Achilles: A Formal Framework of Leaking Secrets from Signature Schemes via Rowhammer

USENIX Security 2025

Junkai Liang*, Zhi Zhang*, Xin Zhang*, Qingni Shen, Yansong Gao, Xingliang Yuan, Haiyang Xue, Pengfei Wu, Zhonghai Wu

A Threat to Signature Schemes

- Signature Scheme serves as backbones for network security

Network protocol

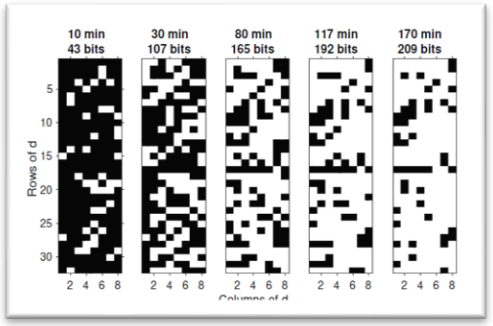
Blockchain Wallet

Software Updates

Secure Message

- Regarding the fault injection attack (FIA), most of the schemes are vulnerable

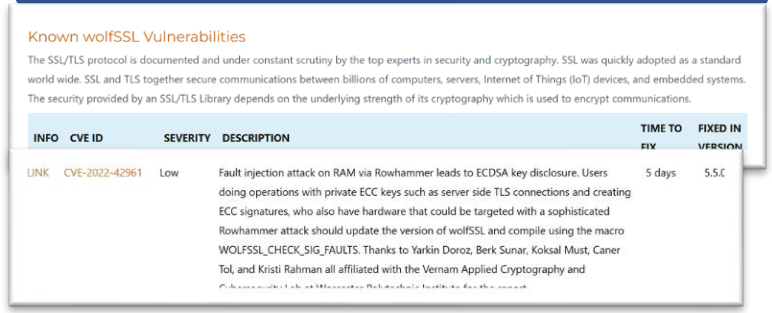
Full Key Recovery



FIA leads to full key recovery of ECDSA, RSA, etc [22].

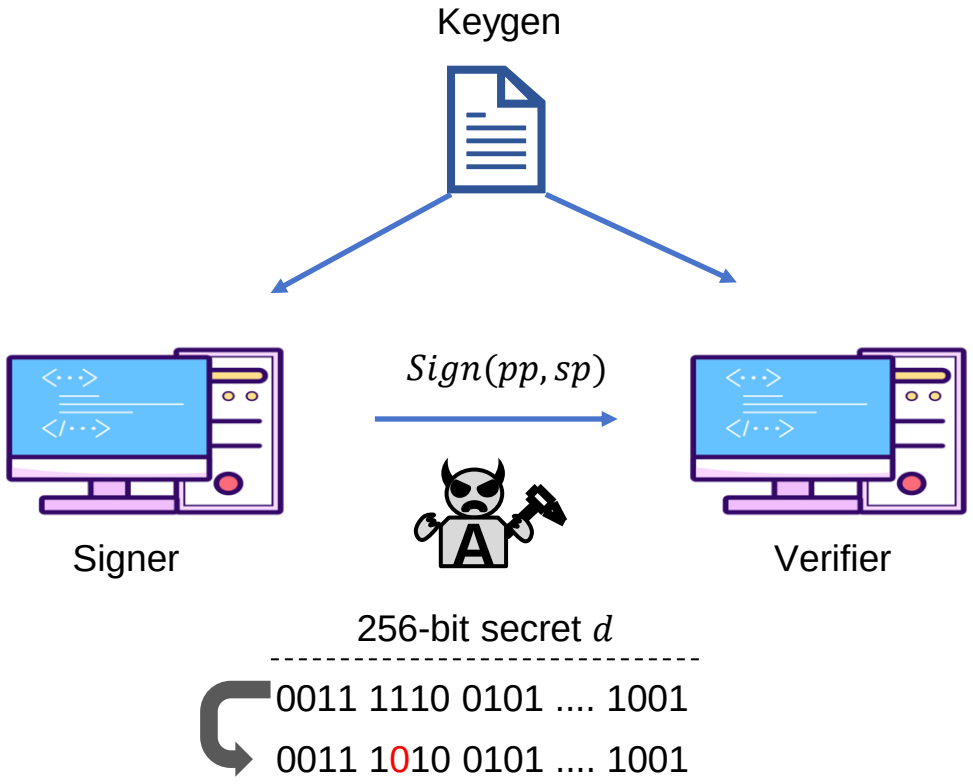
Post Quantum Attacks

Real Impact CVEs



Inject Faults into the parameters of Signature Schemes

- Parameters of the scheme are loaded into memory cells when signing.
  - The attacker exploits Rowhammer bugs to tamper with memory data.
  - Parameters of the scheme can be tampered.



Year	Paper	Scheme	Parameter
2024	Mus's[22]	ECDSA	d (sp)
2023	Saad's[24]	Dilithium	s (sp)
2020	Zane's[11]	RSA	d (sp)
2018	Mus's[23]	LUOV	d (sp)
2018	Podd's[23]	EdDSA	h (pp)
2016	Razavi's[10]	RSA	N (pp)

.....

Problems of Existing works

- Many signature schemes are not analysed, e.g., BLS, BB-Short, Elgamal, MLDSA, etc.

- It is still unclear how we utilise existing analysis for new signature schemes.

Problems of Existing works

- Many signature schemes are not analysed, e.g., BLS, BB-Short, Elgamal, MLDSA, etc.

- It is still unclear how we utilise existing analysis for new signature schemes.

Is there a formal framework ?

Problems of Existing works

- Many signature schemes are not analysed, e.g., BLS, BB-Short, Elgamal, MLDSA, etc.

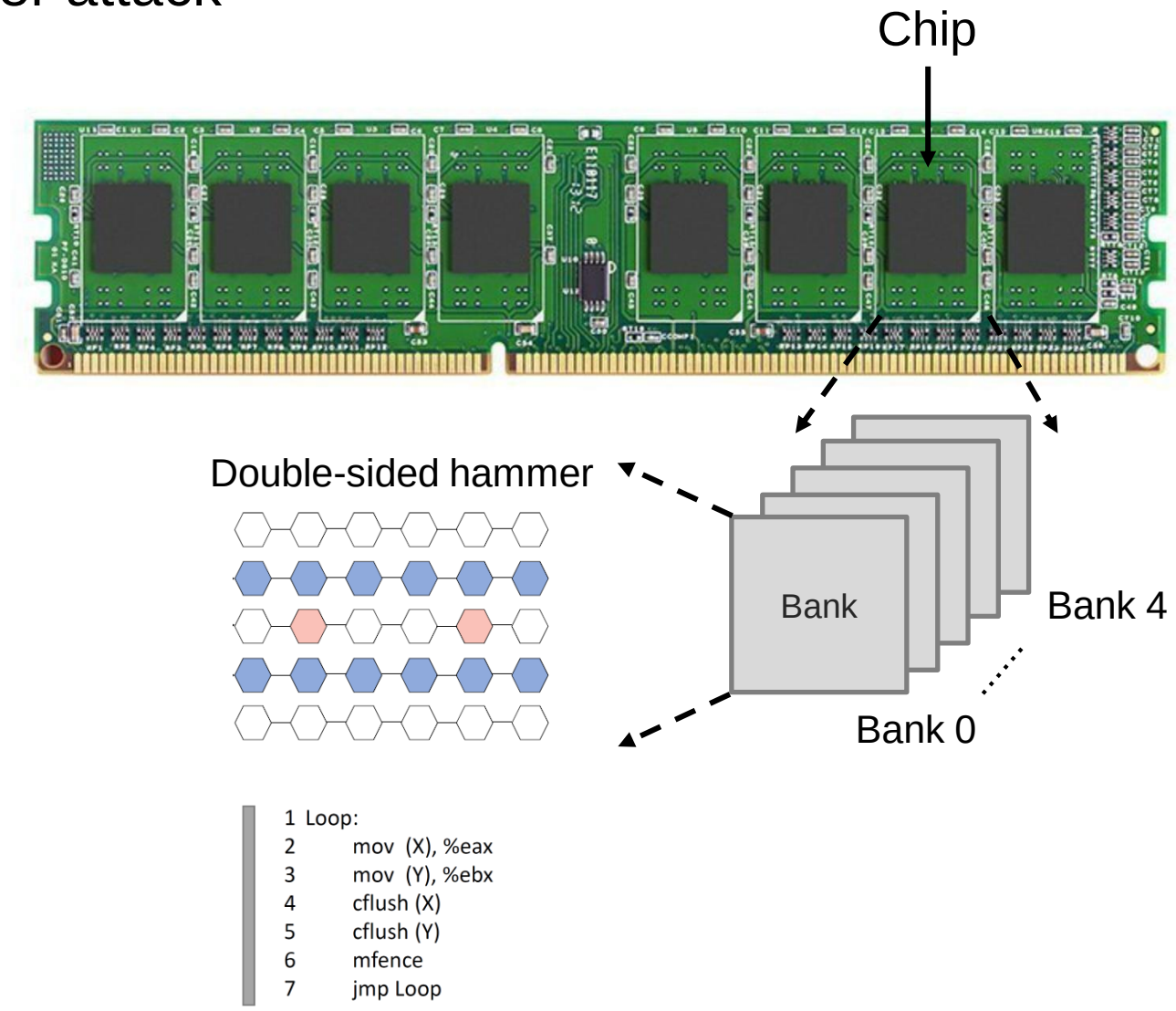
- It is still unclear how we utilise existing analysis for new signature schemes.

Is there a formal framework ?

- Yes! Our framework Achilles targets generalised signature schemes, and can be used for:
  1> discovering new vulnerable parameters even in studied schemes
  2> helping identify new vulnerable schemes

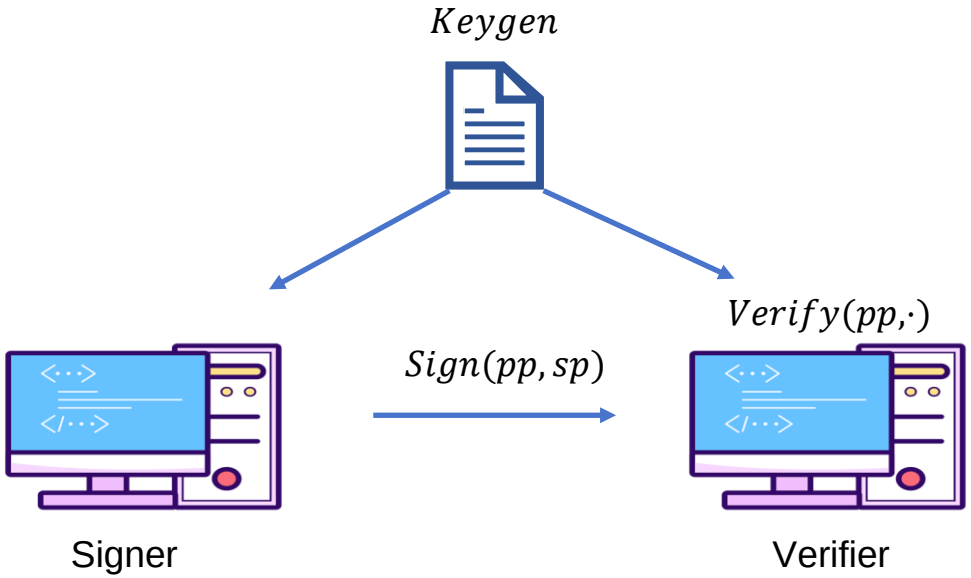
Background

- Rowhammer attack



Background

- Signature Schemes: the idea is simple. The verifier verifies the identity of the signer by using public parameters to verify the signature



Achilles – Step 1

- A formal Treatment of Signature Schemes
  - It is based on the general definition of signature schemes
  - It includes some inevitable key parameters in signature schemes

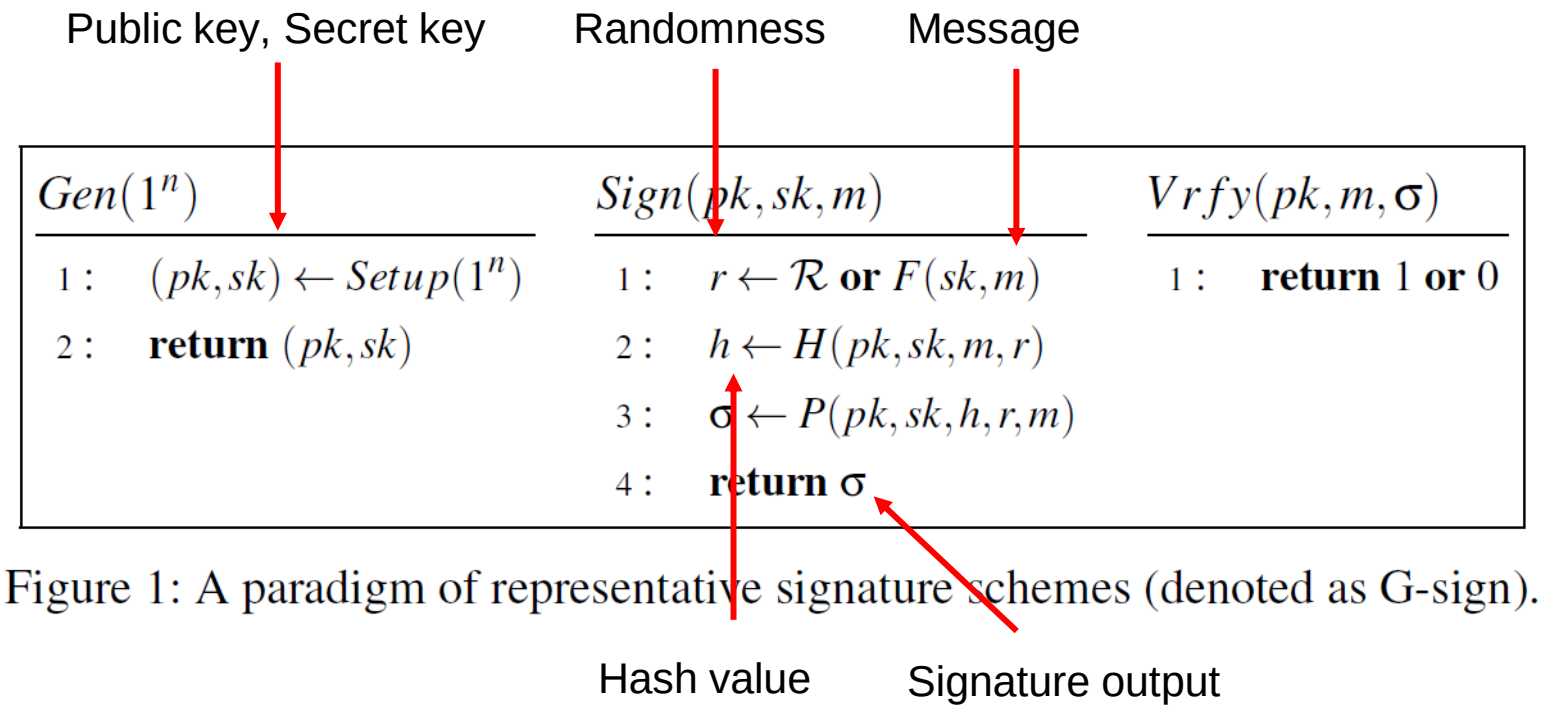


Figure 1: A paradigm of representative signature schemes (denoted as G-sign).

Achilles – Step 2

- A Game-based model against $G\text{-}sign$
  - $OSign$: Sign a message and return the signature

$OSign(pk, sk, m)$

1 : $\sigma \leftarrow Sign(pk, sk, m)$
2 : $M \leftarrow M \cup \{m\}$
3 : **return** σ

Figure 2: Formalizing Rowhammer faults against G-sign.

Achilles – Step 2

- A Game-based model against $G\text{-}sign$
  - $OSign$: Sign a message and return the signature
  - $f_i(x)$: formalization of Rowhammer fault (a random model)

$OSign(pk, sk, m)$

1 : $\sigma \leftarrow Sign(pk, sk, m)$
2 : $M \leftarrow M \cup \{m\}$
3 : **return** σ

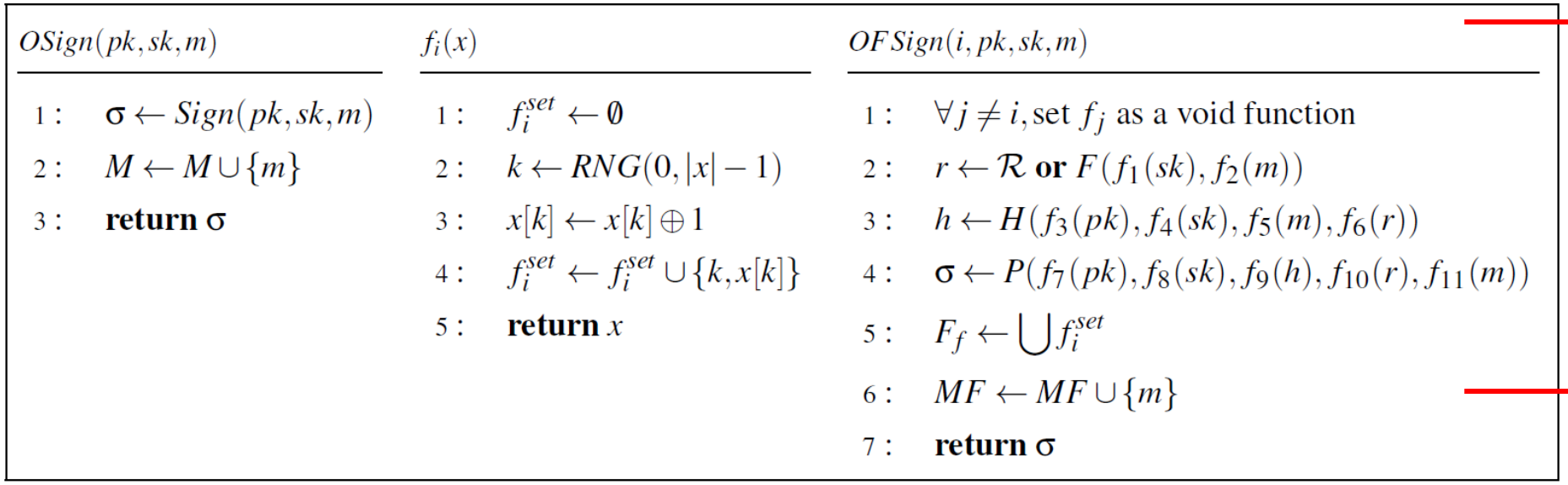
$f_i(x)$

1 : $f_i^{set} \leftarrow \emptyset$
2 : $k \leftarrow RNG(0, |x| - 1)$
3 : $x[k] \leftarrow x[k] \oplus 1$
4 : $f_i^{set} \leftarrow f_i^{set} \cup \{k, x[k]\}$
5 : **return** x

Figure 2: Formalizing Rowhammer faults against G-sign.

Achilles – Step 2

- A Game-based model against $G\text{-}sign$
  - $OSign$: An oracle which signs a message and return the signature
  - $f_i(x)$: formalization of Rowhammer fault (a random model)
  - $OFSign$: An oracle which allows fault occurs in the key parameters.



i is specified by the attacker, indicated which parameter is faulted

Store the fault information in F_f and MF

Figure 2: Formalizing Rowhammer faults against G-sign.

Achilles – Step 2

- A Game-based model against $G\text{-}sign$
  - $OSign$: An oracle which signs a message and return the signature
  - $f_i(x)$: formalization of Rowhammer fault (a random model)
  - $OFSign$: An oracle which allows fault occurs in the key parameters.
  - Exp^F: A Game experiment indicated two kinds of capabilitites of the attacker.

$Exp^F(\mathcal{A})$

1 : $M \leftarrow \emptyset, MF \leftarrow \emptyset, F_f \leftarrow \emptyset$

2 : $(sk, pk) \leftarrow Gen()$

3 : $(m^*, \sigma^*, f^*) \leftarrow \mathcal{A}^{OSign(\cdot), OFSign(i,\cdot)}(pk)$

4 : $v \leftarrow Vrfy(pk, m^*, \sigma^*)$

5 : **return** 1 if $v == 1 \wedge m^* \notin M \wedge m^* \notin MF$

6 : **return** 2 if $v == 1 \wedge m^* \notin M \wedge m^* \in MF \wedge f^* \in F_f$

7 : **return** 0, otherwise

Figure 3: A game-based experiment modelling post-Rowhammer analysis.

Line 5 : The attacker can output a valid signature on a new message

The fault occurs on the public parameters (pp)

Line 6 : The attacker can output a valid signature on an existed message which was queried in $OFSign$

The fault occurs on the secret parameters (sp)

Achilles – Step 3

- We design algorithms to recover the secrets when the fault occurs on sp or pp

Algorithm 1: Post-Rowhammer analysis via DFA

1 Initially, $S = (Gen, Sign, Vrfy)$ is a signature scheme. $OFSign$ and $f_i(x)$ are function oracles defined in Figure 2.
2 Parse S to G-sign
3 **foreach** $p \in \{m, pk, h\}$ **do**
  // get a valid and faulty signature.
4 $\sigma \leftarrow OSign(\cdot)$
5 Use i to denote the index for parameter p
6 $\sigma' \leftarrow OFSign(i, \cdot)$
7 return $sk' = (\sigma - \sigma')/g(pp)$
8 **end**

→ enumerate all possible public parameters

→ get the representation of the faulty signature

↓ judge if it is exploitable by differential analysis

Achilles – Step 3

- We design algorithms to recover the secrets when the fault occurs on sp or pp

Algorithm 2: Post-Rowhammer analysis via SCA

```
3  foreach p ∈ {sk, r} do
       // get a valid and faulty signature.
4      Use i to denote the index for parameter p
5      do
6          σ' ← OFSign(i, ·)
           // enumerate the one-bit fault.
7          Δ ← 0, j ← 0
8          do
9              if Vrfy(pk · g(pp, Δd), m, σ') = 1 then
10                 N++
11                 sk' = sk' ⊕ Δ
12             end
13             Δ ← Δ << 1, j ← j + 1
14         while j < |sk'|
15     while |sk'| − recovered_bits > 50
16 end
17 return sk'
```

enumerate all possible secret parameters

get the representation of the faulty signature

try to correct the faulty signature to a valid one

use Δ as bit leakage

Achilles – Step 3

- To use the algorithms, the attacker need first instantiates with a specified scheme
- An $EdDSA$ example:

$KeyGen$:
$\mathrm{sk} \leftarrow s$
$pk \leftarrow (P, B)$
$[P = sB]$

$Sign$:
$r = H(m)$
$R = rB$
$h = H(R||P||m)$
$\sigma = r + hs$

Scheme	pp				sp
EdDSA	h	m	P, B	r, R	s
G-sign	h	m	pk	r	sk

Table 2: Classification of potentially vulnerable parameters in EdDSA.

DFA: when h is faulted, the faulty signature $\sigma' = r + (h + \Delta)s$ is exploitable

SCA: when s is faulted, the faulty signature $\sigma' = r + h(s + \Delta)$ is exploitable

Achilles – AutoVuln

- The process can be automated by using symbolic operations.

$$./AutoVuln\ -pk\_list = []\ -sk\_list = [] - h\_list = [] - m\_list = [] - r\_list = [] - sign\_list = [] - vrfy\_list = []$$

description of the symbols

Potentially vulnerable parameters

description of the equations

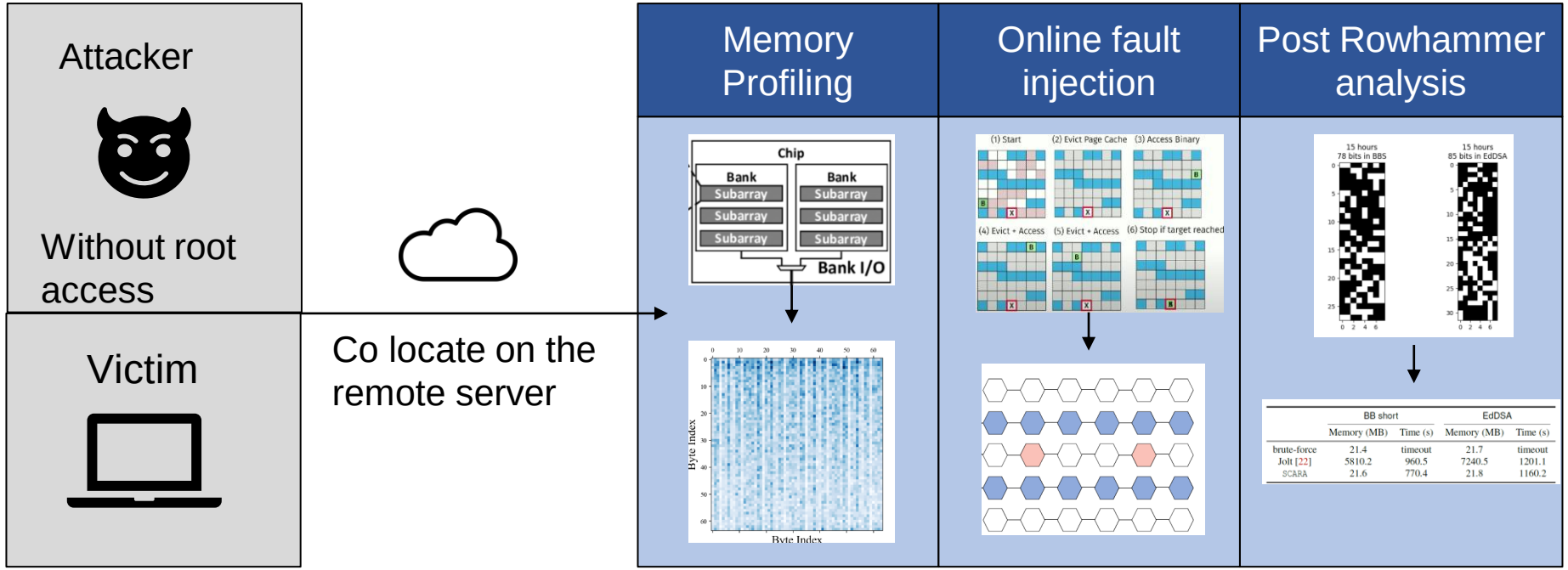
Result:

Scheme	Library	Time	Identified Potentially Vulnerable Parameter				
			pk	*h*	*m*	*sk*	*r*
BB short	relic-toolkit-0.6.0	64.0 ms				s	
EdDSA	wolfSSL-5.6.6	75.2 ms	R,P	h	m	s	
ML-DSA	liboqs-0.10.0	85.0 ms	μ,w	c	m	s_1	
Elgmal	cryptopp-8.9	66.9 ms				x	k
RSA	wolfSSL-5.6.6	54.5 ms				s	
BLS	bls-signatures-2.0.3	52.2 ms				x	

Table 3: The execution time and potentially vulnerable parameters identified automatically by AutoVuln.

Evaluation

- We follow the idea of including offline memory profiling, online fault injection and post Rowhammer analysis for performing an attack.



Evaluation

- Memory profiling:
  - We first run DRAMdig to get the bank function of our DRAM, from which we can know if two addresses are in the same bank

    The bank function of our DRAM: (17,21), (16,20), (15,19), (14,18), (6,13)
  - Then we profile 12GB of physical memory

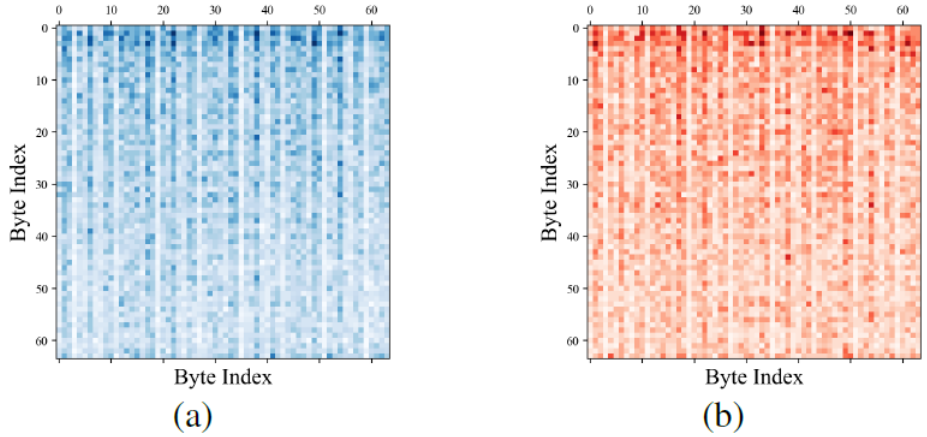


Figure 4: The distribution of flippable-bit offsets over 4 KB-aligned pages. Bit flips from 1 to 0 (blue) and bit flips from 0 to 1 (red) accumulated over 4 KB pages.

Evaluation

- Online Fault Injection:
  - We utilize the behaviour of Linux buddly allocator (first-in-last-out), which allows we to do memory massaging by release vulnerable pages and coerce the victim to use these pages to store the parameters.
  - Challenge: the fault must occur in a limited time window between a parameter's initiation and its subsequent use in computations
  - Our solution: we opt for OS signals. The attacker process sends the registered signal to trick the OS kernel into switching the victim process off the core.

Evaluation

- Post Rowhammer Analysis:
  - From the faulty signatures, we are able to recover the secrets.
  - In DFA, we only need one faulty signature for full recovery (EdDSA).
  - In SCA, we need hundreds of faulty signatures. We also need to find the remaining bits.

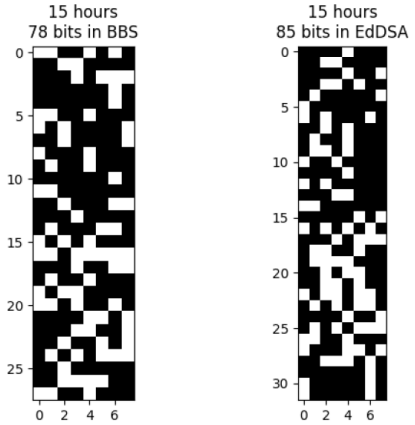


Figure 5: The number of recovered bits in 15 hours for the secret key in BB short and EdDSA is represented in the form of pixels, where white cells indicate recovered bits, and black cells indicate non-recovered bits.

	BB short		EdDSA	
	Memory (MB)	Time (s)	Memory (MB)	Time (s)
brute-force	21.4	timeout	21.7	timeout
Jolt [22]	5810.2	960.5	7240.5	1201.1
SCARA	21.6	770.4	21.8	1160.2

Table 4: The memory and time overhead when recovering the remaining secret bits using brute-force, Jolt [22] and our SCARA, respectively.

Countermeasure

- Specific defenses of signature schemes
  - Leakage resilient cryptographic schemes
  - Verifying after signing
  - Redundancy check

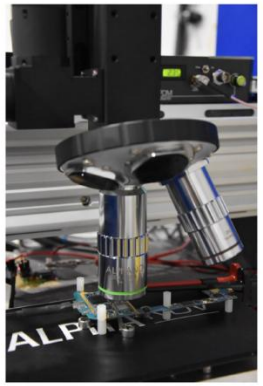
- WolfSSL has adopted our suggestion and implement “verifying after signing” and “Redundancy check” for RSA and EdDSA.

LINK	CVE-2024-2881	Medium	Fault injection attack with EdDSA signature operations. This affects ed25519 sign operations where the system could be susceptible to Rowhammer attacks. Thanks to Junkai Liang, Zhi Zhang, Xin Zhang, Qingni Shen for the report (Peking University, The University of Western Australia). Fixed in this GitHub pull request #7212	0 days	5.7.0
LINK	CVE-2024-1545	Medium	Fault Injection vulnerability in RsaPrivateDecryption function that potentially allows an attacker that has access to the same system with a victims process to perform a Rowhammer fault injection. Thanks to Junkai Liang, Zhi Zhang, Xin Zhang, Qingni Shen for the report (Peking University, The University of Western Australia)." Fixed in this GitHub pull request #7167	9 days	5.7.0

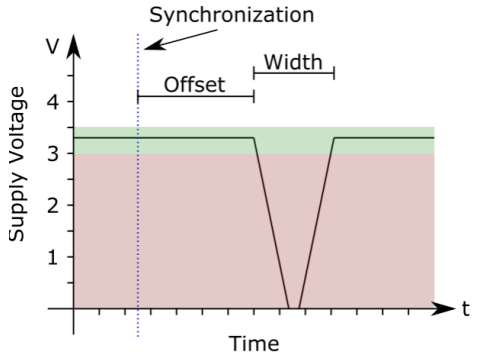
In our work, we have shown that these countermeasures can be bypassed by faulting the opcode

Physical Fault Injection

- Can physical fault injection achieve similar results ?



Optical radiation



Voltage/Clock Glitch

Electromagnetic pulses

Temperature attack

- The temporal and spatial precision of these attacks does not meet our requirements. These attacks do not allow remote attackers.

Thank you !

Questions ?

https://github.com/liang-junkai/Achilles

For details and further info:

Junkai Liang
(ljknjupku@gmail.com)

More Insights, Code and Data In Website