WPI

VERNAM Group

Security & Privacy @ WPI

Jolt: Recovering TLS Signing Keys via Rowhammer Faults

Koksal Mus
kmus@wpi.edu

Yarkın Doröz
ydoroz@wpi.edu

M. Caner Tol
mtol@wpi.edu

Kristi Rahman
krahman@wpi.edu

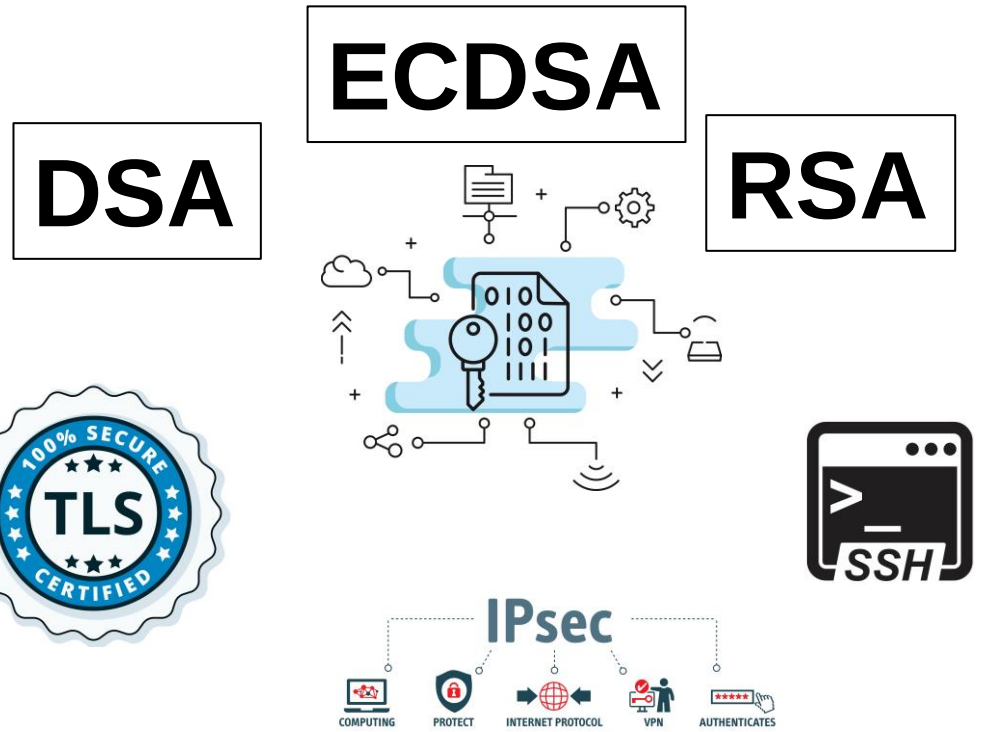
Berk Sunar
sunar@wpi.edu

Worcester Polytechnic Institute

44th IEEE Symposium on Security and Privacy
May 22-25, 2023, San Francisco, CA

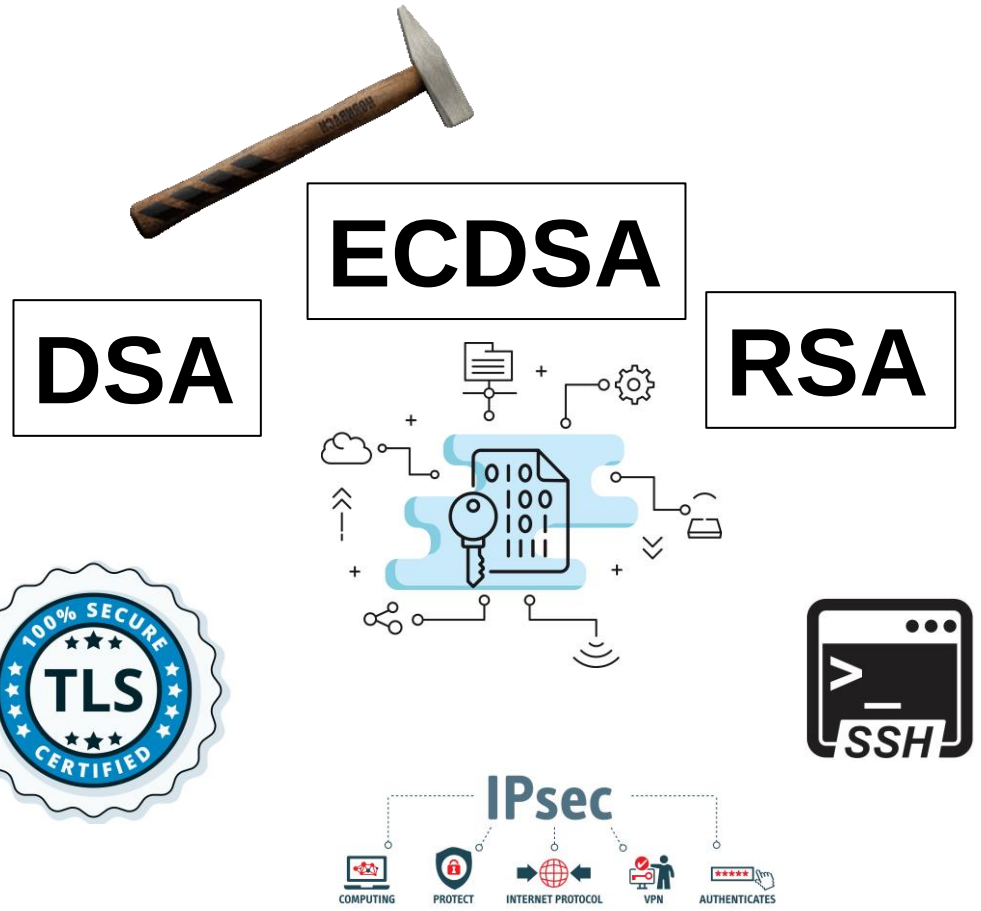
Digital Signature Schemes

- Digital Signature Algorithms are everywhere
- For authenticity and integrity of digital messages
  - E-commerce, banking, software distribution etc.
- Integrity during handshake of protocol parameters
  - TLS, SSH, IPSec
- NIST FIPS 186-x Digital Signature Standard
  - DSA*
  - ECDSA
  - EdDSA
  - RSA



Digital Signature Schemes

- Digital Signature Algorithms are everywhere
- For authenticity and integrity of digital messages
  - E-commerce, banking, software distribution etc.
- Integrity during handshake of protocol parameters
  - TLS, SSH, IPSec
- NIST FIPS 186-x Digital Signature Standard
  - DSA*
  - ECDSA
  - EdDSA
  - RSA



Threat Model

- Active software-only fault attack
- **Victim**
  - Is running in an environment **logically** isolated from attackers
  - Signs messages using digital signature schemes (e.g. ECDSA)
- **Attacker**
  - Co-locates with victim, e.g. shared cloud server, browser etc.
  - Inject faults (Rowhammer) in victim's memory space (key)
  - Collects faulty signatures
  - Recovers private signing key

Jolt – Signature Correction Attack

- Flips key bits in victim memory during signing; collects resulting faulty signatures to recover key bit values

Fault injection

1. **Memory Profiling Phase (Offline):** Identify flippable locations in memory for Rowhammer
2. **Online Phase:** Inject faults while victim signs; collect faulty signatures

Recovery

3. **Post-Processing Phase (Offline) :** Correct faulty signatures to recover key bits

Key Idea

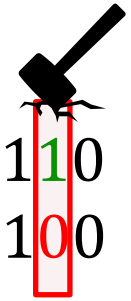
- Assume we are able to flip a bit in an unknown location of memory
- If we are somehow able to learn the error pattern (but **not** the data!), i.e.
- We can deduce the bit value in the erroneous position
- Works in the other direction as well
- Magnitude reveals bit location
- Sign reveals bit value

$d = 01010010\,1\,1110$

$d' = 01010011\,1\,1110$

$\Delta_d = d' - d = 16 = 2^4 = 000000010000$

$d = xxxxxxx0xxxx$



$d = 0101001011\,1\,0$

$d' = 0101001011\,0\,0$

$\Delta_d = d' - d = -2 = -2^1 = 111111111110$

$d = xxxxxxxxxx1x$

Post-Processing Phase on ECDSA

Victim Side

- **1- Key Generation**
  - Select an Elliptic Curve, E
  - Select a base point, P on the curve
  - $Q = dP \in E$
  - Private key: d
  - Public key: Q
- **2- Signature Generation**
  - Select a nonce, k
  - Compute kP and $r = (kP)_x$
  - Compute $\bar{s} = k^{-1}(H(m) + \bar{d}r) \bmod n$
  - Signature : $(r, \bar{s})$

Attacker Side

- **3- Signature Verification**
  - Compute $H(m)$
  - Compute $\bar{w} = \bar{s}^{-1} \bmod n$
  - Compute $\bar{u_1} = H(m)\, \bar{w} \bmod n$
  - Compute $\bar{u_2} = r\, \bar{w} \bmod n$
  - Compute $\bar{R} = \bar{u_1} P + \bar{u_2} Q$
  - $\bar{r} = (\bar{R})_x$
  - Check if $r = \bar{r}$

- Faulty key: $\bar{d} = d + \Delta_d$
- Faulty signature: $(r, \bar{s})$

Post-Processing Phase on ECDSA

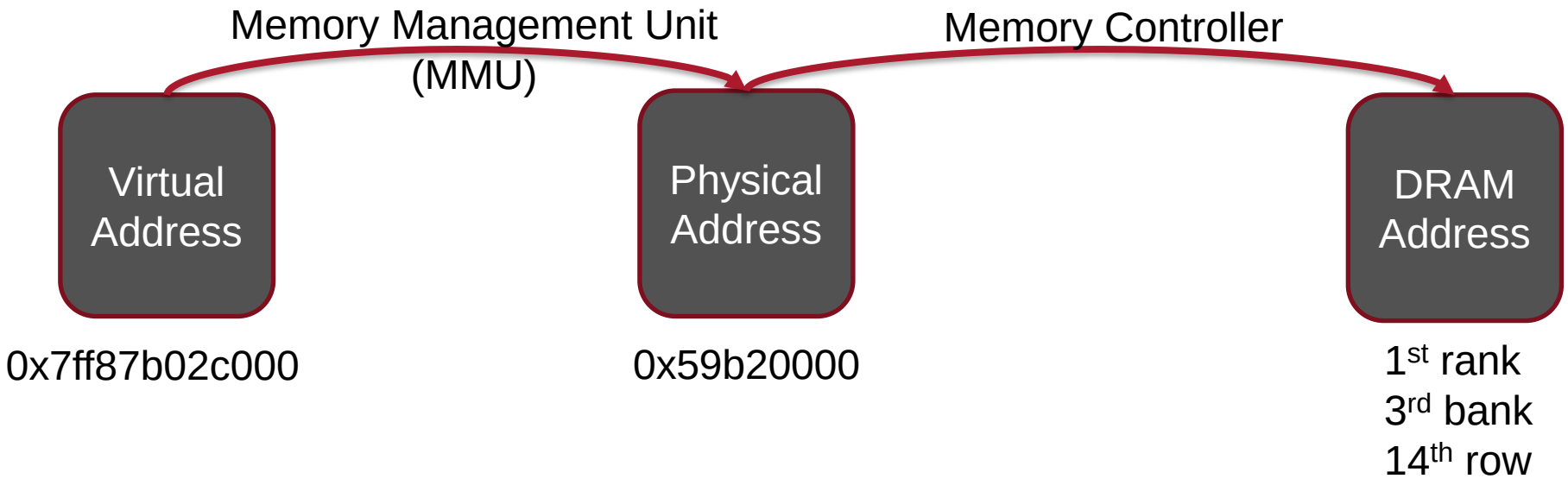
- **1- Key G**
  - Selec
  - Selec
  - Q=d
  - Priva
  - Publi
- **2- Signat**
  - Selec
  - Compute
  - Compute
  - Signature

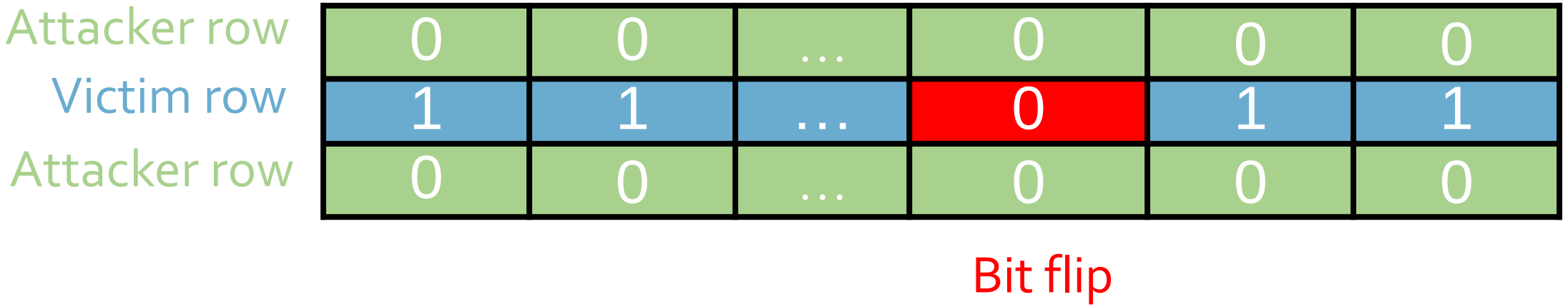
$$\overline{R} + \Delta_d \bar{u}_2 P = R$$

$$2 * length(d) = 512\ trials$$

Slight variation of attack works for
- DSA, RSA and even EdDSA
- Nonce, instead of secret key

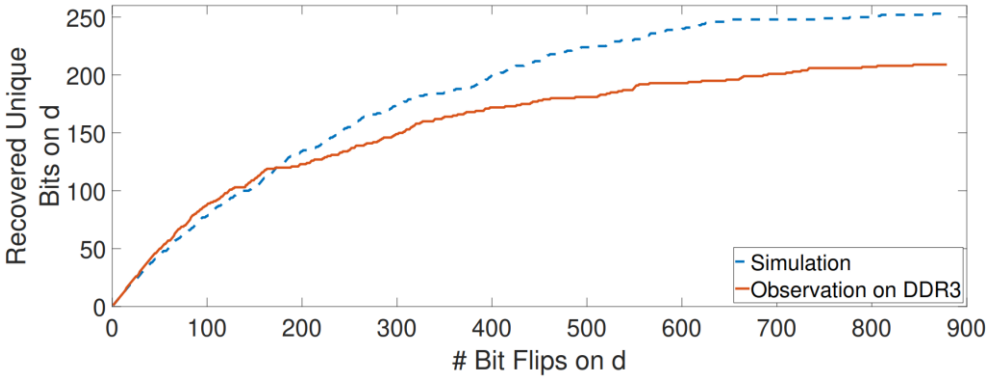
Faulting via Rowhammer

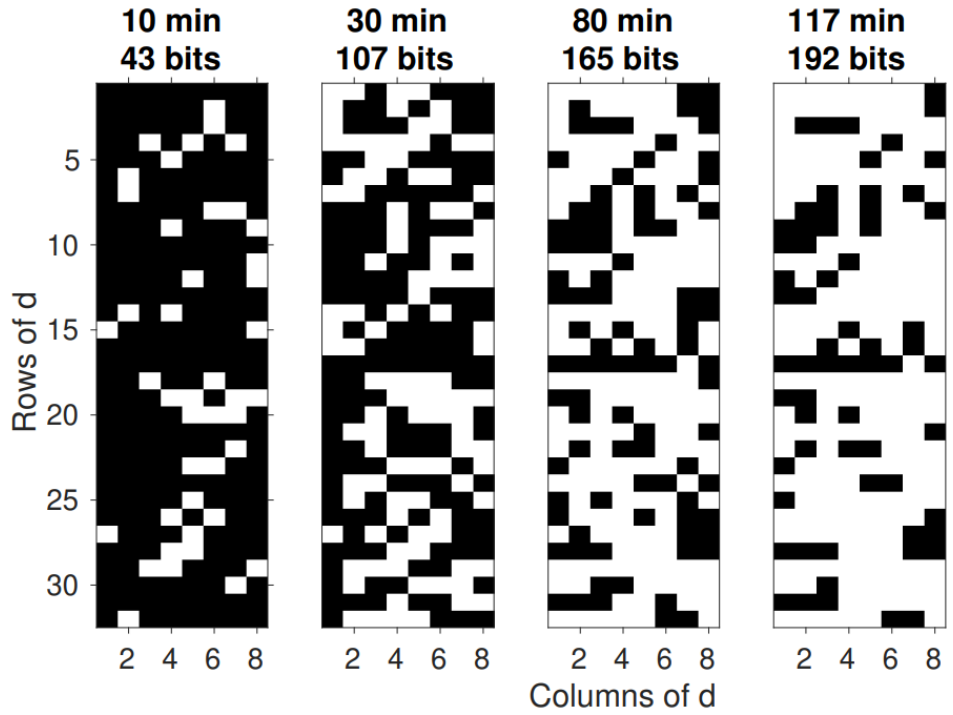




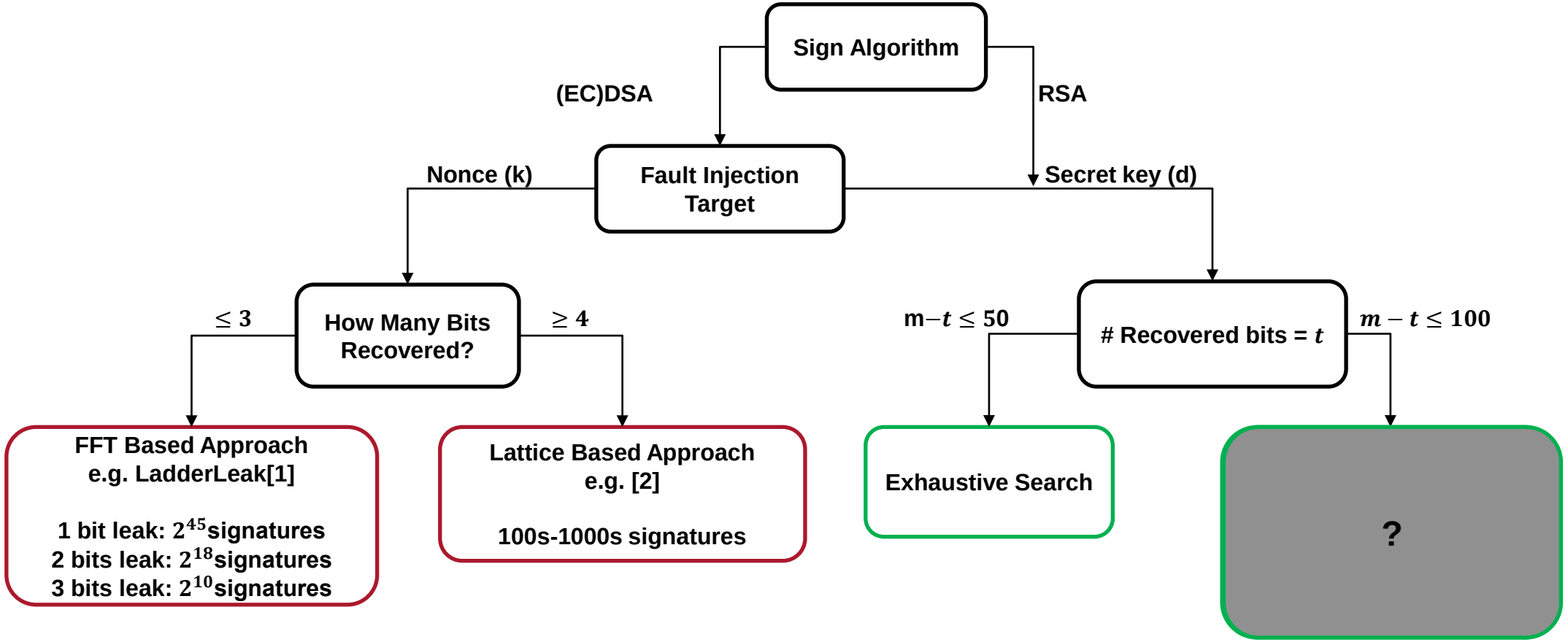
Recovered Key Bits

- OpenSSL ECDSA
- In the first 117 mins of online phase, we collected 515 faulty signatures.
- 1 time triple bit flips, (0.2%)
- 39 times double bit flips (7.6%)
- 475 times single bit flips. (92.2%)
- We did not observe >3-bit flips.
- For multiple flips, more combinations for Δ_d
- Overlapping bit flips





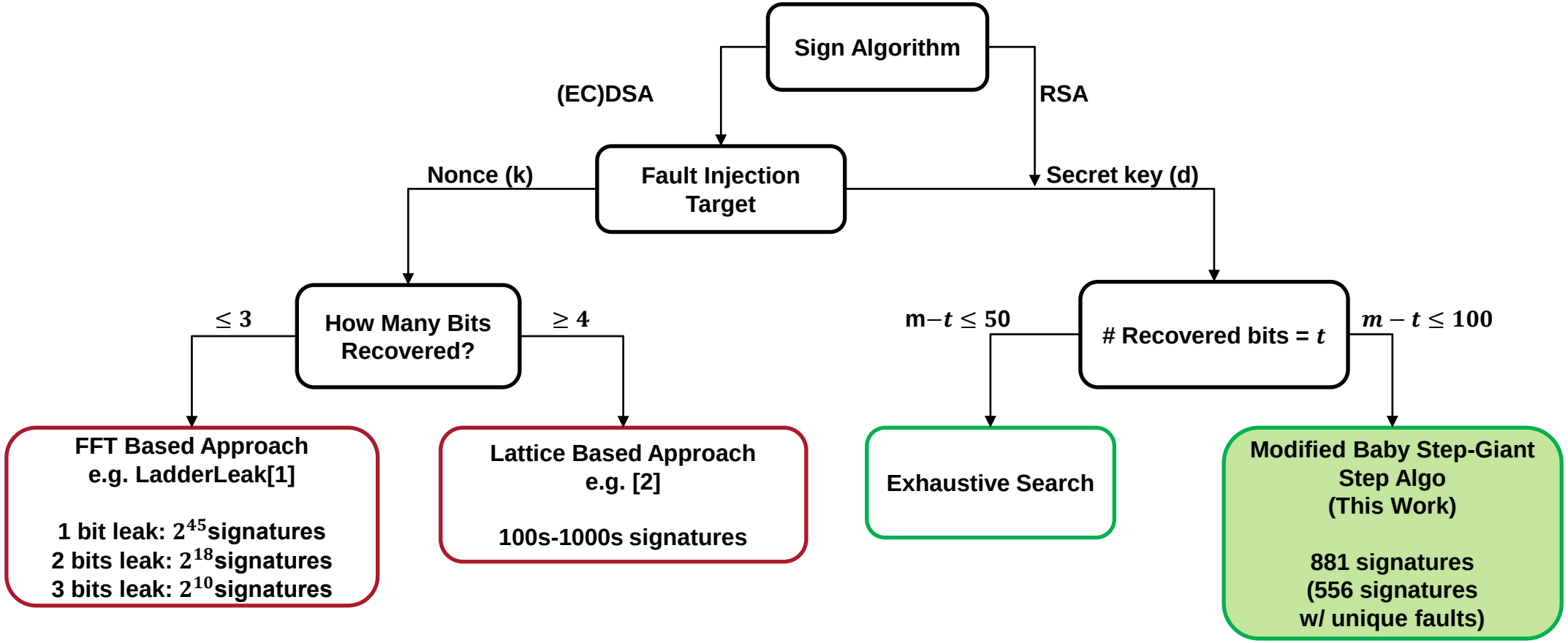
Recovering Remaining Bits



[1] Aranha, D. F., Novaes, F. R., Takahashi, A., Tibouchi, M., & Yarom, Y. (2020, October). LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (pp. 225-242).

[2] Moghimi, D., Sunar, B., Eisenbarth, T., & Heninger, N. (2020, January). TPM-FAIL: TPM meets timing and lattice attacks. In Proceedings of the 29th USENIX Security Symposium.

Recovering Remaining Bits



[1] Aranha, D. F., Novaes, F. R., Takahashi, A., Tibouchi, M., & Yarom, Y. (2020, October). LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (pp. 225-242).

[2] Moghimi, D., Sunar, B., Eisenbarth, T., & Heninger, N. (2020, January). TPM-FAIL: TPM meets timing and lattice attacks. In Proceedings of the 29th USENIX Security Symposium.

Analysis of Crypto Libraries

- Manual review on the low-level signature primitives to look for the countermeasures

	Signature Check	Faulty Sig. Transmitted	Patched	CVE
wolfSSL 5.3.1	✗	✓	✓	CVE-2022-42961
OpenSSL 3.0.4	✗	✓	✗	N/A
OpenSSL-FIPS 2.0.8	✓	✓	✗	N/A
LibreSSL 3.5.3	✗	✓	✓	CVE-2022-42963
Amazon s2n 102.0	✗	✓	✓	CVE-2022-42962
MS SymCrypt 102.0	✗*	N/A	N/A	N/A

*: Only the first signature is detected.

Analysis of Crypto Libraries

- Manual review on the low-level signature primitives to look for the countermeasures
- Run client-server setup and inject faults on the server side and receive the faulty signature by the client

	Signature Check	Faulty Sig. Transmitted	Patched	CVE
wolfSSL 5.3.1	✗	✓	✓	CVE-2022-42961
OpenSSL 3.0.4	✗	✓	✗	N/A
OpenSSL-FIPS 2.0.8	✓	✓	✗	N/A
LibreSSL 3.5.3	✗	✓	✓	CVE-2022-42963
Amazon s2n 102.0	✗	✓	✓	CVE-2022-42962
MS SymCrypt 102.0	✗*	N/A	N/A	N/A

*: Only the first signature is detected.

Analysis of Crypto Libraries

- Manual review on the low-level signature primitives to look for the countermeasures
- Run client-server setup and inject faults on the server side and receive the faulty signature by the client
- Disclosure

> Sent: Wednesday, August 24, 2022 4:27 AM
> Subject: [EXT] Re: [openssl-security] openssl - vulnerability disclosure
>
> Thank you for this report. In general fault injection attacks are outside of our threat model - see https://www.openssl.org/policies/general/security-policy.html

	Signature Check	Faulty Sig. Transmitted	Patched	CVE
wolfSSL 5.3.1	✗	✓	✓	CVE-2022-42961
OpenSSL 3.0.4	✗	✓	✗	N/A
OpenSSL-FIPS 2.0.8	✓	✓	✗	N/A
LibreSSL 3.5.3	✗	✓	✓	CVE-2022-42963
Amazon s2n 102.0	✗	✓	✓	CVE-2022-42962
MS SymCrypt 102.0	✗*	N/A	N/A	N/A

*: Only the first signature is detected.

Countermeasures

- Against Rowhammer
  - Increasing DRAM Refresh rate
  - Using ECC memory
  - Target Row Refresh (TRR)
- Against Signature Correction
  - Verify after Sign
    - Implemented in
      - WolfSSL
      - Amazon-s2n
      - LibreSSL
  - Redundant Signing
  - Masking Sensitive Values

```
80  +  * WOLFSSL_CHECK_SIG_FAULTS
81  +  *    Verifies the ECC signature after signing in case of faults in the
82  +  *    calculation of the signature. Useful when signature fault injection is a
83  +  *    possible attack.
7331          #ifdef HAVE_ECC
7332             if (ssl->hsType == DYNAMIC_TYPE_ECC) {
7333                ret = EccSign(ssl, args->sigData, args->sigDataSz,
7334                    args->verify + HASH_SIG_SIZE + VERIFY_HEADER,
7335                    (word32*)&sig->length, (ecc_key*)ssl->hsKey,
7436  +        #if defined(HAVE_ECC) && defined(WOLFSSL_CHECK_SIG_FAULTS)
7437  +           if (ssl->hsType == DYNAMIC_TYPE_ECC) {
7438  +              ret = EccVerify(ssl,
7439  +                  args->verify + HASH_SIG_SIZE + VERIFY_HEADER,
7440  +                  sig->length, args->sigData, args->sigDataSz,
7441  +                  (ecc_key*)ssl->hsKey,
```

Conclusion

- **Jolt**: a novel attack that targets digital signature schemes
- Signature correction + Modified Baby-Step-Giant-Step algorithm
- Full ECDSA key recovery with <1000 faulty signatures and <2 hrs online time
  - Signing key does not change across sessions
- Also works on DSA, EdDSA and RSA
- WolfSSL, LibreSSL, Amazon s2n, OpenSSL, Microsoft SymCrypt
- Security protocols need thorough review against faults
- Underlying libraries needs to be selected carefully

Thank you!
Questions?

Contact me
mtol@wpi.edu
canertol


WORCESTER POLYTECHNIC INSTITUTE
LEHR UND KUNST
1865

DRAM Analysis

- 15 DDR3 and 4 DDR4 chips
- ~1.5 hours DDR3
- ~8 hours DDR4
- The raw memory profiles of 14 DDR3 DRAM chips are taken from Hammertime[61].

	Brand	Serial Number	Size [GB]	# Flips in d / profile	Vuln?
DDR3	Corsair	CMD16GX3M2A1600C9	16	232 ± 7	✓
	Corsair	CML16GX3M2C1600C9	16	47 ± 7	✓
	Corsair	CML8GX3M2A1600C9W	8	7 ± 3	✗
	Corsair	CMY8GX3M2C1600C9R	8	245 ± 5	✓
	Crucial	BLS2C4G3D1609ES2LX0CEU	8	4 ± 2	✗
	Geil	GPB38GB1866C9DC	8	55 ± 7	✓
	Goodram	GR1333D364L9/8GDC	8	6 ± 3	✗
	GSkill	F3-14900CL8D-8GBXM	8	231 ± 8	✓
	GSkill	F3-14900CL9D-8GBSR	8	53 ± 8	✓
	Hynix	HMT351U6CFR8C-H9	8	253 ± 1	✓
	V7	V73T8GNAJKI	8	37 ± 6	✓
	PNY	MD8GK2D31600NHS-Z	6	37 ± 6	✓
	Integral	IN3T4GNZBIX	4	203 ± 12	✓
	Samsung	M378B5173QH0	4	17 ± 4	✗
	Samsung	M378B5773DH0	2	196 ± 8	✓
DDR4	Corsair	CMU64GX4M4C3200C16	64	255 ± 1	✓
	Corsair	CMK32GX4M2B3200C16	32	1 ± 1	✗
	GSkill	F4-3600C16D-16GVKC	16	196 ± 10	✓
	Crucial	CT8G4DFD824A.C16FF	8	2 ± 2	✗

Modified Baby-Step-Giant-Step DLP Solver

Algorithm 3 Modified Baby-Step Giant-Step DLP Solver

Input: Cyclic group E with generator $|P| = n$, and public key $Q = dP$, recovered t bits of d.
Output: Secret key d.
Initialization
Represent d as $d = d^{(k)} + d'$ // See Table 1
Compute $Q' = Q - d^{(k)}P$ // $Q' = d'P$
Choose w_1, w_2 s.t. $w = w_1 + w_2$ for mem/cycle budgets $2^{w_1}, 2^{w_2}$
Precomputation
$j = 0$, $A = P$ // P generator on E
for $i = 0$ to $m - 1$ **do**
    if d'_i is unknown **then**
        $A_j = A$ // $A_j = 2^i P \in E$
        $a_j = i$ // index of m-bit representation
        Store (j, a_j, A_j) in Precomputation Table
        $j = j + 1$
    end if
    $A = 2A$ // Doubling on E
end for
Baby Step Computations
for $k = 0, 1, \ldots, 2^{w_1} - 1$ **do**
    $P_k = 0$, $b_k = 0$ // $P_k \in E$
    for $j = 0$ to $w_1 - 1$ **do**
        if $k_j = 1$ **then** // $k = (k_{w_1-1}, \cdots, k_0)_2$
            $P_k = P_k + A_j$ // Point Addition on E
            $b_k = b_k + 2^{a_j}$ // Baby step indices
        end if
        Store triplet (k, b_k, P_k) in Lookup Table $\mathbf{T}$.
    end for
end for
Sort Lookup Table $\mathbf{T}$ w.r.t. P_k column.
Giant Step Computations
Set $w_2 = w - w_1$
for $i = 0, 1, \ldots, 2^{w_2} - 1$ **do**
    $B_i = 0$, $c_i = 0$
    for $j = 0$ to $w_2 - 1$ **do**
        if $i_j = 1$ **then** // $i = (i_{w_2-1}, \cdots, i_0)_2$
            $B_i = B_i + A_{j+w_1}$ // Point Addition on E
            $c_i = c_i + 2^{a_{j+w_1}}$ // Giant step indices
        end if
    end for
    if $Q' - B_i$ matches any P_k in $\mathbf{T}$ **then**
        $d' = (c_i + b_k)_2$
        Return $d = d' + d^{(k)}$
    end if
end for

OpenSSL

```
// SERVER SIDE: Fault injection
    ...
// Enable for *simulated* faults on secret key
RandomBitFlip(eckey->priv_key);

if (eckey->meth->sign != NULL)
    return eckey->meth->sign(type, dgst, dlen,
    sig, siglen, kinv, r, eckey);
    ...

```

```
// CLIENT SIDE: Bit recovery algorithm
    ...
BN_mod_mul(u1, m, u2, order, ctx);
BN_mod_mul(u2, sig->r, u2, order, ctx);

for(test_add=0; test_add<2; test_add++)
    for(i =0; i<256; i++){
        //mult with 2**i
        powmul(u3, u2, i, order, ctx);
        if(test_add)
            BN_mod_add(u1,u1,u3,order,ctx);
        else
            BN_mod_sub(u1,u1,u3,order,ctx);

        ...
        /* if the signature is correct
        u1 is equal to sig->r */
        if((BN_ucmp(u1, sig->r))
            return i;
    }
```

ECDSA Private Key - d

Private key bits (d)	d_{m-1}	d_{m-2}	$\cdots$	d_2	d_1	d_0
Positions of w known and t Unknown Bits	u_{w-1}	k_{t-1}	$\cdots$	k_0	u_1	u_0
Known Part $d^{(k)}$	0	$d_{k_{t-1}}$	$\cdots$	d_{k_0}	0	0
Unknown Part d'	$d_{u_{w-1}}$	0	$\cdots$	0	d_{u_1}	d_{u_0}

TABLE 1: First row is the binary representation of d, the second row shows the positions (randomly chosen for demonstration) of t known and w unknown bits of d, the third row shows the known bits of d and the positions, the fourth row shows the unknown bits of d and the positions.

Crypto Libraries

Library/Version	Is Faulty Signature Detected?	Is Faulty Signature Transmitted by Server?	Is it a Standalone Library	Does library perform Signature check?
wolfSSL 5.3.1	No	Yes	Yes	None
OpenSSL 3.0.4	No	Yes	Yes	None
OpenSSL-FIPS 2.0.8	Yes	Yes	Yes	PK-SK pair is checked with fix message
LibreSSL 3.5.3	No	Yes	forked from OpenSSL	None
Amazon s2n 1.3.18	No	Yes	uses OpenSSL for crypto layer	None
MS SymCrypt 102.0	No (only first sign. is detected)	No TLS layer	Yes	PK-SK pair is checked with fix message (for first signature only)

Rowhammer slides