.ccc.de/v/36c3-10942-uncover_understand_own_-_regaining_control_over_your_amd_cpu

https://github.com/PSPReverse/PSPTool

PSPTOO

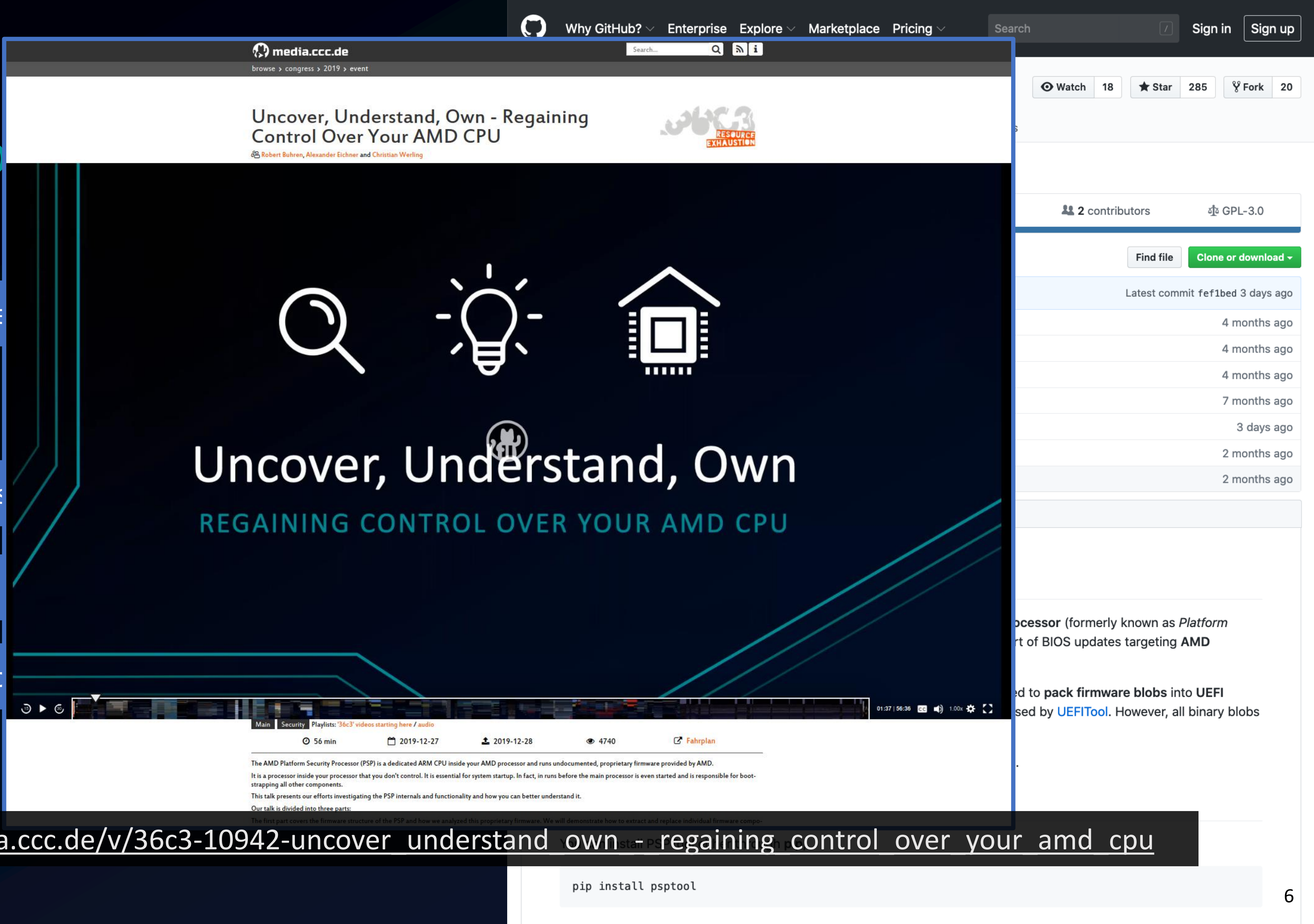
Python-base

Parsing

Decompress

PEM export

Signature up



https://media.ccc.de/v/36c3-10942-uncover_understand_own_-_regaining_control_over_your_amd_cpu

https://github.com/PSPReverse/PSPTool

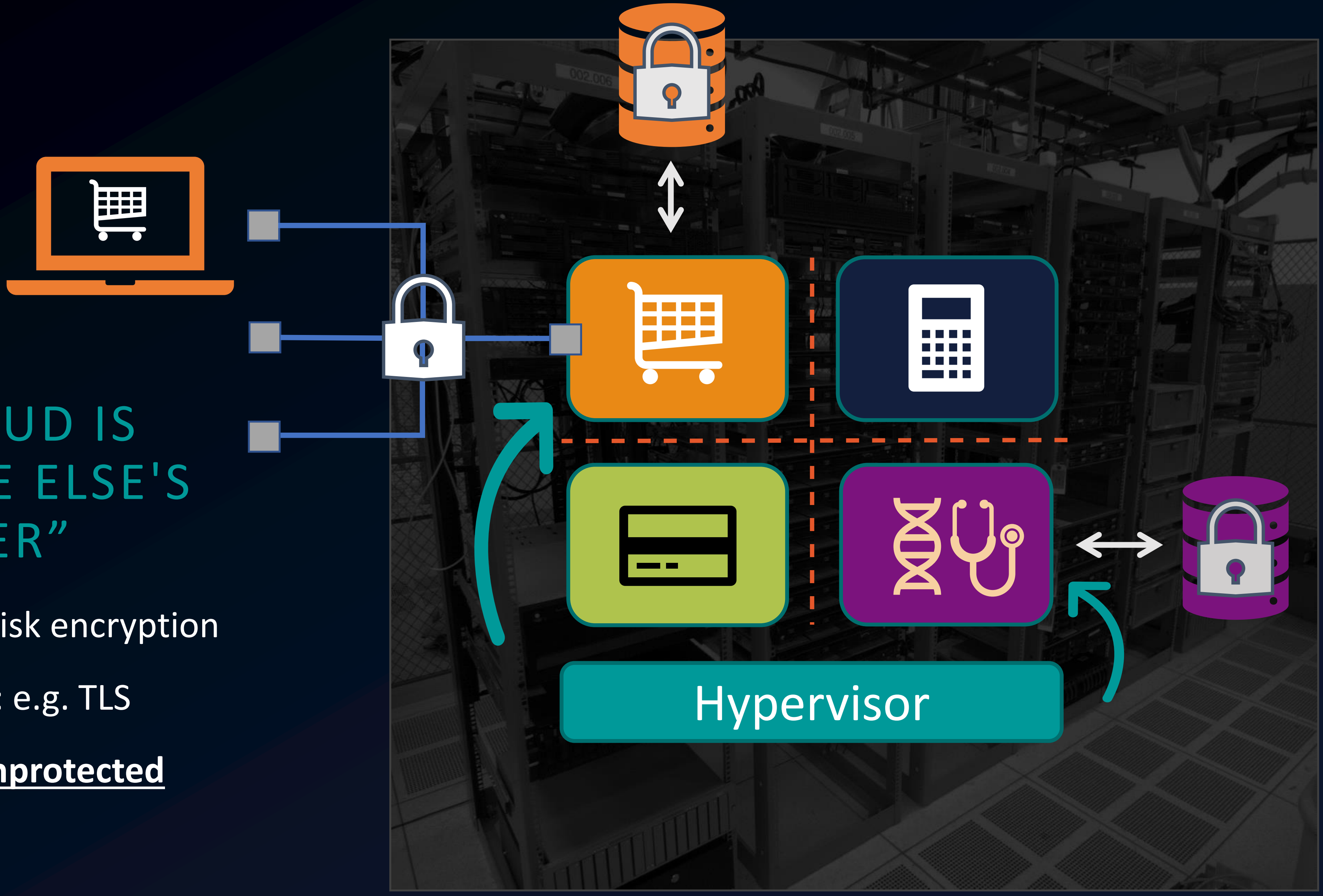
AMD
Secure Encrypted Virtualization (SEV)

“THE CLOUD IS SOMEONE ELSE'S COMPUTER”

Data-At-Rest: disk encryption

Data-In-Transit: e.g. TLS

Data-In-Use: **unprotected**

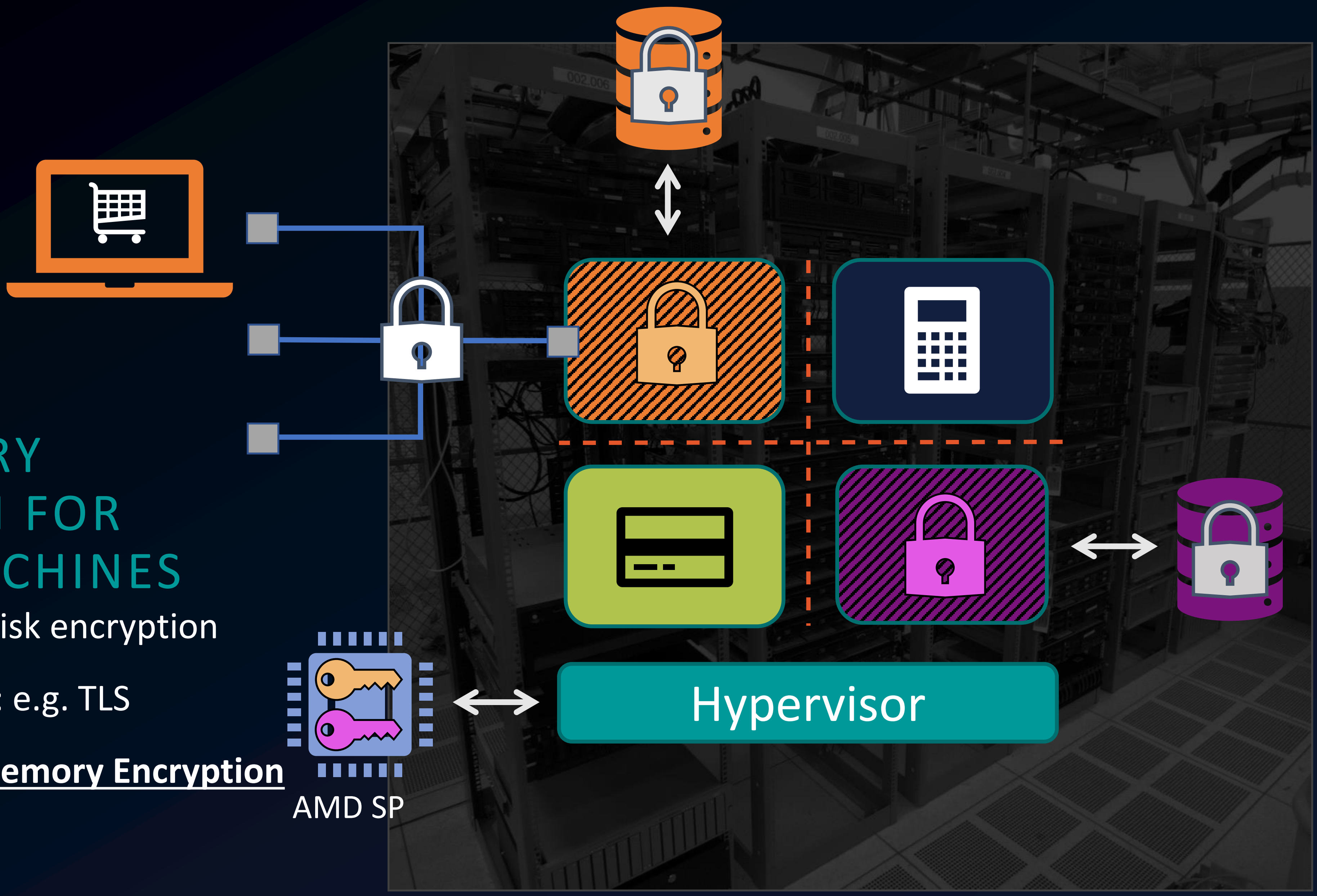


SEV: MEMORY ENCRYPTION FOR VIRTUAL MACHINES

Data-At-Rest: disk encryption

Data-In-Transit: e.g. TLS

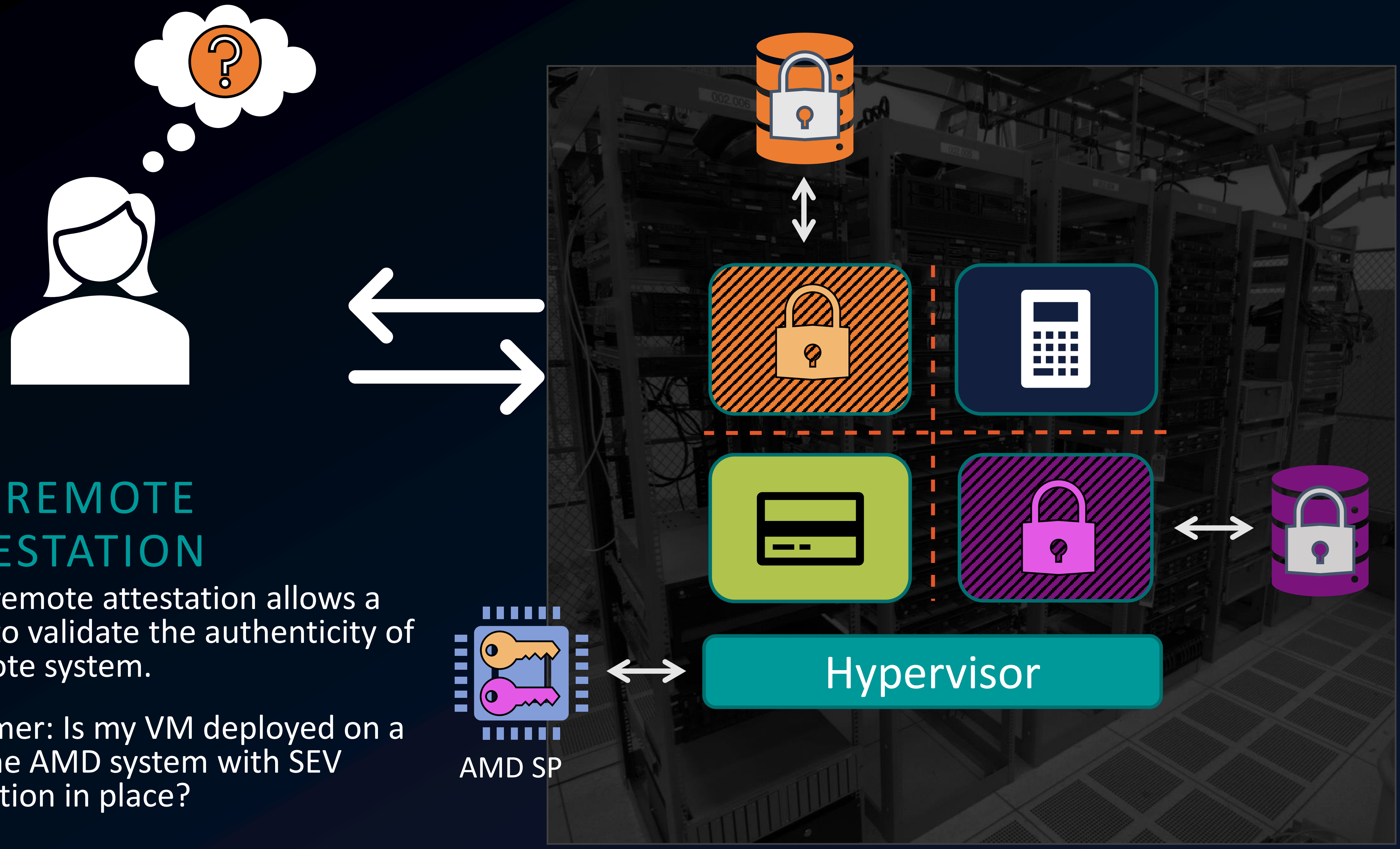
Data-In-Use: **Memory Encryption** (AES-128)



SEV REMOTE ATTESTATION

SEV's remote attestation allows a party to validate the authenticity of a remote system.

Customer: Is my VM deployed on a genuine AMD system with SEV protection in place?

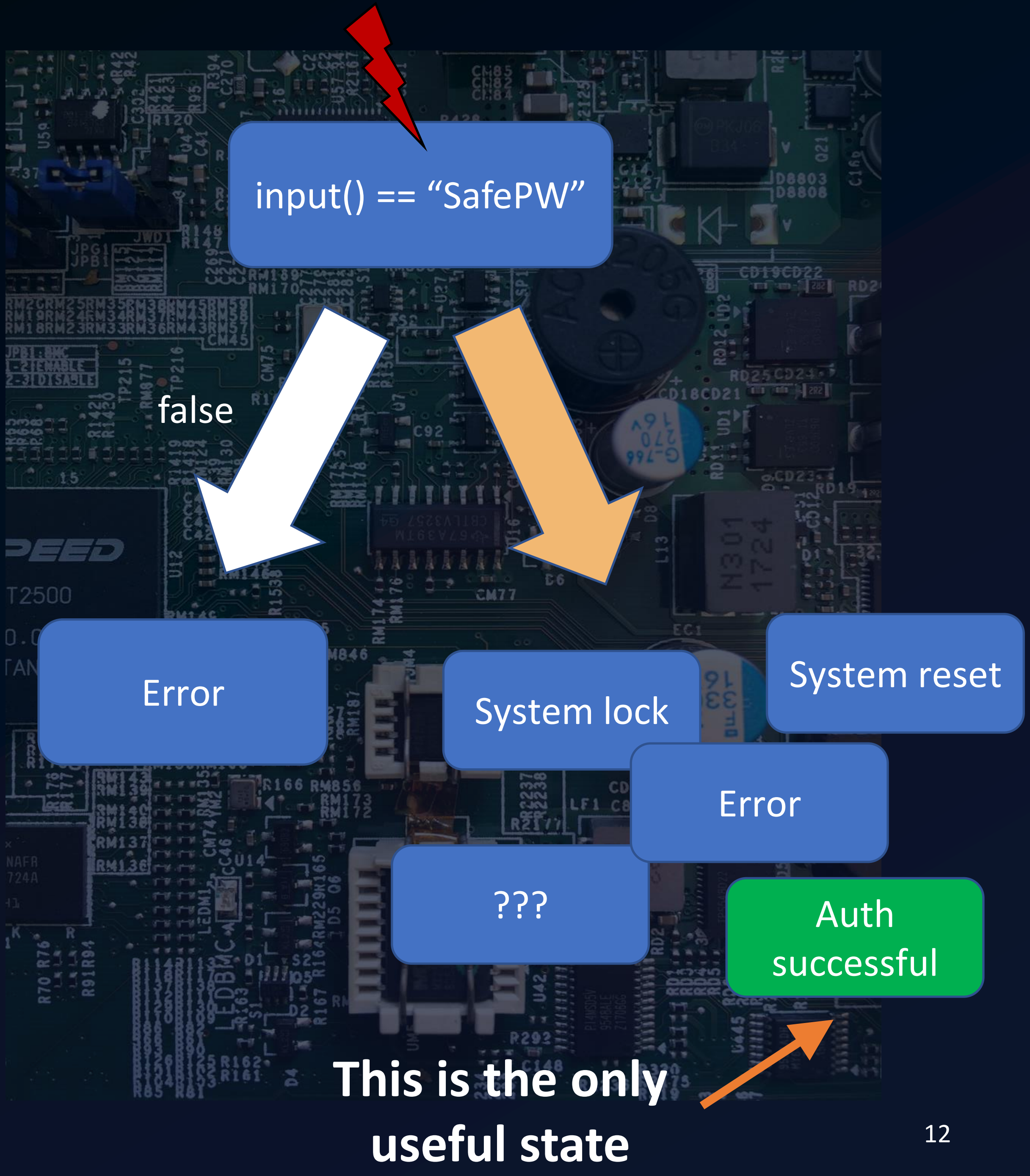


Faulting the AMD-SP

FAULT INJECTION ATTACKS

Modifications of an ICs environment can cause errors in the ICs operation

- Lower voltage rails
  → Voltage fault injection
- Hit IC with electro magnetic radiation
  → EM fault injection
- Hit IC with laser
  → Laser fault injection …
- Most faults are useless for an attacker



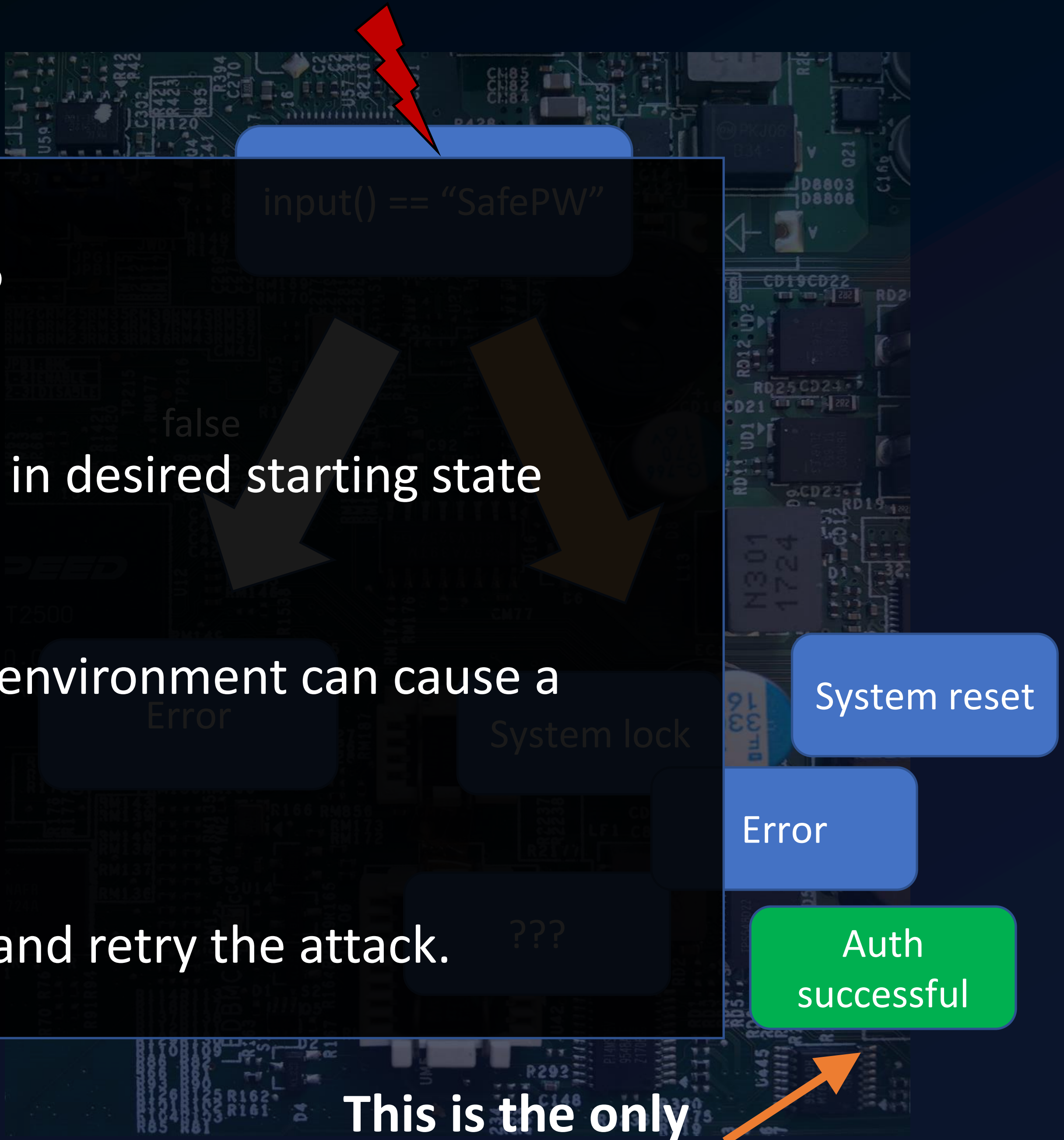
FAULT INJECTION ATTACKS

Modifications of an ICs environment can cause errors in the ICs operation

- Lower voltage rails
  → Voltage fault injection
- Hit IC with electro magnetic radiation
  → EM fault injection
- Hit IC with laser
  → Laser fault injection …
- Most faults are useless for an attacker

Key Challenges

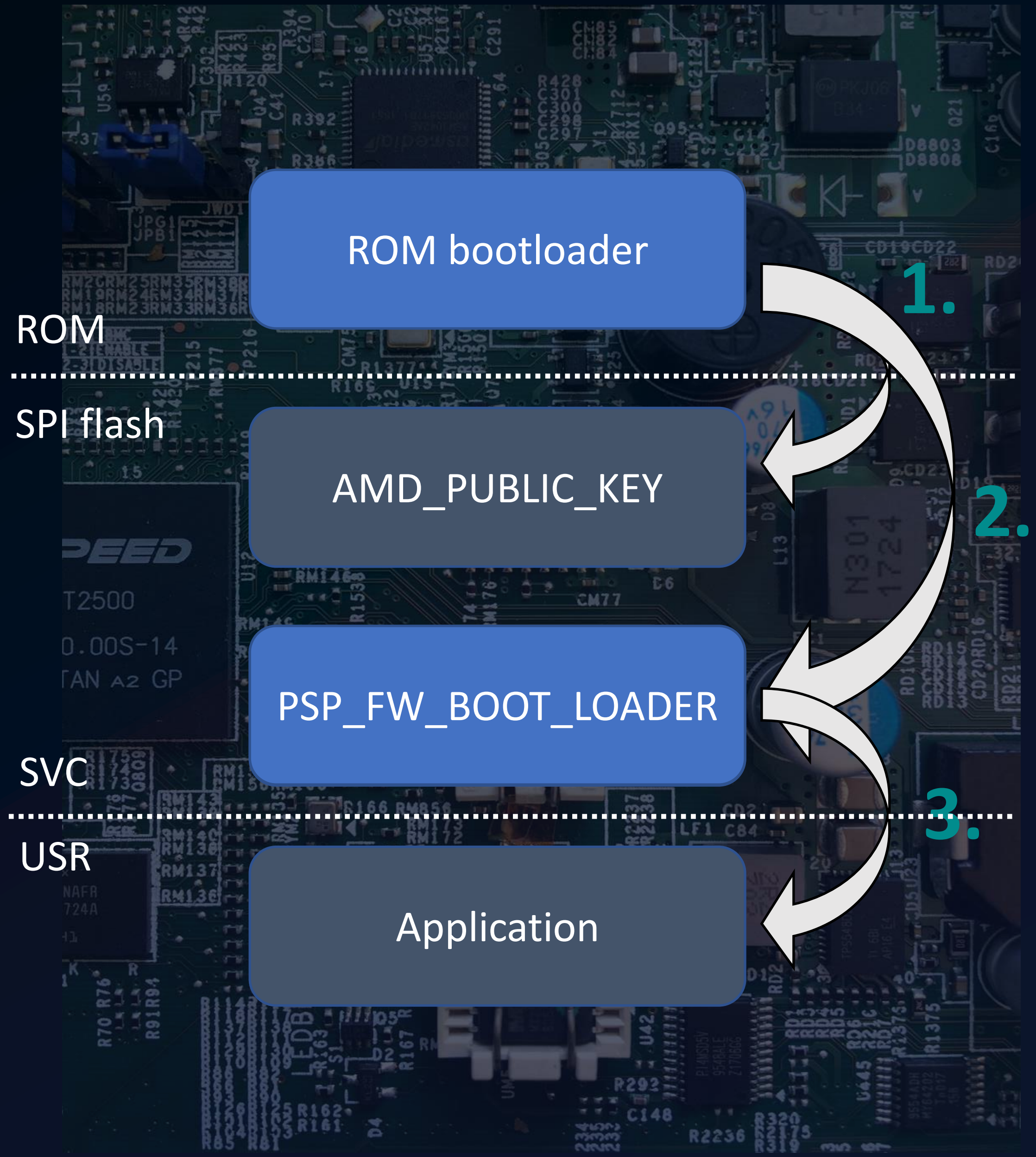
- **Trigger:**
  Identify when the IC is in desired starting state
- **Parameters:**
  Which changes to the environment can cause a useful fault
- **Reset/success:**
  Identify failed attacks and retry the attack.

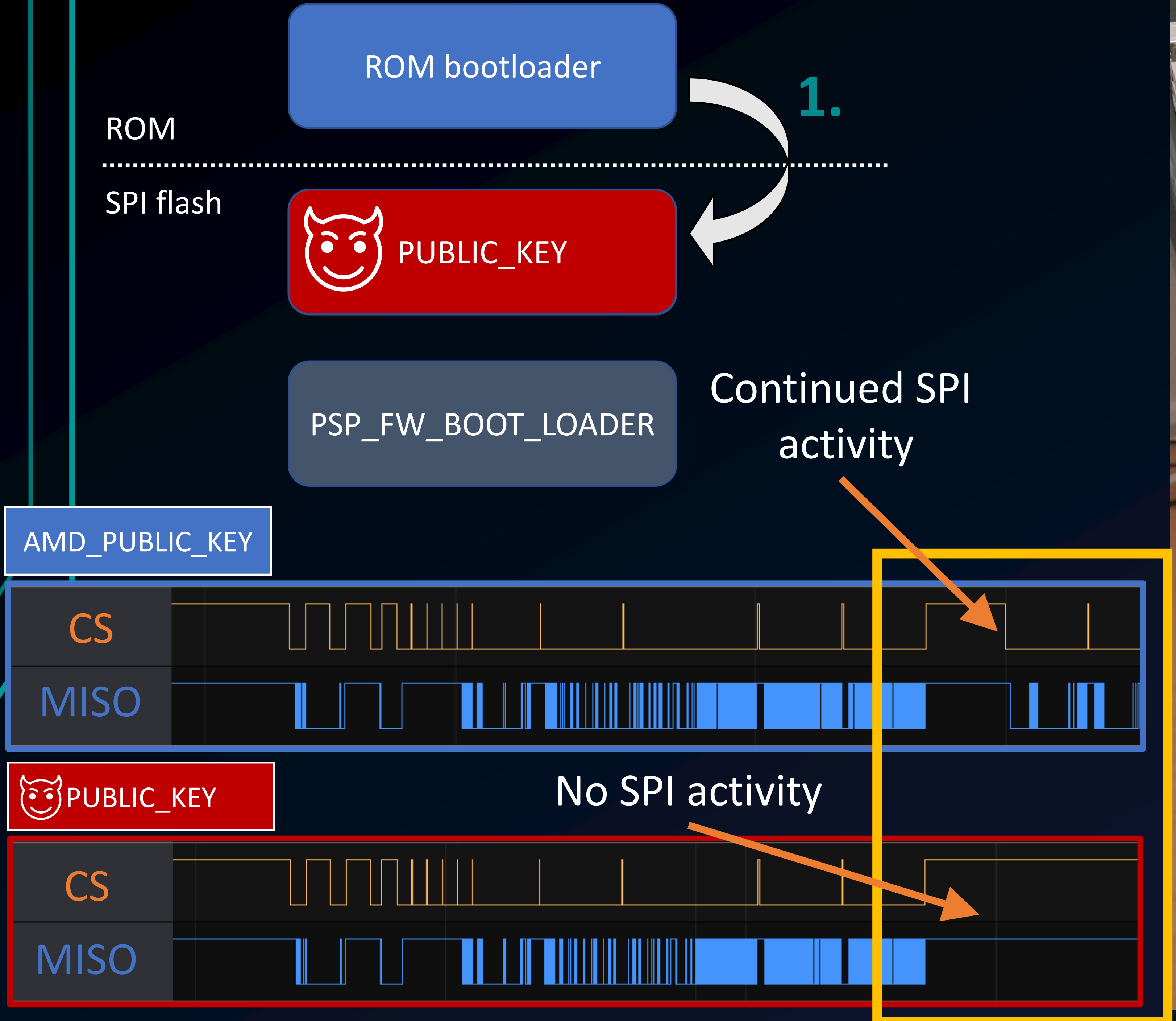


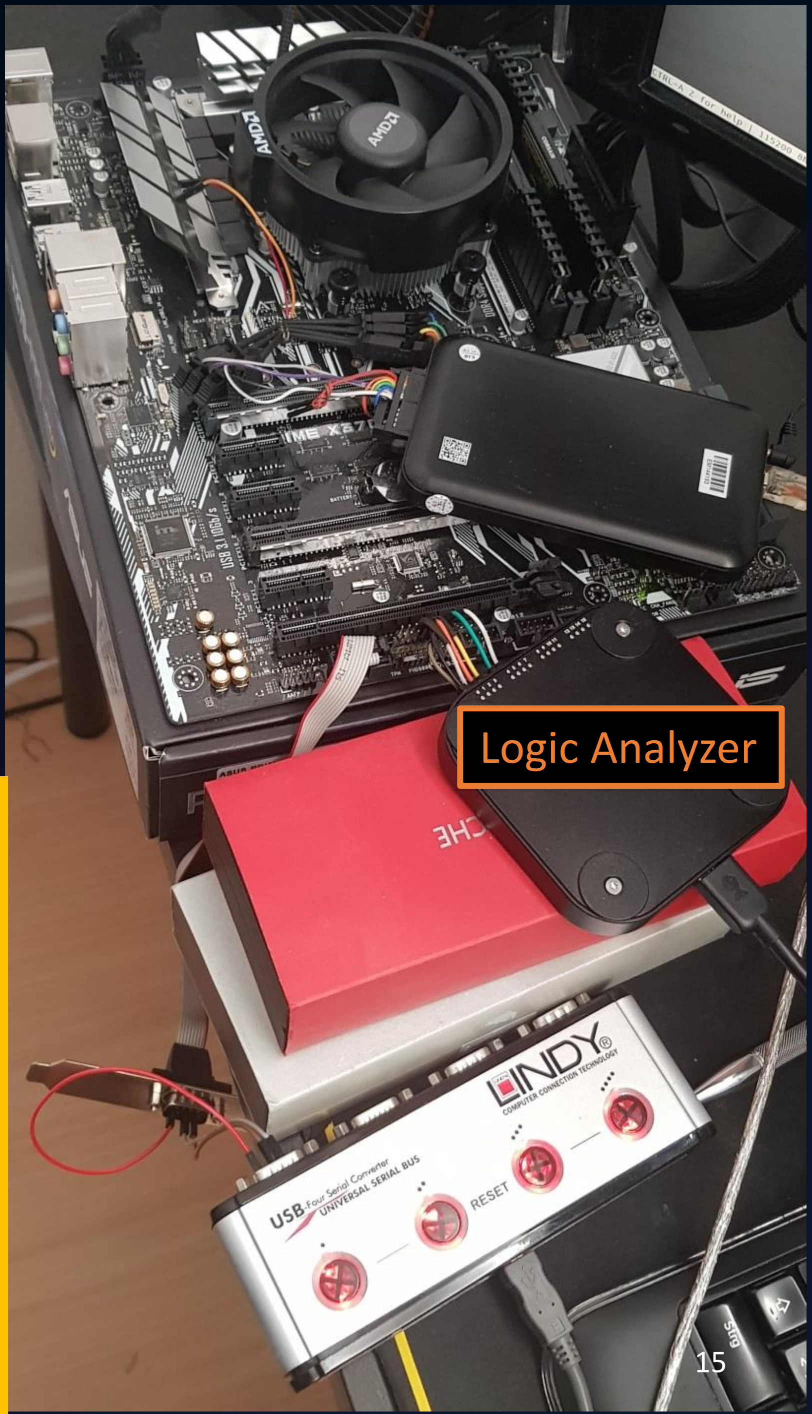
This is the only useful state

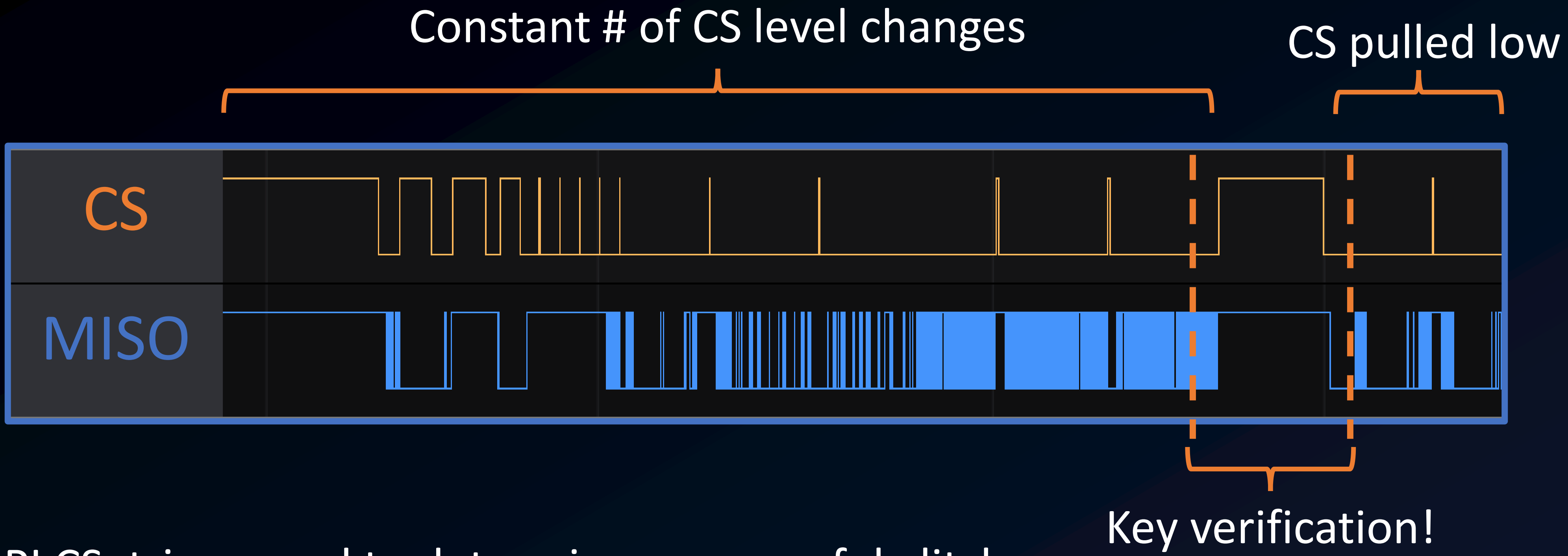
AMD-SP BOOT

1. Load & verify AMD_PUBLIC_KEY
   - verify using hash
2. Load & verify PSP_FW_BOOT_LOADER
   - verify using public key
3. Load & verify additional applications
   - verify using public key




ROM bootloader
1.
ROM
SPI flash
PUBLIC_KEY
PSP_FW_BOOT_LOADER
Continued SPI activity
AMD_PUBLIC_KEY
CS
MISO
PUBLIC_KEY
No SPI activity
CS
MISO


Logic Analyzer

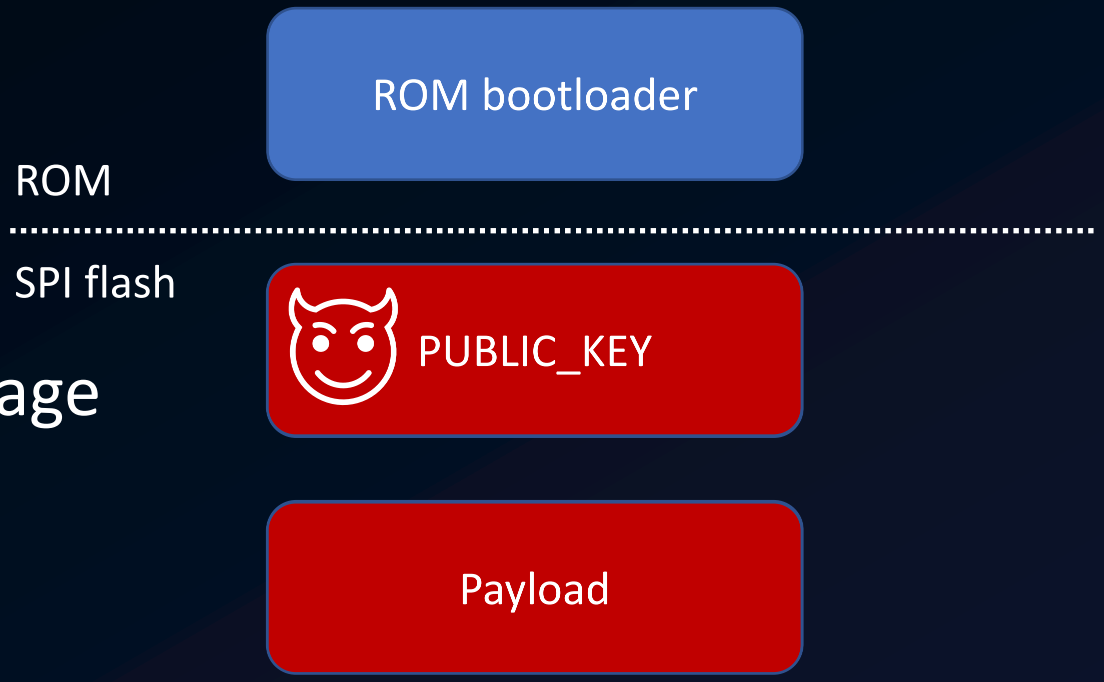


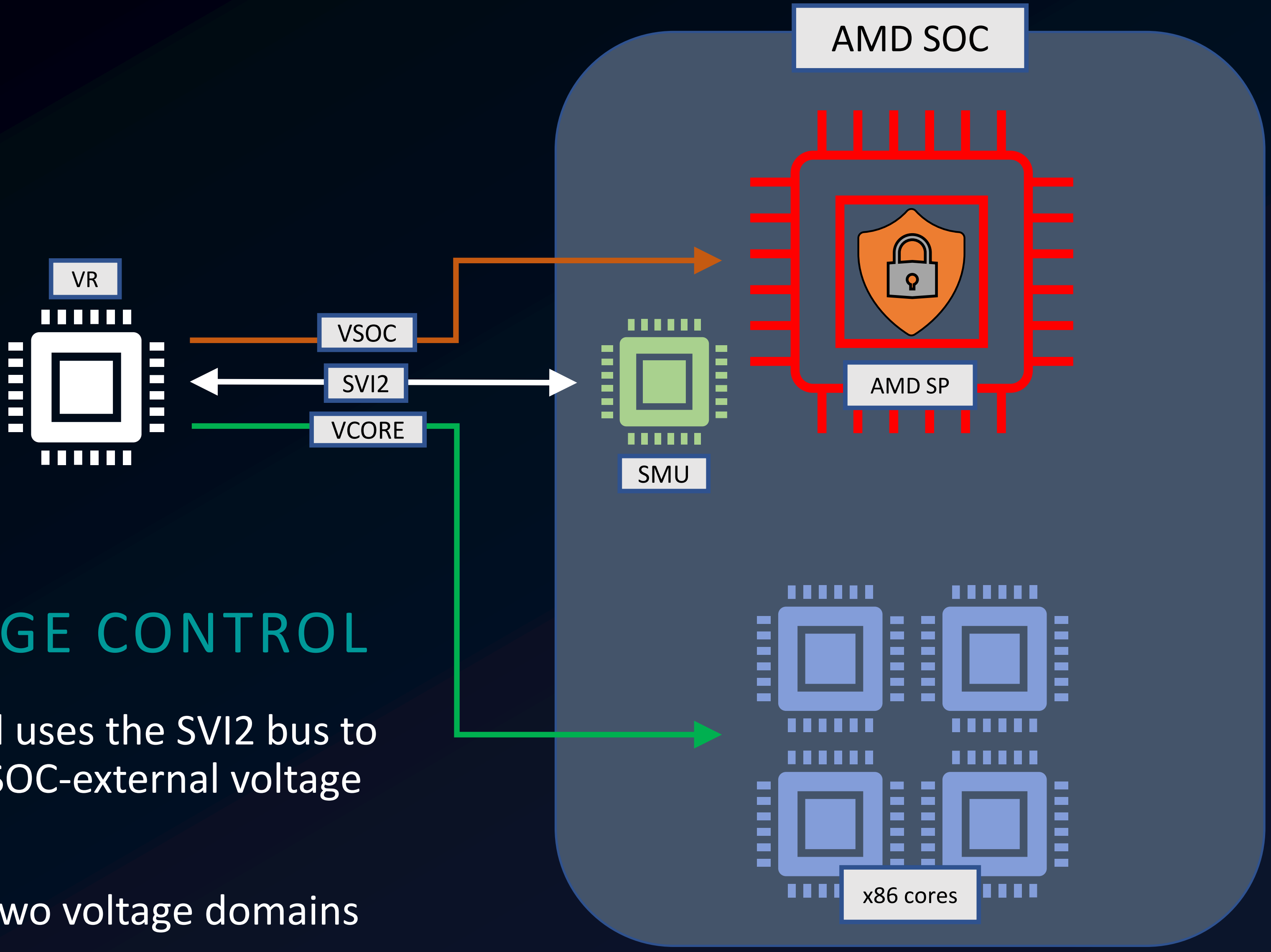
SPI CS: trigger and to determine a successful glitch

ATTACK OVERVIEW

Our goal is to execute our payloads right after the ROM bootloader.

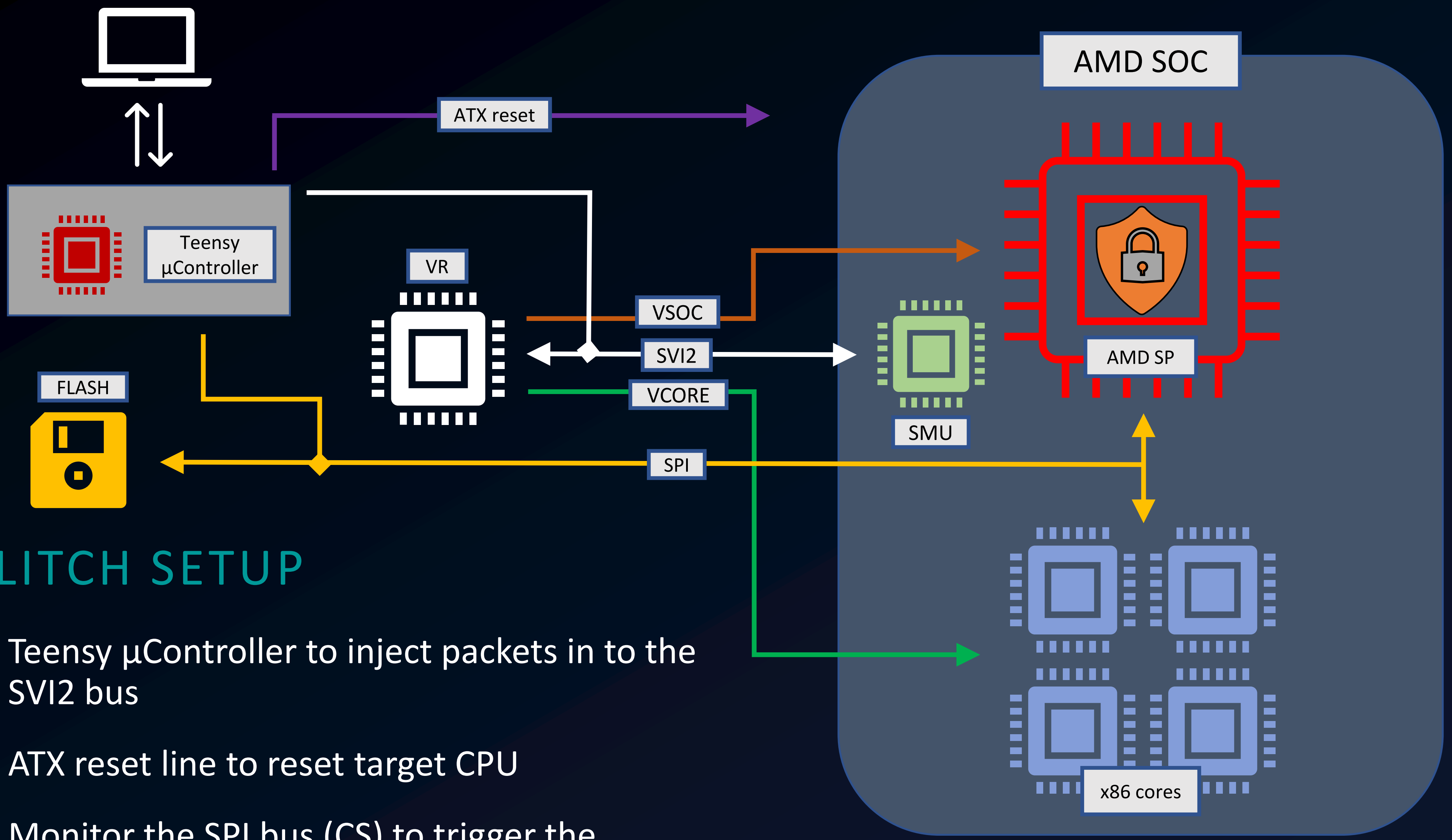
1. Replace AMD_PUBLIC_KEY in UEFI image
2. Replace PSP_FW_BOOT_LOADER component with payload
3. Sign payload with custom key
4. Glitch key verification





DYNAMIC VOLTAGE CONTROL

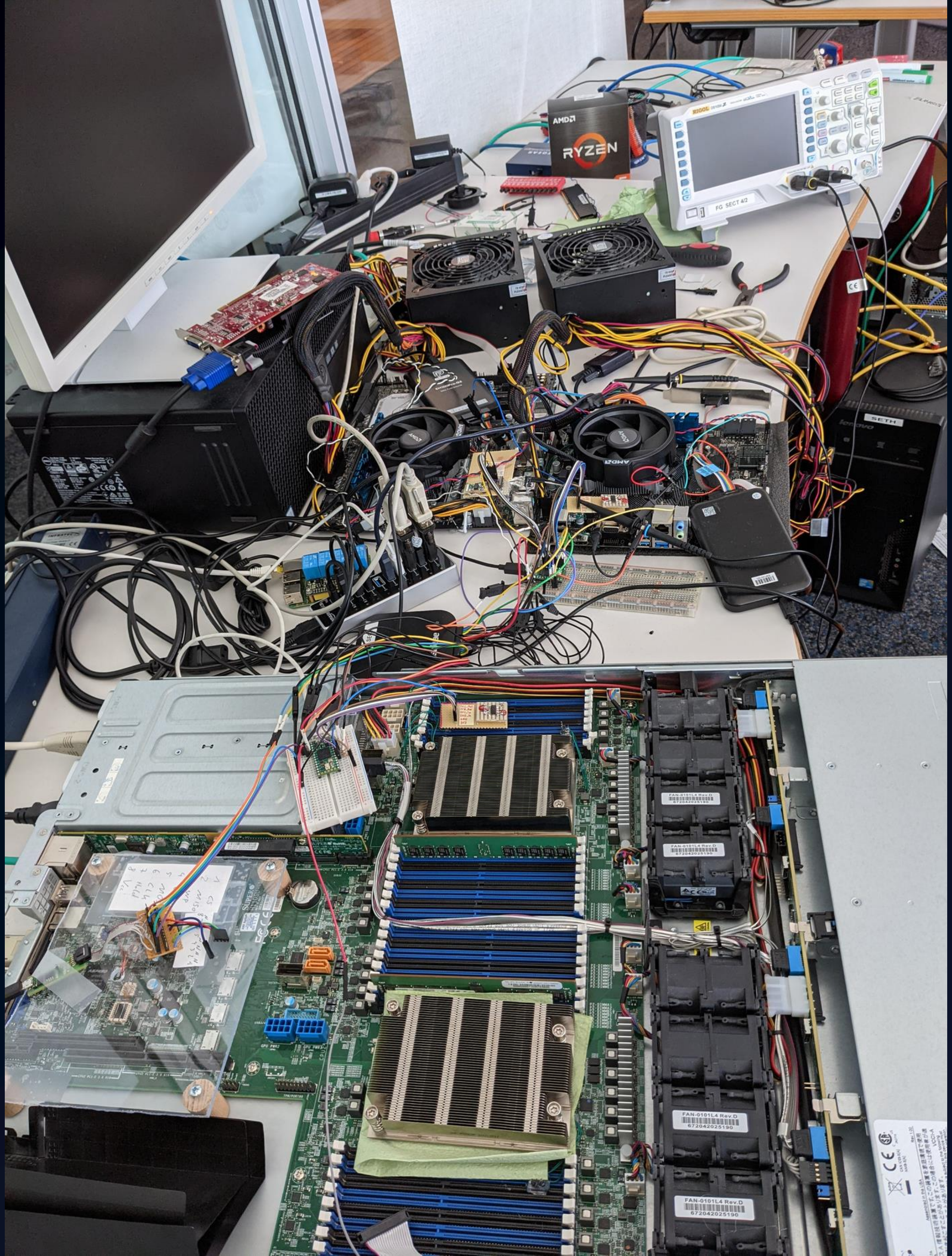
- SMU monitors SOC and uses the SVI2 bus to communicate with an SOC-external voltage regulator
- SVI2 allows to control two voltage domains per VR
- Ryzen uses single VR, Epyc dedicated VR for each domain



GLITCH SETUP

- Teensy µController to inject packets in to the SVI2 bus
- ATX reset line to reset target CPU
- Monitor the SPI bus (CS) to trigger the voltage glitch
- Control glitch parameters via external PC

Chen, Zitai, et al. "VoltPillager: Hardware-based fault injection attacks against Intel {SGX} Enclaves using the {SVID} voltage scaling interface." *30th {USENIX} Security Symposium ({USENIX} Security 21)*. 2021.

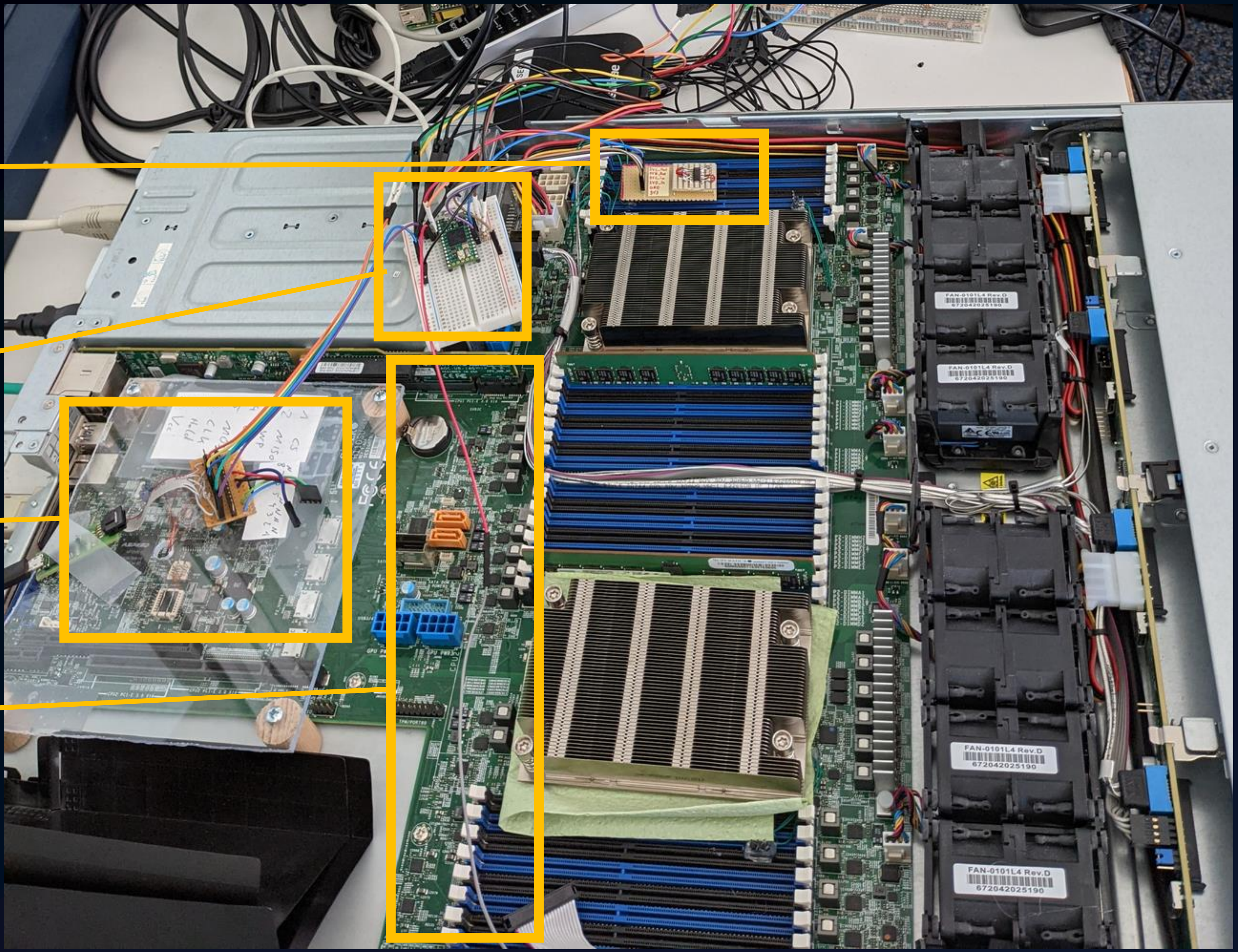

AMD
RYZEN

SVI2 access

Teensy µController

SPI access

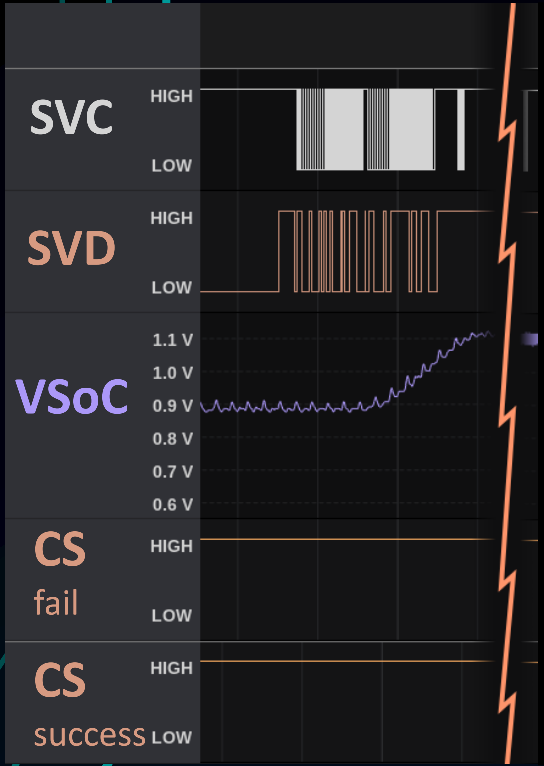
ATX reset



Glitch steps

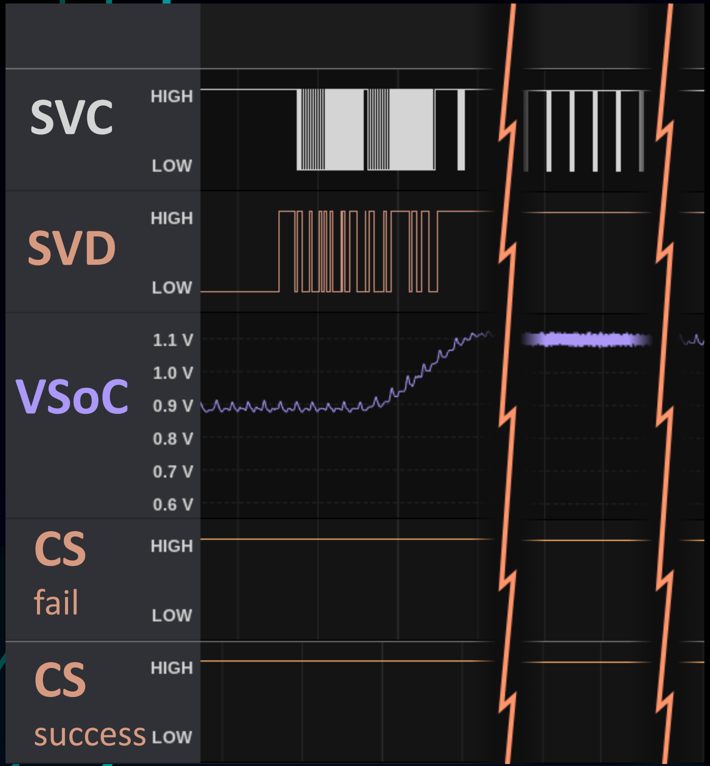
- SVI2 SVC – clock, CPU/VR (shared)
- SVI2 SVD – data from CPU, pulled low when inactive
- VSoC – target input voltage
- SPI CS – SPI’s chip-select signal (successful/failed pubkey verification)

Glitch steps



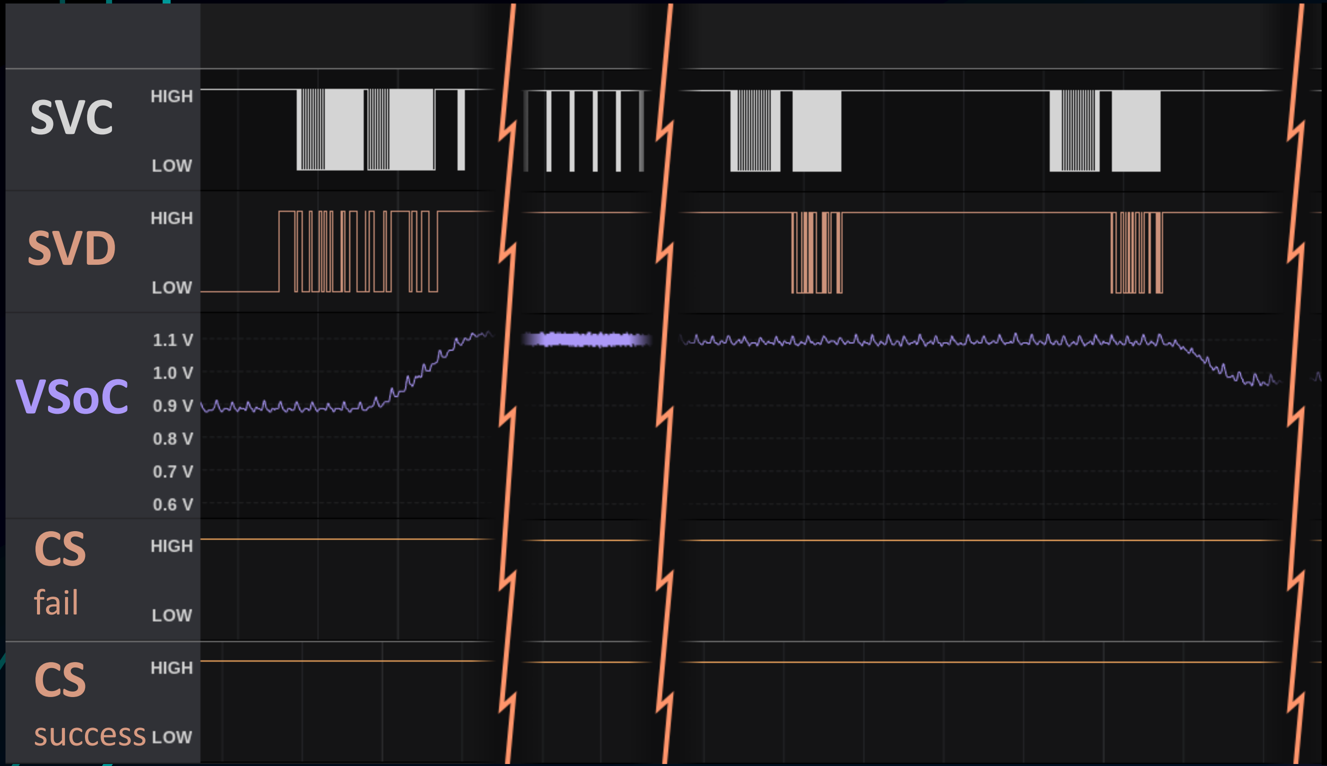
- SVI2 SVD: becomes high -> start attack logic
- CPU initially configures voltage

Glitch steps



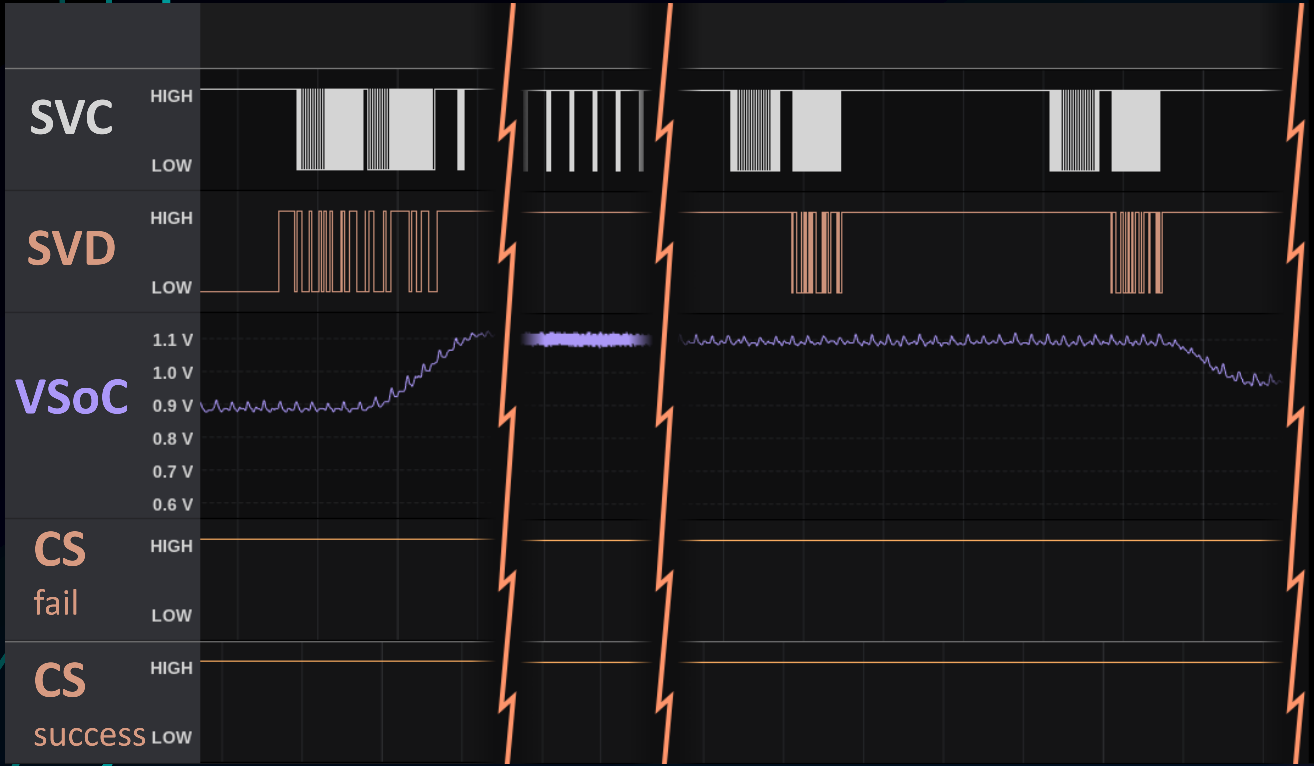
- SVI2 SVD: becomes high -> start attack logic
- CPU initially configures voltage
- VR constantly sends telemetry data to CPU

Glitch steps



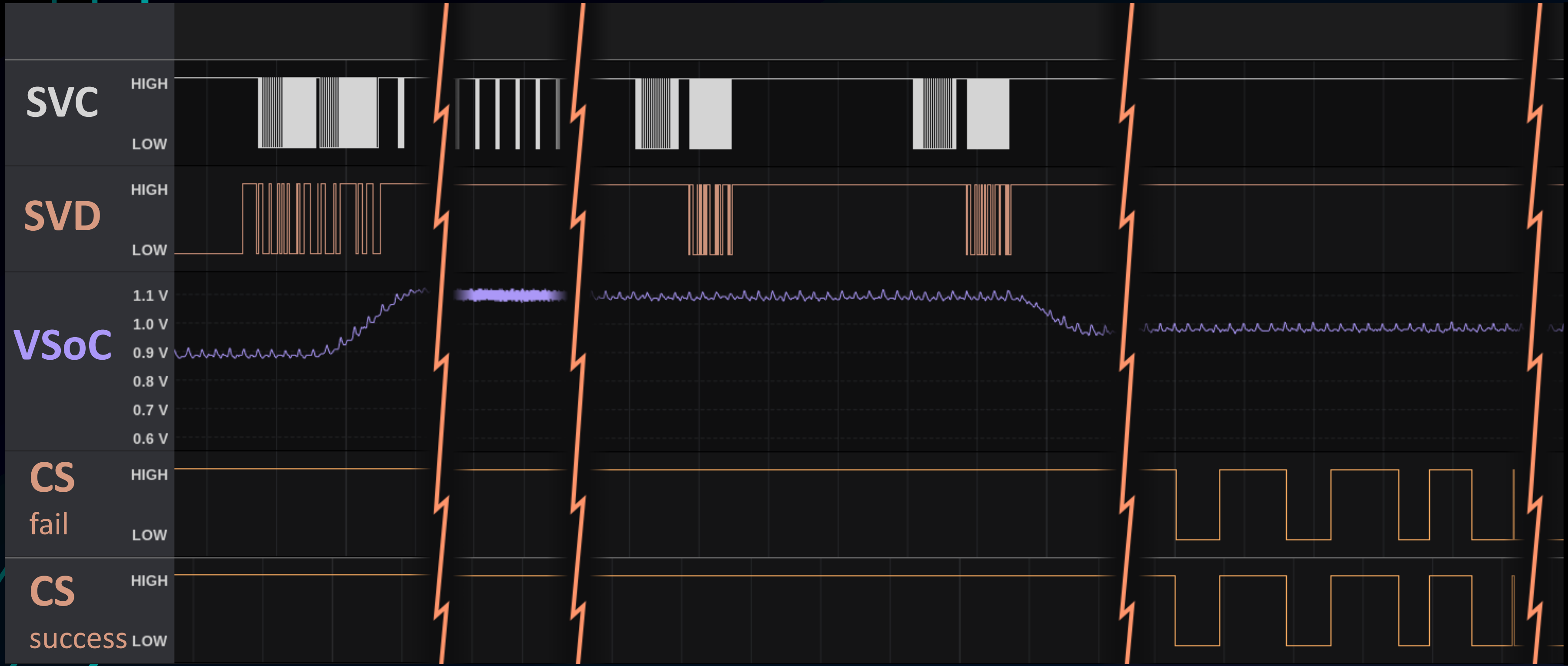
- SVI2 SVD: becomes high -> start attack logic
- CPU initially configures voltage
- VR constantly sends telemetry data to CPU
- Inject packets to disable telemetry -> avoids packet collision

Glitch steps



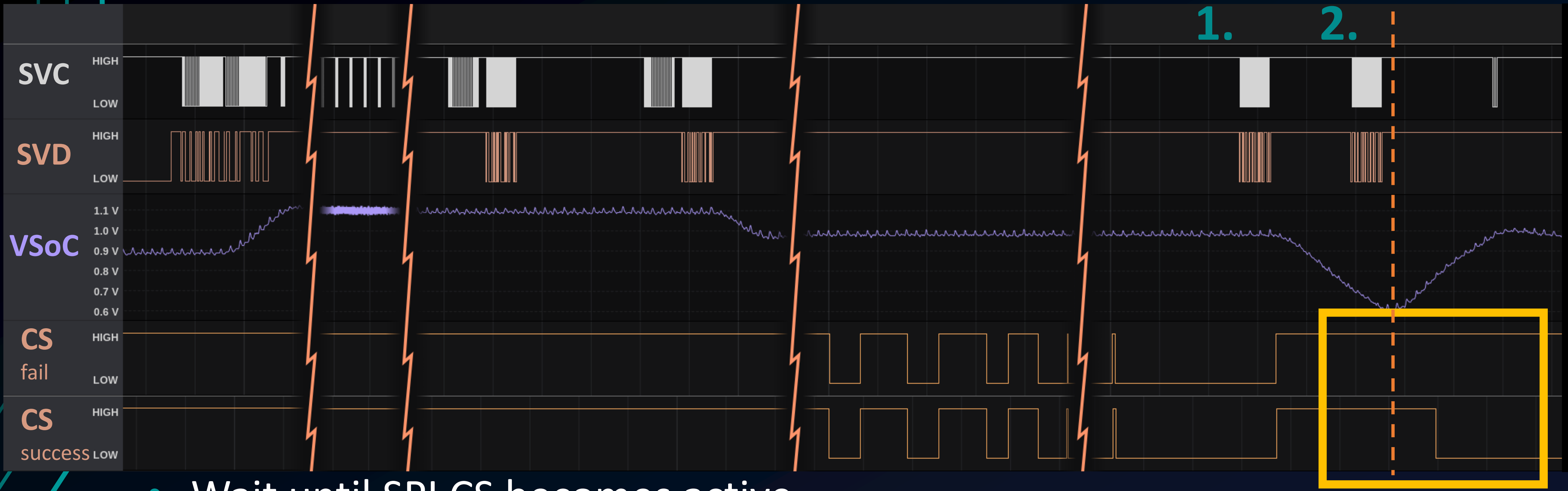
- Wait until SPI CS becomes active

Glitch steps



- Wait until SPI CS becomes active
- Count # of CS level changes to time glitch

Glitch steps



- Wait until SPI CS becomes active
- Count # of CS level changes to time glitch
- Inject packet to drop voltage and to revert to the original voltage level
- Verify success by observing CS again -> reset if CS not “low” after timeout

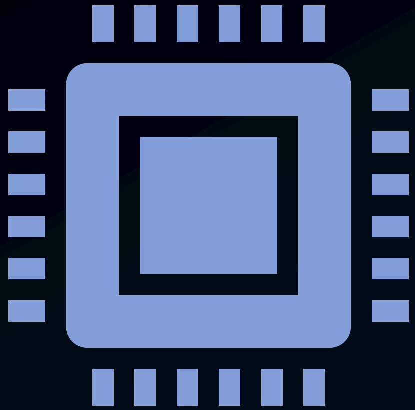
RESULTS

- Epyc and Ryzen CPUs are affected
- Successful glitch between every ~13min (Zen 1) and every ~46min (Zen 3)

Payloads:

- SPI “Hello World”
- Decrypt firmware (Zen 3)
- Dump ROM bootloader to SPI bus
- Deploy custom SEV firmware
- Dump (V)CEK secrets to the SPI bus

https://github.com/PSPReverse/amd-sp-glitch

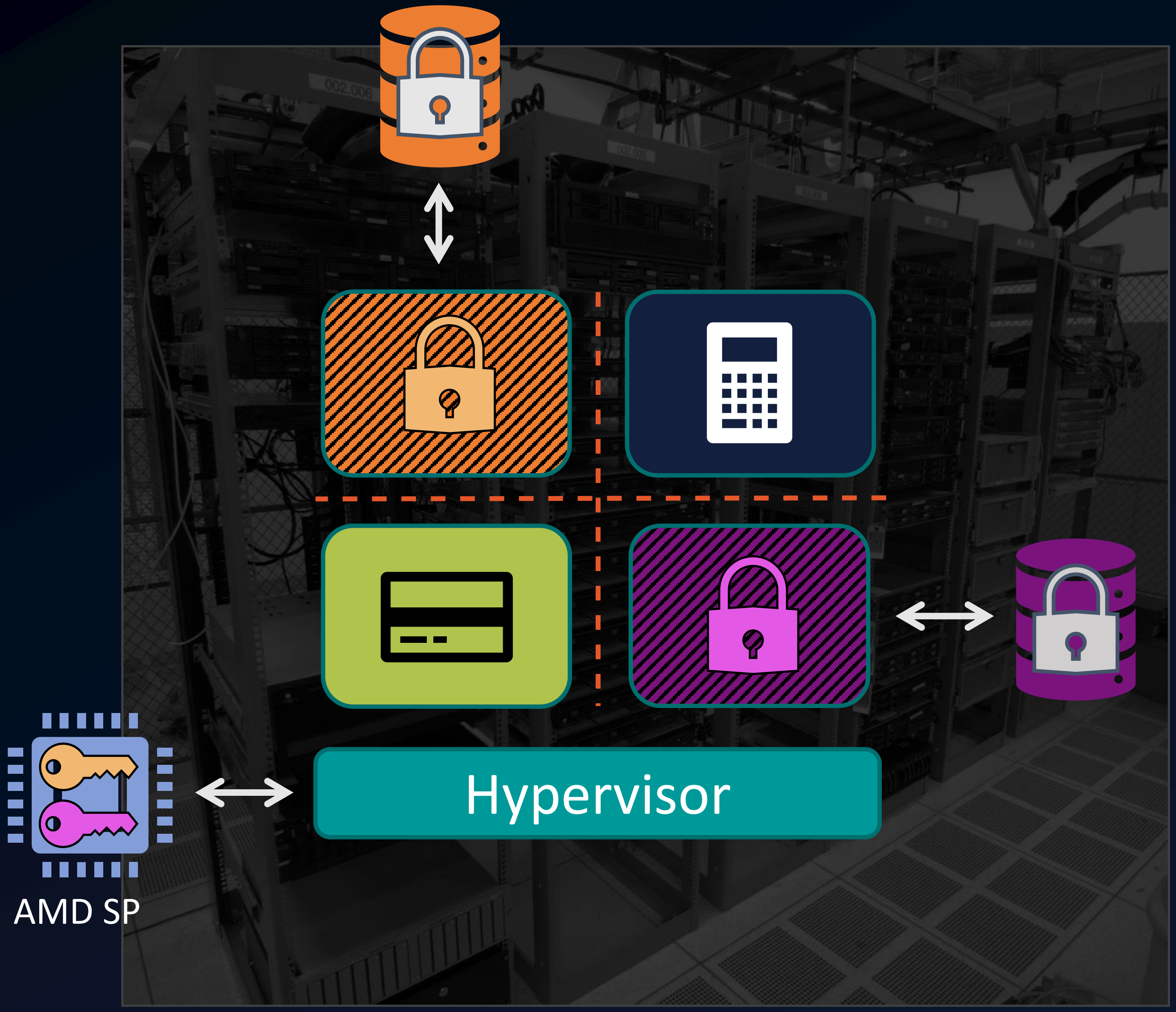


SEV: AMD-SP

Hosts the SEV firmware that implements the SEV API

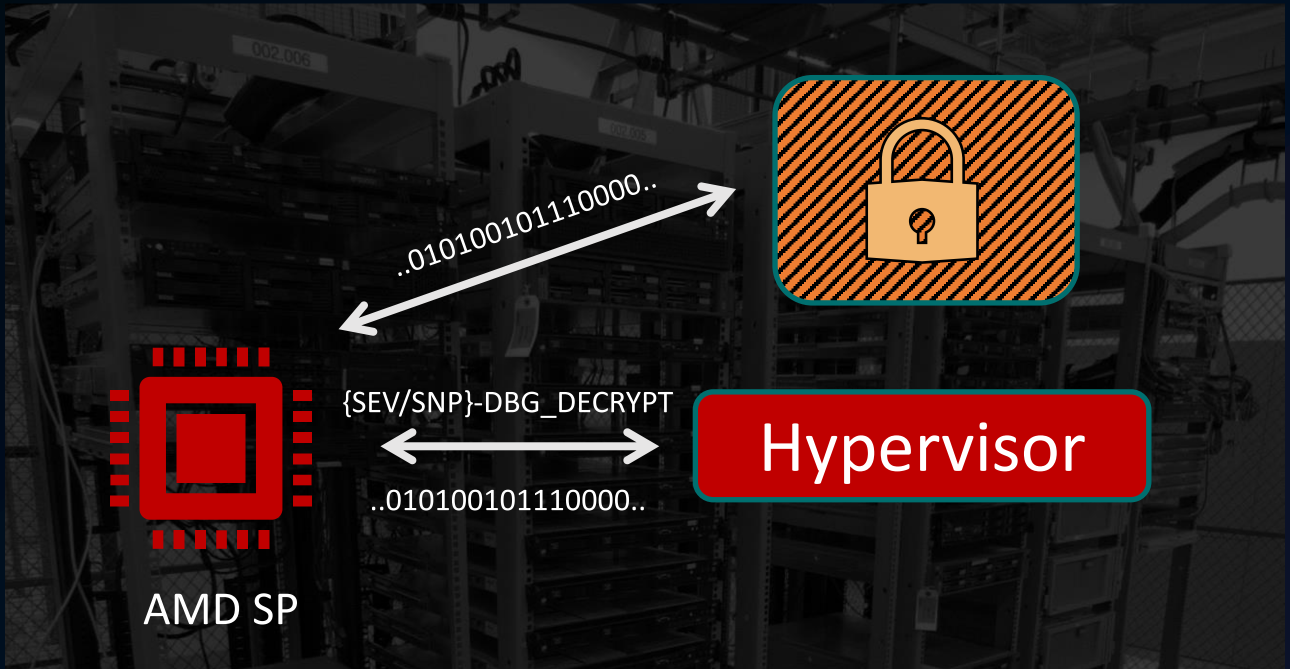
Memory encryption keys

Endorsement keys (CEK / VCEK)



MALICIOUS CLOUD ADMINISTRATOR

- **Debug override**



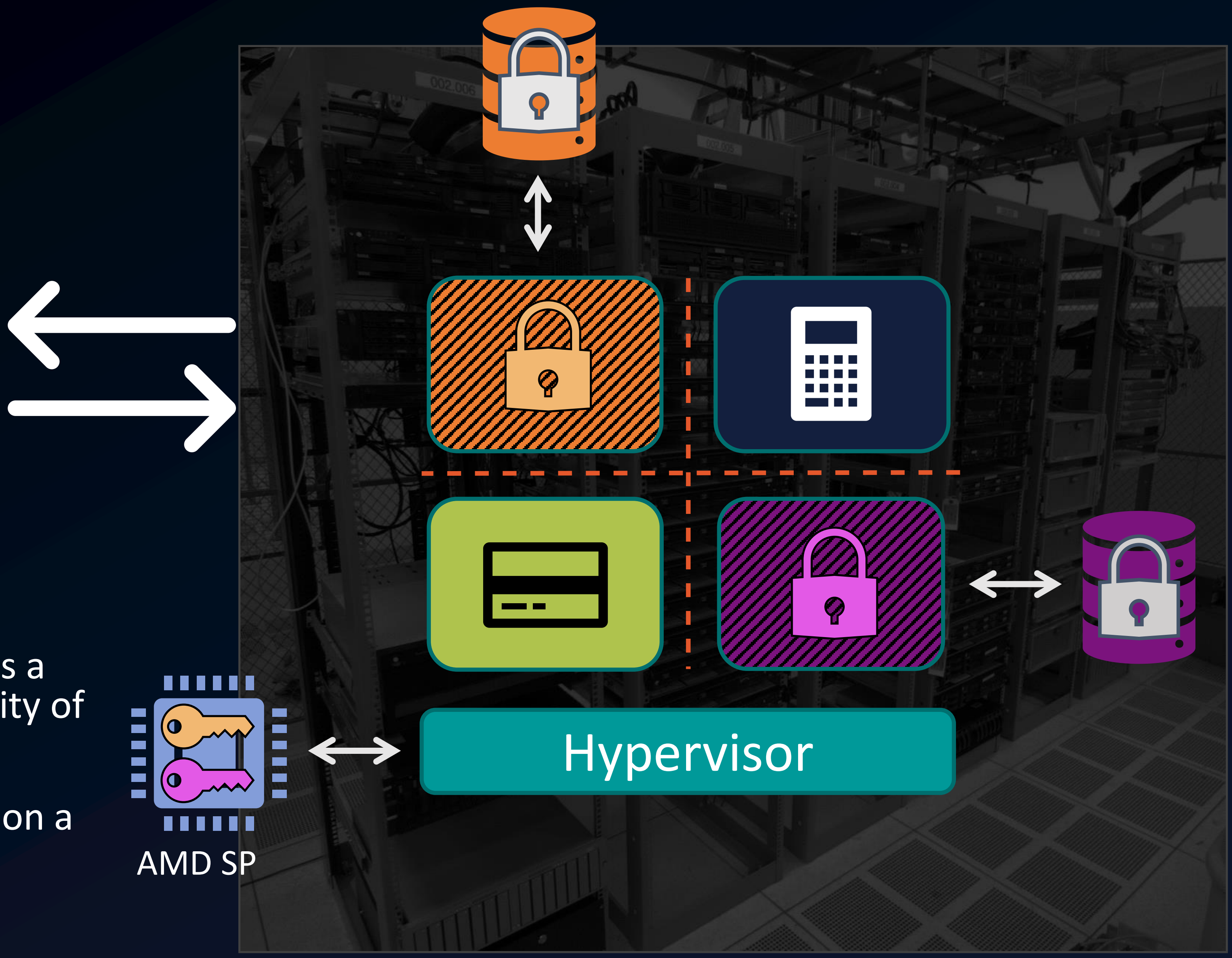
1. Boot system with patched SEV firmware: Enables the "DBG_DECRPYT" SEV API command regardless of a guest's SEV policy
2. Decrypt the VM's memory

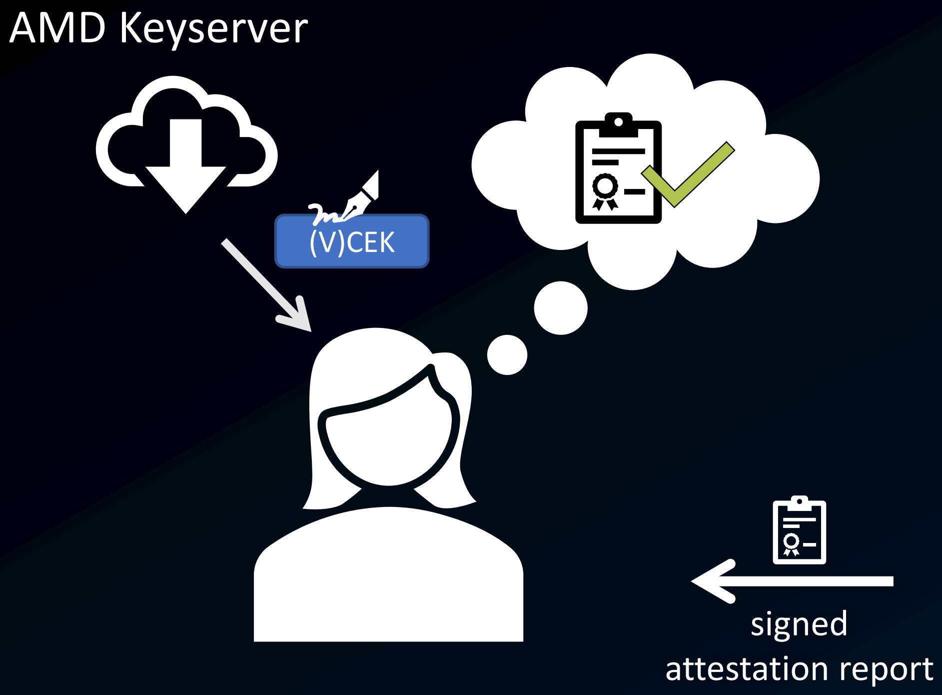
Works with SEV / SEV-ES and SEV-SNP

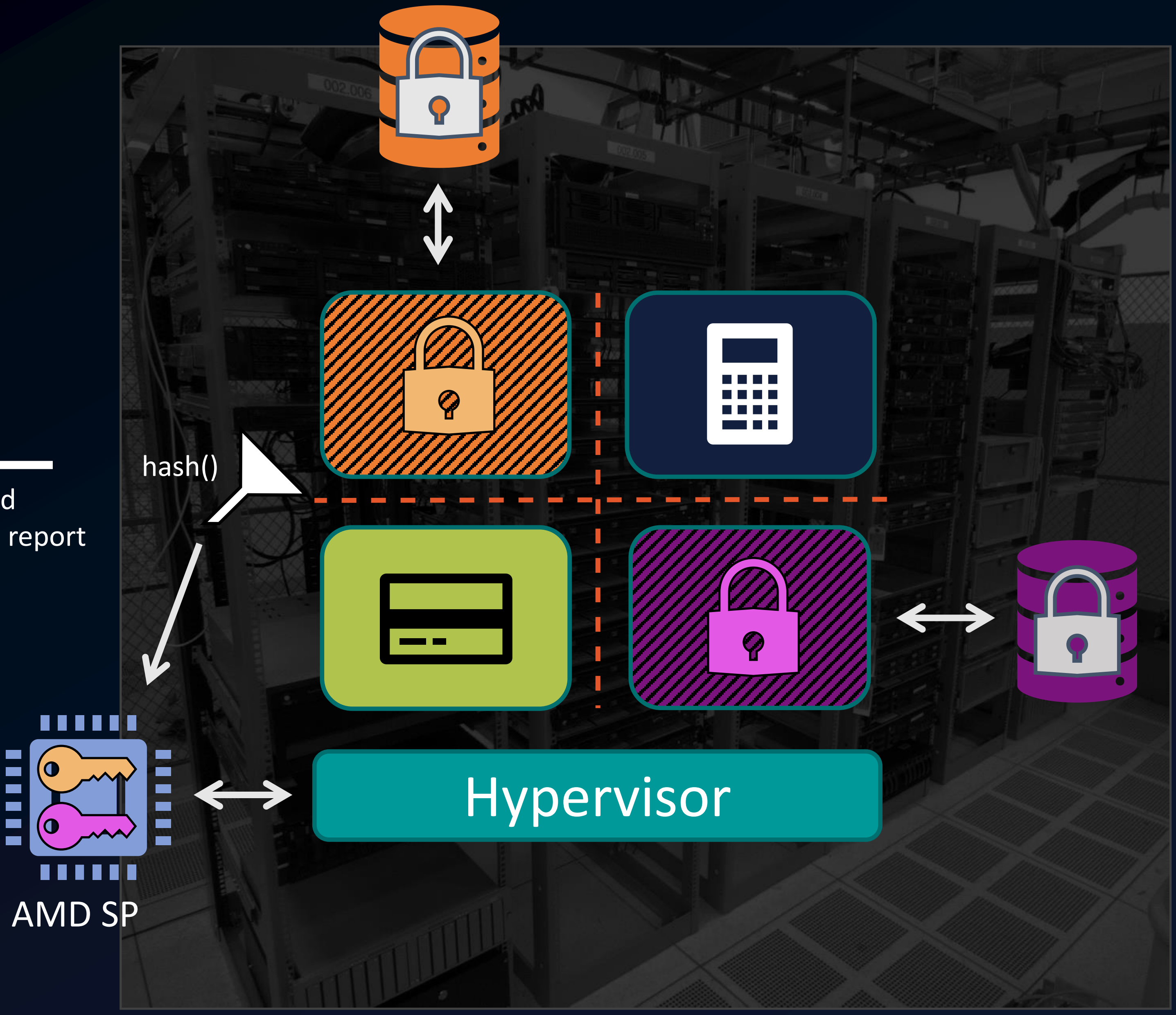
SEV REMOTE ATTESTATION

SEV's remote attestation allows a party to validate the authenticity of a remote system.

Customer: Is my VM deployed on a genuine AMD system with SEV protection in place?

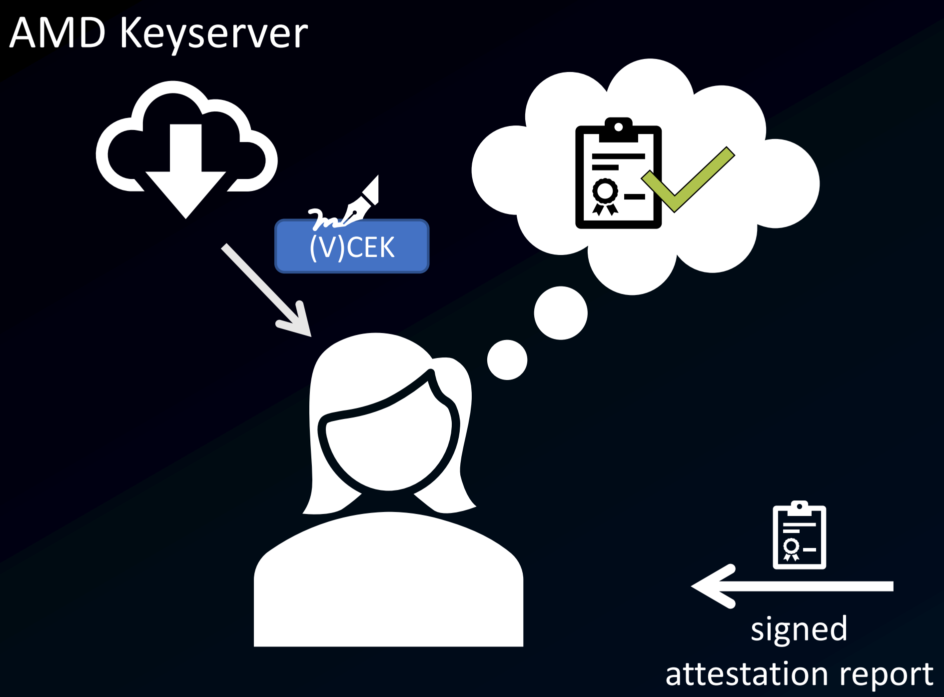






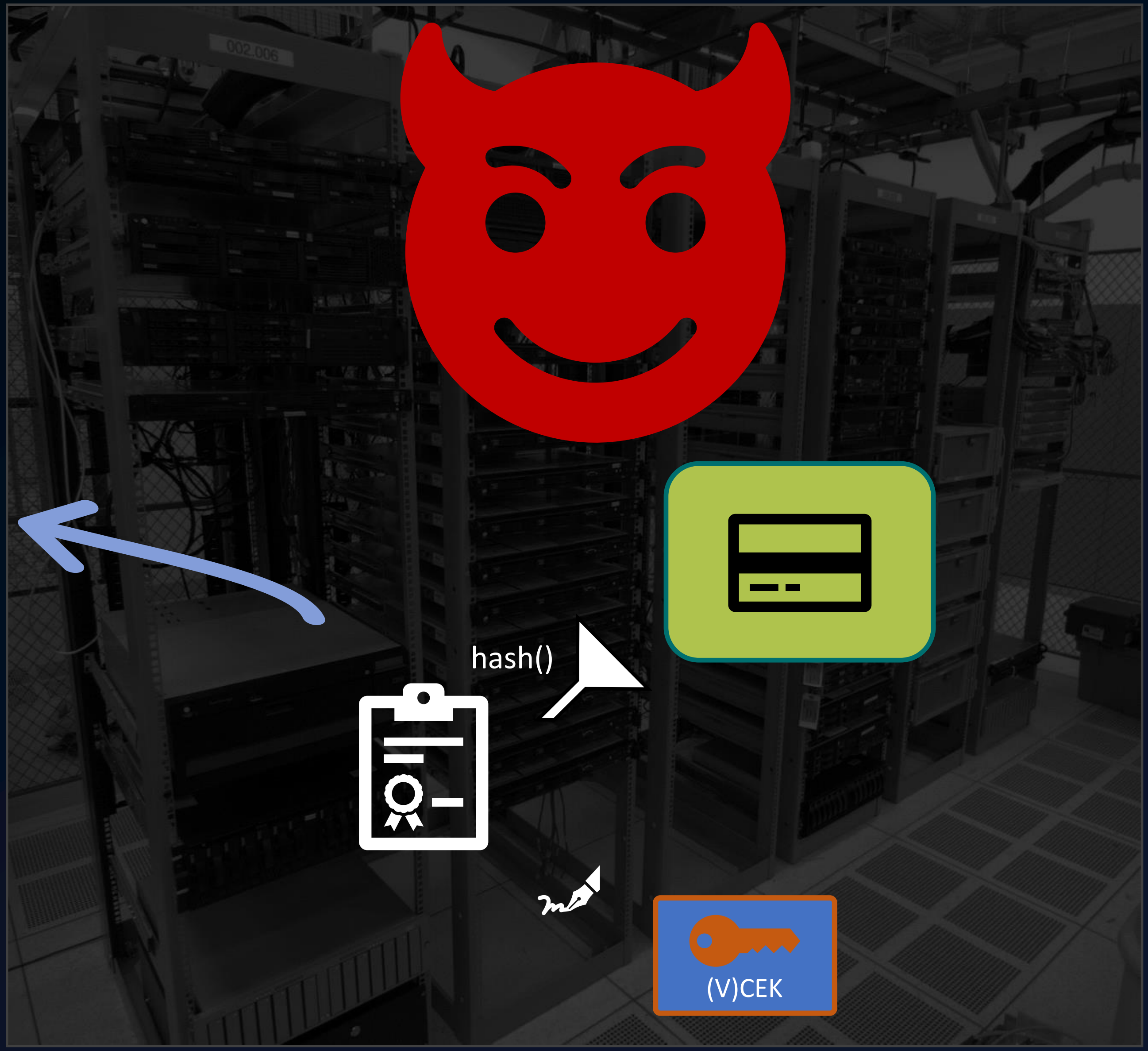
SEV REMOTE ATTESTATION

1. AMD-SP creates measurement of SEV protected VM
2. Customer receives signed attestation report including measurement
3. Customer validates attestation report by verifying its signature using a key from an AMD keyserver: (V)CEK



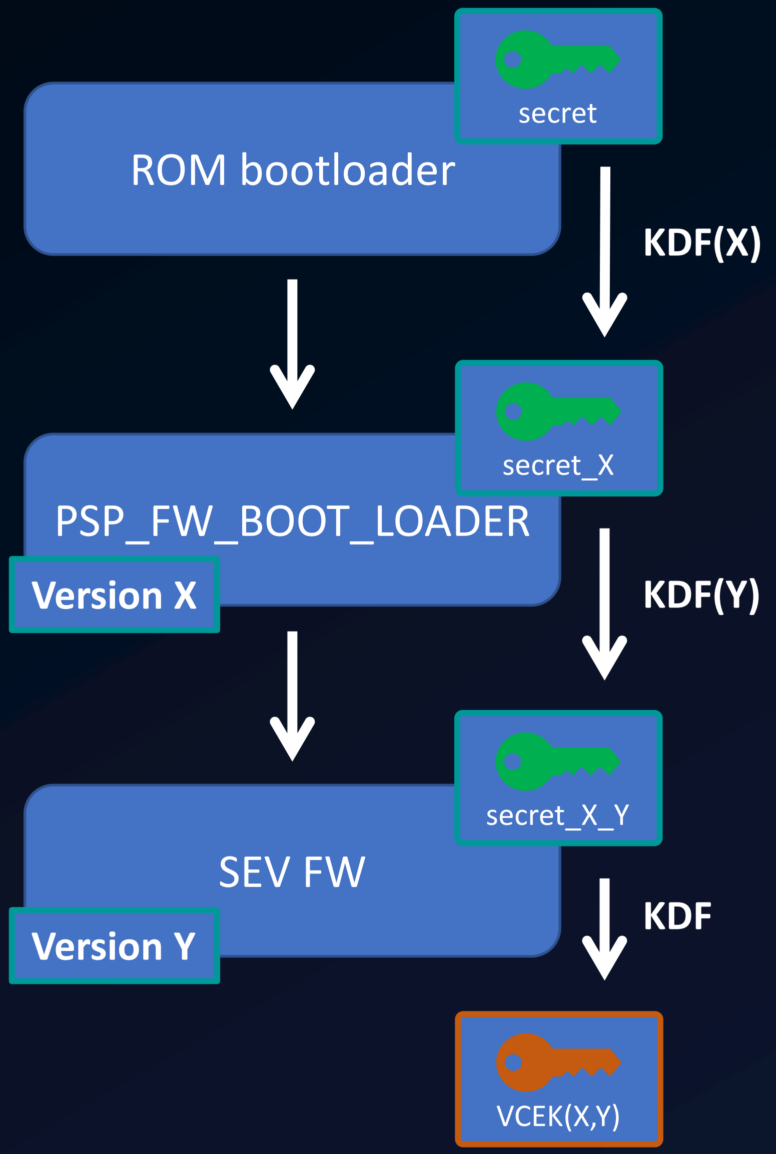
SEV REMOTE ATTESTATION

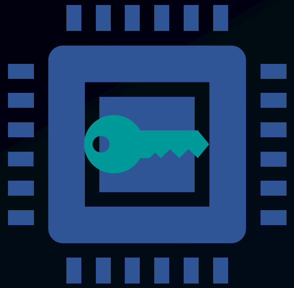
Extracted endorsement keys allow an attacker to, e.g., fake the presence of SEV!



VERSIONED CEK (VCEK) SIMPLIFIED

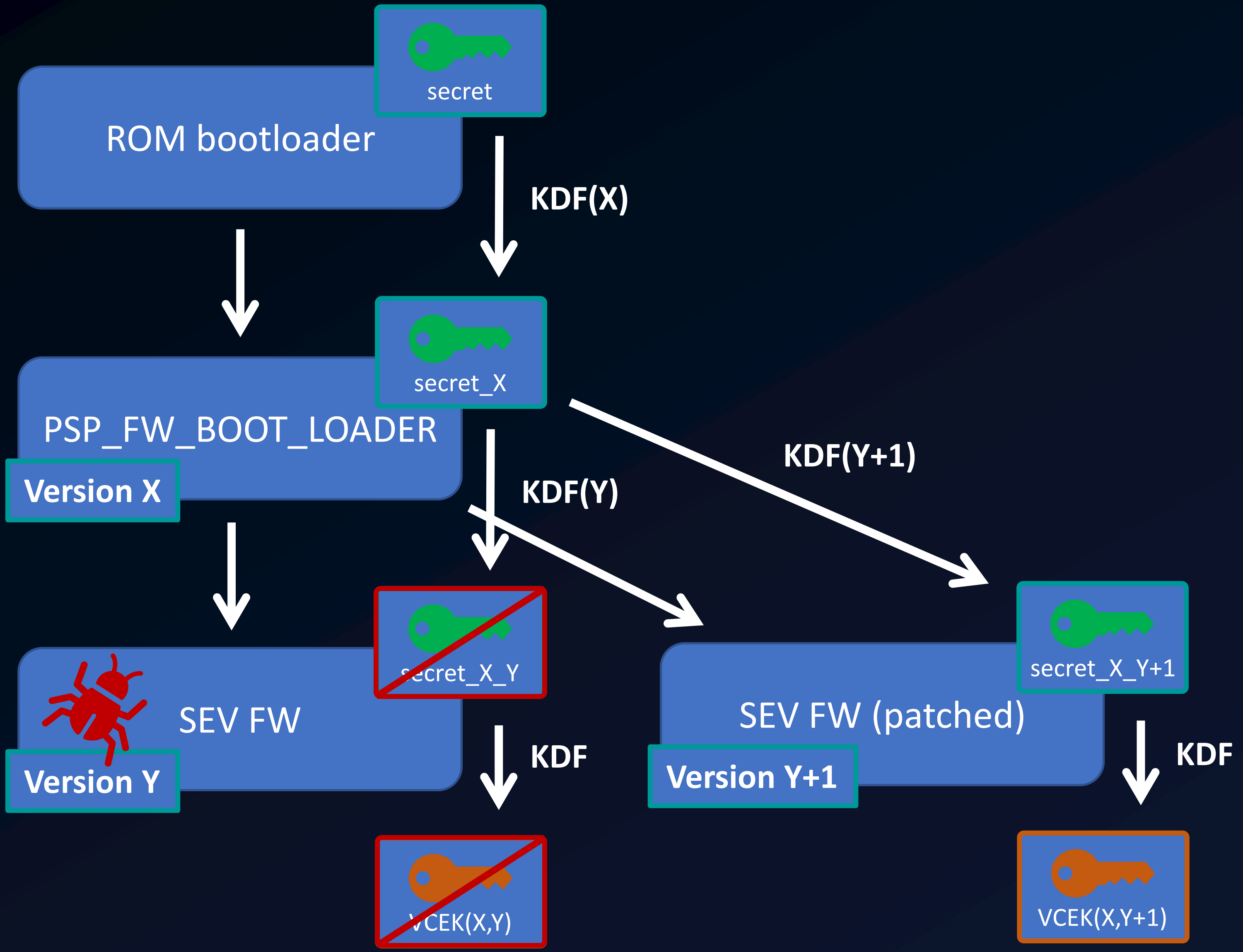
“[VCEK is] derived from chip-unique secrets and current TCB version”

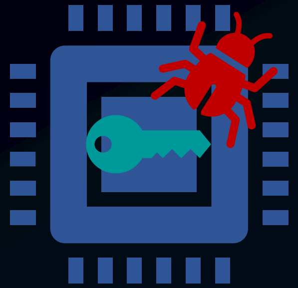




VCEK ATTACK

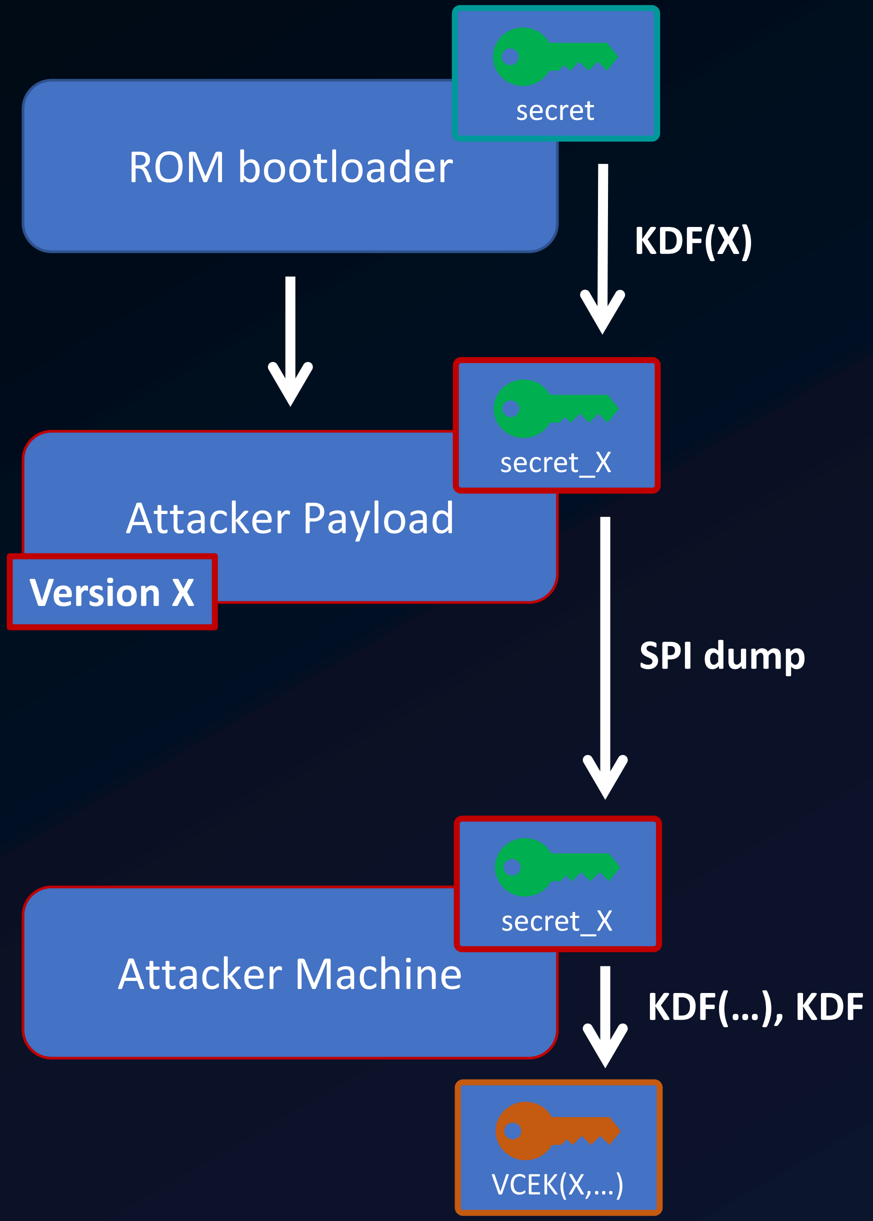
What if there is a bug?





OUR ATTACK

- Version is part of the header
- We get VCEK for any TCB
- SEV-SNPs allows TCB downgrade
  → attack needs only one glitch



Summary

AMD-SP IS SUSCEPTIBLE TO VOLTAGE FAULT INJECTION ATTACKS

- **Ryzen** and **Epyc** Zen 1, Zen 2 and Zen 3 systems are affected
  - ThreadRipper most probably
- Allows an attacker to **execute payloads** on the AMD-SP right after the ROM bootloader
- **Reliable code-execution** between every ~13min (Zen 1) and every ~46min (Zen 3)
- **SEV's** protection mechanism can be circumvented
- **fTPMs** most probably compromised
  - not tested yet
- **Mitigations:** none

→ Future CPU generations might include HW and SW mitigations

RESOURCES

https://arxiv.org/abs/2108.04575

- Paper: One Glitch to Rule Them All: Fault Injection Attacks Against AMD SEV

https://github.com/PSPReverse/amd-sp-glitch

- Supplemental data and code:
  - Glitch setup and code
  - (V)CEK key derivation implementation
  - Firmware decryption implementation

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

- Proof-of-concept implementation of the migration attack for SEV / SEV-ES

https://github.com/PSPReverse/PSPTool

- psptool & psptrace

https://github.com/PSPReverse/PSPEmu

- PSPEmulator: Emulator for the AMD-SP
- QEMU port: https://github.com/RobertBuhren/qemu/tree/pspemu

THANK YOU

Robert Buhren: robert.buhren@sect.tu-berlin.de

Hans Niklas Jacob: hnj@sect.tu-berlin.de