Detection & Mitigation of Ciphertext Side-Channels in Trusted Execution Environments

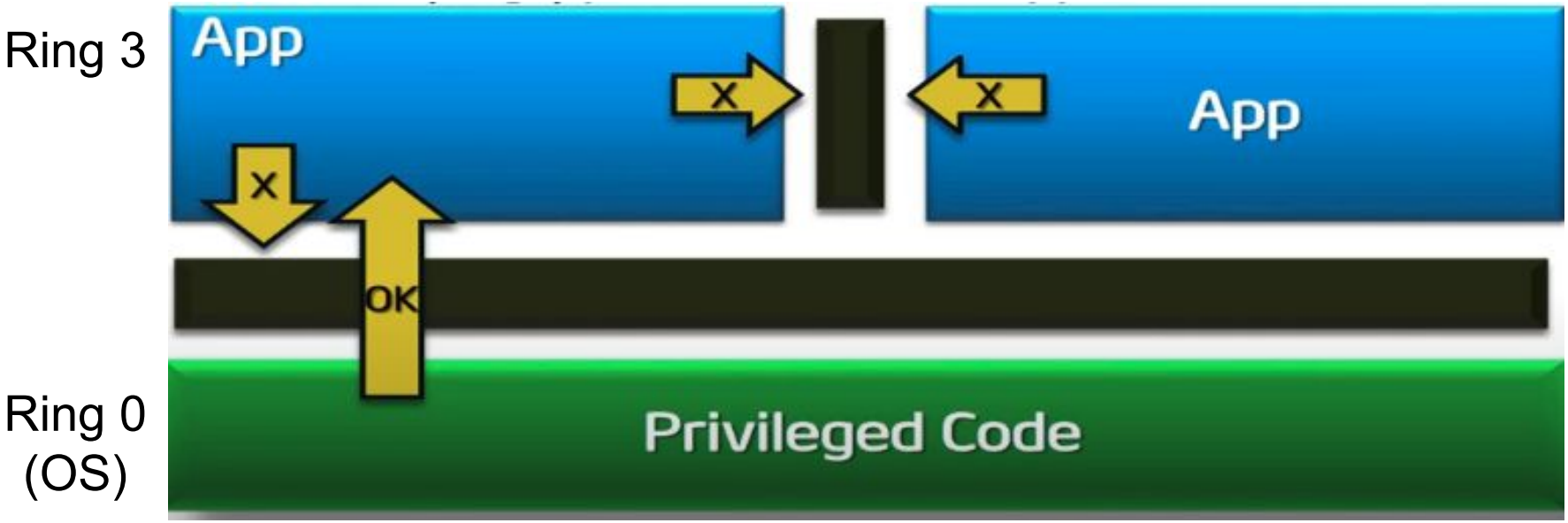
Viraj Lunani, Vedant Iyer, Wei-Lun Huang,
Taeyoon Kim, and Aayush Singh

Outline

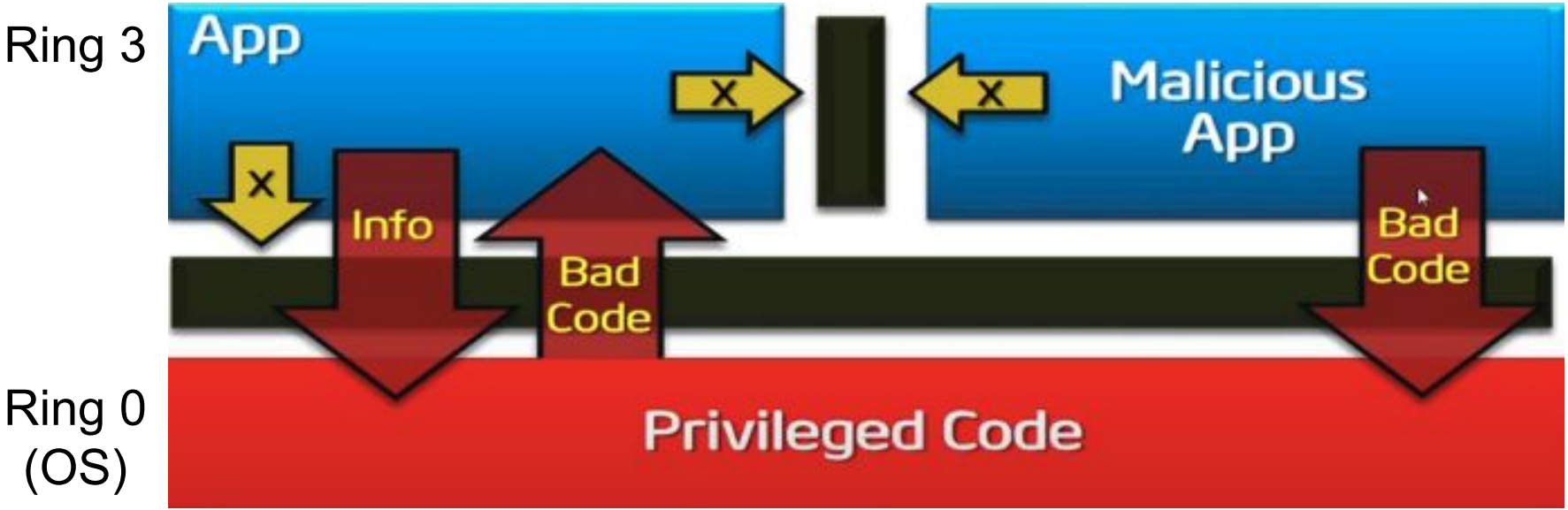
- [Background] Trusted Execution Environments
- [Background] Ciphertext Side Channels
- [Comparison] CipherH vs. Cipherfix
- [CipherH] Automated Detection of Ciphertext Side-channel Vulnerabilities in Cryptographic Implementations
- [Cipherfix] Mitigating Ciphertext Side-Channel Attacks in Software

Trusted Execution Environments (TEE)

Existing Ring Architecture Problem



Existing Ring Architecture Problem



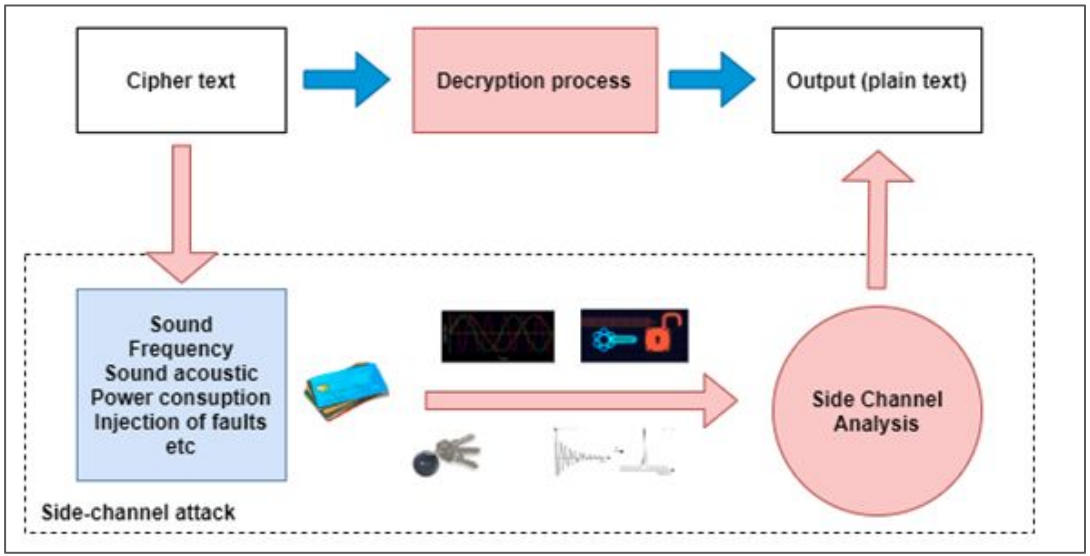
Trusted Execution Environment (TEE)

- Assume the OS is compromised.
- Provide isolation from the hypervisor.
  - Access Rights
  - Cryptography

Issues with TEEs

- Shared Hardware = Shared Resources
- Exploitable Side Channels

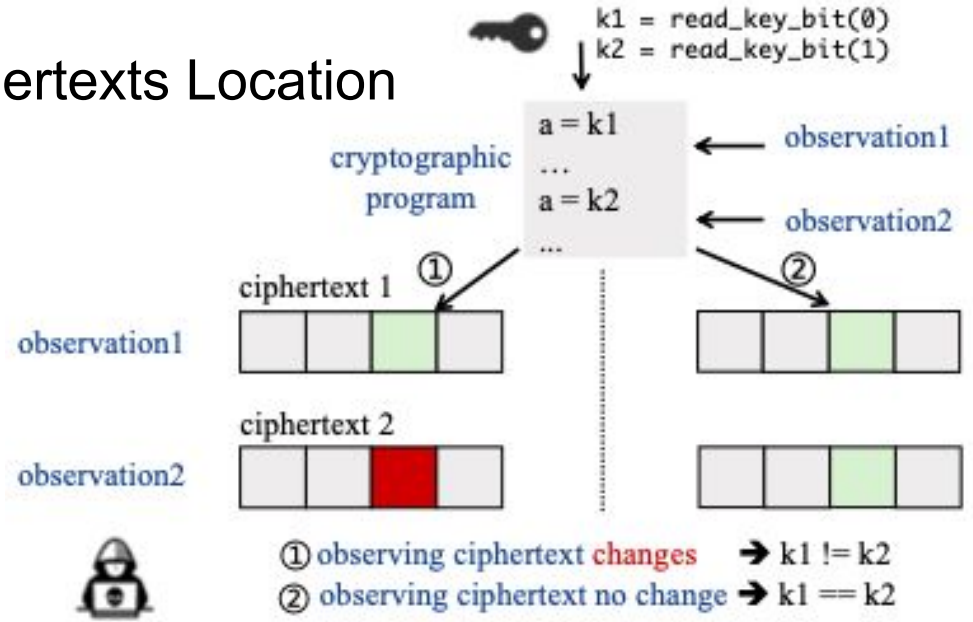
Side-Channel
Attack Overview



Ciphertext Side Channels

Deterministic Encryption

- Fixing at One Memory Cell
- Identical Plaintexts → Ciphertexts Location



[1] CIPHERH: Automated Detection of Ciphertext Side-channel Vulnerabilities in Cryptographic Implementations

CT: Ciphertext

PT: Plaintext

Dictionary and Collision Attacks

- Dictionary Attacks
  - All Possible CTs Observed
  - CT-PT Relations Inferred
- Collision Attacks
  - Before/After a Memory Write
  - Identical vs. Different CTs

```
cswap(p, q, b):
  c = ~(b - 1);    // b = 0 -> c = 00...00
  t = c & (p ^ q);
  p ^= t;
  q ^= t;
```

(a) Constant-time swap of p and q, depending on bit b.

	Ciphertext of p	
b	before cswap	after cswap
0	e4c80f2a	e4c80f2a
1	e4c80f2a	aa2f2a61

(b) Ciphertext of p, before and after calling cswap.

[2] Cipherfix: Mitigating Ciphertext Side-Channel Attacks in Software

CipherH vs. Cipherfix

Patching Ciphertext Side-Channels

- At the Software Level
  - AMD SEV cannot afford hardware patches.
- In Constant-Time Crypto Software
  - Different Secrets → Identical Control Flows + Memory Cells Accessed
- CipherH for Side-Channel Detection
  - USENIX Security 2023
- Cipherfix for Side-Channel Mitigation
  - USENIX Security 2023

Approaches

Paper	CipherH	Cipherfix
Stage 1	Dynamic Taint Analysis, Instrumented at the LLVM-IR Level	Dynamic Taint Analysis, Instrumented at the Binary Level
Outputs	Tainted Functions	Tainted Memory Accesses (Instructions + Addresses)
Stage 2	Intra-procedural Symbolic Execution + …	Masking the Tainted Memory Writes
Outputs	Vulnerable Instructions	Protected Software Binary

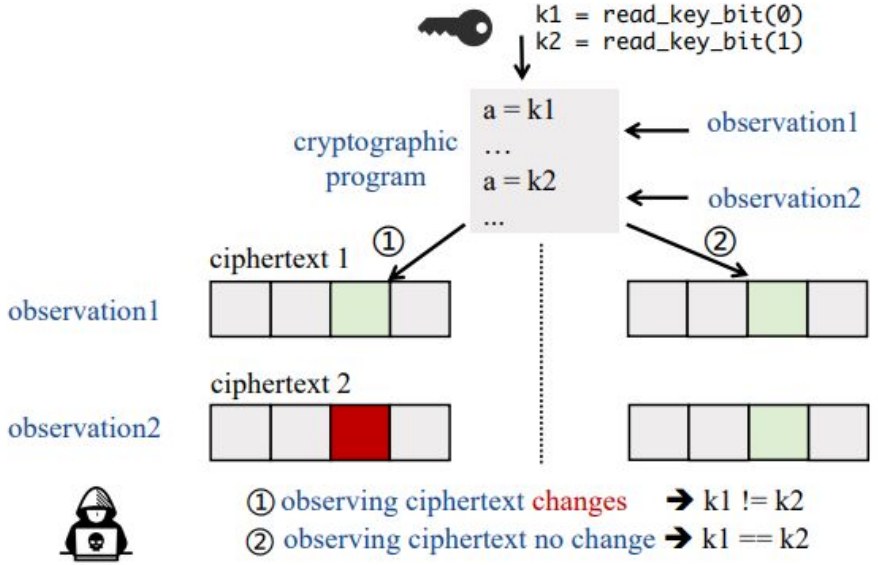
Authors

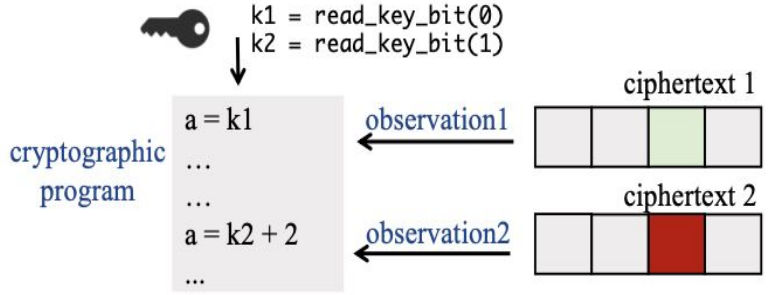
- CipherH Authors
  - ≈ CipherLeaks Authors
  - ≈ ½ Authors of the S&P'22 Follow-up
- Cipherfix Authors
  - ≈ ½ Authors of the S&P'22 Follow-up
- Advantages
  - Knowing Ciphertext Side Channels Better
  - More Familiar with Implementations: e.g., CipherH
  - Expensive & Uncommon AMD EPYC Processors, with Root Privileges

CipherH

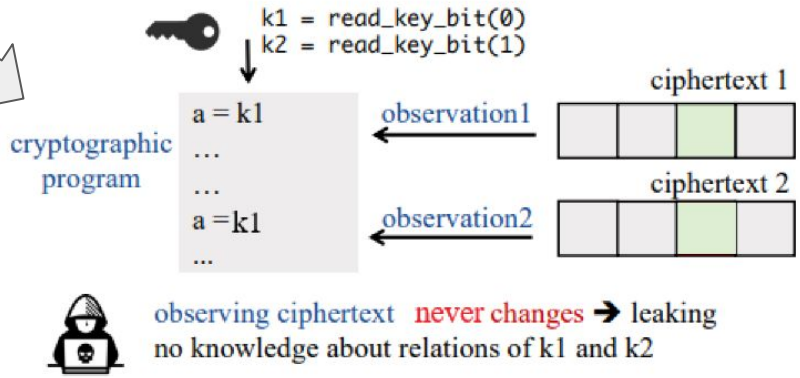
Automated Detection of Ciphertext Side-channel Vulnerabilities in Cryptographic Implementations

Safe Cases





$$\forall k_1, k_2 \in K, W_1(k_1) \neq W_2(k_2)$$



$$\forall k_1, k_2 \in K, W_1(k_1) = W_2(k_2)$$

[1] CIPHERH: Automated Detection of Ciphertext Side-channel Vulnerabilities in Cryptographic Implementations

Path Constraint

$$\exists k_1, k_2, k_1', k_2' \in K, W_1(k_1) = W_2(k_2) \wedge W_1(k_1') \neq W_2(k_2')$$

$$\exists k_1, k_2, k_1', k_2' \in K, W_1(k_1) = W_2(k_2) \wedge W_1(k_1') \neq W_2(k_2') \wedge C$$

- Unsafe Case: two different executions following the same path
  - In one execution, the two memory writes give identical ciphertexts.
  - In the other, the two memory writes give different ciphertexts.
- Some program may not cover all two memory accesses
- C = Conjunction of All Branch Conditions: from program entry to the second memory write.
  - If C contains only public symbols, two executions mentioned above follow the same path.
  - If C contains secret symbols, secret symbol k is renamed and constraint solver will search for secrets used by two executions.

[1] CIPHERH: Automated Detection of Ciphertext Side-channel Vulnerabilities in Cryptographic Implementations

Evaluation/Weakness

Implementation	Algorithm	Opt.	Intraprocedural Symbolic Execution		Interprocedural Symbolic Execution		Function
			(Vulnerable/Analyzed) Functions	Vulnerable Program Points	(Vulnerable/Analyzed) Functions	Vulnerable Program Points	(Tainted/Covered)
WolfSSL 5.3.0	ECDSA	-O2	3/53	6	1/2	12	53/92
WolfSSL 5.3.0	RSA	-O2	3/30	14	3/5	30	30/78
OpenSSL 3.0.2	ECDSA	-O3	4/68	6	4/11	29	68/1061
OpenSSL 3.0.2	RSA	-O3	9/142	53	11/38	55	142/1296
MbedTLS 3.1.0	ECDH	-O2	2/37	2	2/5	5	37/87
MbedTLS 3.1.0	RSA	-O2	2/39	2	4/7	22	39/83
Total			23/369	83	25/68	153	369/ 2697

- Inter-procedural → False Positives in Some Cases
  - **142** findings are **true positives** out of **153** findings (142/153)
  - They had **9** **false positives** (9/153)
- Manual Checking - For intra-procedural, we use automated process, but for inter-procedural, the developers have to check manually
- Would be problematic if no prior experience or if the library is too big

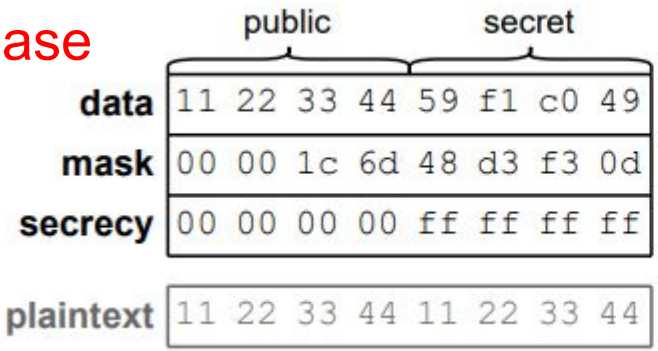
[1] CIPHERH: Automated Detection of Ciphertext Side-channel Vulnerabilities in Cryptographic Implementations

Cipherfix

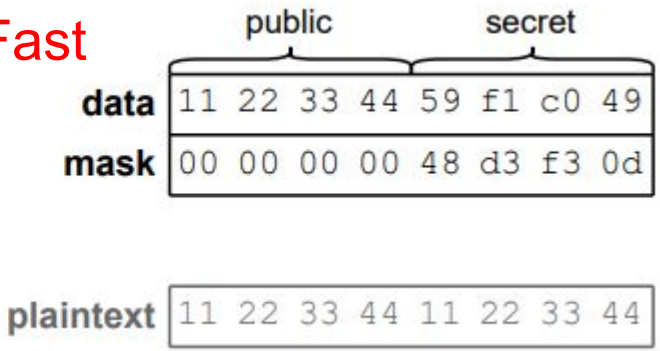
Mitigating Ciphertext Side-Channel Attacks in Software

How do we encrypt memory randomly?

CF-Base



CF-Fast



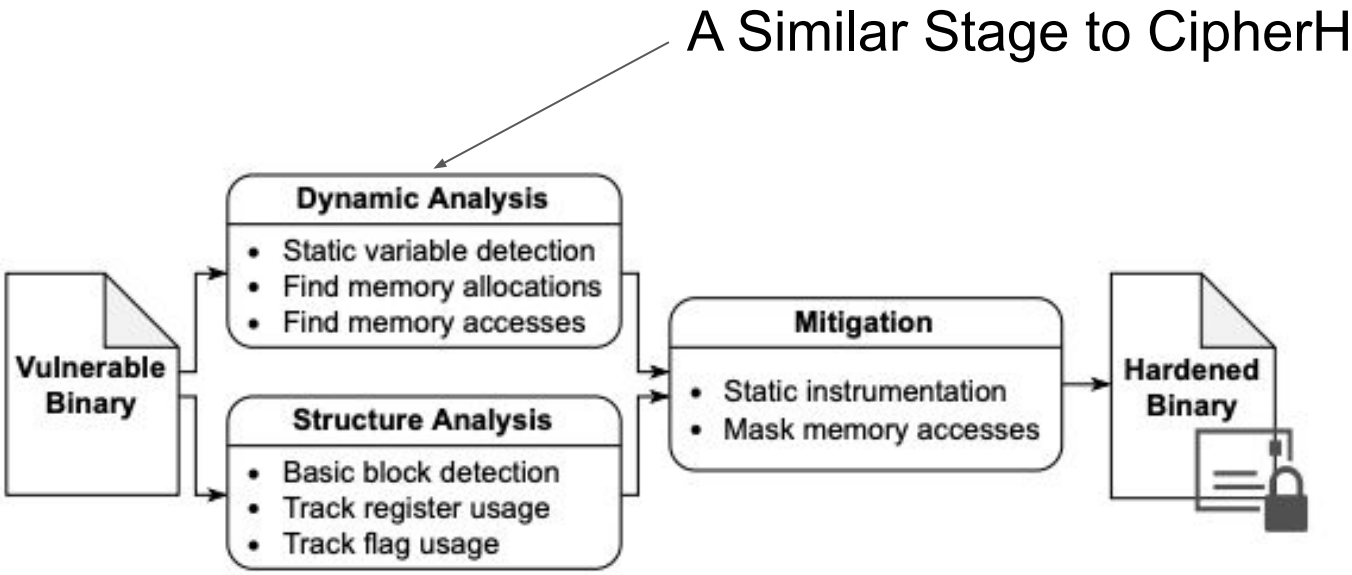
	AES
FAST	2.4x
BASE	3.9x
ENHANCED	5.1x

CF-Enhanced reduces collisions by guaranteeing that all data and masks are at least length-*w*'.

Smaller memory accesses will be modified to access surrounding bytes.

[2] Cipherfix: Mitigating Ciphertext Side-Channel Attacks in Software

Cipherfix process



[2] Cipherfix: Mitigating Ciphertext Side-Channel Attacks in Software

Pros

- Apply a random mask to each memory write.
- Binary-Level Implementation
  - Static + Dynamic
  - Compilation Not Required

Cons

- Assumption: The control flow does not depend on secrets.
- Large Runtime Overhead
- CF-Fast/Base is not as secure as CF-Enhanced.
  - Mask Collision

Readings

Questions/Feedback?

- CipherLeaks [USENIX Security 2021]
  - Ciphertext (CT) Side Channels Found
- A Systematic Look at Ciphertext Side Channels on AMD SEV-SNP
  - Follow-up of CipherLeaks [IEEE S&P 2022]
- [1] CipherH [USENIX Security 2023]
  - Automated CT Side-Channel Detection
- [2] Cipherfix [USENIX Security 2023]
  - Automated CT Side-Channel Mitigation