

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

SECT

Technische
Universität
Berlin

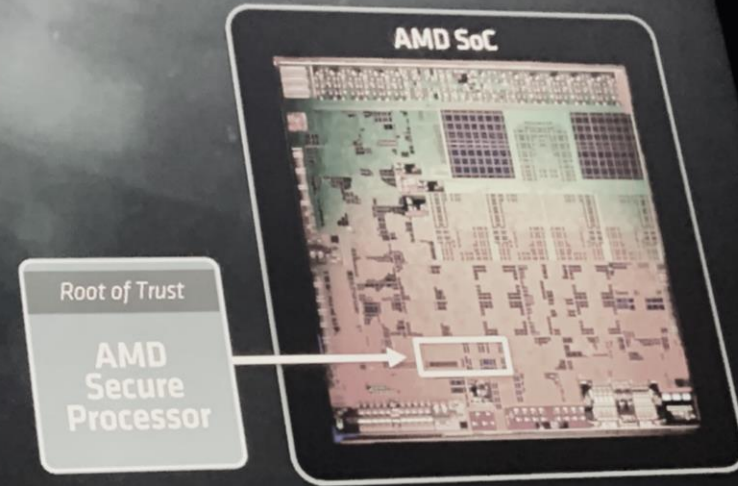


AMD SECURE PROCESSOR

A Dedicated Security Subsystem

- AMD Secure Processor integrated within SoC
 - 32-bit microcontroller (ARM Cortex-A5)
- Runs a secure OS/kernel
- Secure off-chip NV storage for firmware and data (i.e. SPI ROM)
- Provides cryptographic functionality for secure key generation and key management
- Enables hardware validated boot

Hardware Root of Trust Provides Foundation for Platform Security



AMD

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

Applications

SECURE ENCRYPTED VIRTUALIZATION (EPYC)

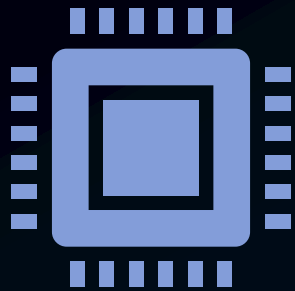
- **SEV** protects virtual machines in **untrusted** environments by encrypting VM memory
- The AMD SP is responsible for key management
- Paper: “Insecure Until Proven Updated: Analyzing AMD SEV's Remote Attestation”

SECURE OS (RYZEN / TR)

- Firmware **TPM**
- ...

TRUSTED EXECUTION ENVIRONMENT

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



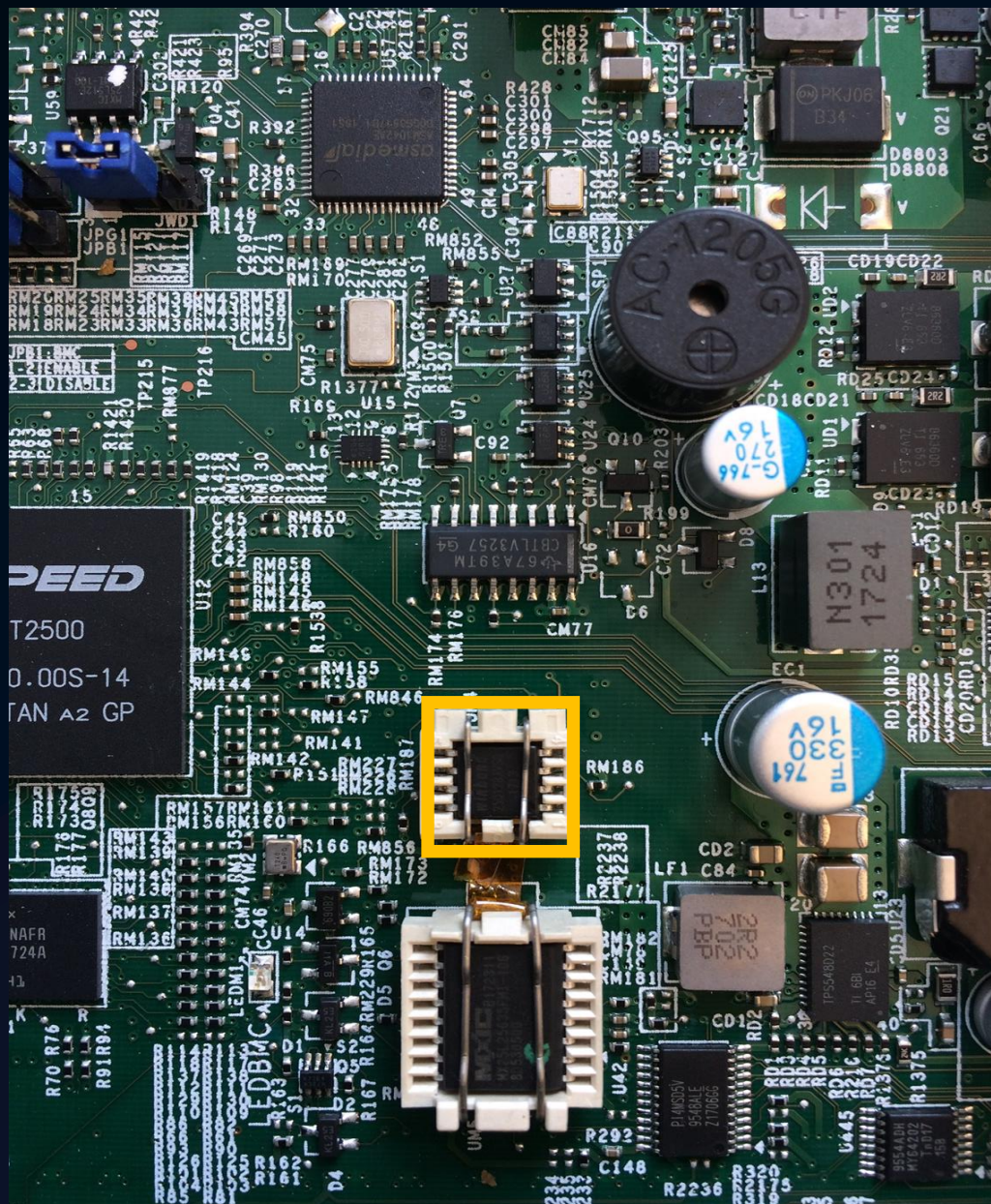
FIRMWARE ANALYSIS

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_star-psptool_regaining_control_over_your_amd_cpu

Display, extract, and manipulate PSP firmware inside UEFI images

76 commits 3 branches 0 packages 0 releases 2 contributors GPL-3.0

Branch: master New pull request Find file Clone or download

File	Description	Time
bin	Finally discard legacy psptool and rename psptool2 to psptool	4 months ago
psptool	Show MD5 sums of Entries in verbose mode (-v)	4 months ago
.gitignore	Finally discard legacy psptool and rename psptool2 to psptool	4 months ago
LICENSE	Add GPLv3 license	7 months ago
README.md	Update README.md	3 days ago
setup.cfg	Update configs to upload to PyPI	2 months ago
setup.py	Update configs to upload to PyPI	2 months ago

README.md

PSPTool

PSPTool is a Swiss Army knife for dealing with firmware of the **AMD Secure Processor** (formerly known as *Platform Security Processor* or **PSP**). It locates AMD firmware inside **UEFI images** as part of BIOS updates targeting **AMD platforms**.

It is based on reverse-engineering efforts of AMD's **proprietary filesystem** used to **pack firmware blobs** into **UEFI Firmware Images**. These are usually 16MB in size and can be conveniently parsed by **UEFITool**. However, all binary blobs by AMD are located in padding volumes unparsable by UEFITool.

PSPTool favourably works with UEFI images as obtained through BIOS updates.

Installation

```
pip install psptool
```

PSPTOO

Python-base

Parsing

Decompress

PEM export

Signature up

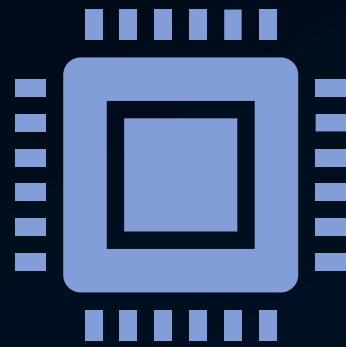
The screenshot shows a video player interface. The main content is a presentation slide with a dark blue background and white text. At the top, it says 'Uncover, Understand, Own - Regaining Control Over Your AMD CPU' by Robert Buhren, Alexander Eichner, and Christian Werling. Below the title are three icons: a magnifying glass, a lightbulb, and a house with a chip inside. The main title 'Uncover, Understand, Own' is in large white font, with 'REGAINING CONTROL OVER YOUR AMD CPU' in smaller teal font below it. At the bottom of the slide, there is a small circular logo with a person icon. The video player interface includes a progress bar at the bottom, showing a duration of 56 minutes and a current time of 01:37. There are also navigation icons and a volume control icon.

The screenshot shows a GitHub repository page for 'PSPReverse/PSPTool'. The repository has 18 watchers, 285 stars, and 20 forks. It has 2 contributors and is licensed under GPL-3.0. The 'Clone or download' button is highlighted in green. The 'Latest commit' is 'fef1bed' from 3 days ago. Below the commit list, there is a description of the repository: 'processor (formerly known as Platform part of BIOS updates targeting AMD' and 'ed to pack firmware blobs into UEFI used by UEFITool. However, all binary blobs'. The repository is part of a larger project, as indicated by the 'Star PSP' link.

<https://media.ccc.de/v/36c3-10942-uncover-understand-own-regaining-control-over-your-amd-cpu>

<https://github.com/PSPReverse/PSPTool>

```
pip install psptool
```



AMD

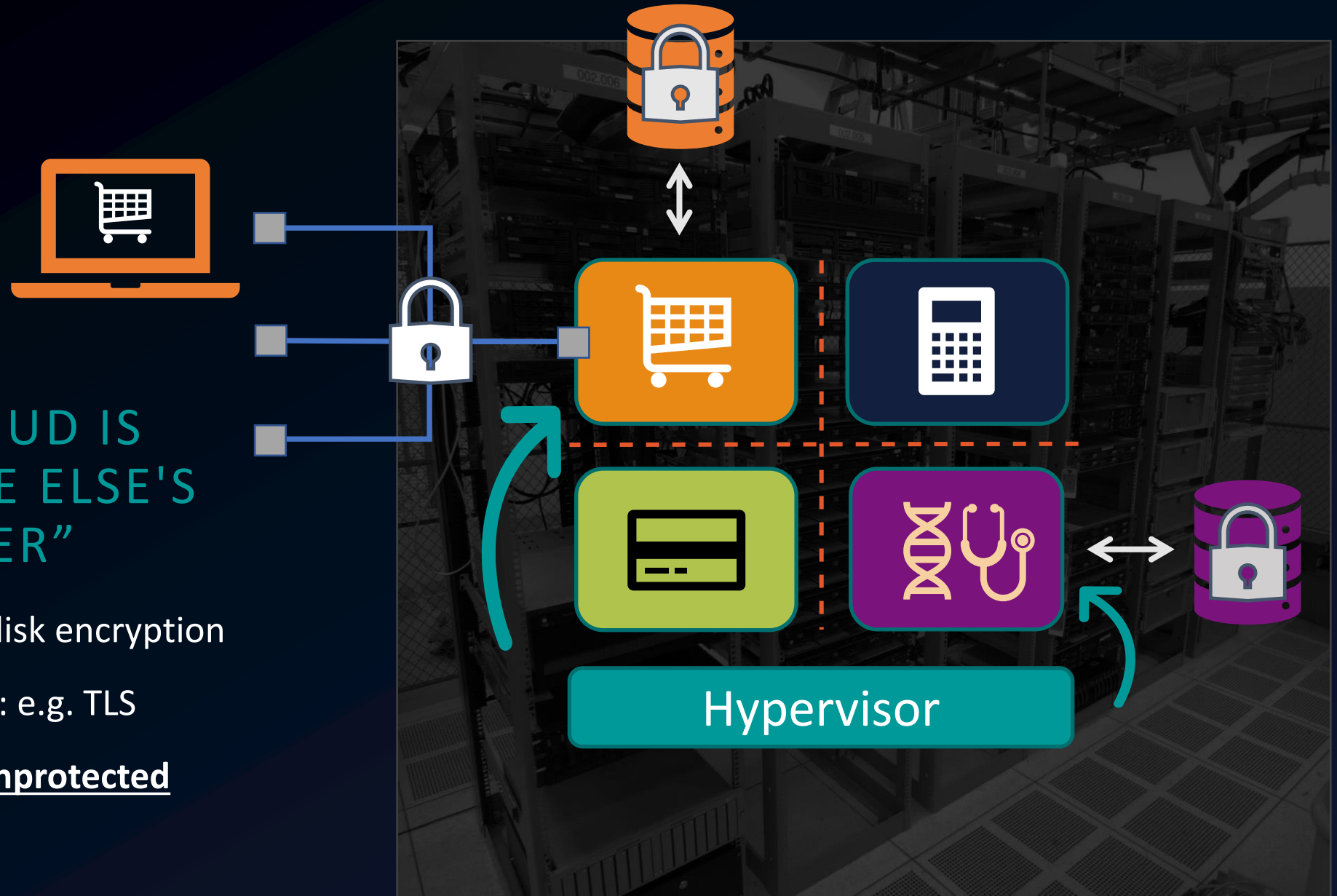
Secure Encrypted Virtualization (SEV)

“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

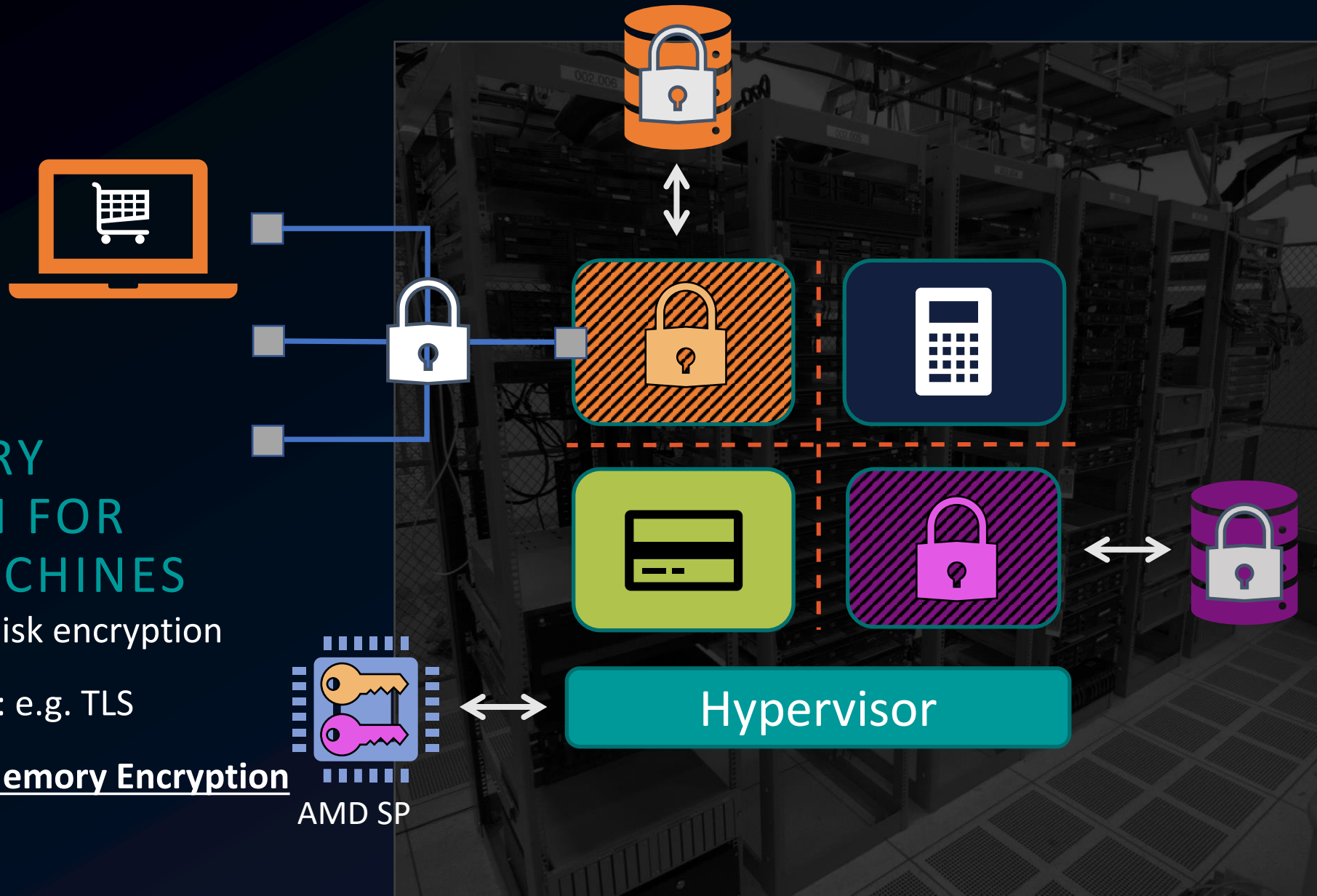
Data-At-Rest: disk encryption

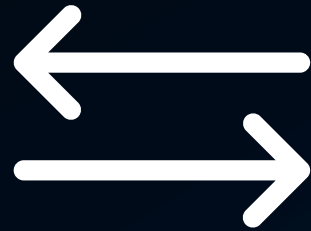
Data-In-Transit: e.g. TLS

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



AMD SP

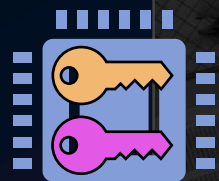




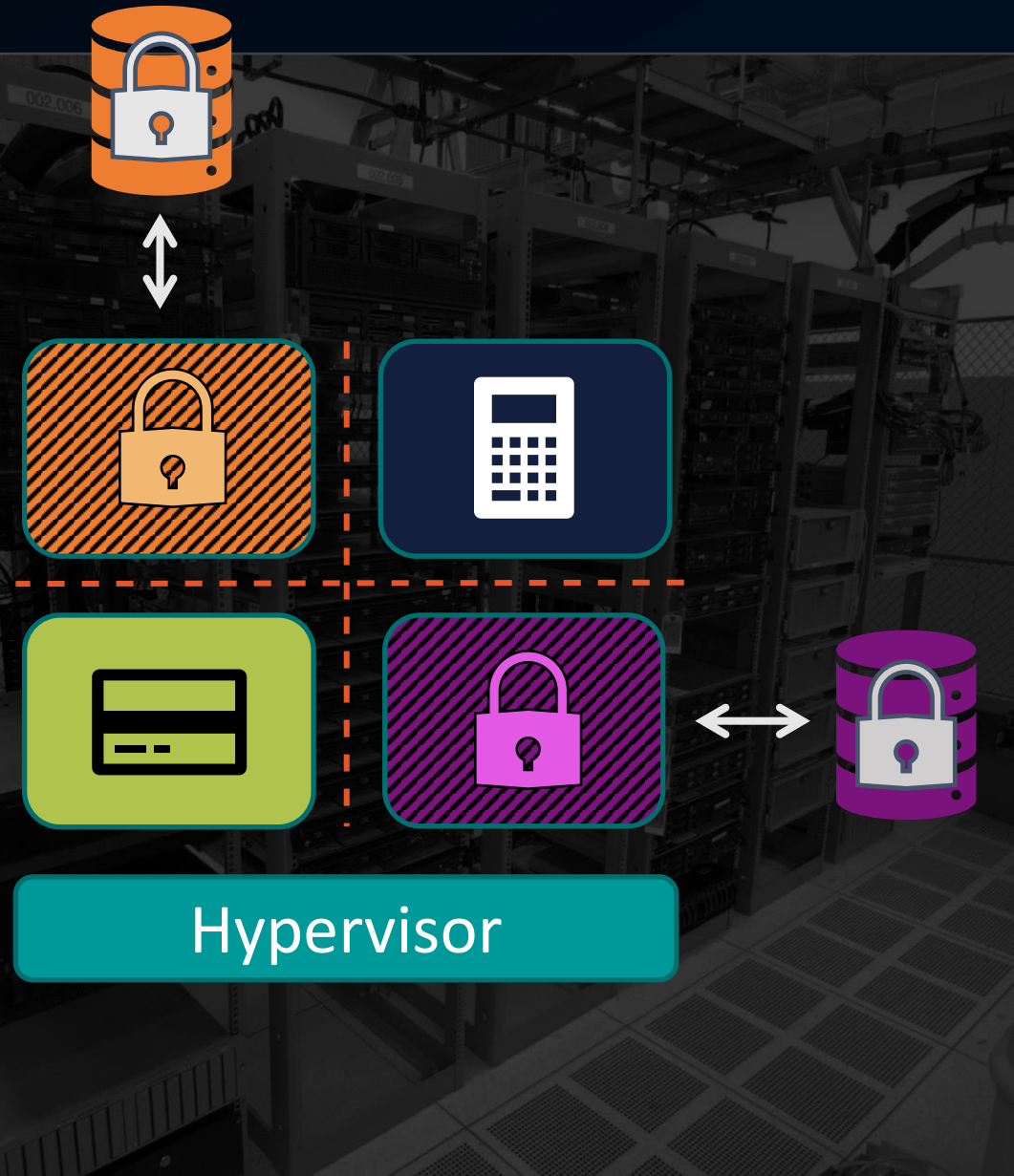
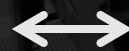
SEV REMOTE ATTESTATION

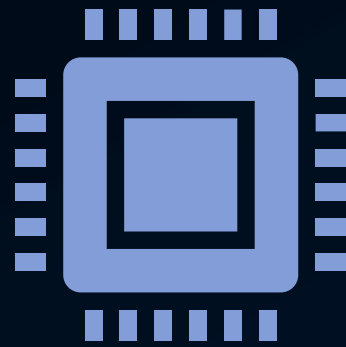
SEV's remote attestation allows a party to validate the authenticity of a remote system.

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



AMD SP



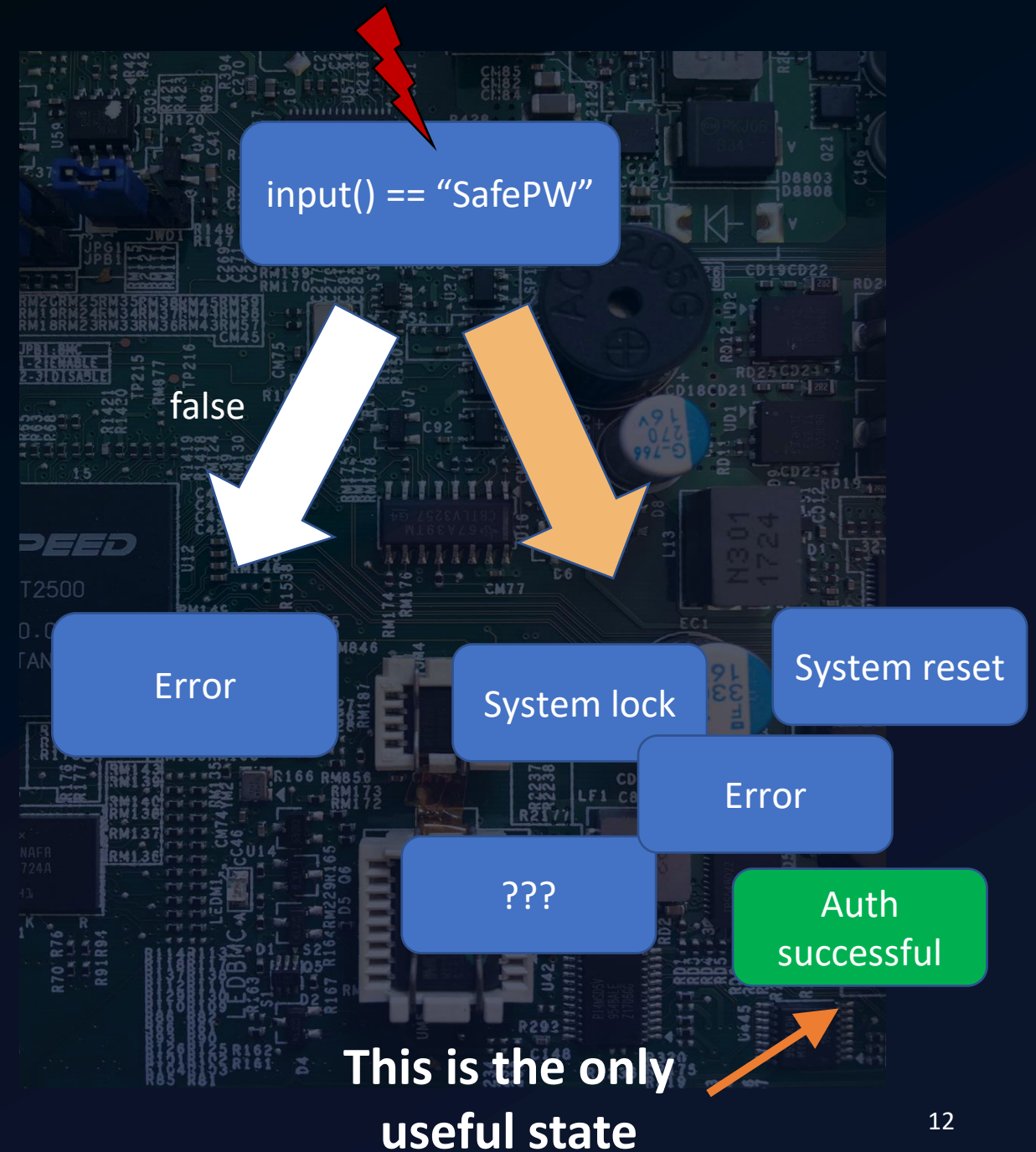


Faulting the AMD-SP

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

Key Challenges

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

- **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.
- Lower voltage rails
→ Voltage fault injection
 - Hit IC with electromagnetic radiation
→ EM fault injection
 - Hit IC with laser
→ Laser fault injection ...
 - Most faults are useless for an attacker

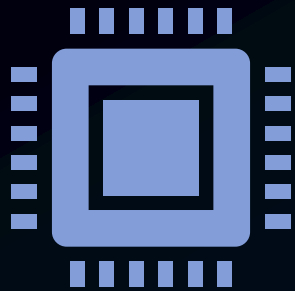
input() == "SafePW"

System reset

Error

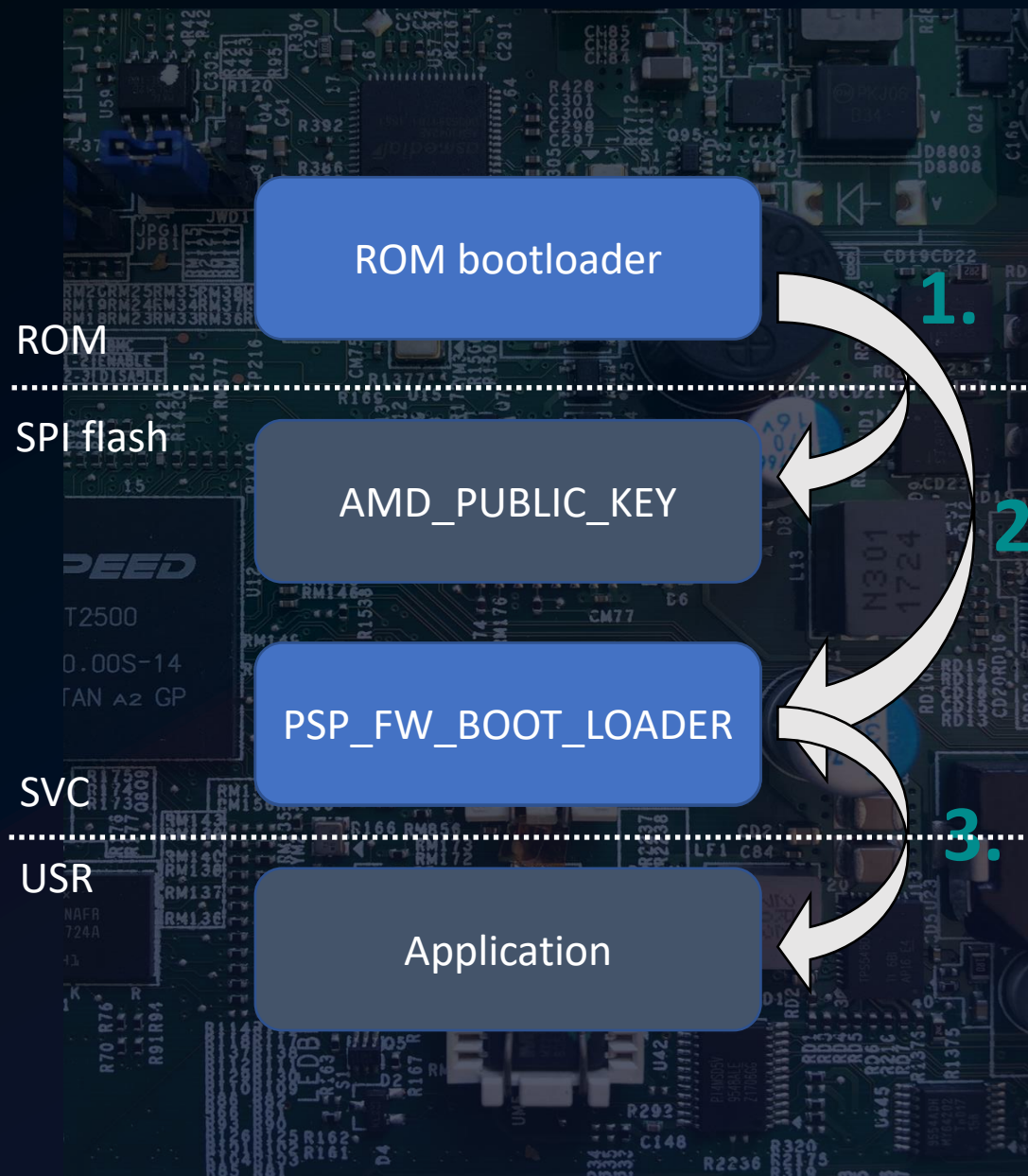
Auth
successful

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

ROM

SPI flash



PUBLIC_KEY

PSP_FW_BOOT_LOADER

1.

Continued SPI activity

AMD_PUBLIC_KEY

CS

MISO

PUBLIC_KEY

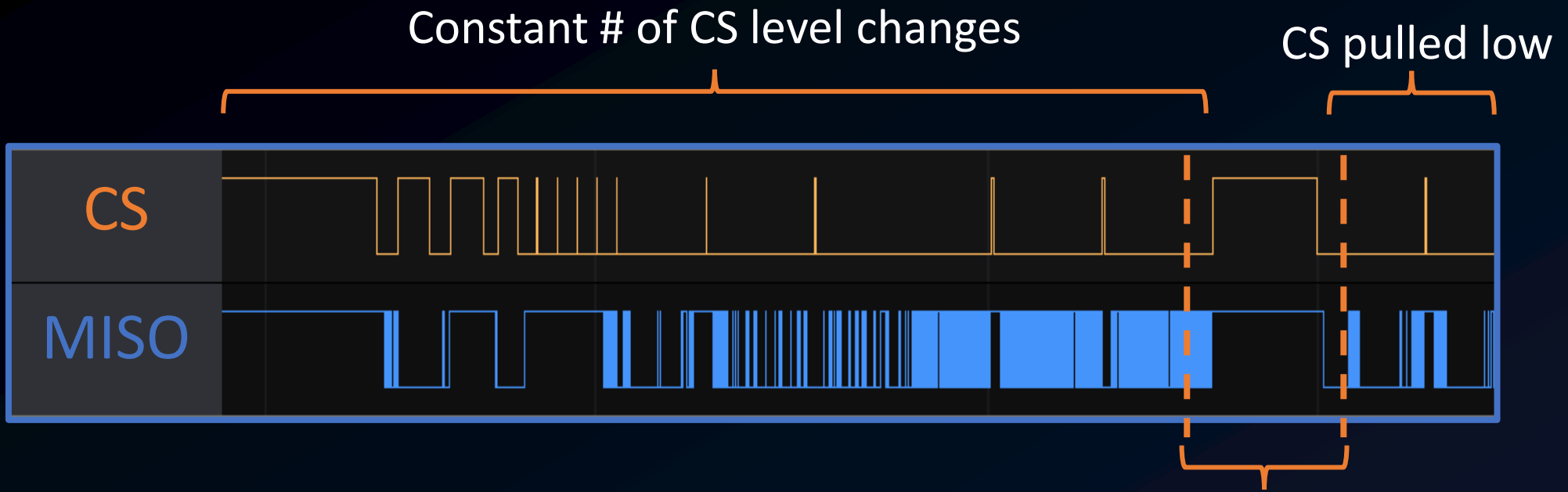
No SPI activity

CS

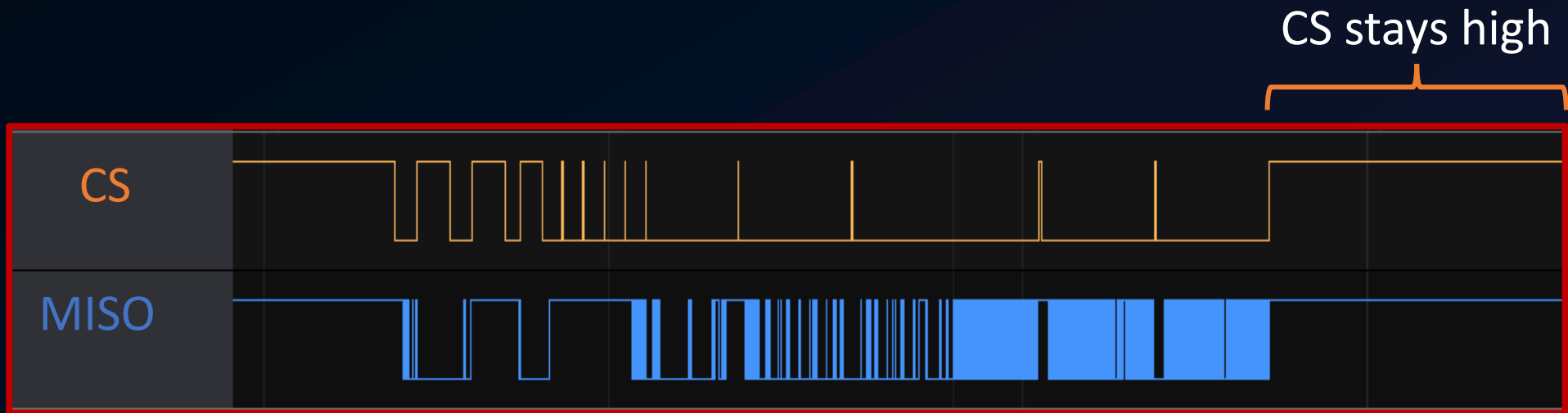
MISO

Logic Analyzer





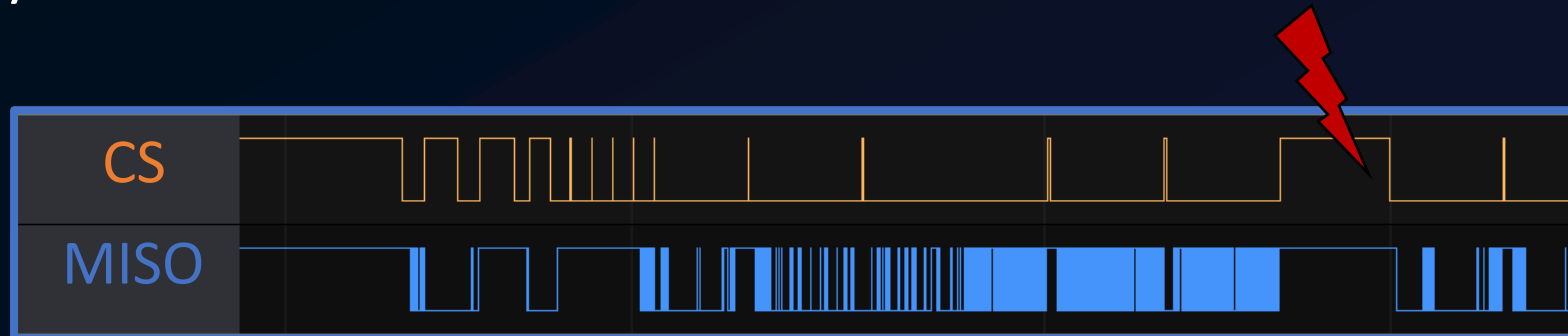
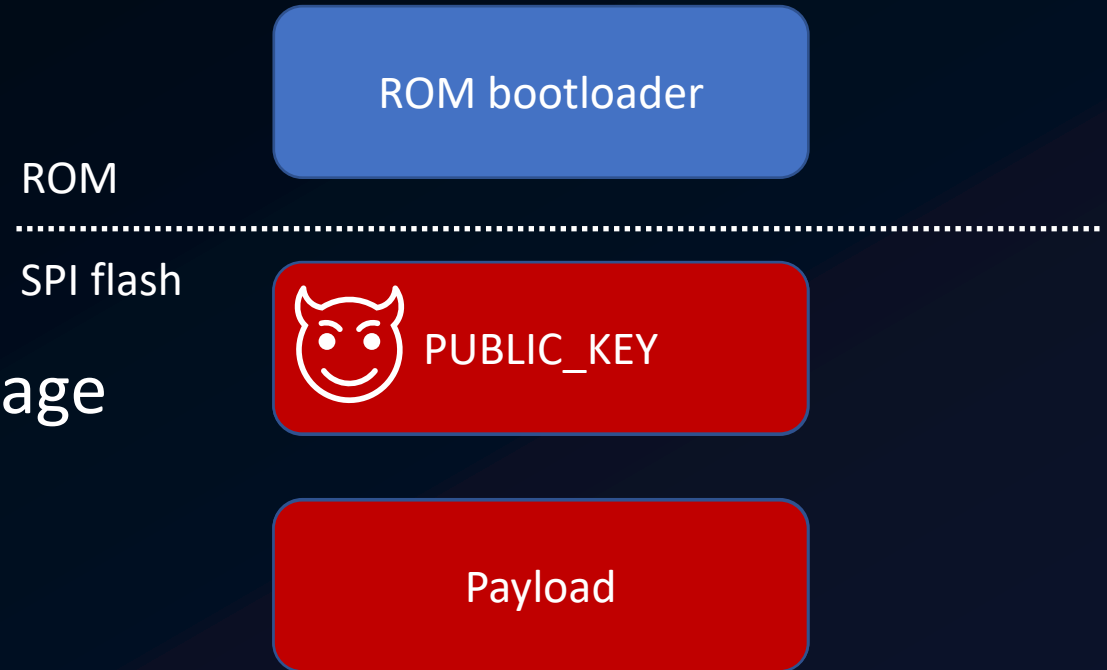
SPI CS: trigger and to determine a successful glitch

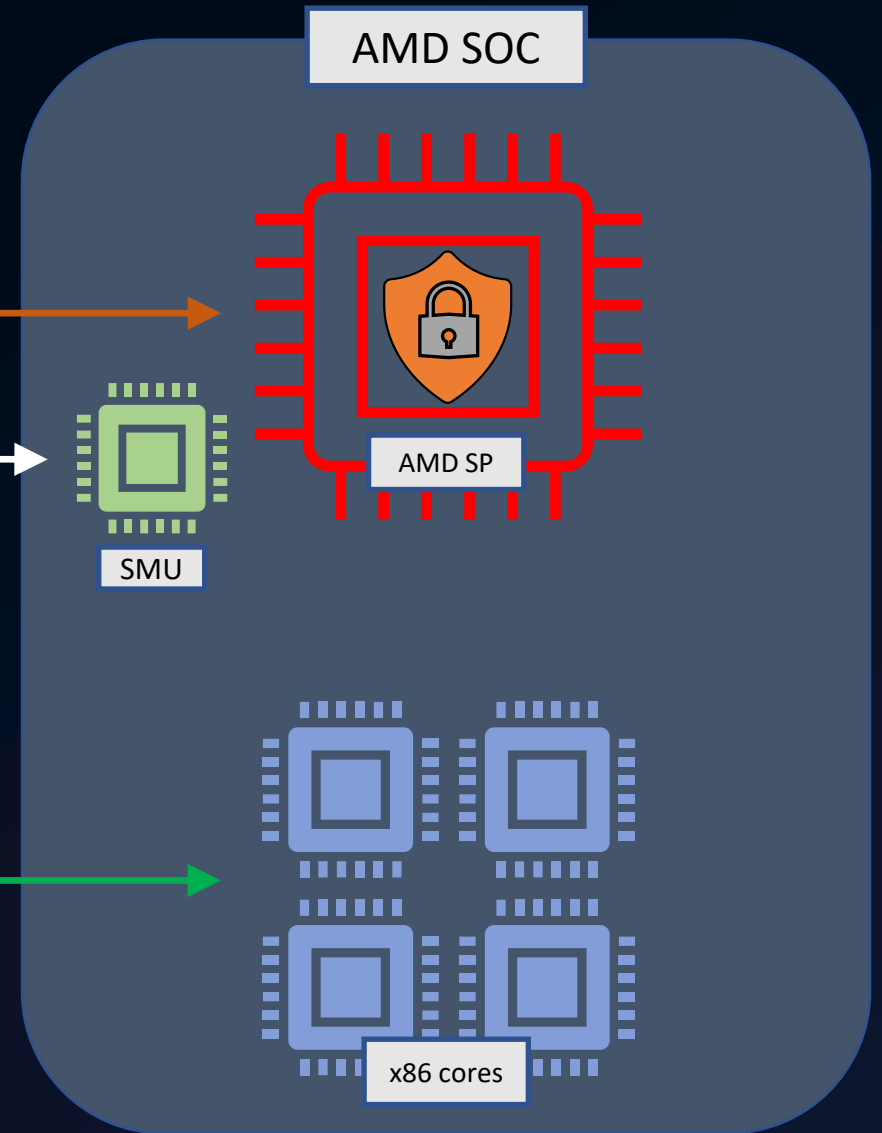
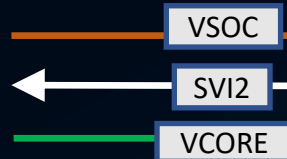
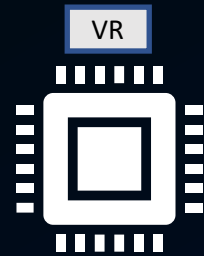
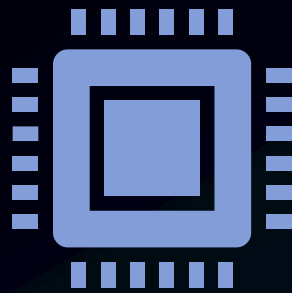


ATTACK OVERVIEW

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

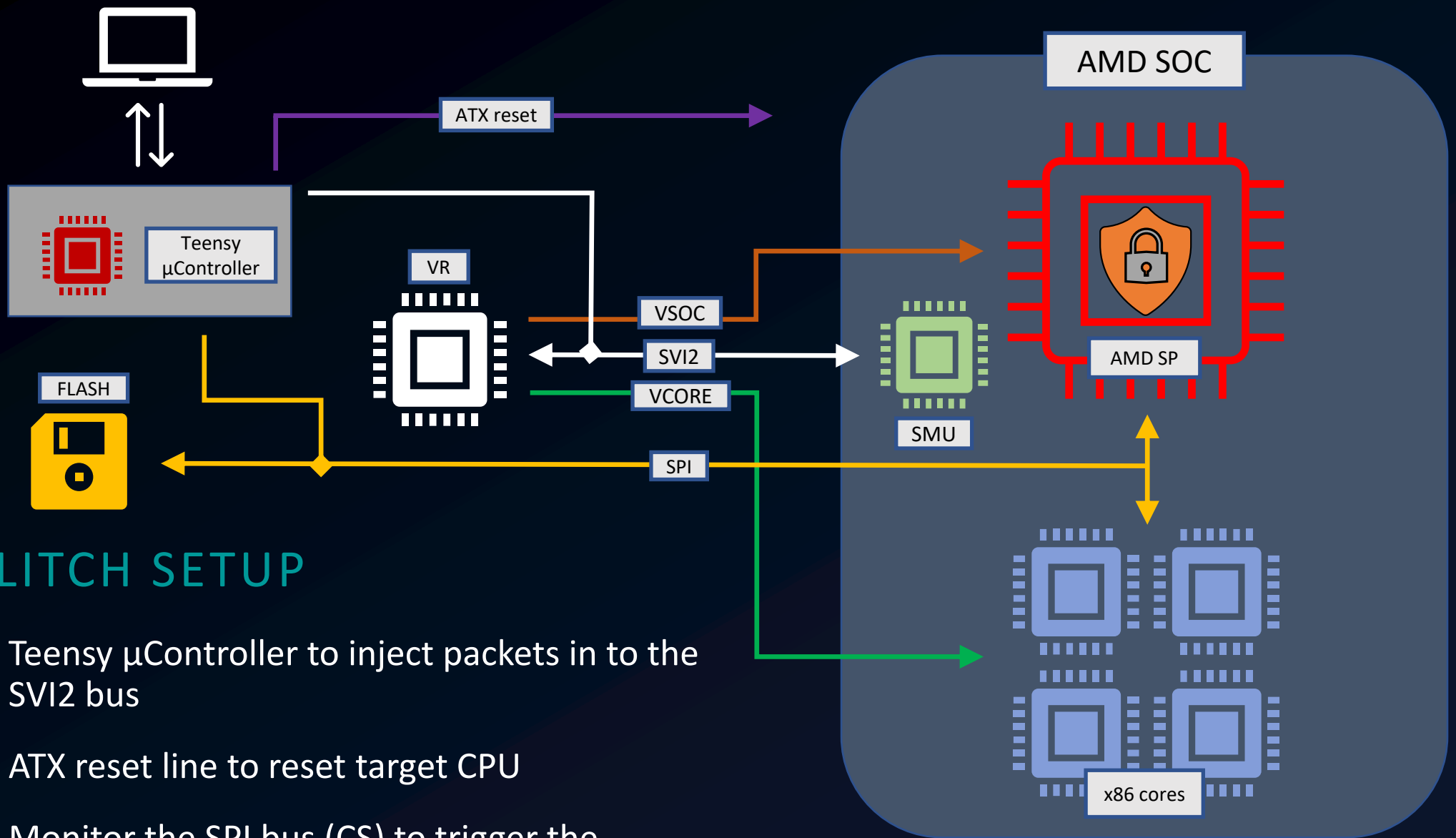
1. Replace AMD_PUBLIC_KEY in UEFI image
2. Replace PSP_FW_BOOT_LOADER component with payload
3. Sign payload with custom key
4. Glitch key verification





DYNAMIC VOLTAGE CONTROL

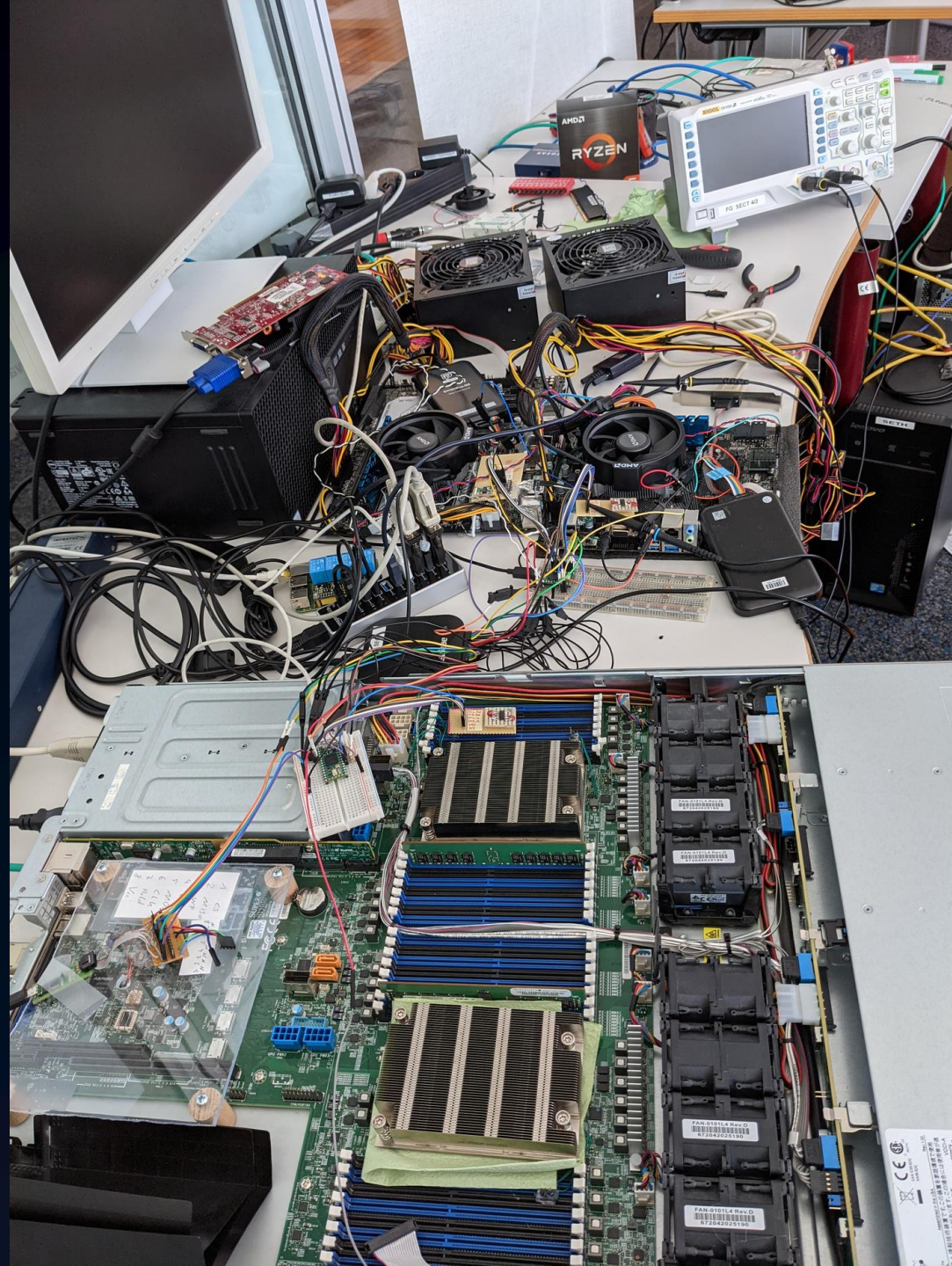
- SMU monitors SOC and uses the SVI2 bus to communicate with an SOC-external voltage regulator
- SVI2 allows to control two voltage domains per VR
- Ryzen uses single VR, Epyc dedicated VR for each domain



GLITCH SETUP

- Teensy μ Controller 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.

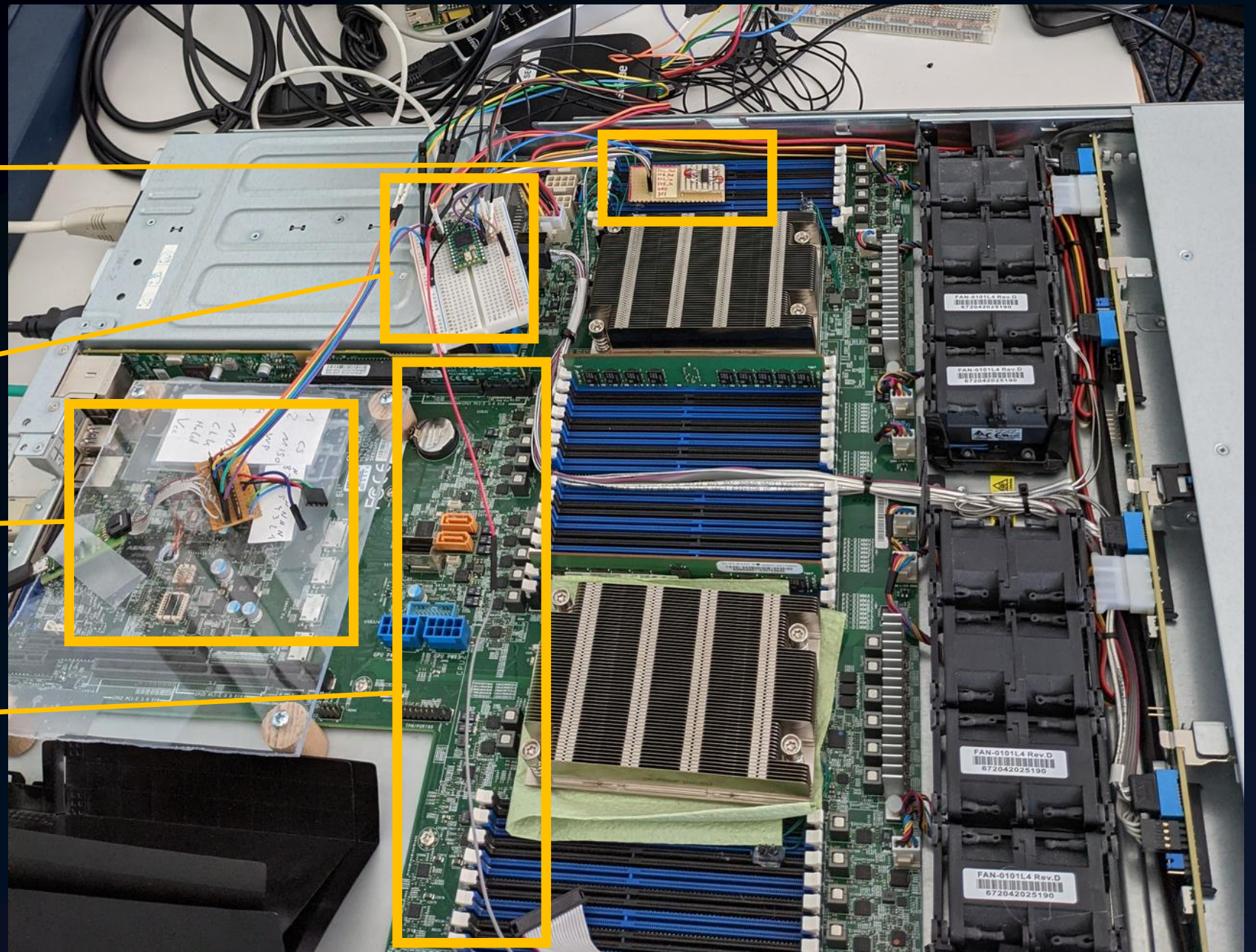


SVI2 access

Teensy
 μ Controller

SPI access

ATX reset

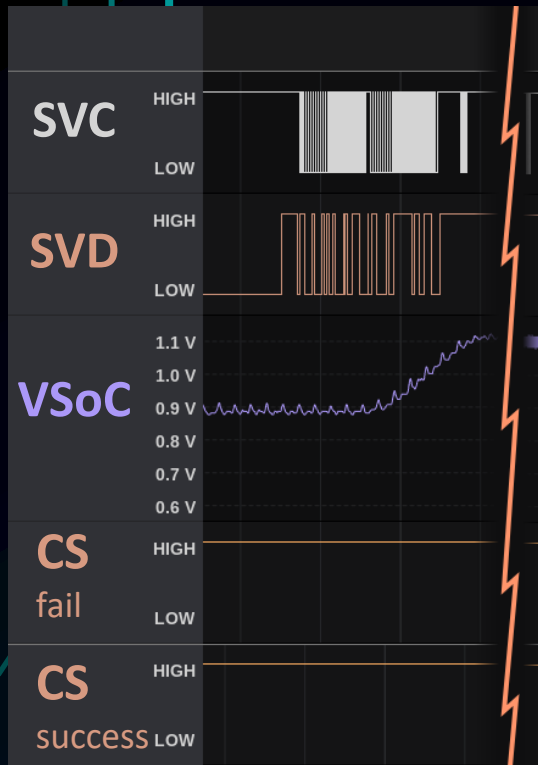


Glitch steps



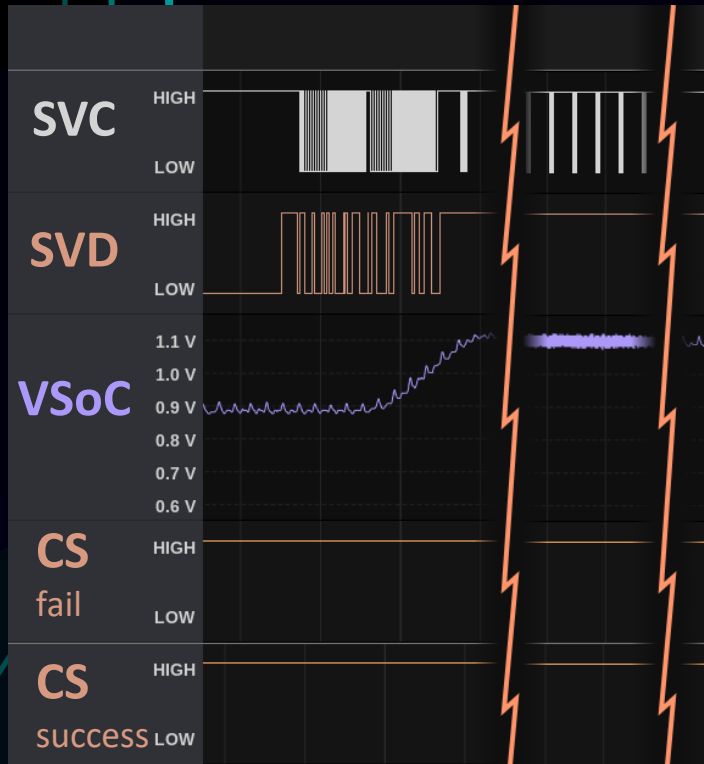
- SVI2 SVC – clock, CPU/VR (shared)
- SVI2 SVD – data from CPU, pulled low when inactive
- VSoC – target input voltage
- SPI CS – SPI's chip-select signal (successful/failed pubkey verification)

Glitch steps



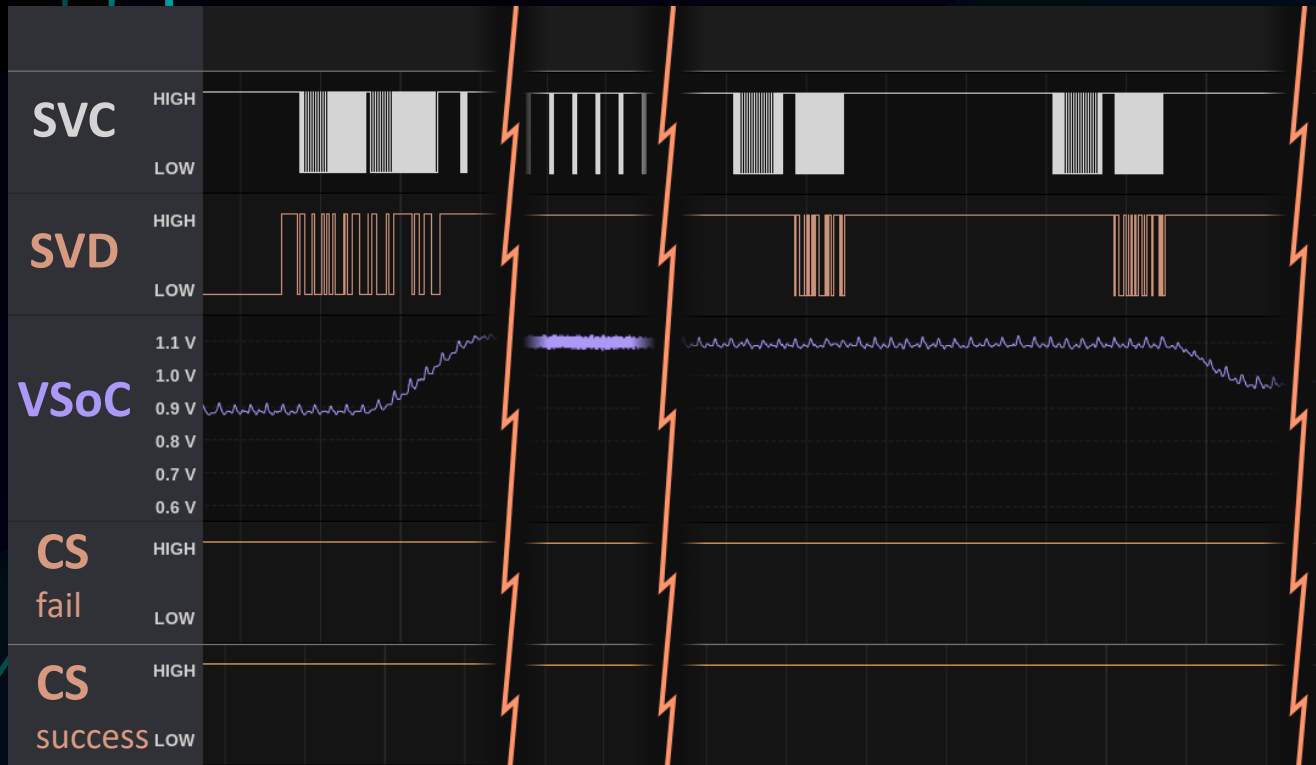
- SVI2 SVD: becomes high -> start attack logic
- CPU initially configures voltage

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



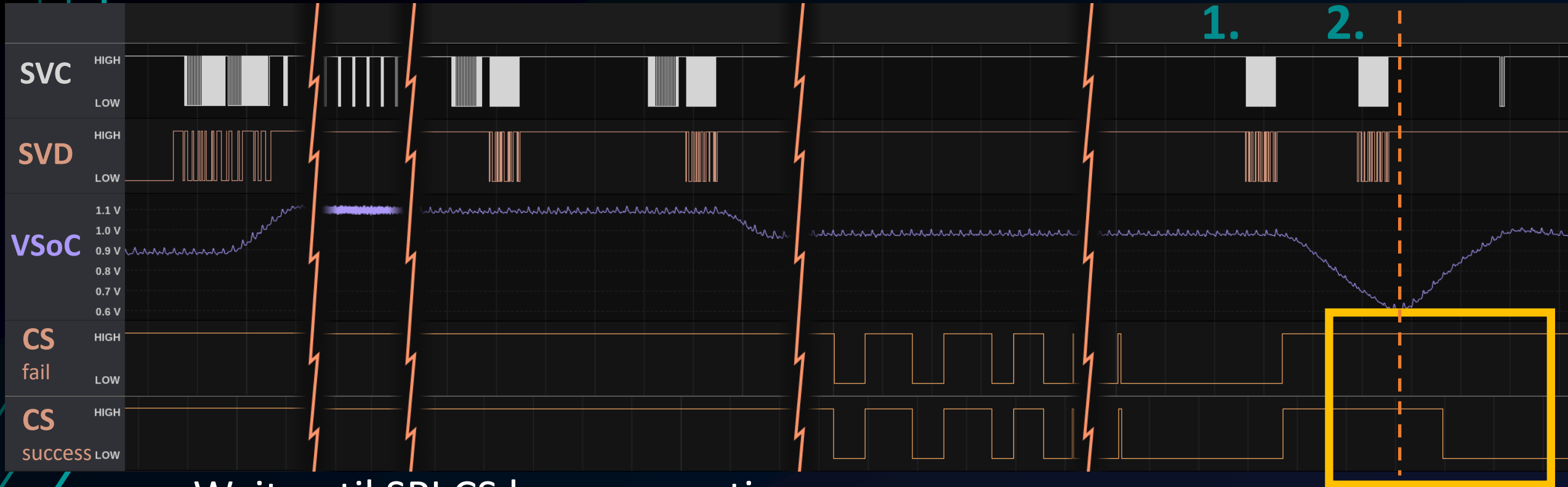
- Wait until SPI CS becomes active

Glitch steps

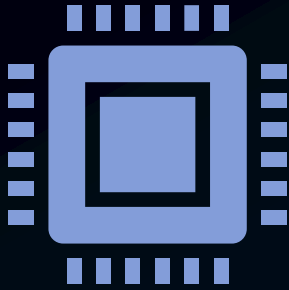


- Wait until SPI CS becomes active
- Count # of CS level changes to time glitch

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



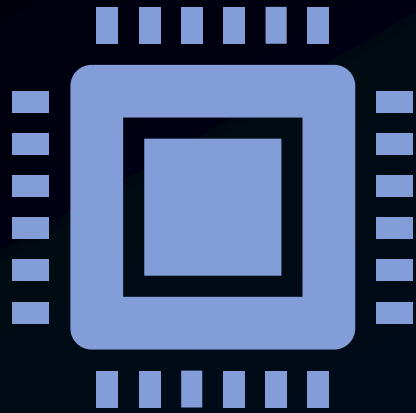
RESULTS

- 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>

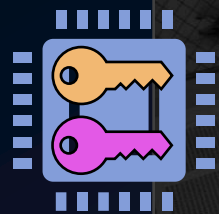


SEV: AMD-SP

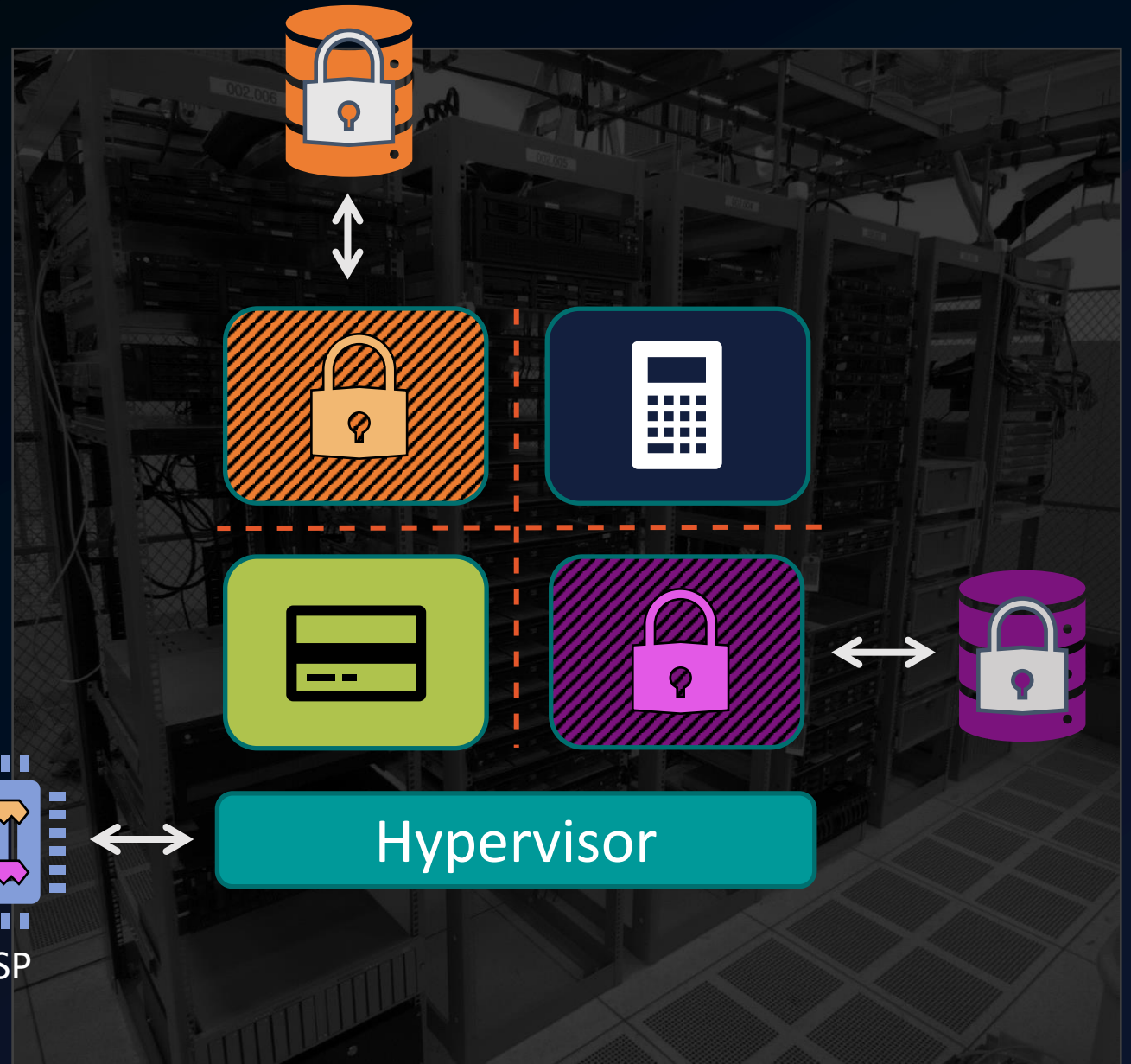
Hosts the SEV firmware that implements the SEV API

Memory encryption keys

Endorsement keys (CEK / VCEK)



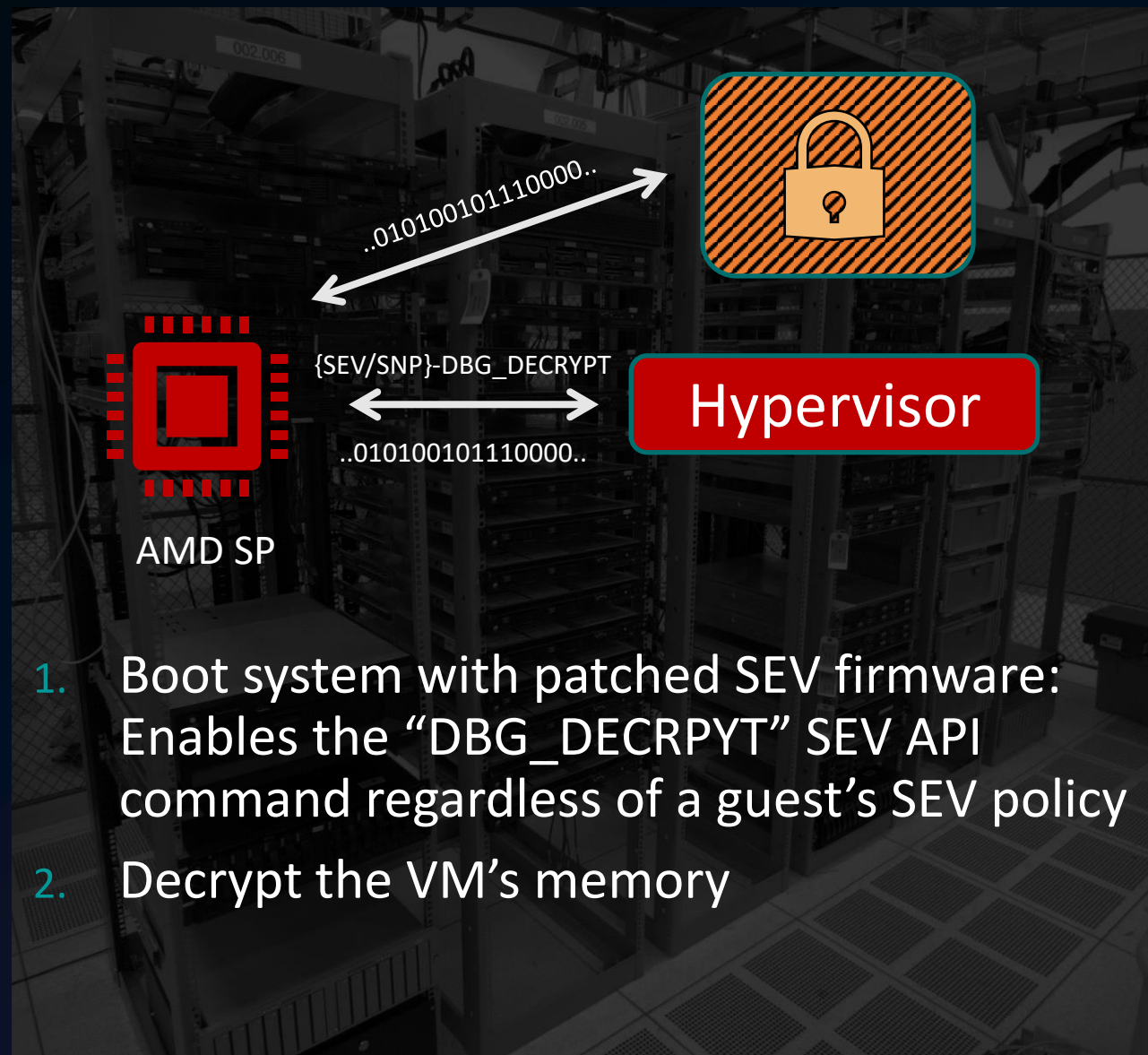
AMD SP

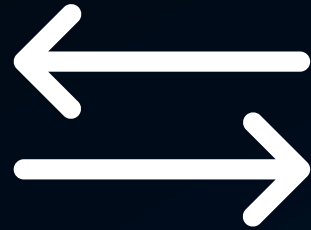




MALICIOUS CLOUD ADMINISTRATOR

- Debug override

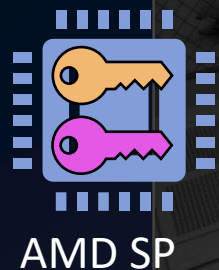




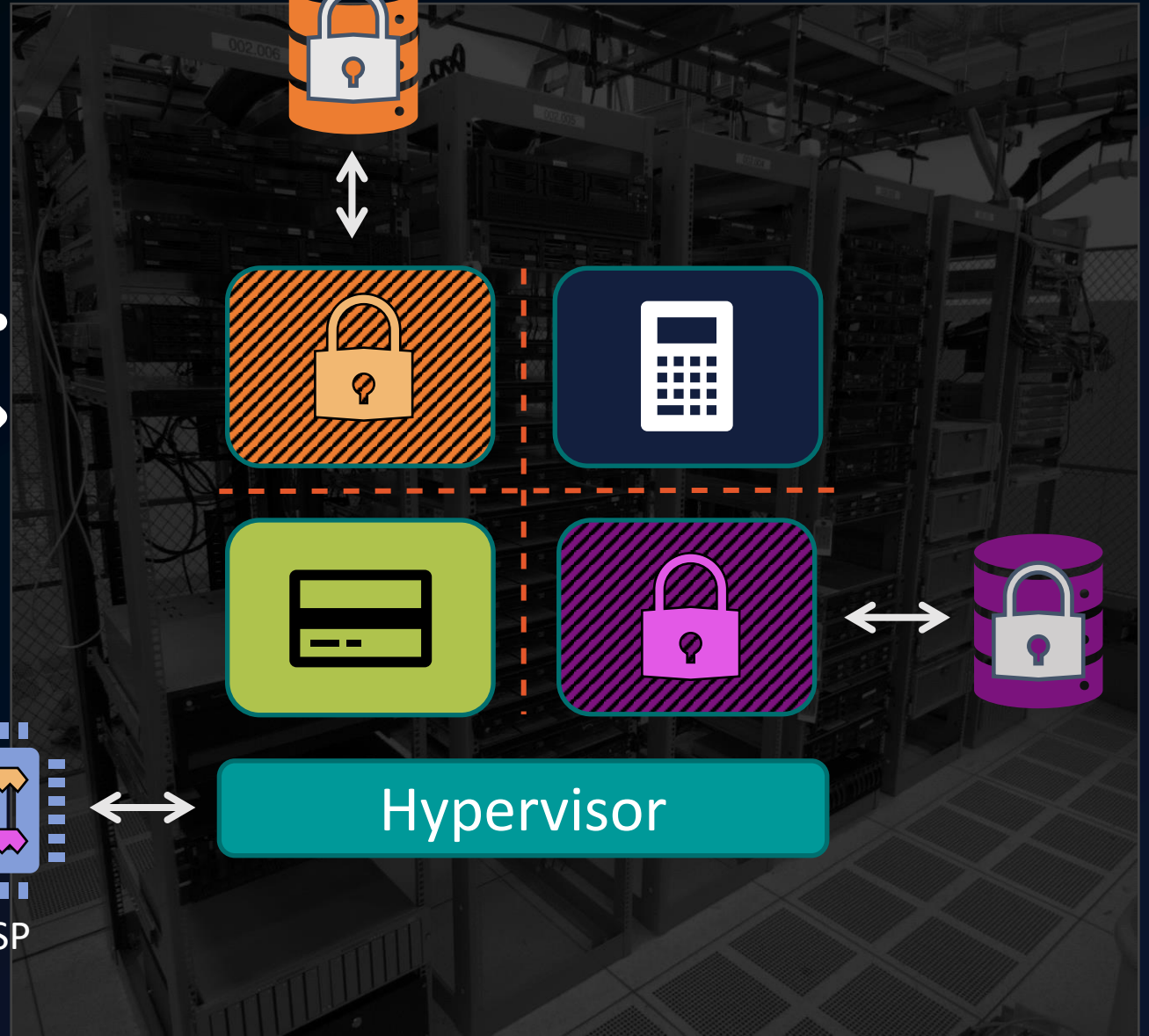
SEV REMOTE ATTESTATION

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

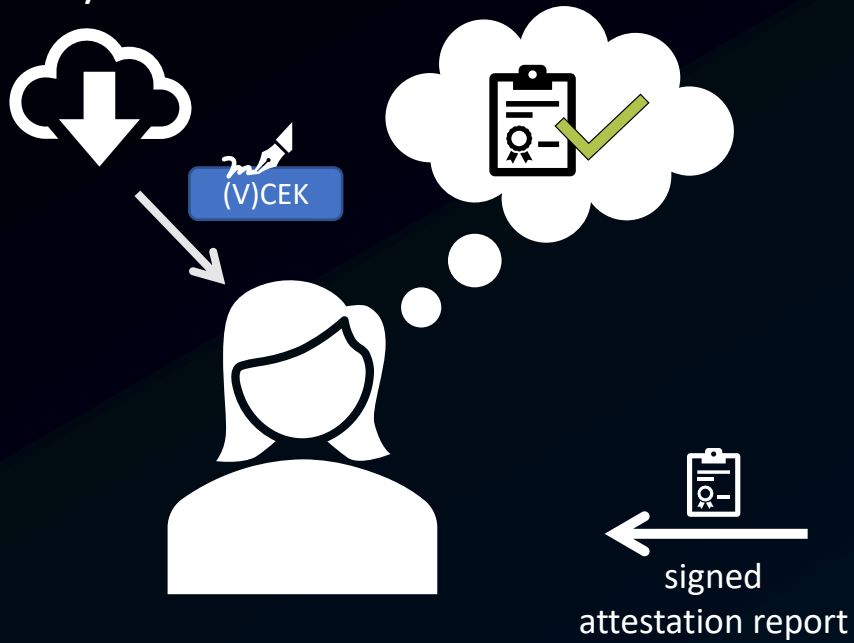
Customer: Is my VM deployed on a genuine AMD system with SEV protection in place?



AMD SP

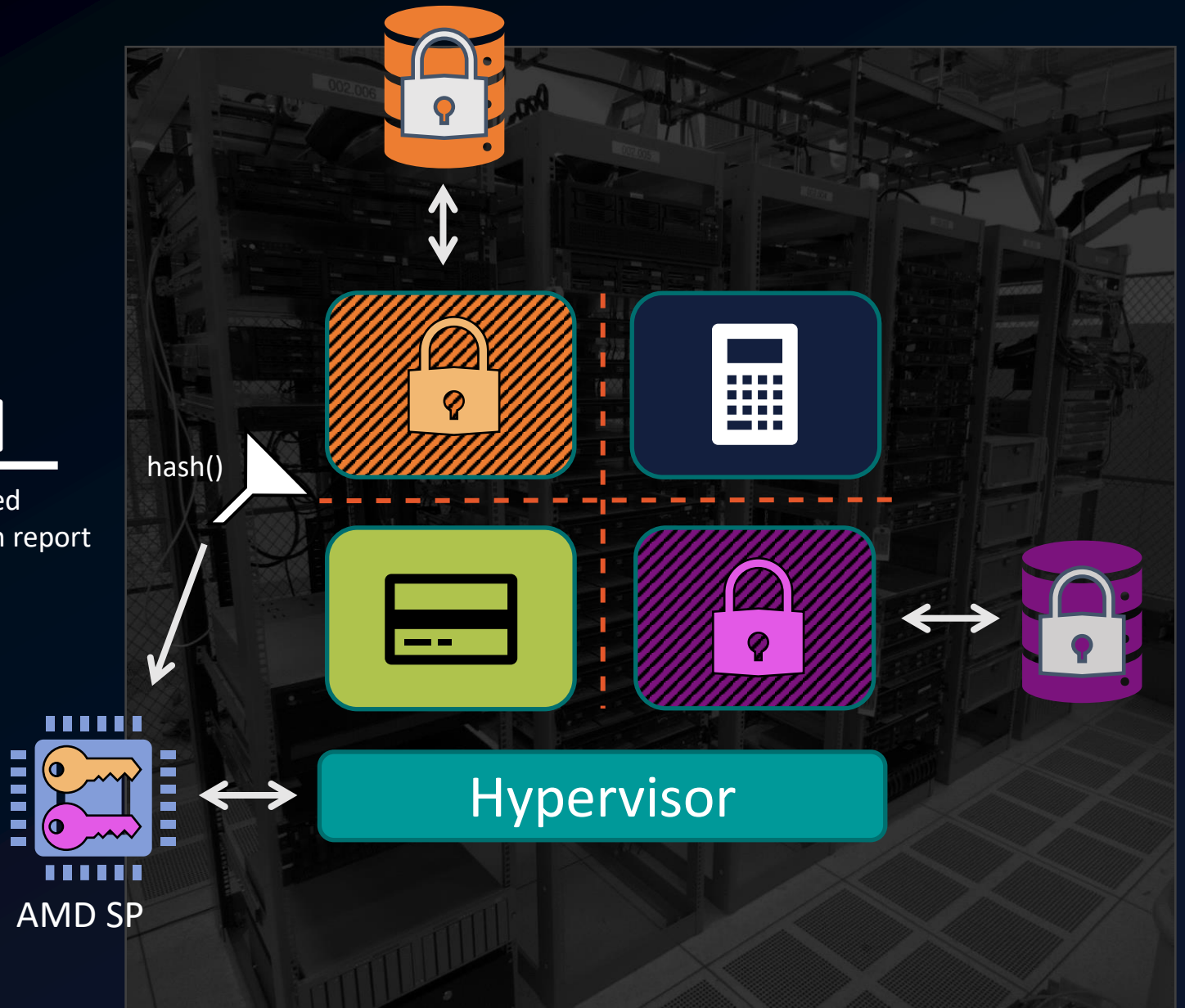


AMD Keyserver

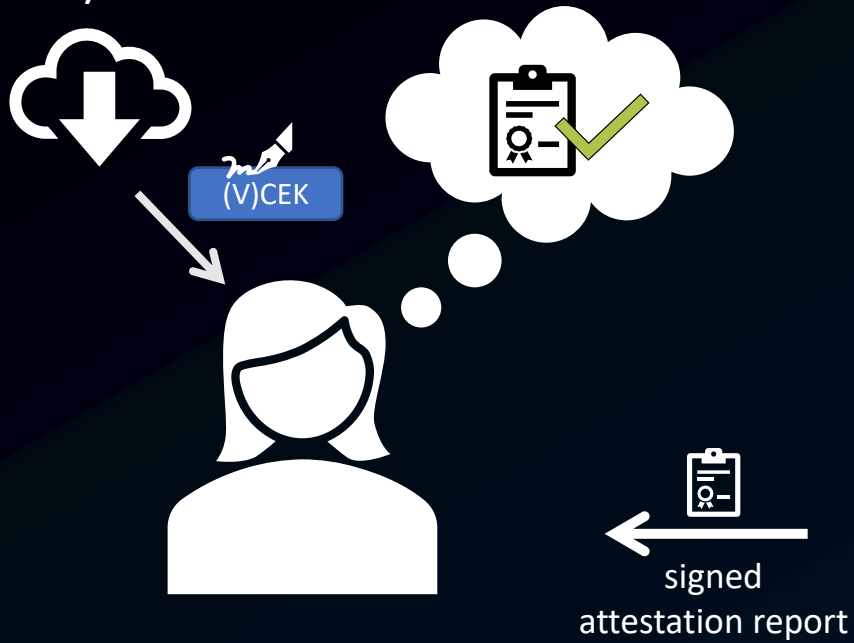


SEV REMOTE ATTESTATION

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

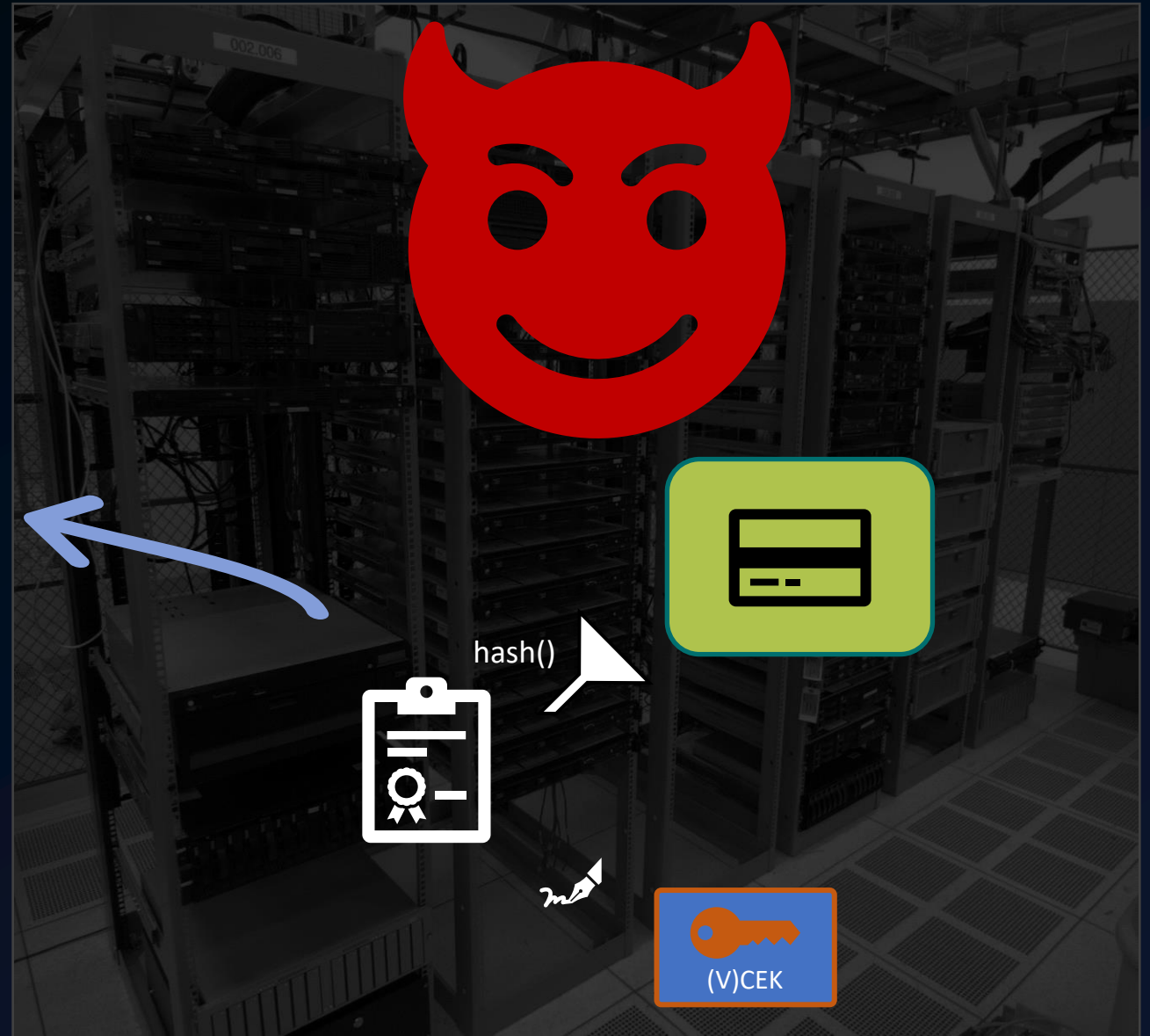


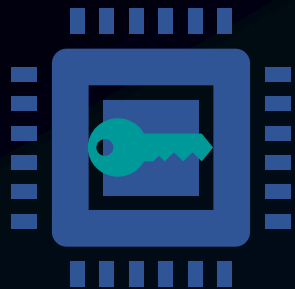
AMD Keyserver



SEV REMOTE ATTESTATION

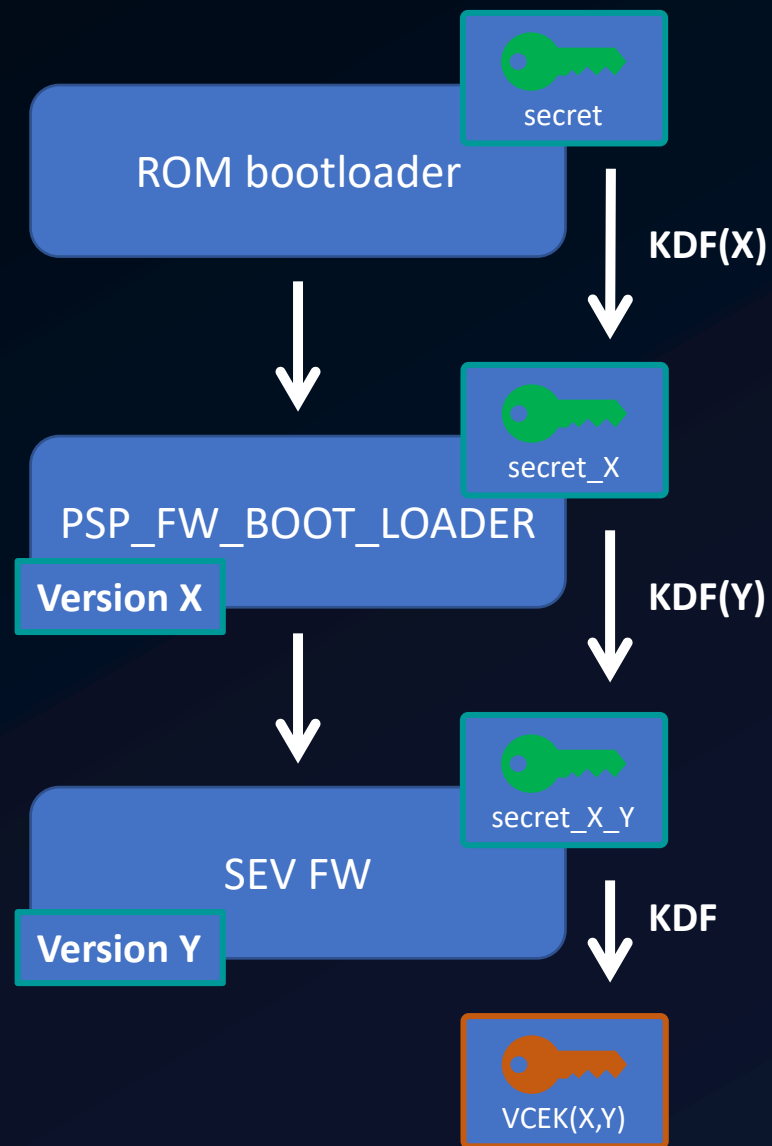
Extracted endorsement keys allow an attacker to, e.g., fake the presence of SEV!

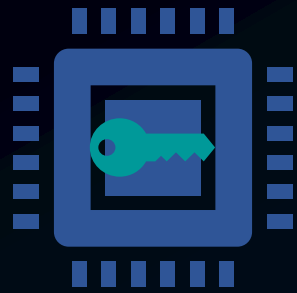




VERSIONED CEK (VCEK) SIMPLIFIED

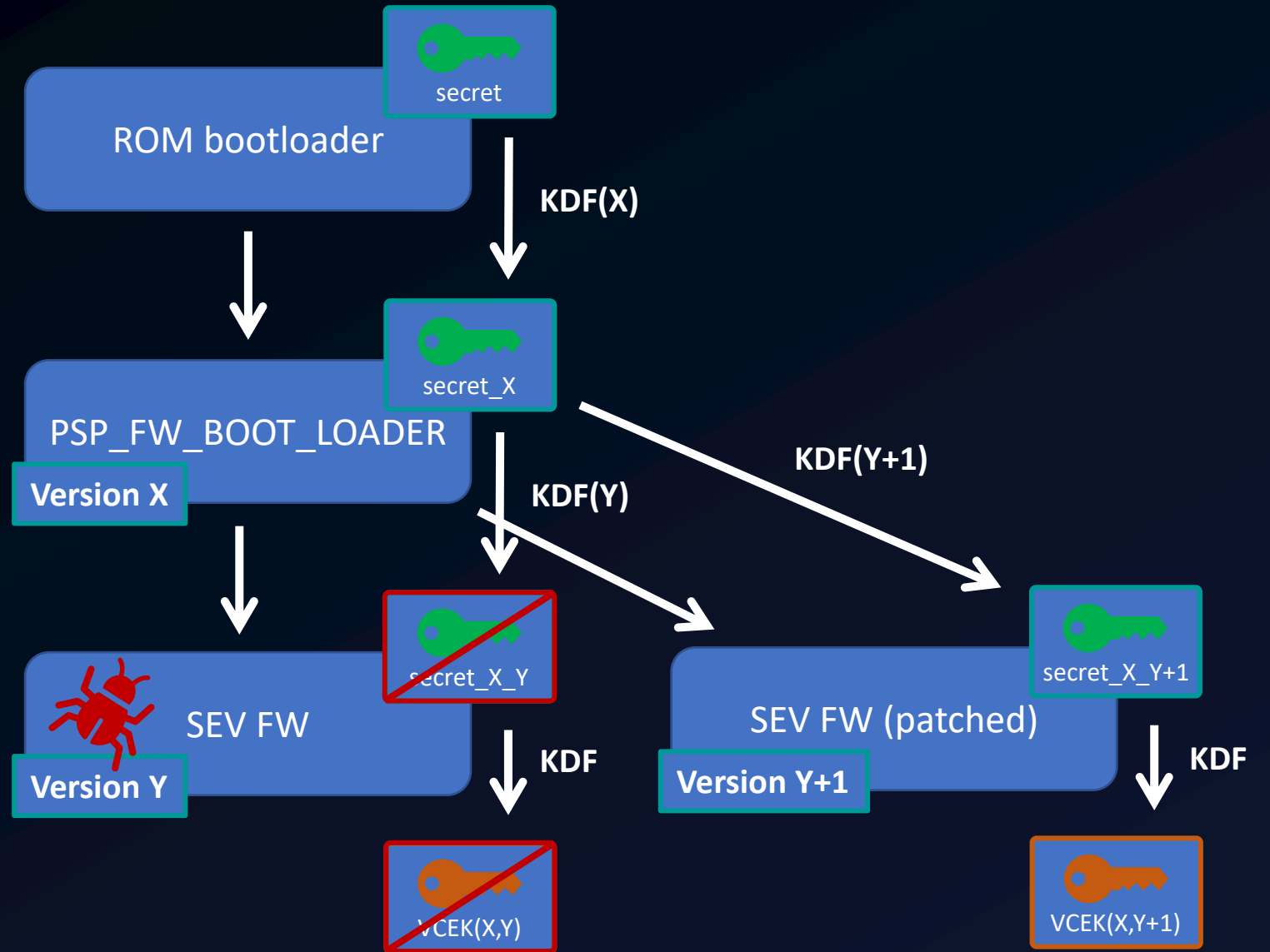
“[VCEK is] derived from chip-unique secrets and current TCB version”

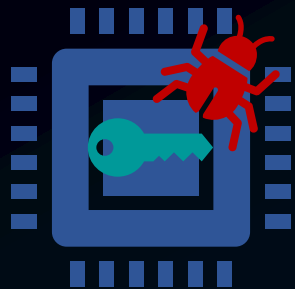




VCEK ATTACK

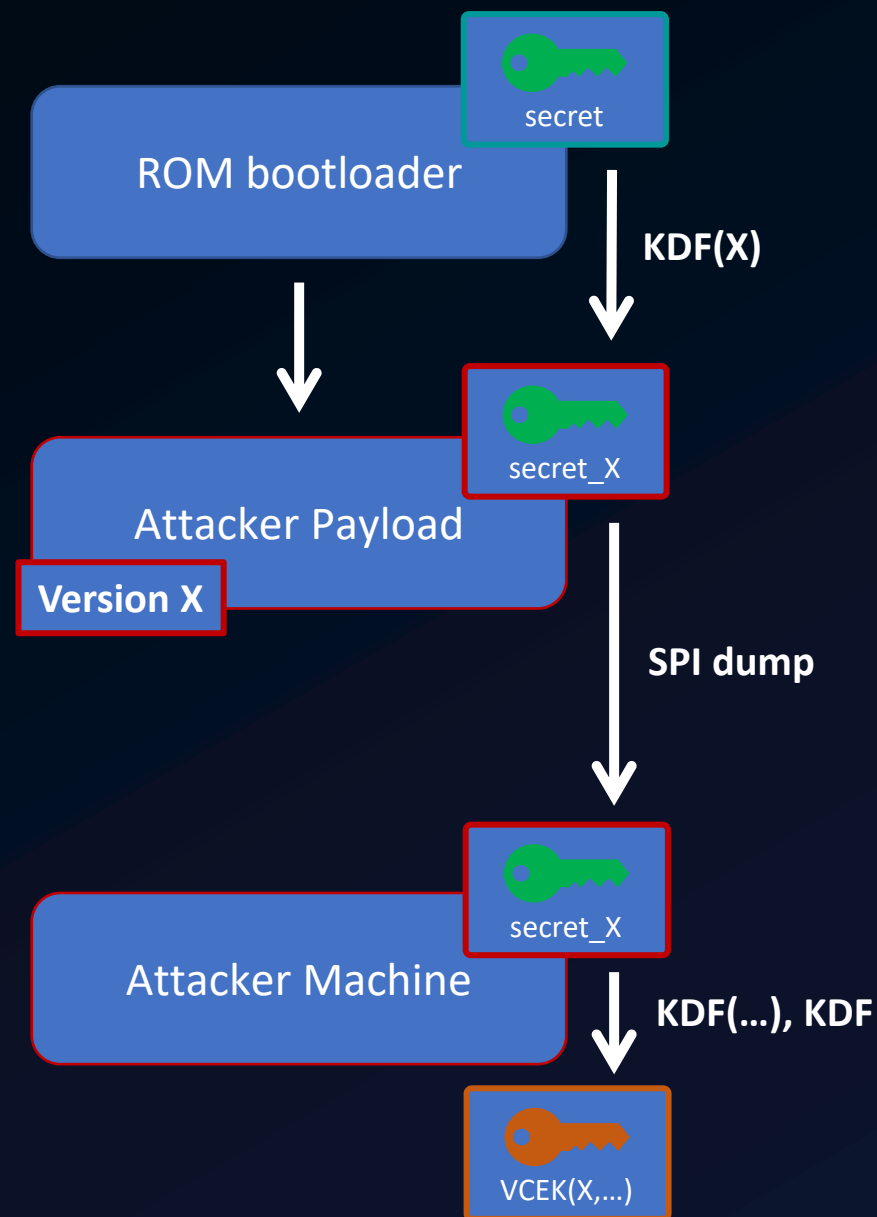
What if there is a bug?





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

