HOW SGX AMPLIFIES THE POWER OF CACHE ATTACKS

MS Thesis Presentation

By

Ahmad “Daniel” Moghimi

Advisor: Dr. Thomas Eisenbarth

Thesis Reader: Dr. Robert J. Walls

OUTLINE

- Motivation: Why is it important to attack Intel SGX?
- Intel Softwate Guard eXtension
- Prime+Probe Cache Attack
- CacheZoom
- Breaking AES
- Conclusion

UNTRUSTED COMPUTING

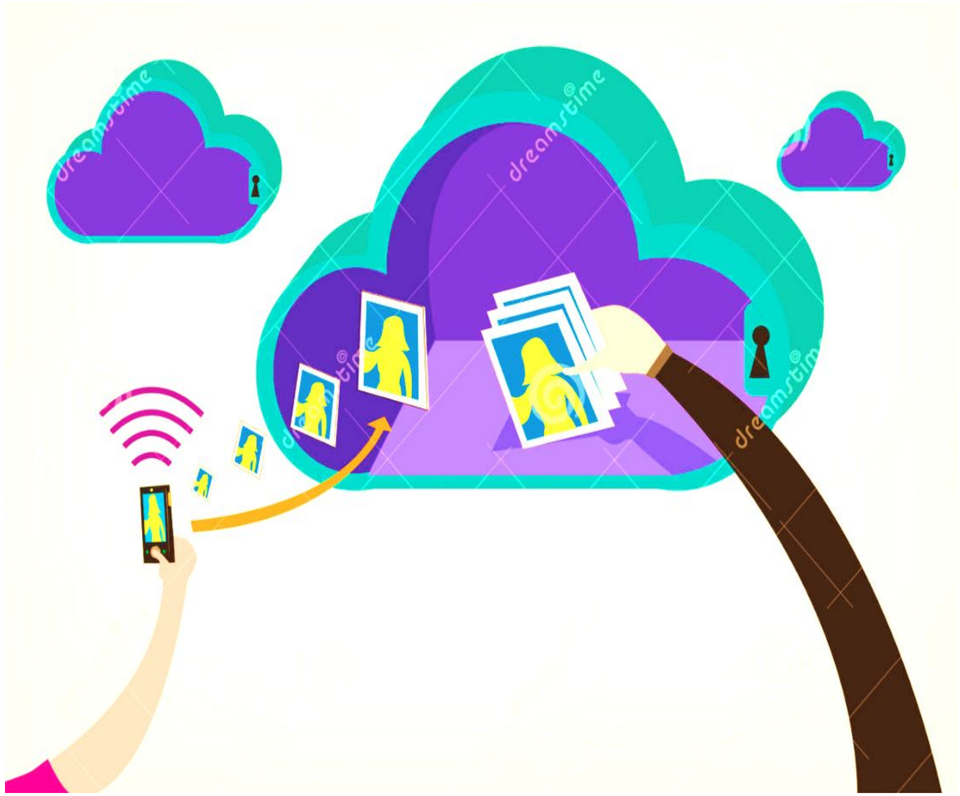
OS/SMM Rootkits

Untrusted Cloud Provider

Cross-VM Attacks[1]

OS & Hypervisor are not trusted

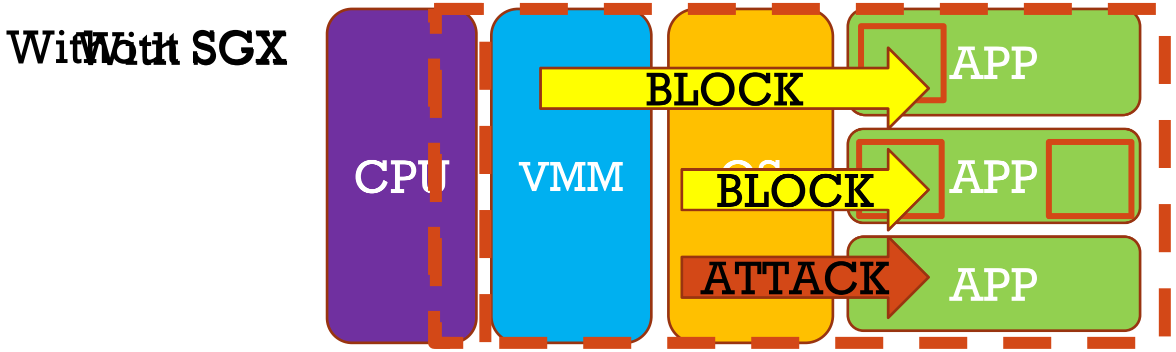
We need hardware supported
Trusted Computing Base



[1] Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. S $ A: A Shared Cache Attack That Works across Cores and Defies VM Sandboxing–and Its Application n to AES. In 2015 IEEE Symposium on Security and Privacy, pages 591–604. IEEE, 2015.

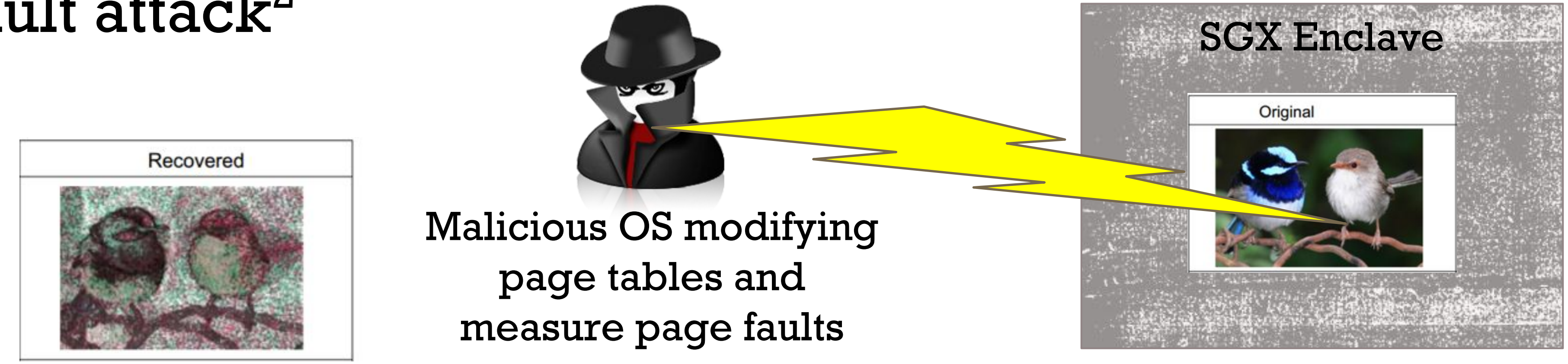
INTEL SOFTWARE GUARD EXTENSION (SGX)

- Trusted Execution Environment (TEE)
- **Enclave:** Hardware protected user-level software module
  - Loaded by the user program
  - Mapped by the Operating System
  - Authenticated and Encrypted by CPU
- Memory accesses are protected by the hardware



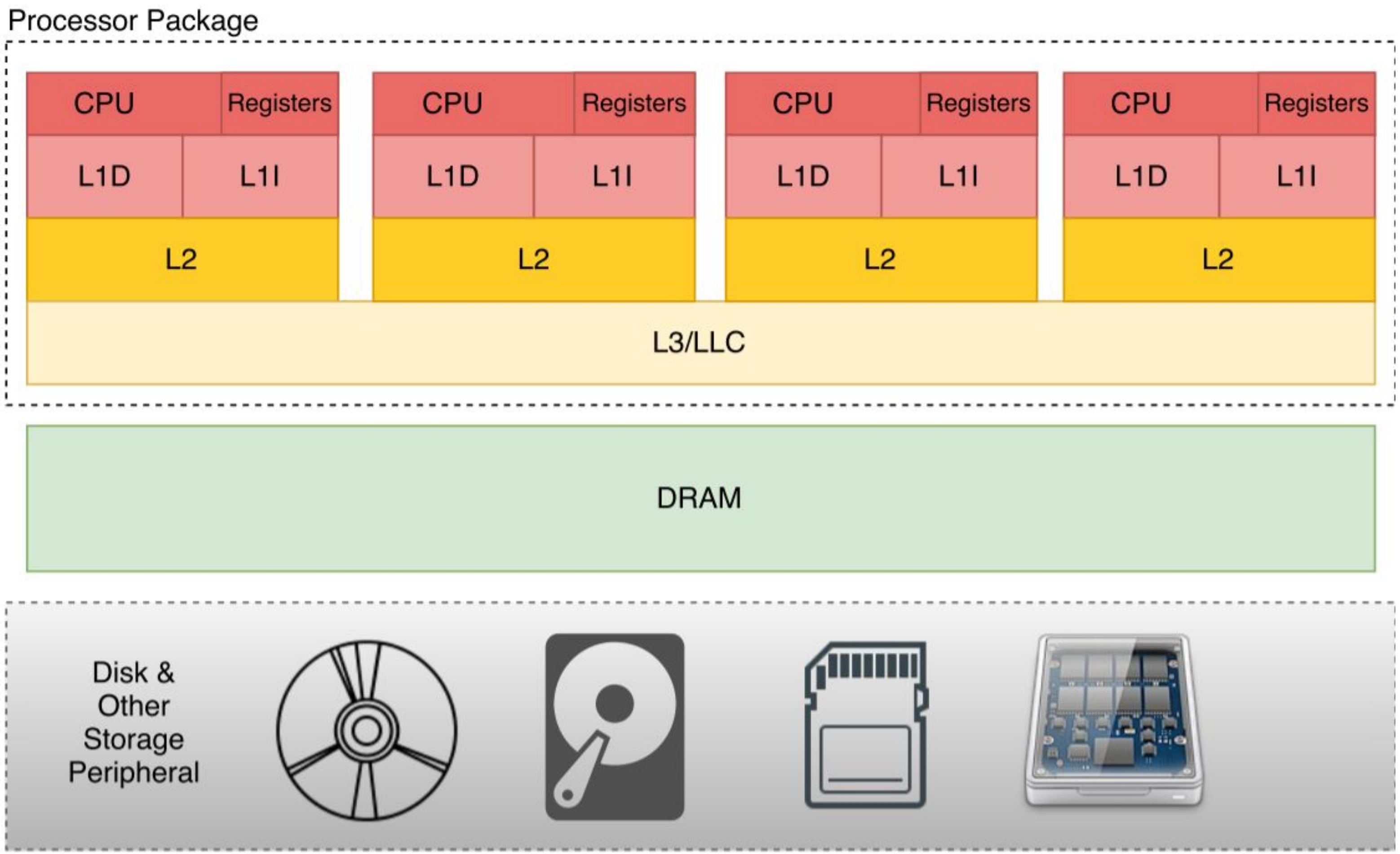
SIDE-CHANNEL ATTACK ON SGX

- SGX threat model:
  - Protecting against side-channel attacks is not our business!!!
- OS initiated side channels are powerful
- Page fault attack[2]

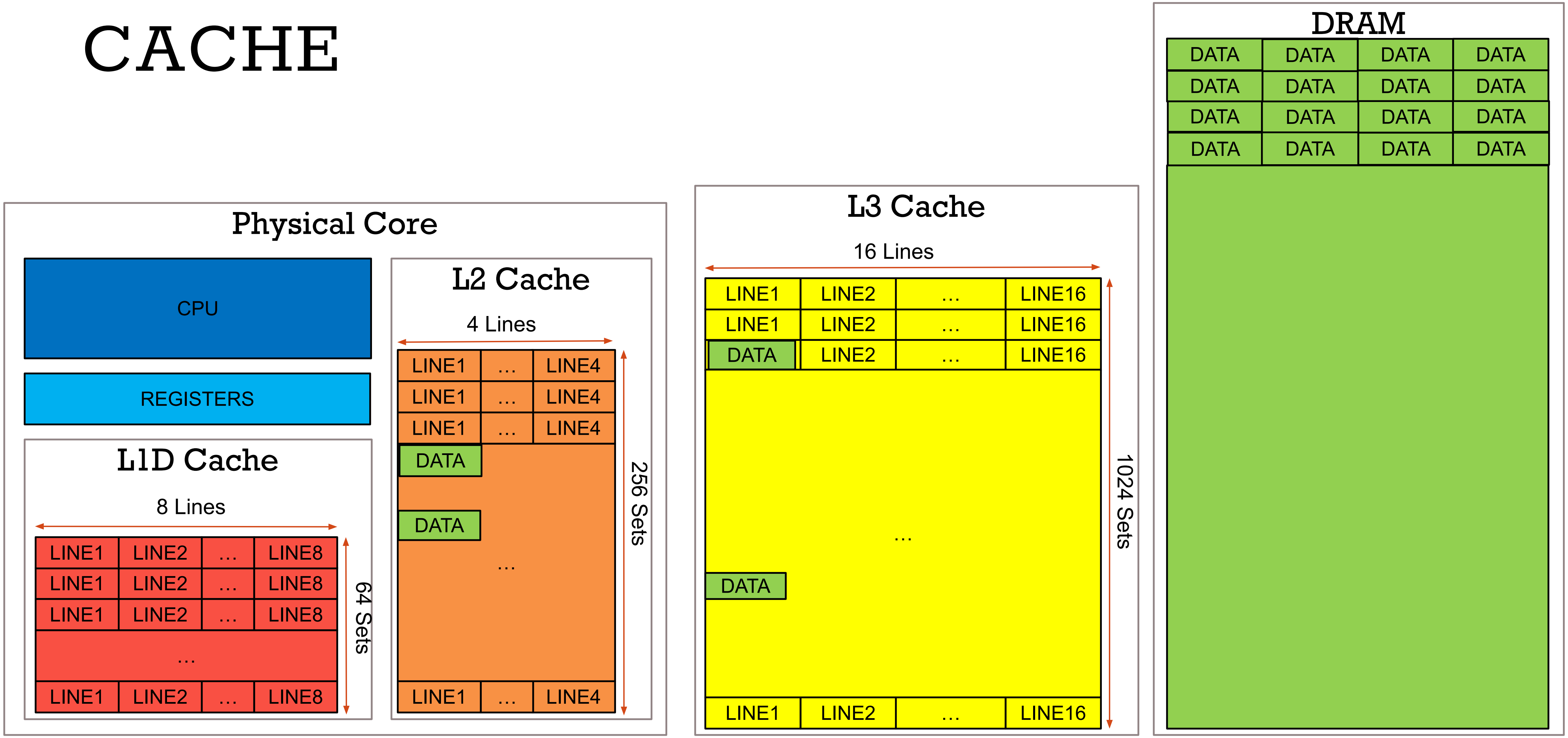


[2] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In 2015 IEEE Symposium on Security and Privacy, pages 640–656. IEEE, 2015.

MEMORY HIERARCHY



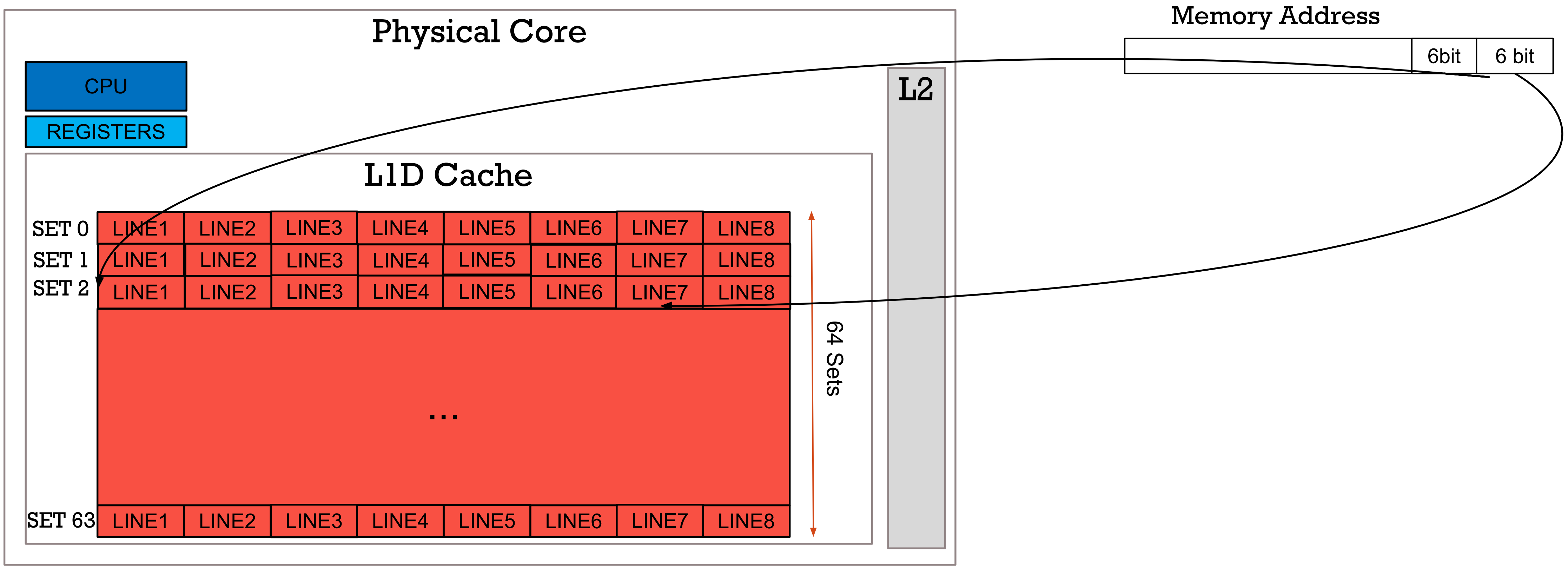
CACHE



Physical Core
CPU
REGISTERS
L1D Cache
8 Lines
LINE1 LINE2 … LINE8
64 Sets
L2 Cache
4 Lines
LINE1 … LINE4
DATA
256 Sets
L3 Cache
16 Lines
LINE1 LINE2 … LINE16
DATA
1024 Sets
DRAM
DATA

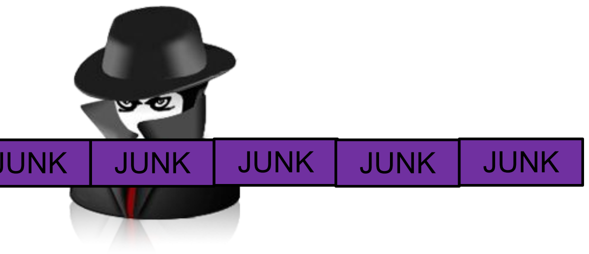
SGX – Memory Encryption Engine (MEE)

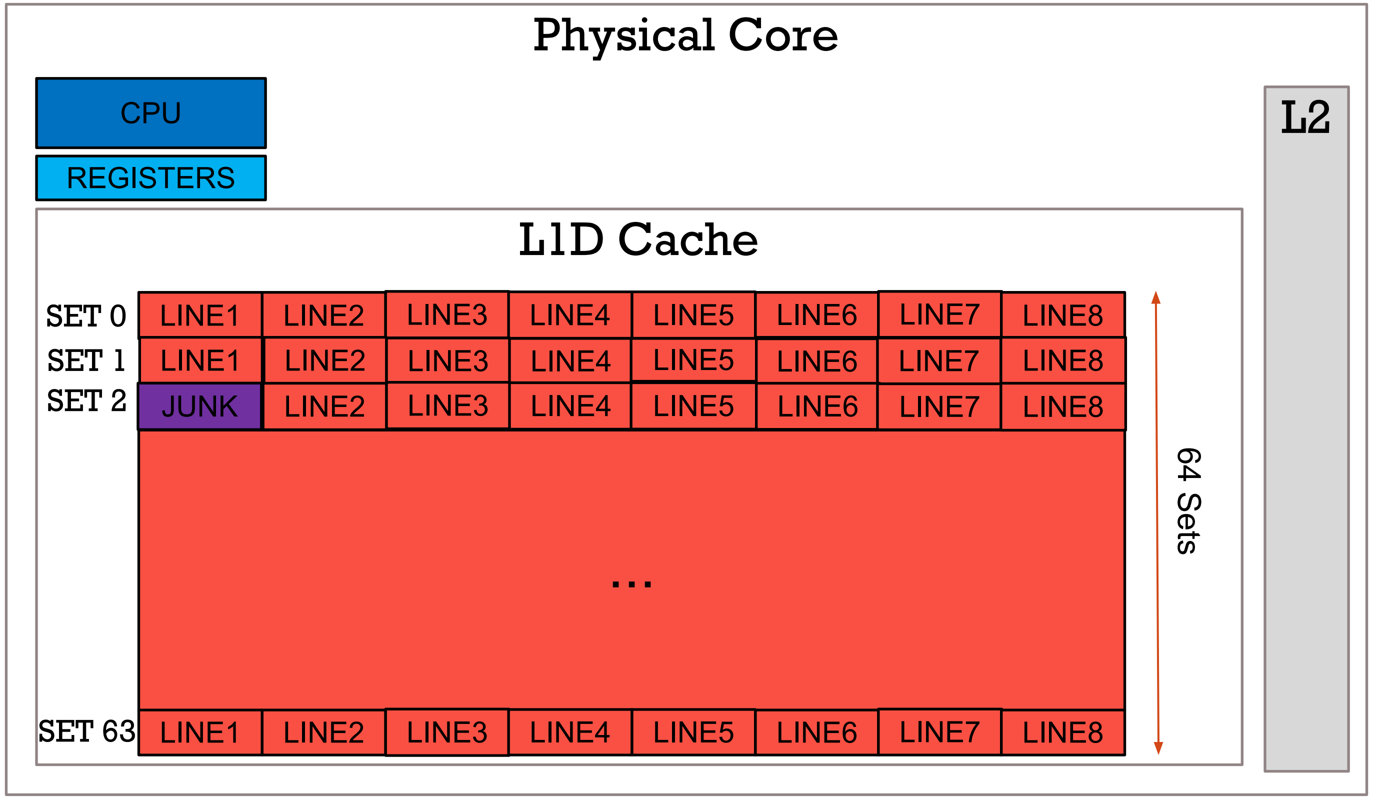
PRIME+PROBE

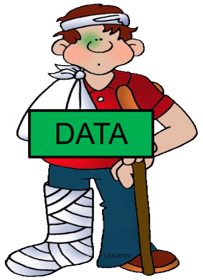


Physical Core
CPU
REGISTERS
L2
L1D Cache
SET 0
SET 1
SET 2
SET 63
LINE1
LINE2
LINE3
LINE4
LINE5
LINE6
LINE7
LINE8
...
64 Sets
Memory Address
6bit
6 bit

PRIME+PROBE







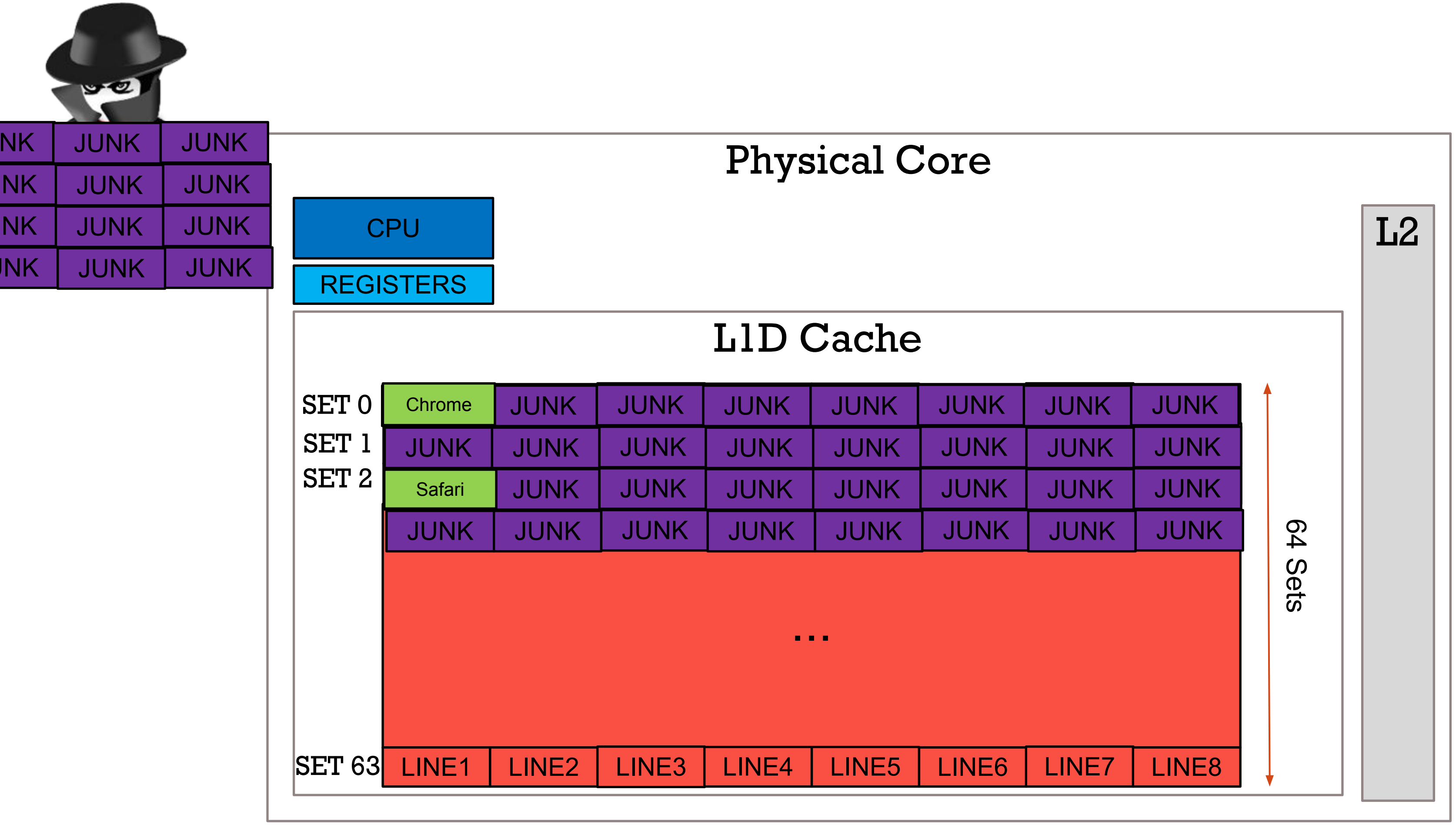
Step 1:
Attacker fills all the memory lines of a set

Step 2:
Victim accesses a memory that maps to the set and evicts a line

Step 3:
Attacker reads the lines and measures access times

Attacker slower access time → Victim has accessed the memory

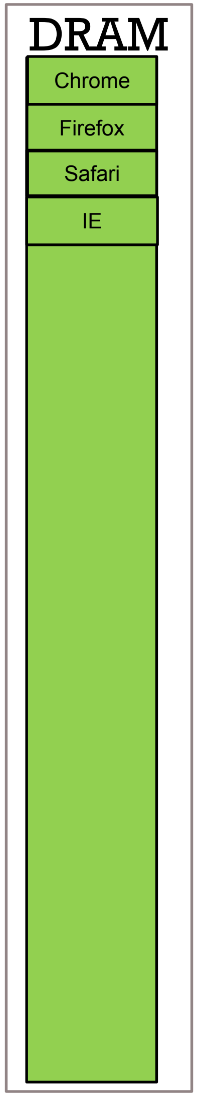
PRIME+PROBE




```
const const char browser_table[4][64] = {
    "CHROME",
    "FIREFOX",
    "SAFARI",
    "IE",
};

  char * my_browser;
  my_browser = browser_table[2];

  my_browser = browser_table[0];
```



PRIME+PROBE ON SGX AKA CACHEZOOM

1. Isolation of the target & victim cache
2. Stabilize the processor clock cycle
3. Craft noise free Prime+Probe code stub
4. Perform the attack in small execution units
5. Measure and filter the remaining noise

CACHEZOOM: CACHE ISOLATION

- Manipulate the OS page table entries to partition cache sets (LLC)
- Manipulate the kernel task scheduler to isolate cores (Core-Private Cache)

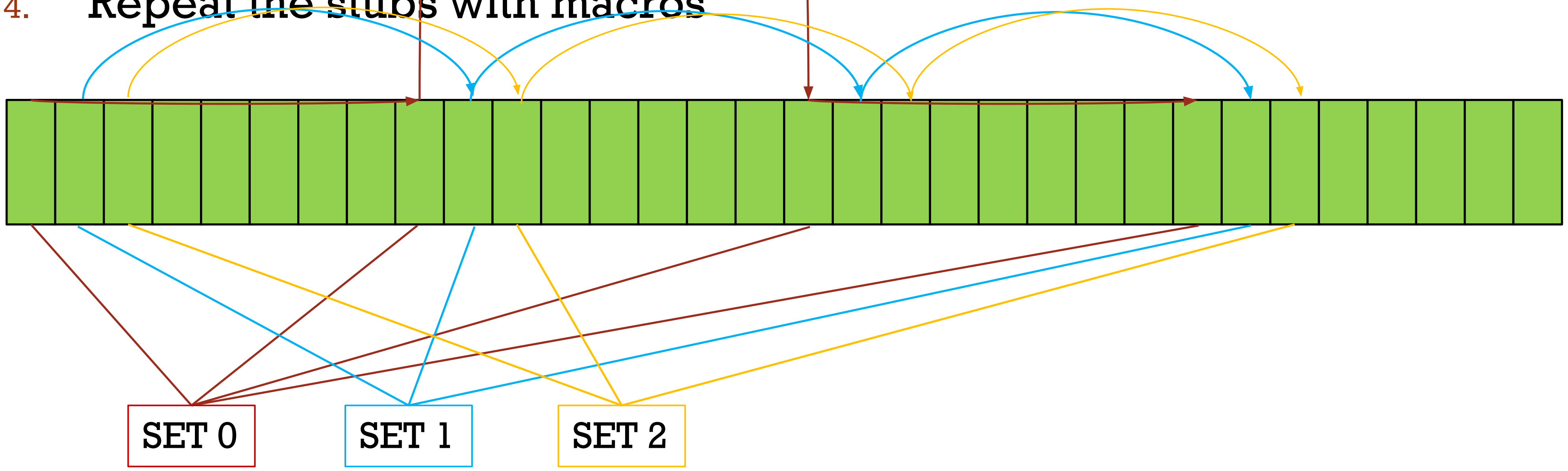
CACHEZOOM: PROCESSOR SPEED

- SpeedStep: Dynamic Frequency Scaling
- C-State: Power Saving Mode
- instructions/cycle depends on CPU frequency

- Configured from OS → Stable CPU Frequency → Reduced noise

CACHEZOOM: PRIME+PROBE CODE STUB

- Pointer chasing Approach
  1. A buffer with link-lists of pointers associated to the same set
  2. Traverse the pointers using a minimal machine code (Prime)
  3. Traverse the pointers and measure time (Probe)
  4. Repeat the stubs with macros



CACHEZOOM: PRIME+PROBE CODE STUB

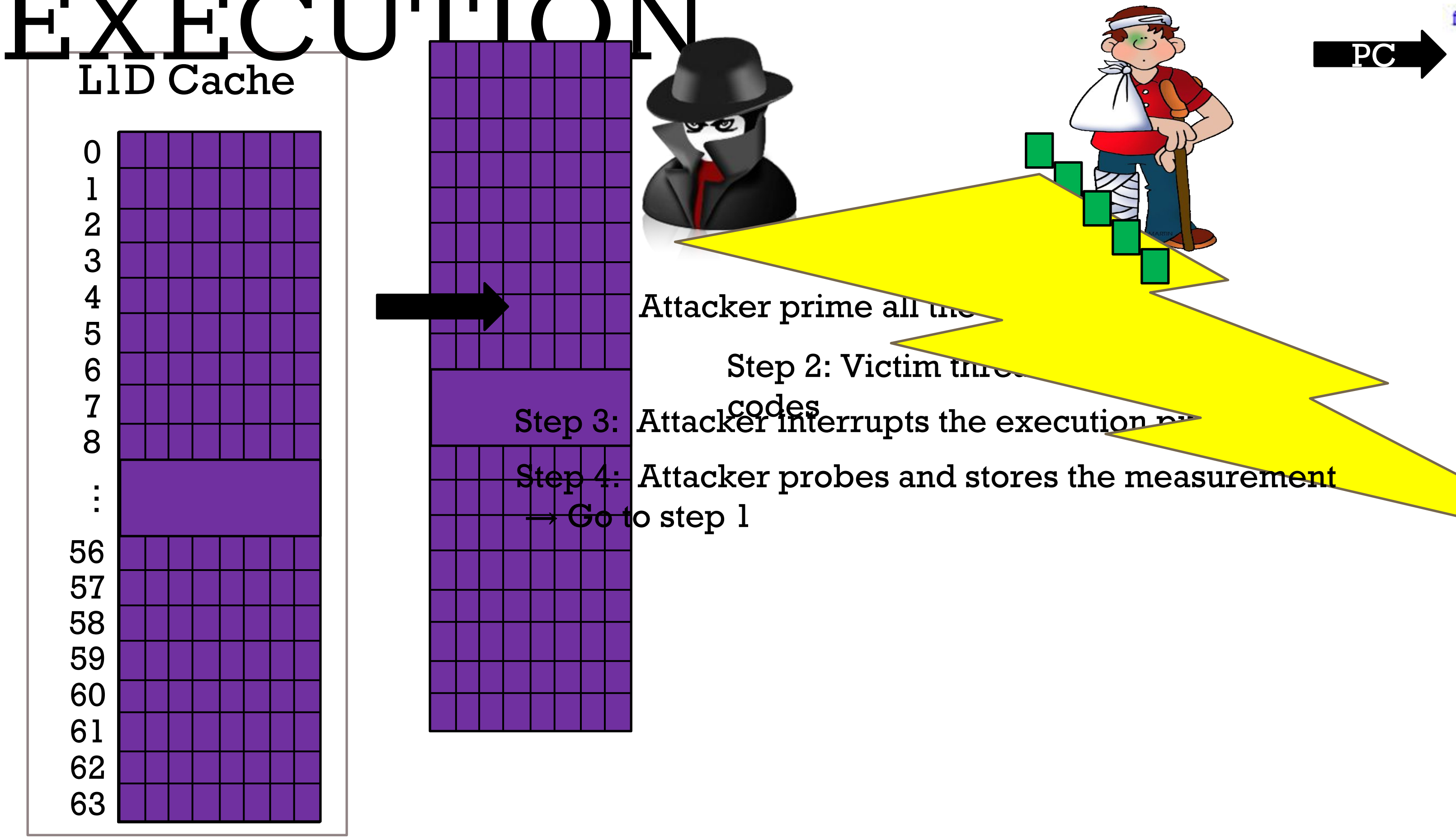
```
mfence ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
```

```
mov %rax , %r10 ;
mfence ;
rdtsc ;
mov %eax , %ecx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
mov (%rbx ) , %rbx ;
lfence ;
rdtsc ;
sub %rax , %rcx ;
mov %r10 , %rax ;
neg %rcx ;
```

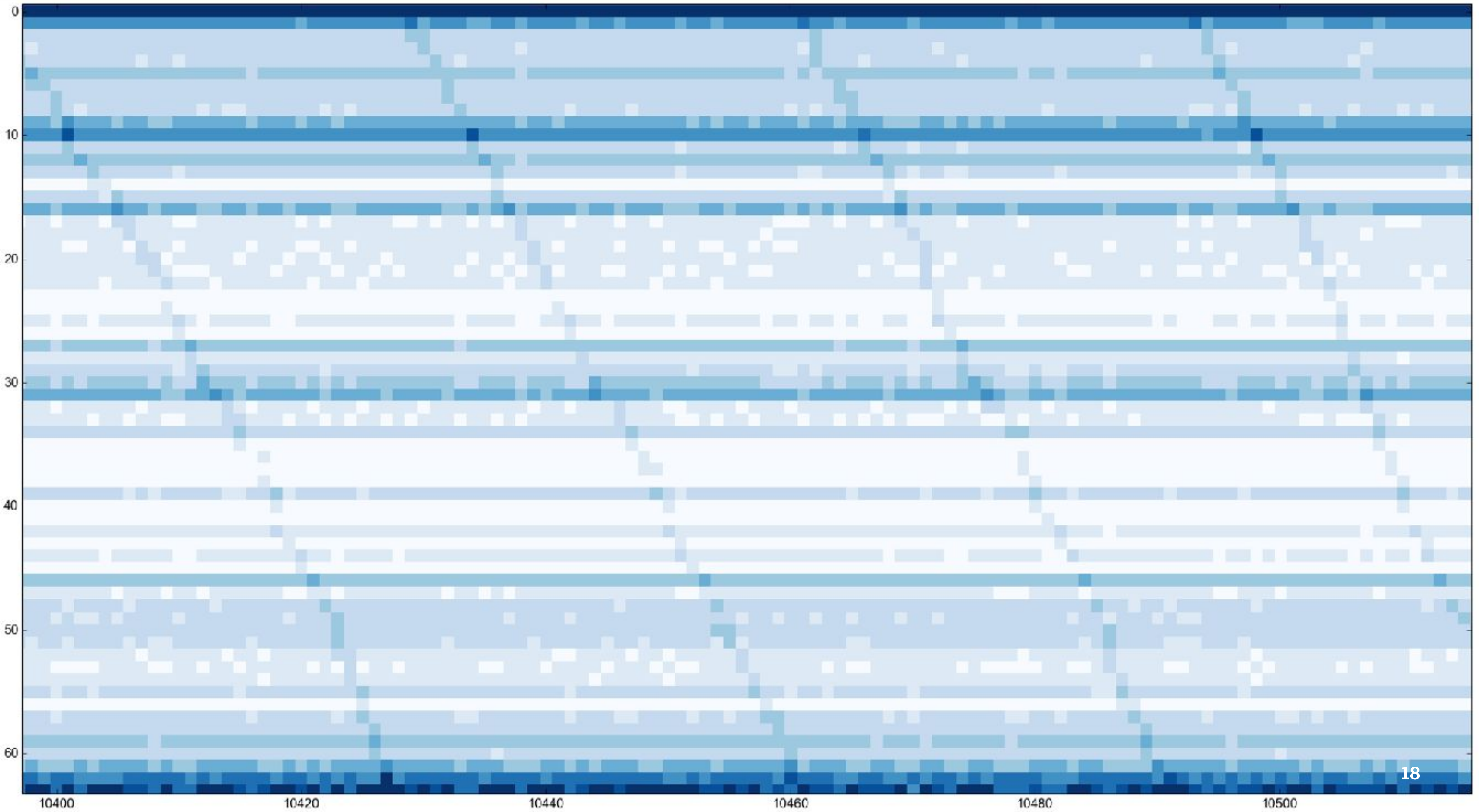
```
#define PRIME prime(_SPY_TABLE_, _CURRENT_SET_);
#define PRIME_2 PRIME; PRIME
#define PRIME_4 PRIME_2; PRIME_2
#define PRIME_8 PRIME_4; PRIME_4
#define PRIME_16 PRIME_8; PRIME_8
#define PRIME_32 PRIME_16; PRIME_16
#define PRIME_64 PRIME_32; PRIME_32

#define PROBE probe(_SPY_TABLE_, _CURRENT_SET_);
#define PROBE_2 PROBE; PROBE
#define PROBE_4 PROBE_2; PROBE_2
#define PROBE_8 PROBE_4; PROBE_4
#define PROBE_16 PROBE_8; PROBE_8
#define PROBE_32 PROBE_16; PROBE_16
#define PROBE_64 PROBE_32; PROBE_32
```

CACHEZOOM: INTERRUPTED EXECUTION

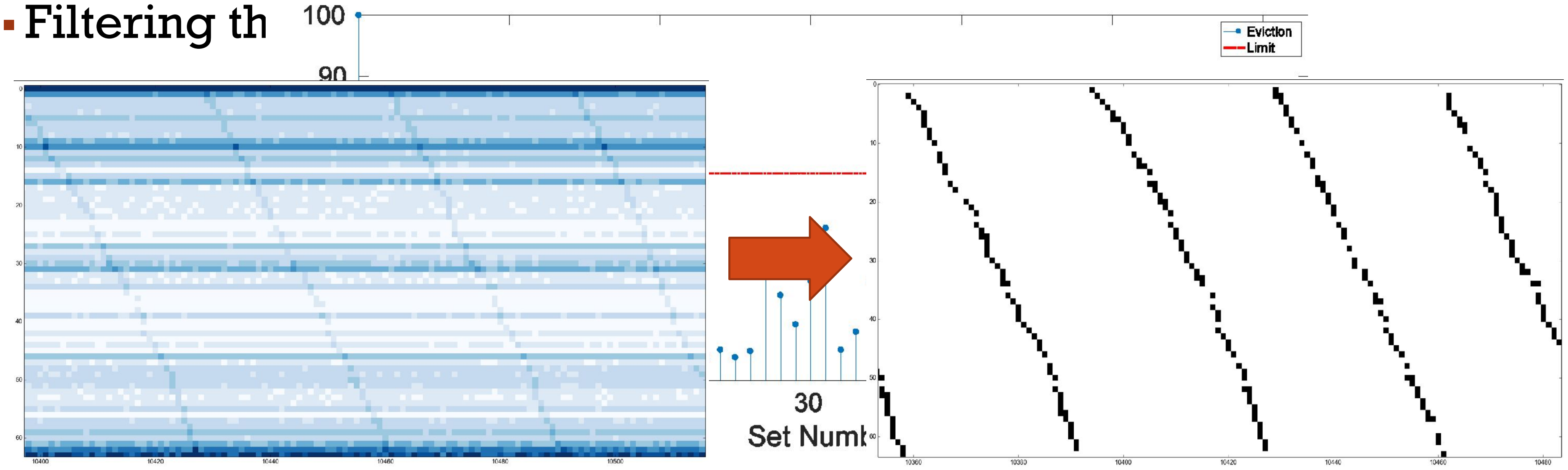



```
for (;;) {
    t0 =
        Te0[(s0 >> 24)       ] ^
        Te1[(s1 >> 16) & 0xff] ^
        Te2[(s2 >>  8) & 0xff] ^
        Te3[(s3      ) & 0xff] ^
        rk[4];
    t1 =
        Te0[(s1 >> 24)       ] ^
        Te1[(s2 >> 16) & 0xff] ^
        Te2[(s3 >>  8) & 0xff] ^
        Te3[(s0      ) & 0xff] ^
        rk[5];
    t2 =
        Te0[(s2 >> 24)       ] ^
        Te1[(s3 >> 16) & 0xff] ^
        Te2[(s0 >>  8) & 0xff] ^
        Te3[(s1      ) & 0xff] ^
        rk[6];
    t3 =
        Te0[(s3 >> 24)       ] ^
        Te1[(s0 >> 16) & 0xff] ^
        Te2[(s1 >>  8) & 0xff] ^
        Te3[(s2      ) & 0xff] ^
        rk[7];
    rk += 8;
    x |= PreFetchTe();
    if (--r == 0) {
        break;
    }
    s0 =
        Te0[(t0 >> 24)       ] ^
        Te1[(t1 >> 16) & 0xff] ^
        Te2[(t2 >>  8) & 0xff] ^
        Te3[(t3      ) & 0xff] ^
```


0
10
20
30
40
50
60
10400
10420
10440
10460
10480
10500

CACHEZOOM: NOISE FILTERING

- Context-switch noise: unavoidable but predictable
- Cycles based on the number of evictions
- evictions/set caused by an empty enclave
- Filtering th



AES S-BOX & T-TABLE

- SubBytes replace each byte with the output of a non-linear function.
  - A precomputed 256 entries S-Box table is used.
  - Main source of leakage
- MixColumns+SubBytes in a single table → T-Table
- We attack:
  - 4 T-tables: 256 entries each, each entry is 32 bits long
  - Big T-table: A single 256 entries table, each entry is 64 bits long.
  - S-Box: A 256 entries each, each entry is 8 bits long

Algorithm 1 AES Encrption

```
procedure ENCRYPT
    i ← 0
    ExpandKeys
    AddRoundKey(i)
round:
    SubBytes
    ShiftRows
    MixColumns
    AddRoundKey(i)
    i ← i + 1
    if i < n − 1 then
        goto round
    SubBytes
    ShiftRows
    AddRoundKey(i)
```

```
t0 =
    Te0[(s0 >> 24)       ] ^
    Te1[(s1 >> 16) & 0xff] ^
    Te2[(s2 >>  8) & 0xff] ^
    Te3[(s3      ) & 0xff] ^
    rk[4];
```

CACHEZOOMING AES INSIDE ENCLAVE

- Attack on Encryption
- Assumptions:
  - Access to OS resources & the target enclave binary
  - The execution is protected by SGX Enclave
  - No knowledge of the cipher key used inside enclave
    (Keys are generated at runtime)
  - Knowledge of Plaintext (Known Plaintext Attack)
    or Ciphertext (Known Ciphertext Attack)

AES T-TABLE: KPA

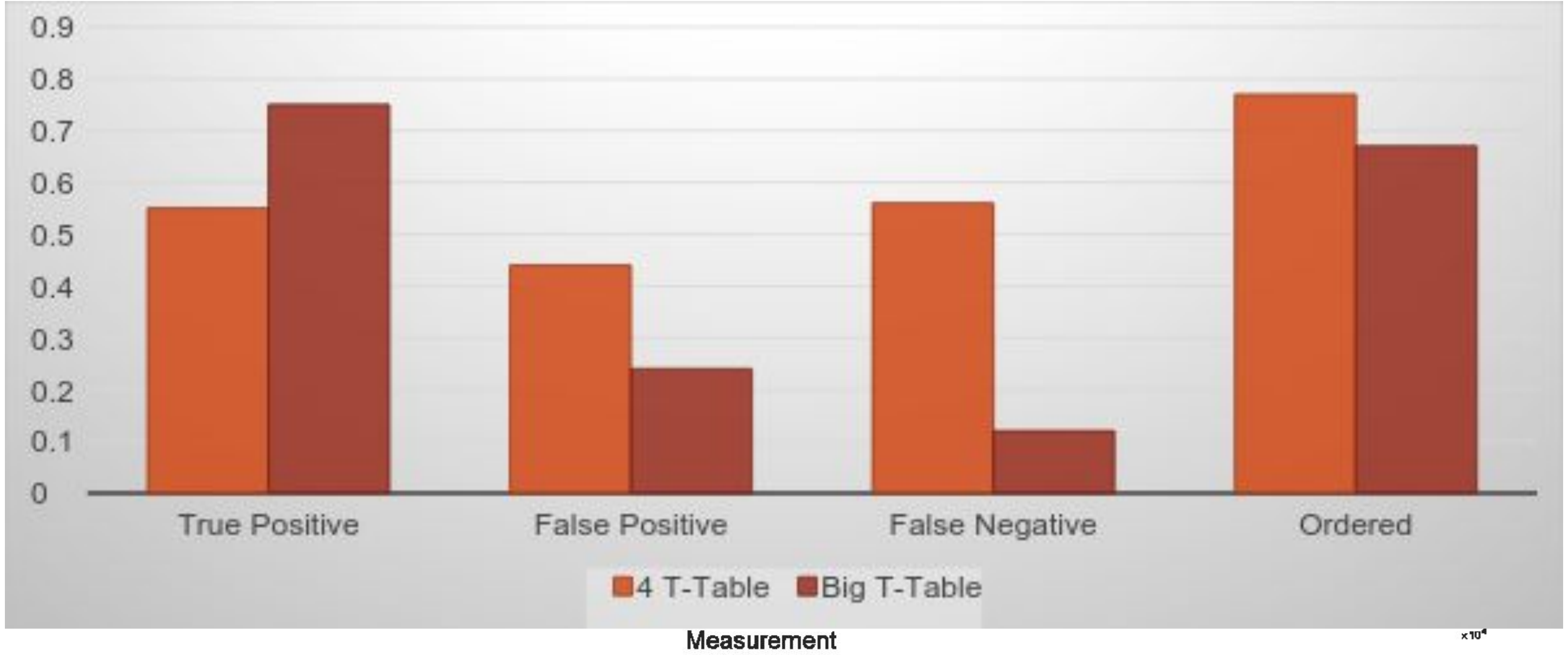
- Input:
  - Knowledge of the plaintext
  - Memory trace of the 1^{st} round after initial key addition (16 table lookups)
  - Memory trace of the 2^{nd} round. (16 table lookups)
- Output:
  - Small Key Space

- A perfect trace of the first round information outputs
  - 4 bits of each key byte in 4 T-Table (total of 64 bits)
  - 5 bits of each key byte in big T-table (total of 80 bits)
- Solving key relations of 1^{st} and 2^{nd} rounds with 2^{nd} round leakage reduce the key space to 8-16 bits [3].

[3] A. C, R. P. Giri, and B. Menezes. Highly efficient algorithms for aes key retrieval in cache access attacks. In 2016 IEEE European Symposium on Security and Privacy (EuroS P), pages 261–275, March 2016.

AES T-TABLE: KPA CHALLENGES

- We don't live in a perfect world, and we have some noises.
  - Out-of-order execution
  - Repeated accesses to the same set
  - Blind Sets



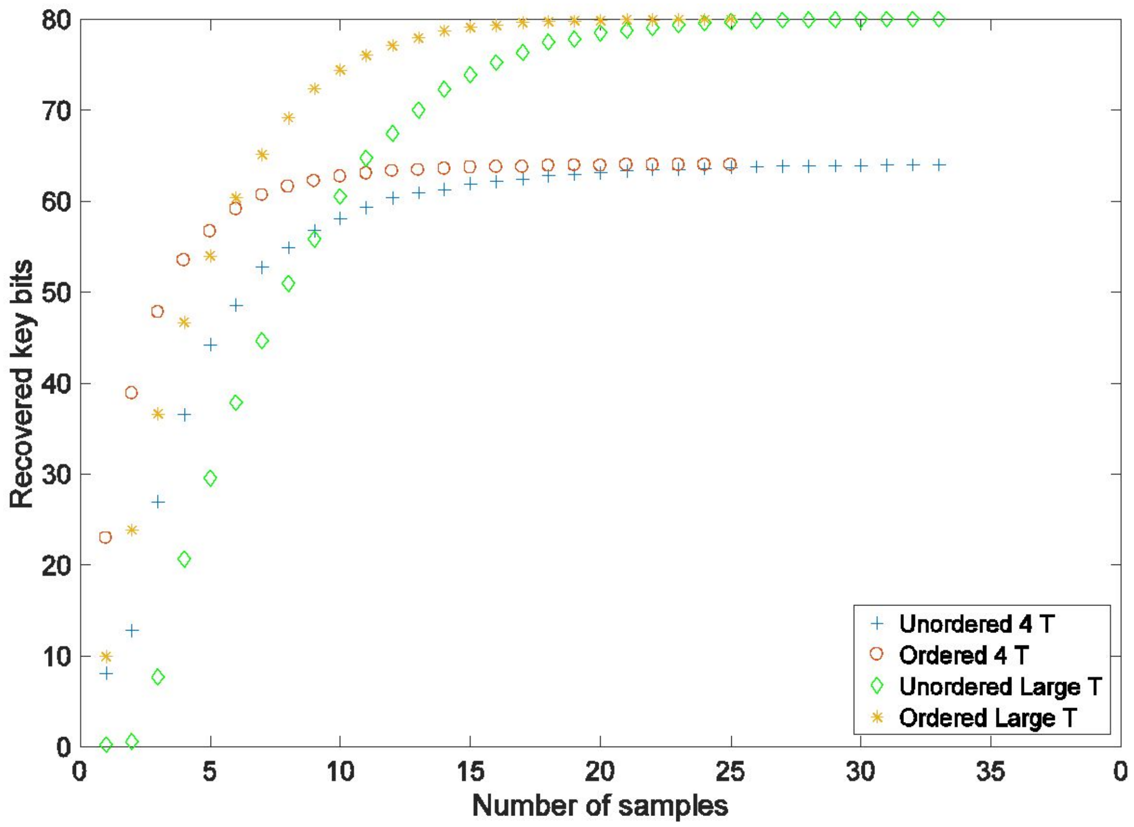
AES T-TABLE: KPA RESULTS

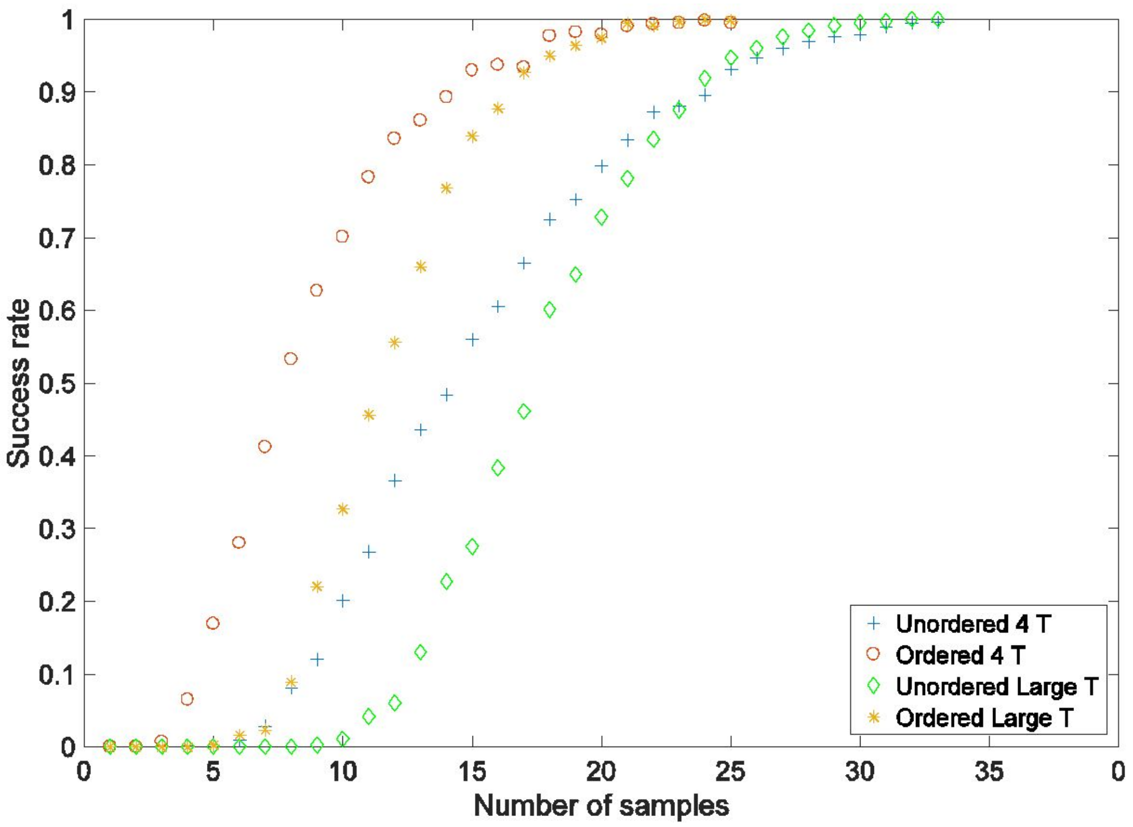
- Different plaintexts → different memory traces
- The scoring algorithm finds the cipher key:
  - Input : 15-25 (cache based traces + plaintexts)
  - Output: cipher key
- Partial order of traces improves key recovery

  (Reduce the number of required measurements)

- First practical implementation on a real scenario

AES T-TABLE: KPA RESULTS



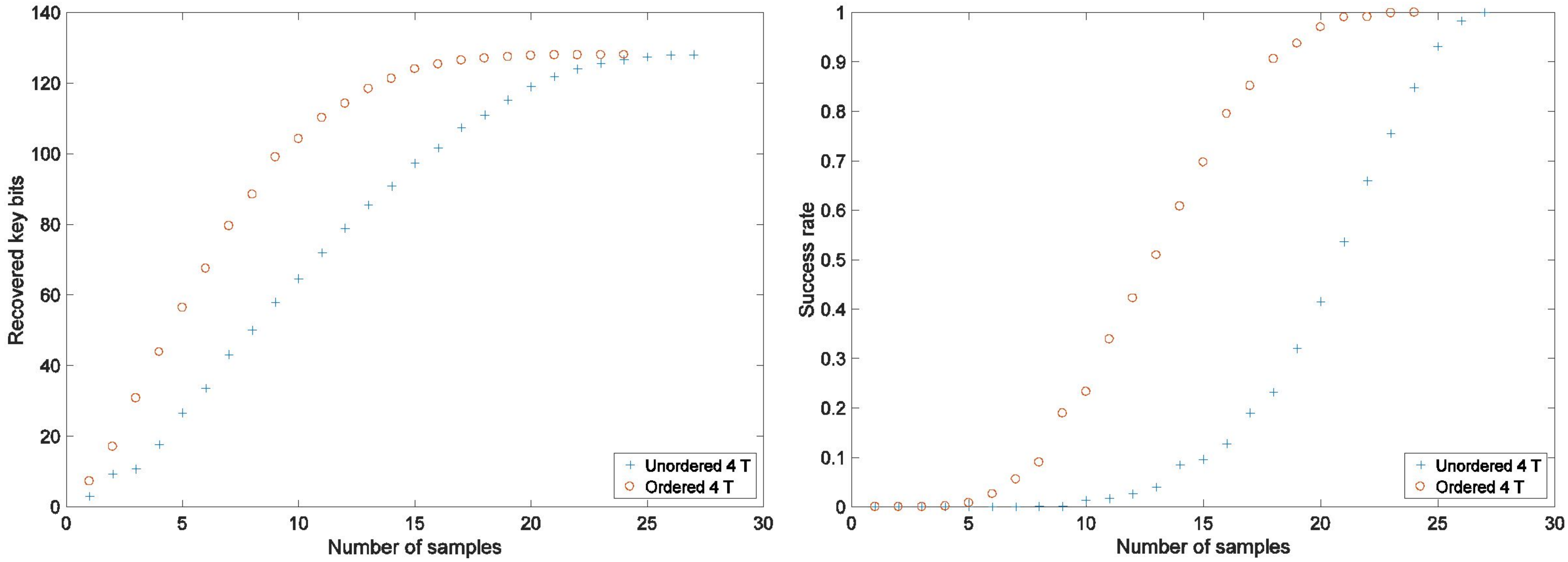


AES T-TABLE: KCA — WE CAN DO BETTER

- Last Round Only Attack
  - First round only leaks 64 bits (80 bits in Big T)
  - Last round observations with different ciphertext recover the entire key.
  - Faster key recovery than using $1^{st}+2^{nd}$ rounds information

- 9^{th} + Last Round Attack
  - The leakage of the 9^{th} round improves the last round attack.
  - $1^{st}+2^{nd}$ rounds attack reduces the key to 8-16 bits.
  - It recovers the exact key with an ideal trace.

AES T-TABLE: LAST ROUND ONLY RESULTS



AES T-TABLE: 9TH+LAST ROUND

- Hypothesis:
  - Last Round Only, 7 observations is enough
  - Can we reduce it?
- Goal:
  - Voiding the assumption that changing symmetric key makes it secure.
- Our constructed key relations and algorithm
  - Recover the key in 2 hours with ideal data
  - Verifies the exact key

AES S-BOX: KCA

- S-BOX table (256 bytes) only affects 4 cache sets.
- No clean order information for 16 accesses within 4 sets
  - Out-of-order execution
  - Repeated set accesses
- Harder to exploit but possible
- accesses/set depends on the key bytes
- Hundreds of measurements + DPA correlation attack

AES S-BOX: KCA RESULTS

- Hundreds of measurement
- Correlation between expected and observed accesses
- Higl
- 150(

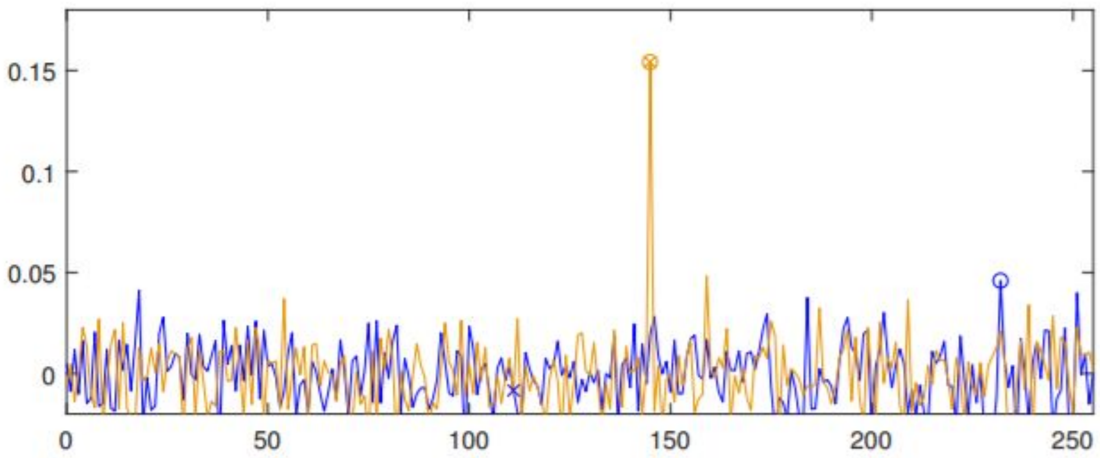
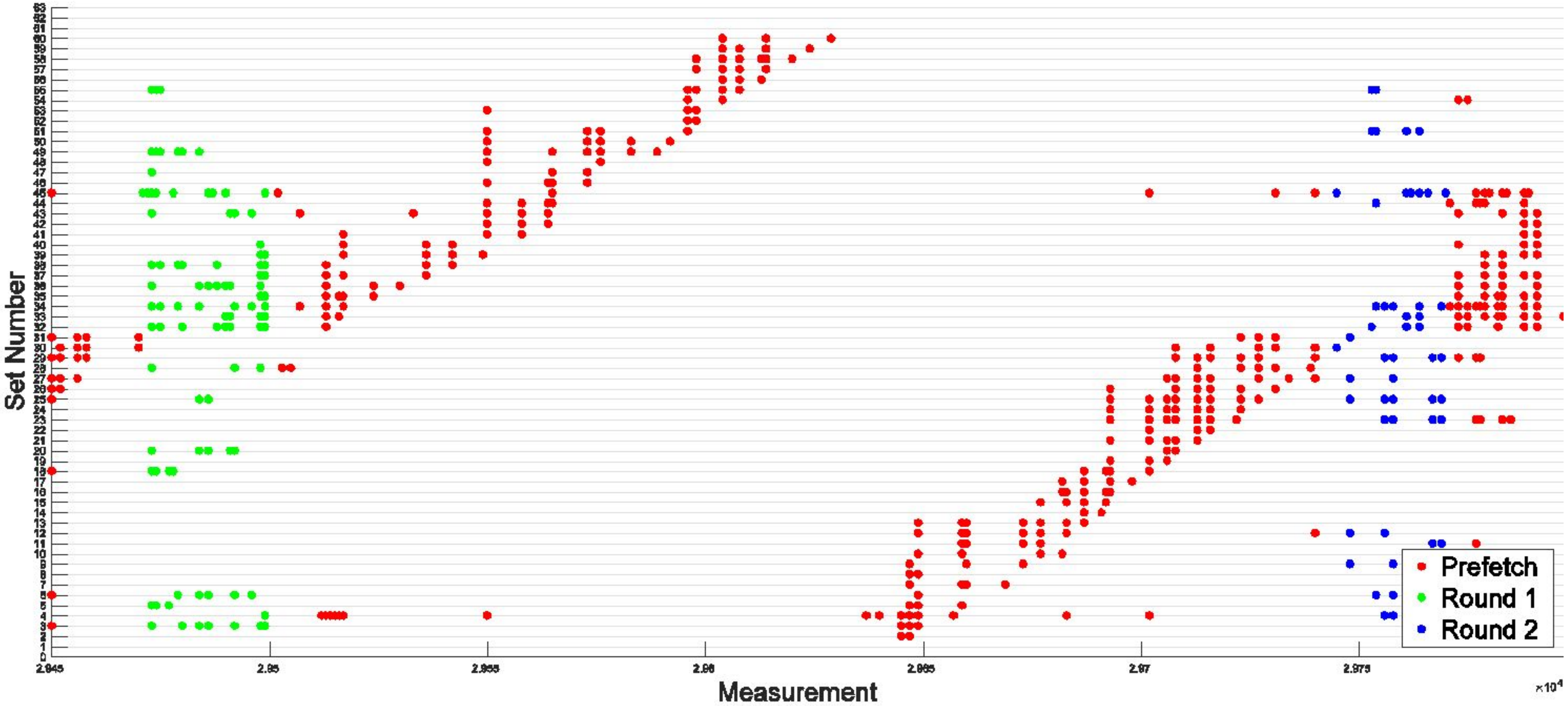


Figure 5.12: Correlation over key value for the best (k_{15}, amber) and worst (k_0, blue) byte positions based on 1500 traces. The guess with the highest correlation (o) and the correct key (x) a match only for k_{15}.

CACHE PREFETCHING

- Resistant to previous cache side-channels
- Vulnerable to our attack



DEFENSE AGAINST CACHEZOOM

- Avoiding data dependent computations
  - Costly and not always practical
- Measuring delay of execution
  - Needs to be done inside the enclave
- Separate caches for enclave
  - Core-private: Require hardware changes
  - LLC: Cache Allocation Technology (CAT)
- Flushing core-private caches?
  - Slow execution inside enclave

CONCLUSION

- Cache side-channel attacks are devastating in the SGX environment.
- Side-channels on intermediate operations of T-Table are powerful.
- AES S-BOX implementation and cache prefetching are not secure.
- Vulnerable software implementations are always bad.

THANKS

- Questions?

AES T-TABLE: 9TH+LAST ROUND EQUATIONS

G1

$$2s(x_0^9) \oplus 3s(x_1^9) \oplus s(x_2^9) \oplus s(x_3^9) = s^{-1}(c_0 \oplus k_{160}) \oplus k_{160} \oplus s(k_{173} \oplus k_{169}) \oplus 0x36$$
$$s(x_0^9) \oplus 2s(x_1^9) \oplus 3s(x_2^9) \oplus s(x_3^9) = s^{-1}(c_{13} \oplus k_{173}) \oplus k_{161} \oplus s(k_{174} \oplus k_{170})$$
$$s(x_0^9) \oplus s(x_1^9) \oplus 2s(x_2^9) \oplus 3s(x_3^9) = s^{-1}(c_{10} \oplus k_{170}) \oplus k_{162} \oplus s(k_{175} \oplus k_{171})$$
$$3s(x_0^9) \oplus s(x_1^9) \oplus s(x_2^9) \oplus 2s(x_3^9) = s^{-1}(c_7 \oplus k_{167}) \oplus k_{163} \oplus s(k_{172} \oplus k_{168})$$

G2

$$2s(x_4^9) \oplus 3s(x_5^9) \oplus s(x_6^9) \oplus s(x_7^9) = s^{-1}(c_4 \oplus k_{164}) \oplus k_{160} \oplus k_{164}$$
$$s(x_4^9) \oplus 2s(x_5^9) \oplus 3s(x_6^9) \oplus s(x_7^9) = s^{-1}(c_1 \oplus k_{161}) \oplus k_{161} \oplus k_{165}$$
$$s(x_4^9) \oplus s(x_5^9) \oplus 2s(x_6^9) \oplus 3s(x_7^9) = s^{-1}(c_{14} \oplus k_{174}) \oplus k_{162} \oplus k_{166}$$
$$3s(x_4^9) \oplus s(x_5^9) \oplus s(x_6^9) \oplus 2s(x_7^9) = s^{-1}(c_{11} \oplus k_{171}) \oplus k_{163} \oplus k_{167}$$

G3

$$2s(x_8^9) \oplus 3s(x_9^9) \oplus s(x_{10}^9) \oplus s(x_{11}^9) = s^{-1}(c_8 \oplus k_{168}) \oplus k_{164} \oplus k_{168}$$
$$s(x_8^9) \oplus 2s(x_9^9) \oplus 3s(x_{10}^9) \oplus s(x_{11}^9) = s^{-1}(c_5 \oplus k_{165}) \oplus k_{165} \oplus k_{169}$$
$$s(x_8^9) \oplus s(x_9^9) \oplus 2s(x_{10}^9) \oplus 3s(x_{11}^9) = s^{-1}(c_2 \oplus k_{162}) \oplus k_{166} \oplus k_{170}$$
$$3s(x_8^9) \oplus s(x_9^9) \oplus s(x_{10}^9) \oplus 2s(x_{11}^9) = s^{-1}(c_{15} \oplus k_{175}) \oplus k_{167} \oplus k_{171}$$

G4

$$2s(x_{12}^9) \oplus 3s(x_{13}^9) \oplus s(x_{14}^9) \oplus s(x_{15}^9) = s^{-1}(c_{12} \oplus k_{172}) \oplus k_{168} \oplus k_{172}$$
$$s(x_{12}^9) \oplus 2s(x_{13}^9) \oplus 3s(x_{14}^9) \oplus s(x_{15}^9) = s^{-1}(c_9 \oplus k_{169}) \oplus k_{169} \oplus k_{173}$$
$$s(x_{12}^9) \oplus s(x_{13}^9) \oplus 2s(x_{14}^9) \oplus 3s(x_{15}^9) = s^{-1}(c_6 \oplus k_{166}) \oplus k_{170} \oplus k_{174}$$
$$3s(x_{12}^9) \oplus s(x_{13}^9) \oplus s(x_{14}^9) \oplus 2s(x_{15}^9) = s^{-1}(c_3 \oplus k_{163}) \oplus k_{171} \oplus k_{175}$$

AES T-TABLE: 9TH+LAST ROUND EQUATIONS

G2

4 4 4 4 4 4 4

$$2s(x_4^9) \oplus 3s(x_5^9) \oplus s(x_6^9) \oplus s(x_7^9) = s^{-1}(c_4 \oplus k_{164}) \oplus k_{160} \oplus k_{164} \rightarrow (x_4^9, x_5^9, x_6^9, x_7^9, k_{160}, k_{164})$$

$$s(x_4^9) \oplus 2s(x_5^9) \oplus 3s(x_6^9) \oplus s(x_7^9) = s^{-1}(c_1 \oplus k_{161}) \oplus k_{161} \oplus k_{165} \rightarrow (x_4^9, x_5^9, x_6^9, x_7^9, k_{161}, k_{165})$$

$$s(x_4^9) \oplus s(x_5^9) \oplus 2s(x_6^9) \oplus 3s(x_7^9) = s^{-1}(c_{14} \oplus k_{174}) \oplus k_{162} \oplus k_{166} \rightarrow (x_4^9, x_5^9, x_6^9, x_7^9, k_{162}, k_{166}, k_{174})$$

$$3s(x_4^9) \oplus s(x_5^9) \oplus s(x_6^9) \oplus 2s(x_7^9) = s^{-1}(c_{11} \oplus k_{171}) \oplus k_{163} \oplus k_{167} \rightarrow (x_4^9, x_5^9, x_6^9, x_7^9, k_{163}, k_{167}, k_{171})$$

$\rightarrow$ 24 bits, $(k_{160}, k_{161}, k_{162}, k_{163}, k_{164}, k_{165}, k_{166}, k_{167}, k_{171}, k_{174})$

Step 1:
Reduce $(x_4^9, x_5^9, x_6^9, x_7^9)$ using all equations
$\rightarrow$14 bits, $(x_4^9, x_5^9, x_6^9, x_7^9)$

Step 2:
Solve each equation using reduced
$(x_4^9, x_5^9, x_6^9, x_7^9)$

AES T-TABLE: 9TH+LAST ROUND EQUATIONS

G1

$$2s(x_0^9) \oplus 3s(x_1^9) \oplus s(x_2^9) \oplus s(x_3^9) = s^{-1}(c_0 \oplus k_{160}) \oplus k_{160} \oplus s(k_{173} \oplus k_{169}) \oplus 0x36$$
$$s(x_0^9) \oplus 2s(x_1^9) \oplus 3s(x_2^9) \oplus s(x_3^9) = s^{-1}(c_{13} \oplus k_{173}) \oplus k_{161} \oplus s(k_{174} \oplus k_{170})$$
$$s(x_0^9) \oplus s(x_1^9) \oplus 2s(x_2^9) \oplus 3s(x_3^9) = s^{-1}(c_{10} \oplus k_{170}) \oplus k_{162} \oplus s(k_{175} \oplus k_{171})$$
$$3s(x_0^9) \oplus s(x_1^9) \oplus s(x_2^9) \oplus 2s(x_3^9) = s^{-1}(c_7 \oplus k_{167}) \oplus k_{163} \oplus s(k_{172} \oplus k_{168})$$

G2

$$2s(x_4^9) \oplus 3s(x_5^9) \oplus s(x_6^9) \oplus s(x_7^9) = s^{-1}(c_4 \oplus k_{164}) \oplus k_{160} \oplus k_{164}$$
$$s(x_4^9) \oplus 2s(x_5^9) \oplus 3s(x_6^9) \oplus s(x_7^9) = s^{-1}(c_1 \oplus k_{161}) \oplus k_{161} \oplus k_{165}$$
$$s(x_4^9) \oplus s(x_5^9) \oplus 2s(x_6^9) \oplus 3s(x_7^9) = s^{-1}(c_{14} \oplus k_{174}) \oplus k_{162} \oplus k_{166}$$
$$3s(x_4^9) \oplus s(x_5^9) \oplus s(x_6^9) \oplus 2s(x_7^9) = s^{-1}(c_{11} \oplus k_{171}) \oplus k_{163} \oplus k_{167}$$

$\rightarrow$ 24 bits, $(k_{160}, k_{161}, k_{162}, k_{163}, k_{164}, k_{165}, k_{166}, k_{167}, k_{171}, k_{174})$

G3

$$2s(x_8^9) \oplus 3s(x_9^9) \oplus s(x_{10}^9) \oplus s(x_{11}^9) = s^{-1}(c_8 \oplus k_{168}) \oplus k_{164} \oplus k_{168}$$
$$s(x_8^9) \oplus 2s(x_9^9) \oplus 3s(x_{10}^9) \oplus s(x_{11}^9) = s^{-1}(c_5 \oplus k_{165}) \oplus k_{165} \oplus k_{169}$$
$$s(x_8^9) \oplus s(x_9^9) \oplus 2s(x_{10}^9) \oplus 3s(x_{11}^9) = s^{-1}(c_2 \oplus k_{162}) \oplus k_{166} \oplus k_{170}$$
$$3s(x_8^9) \oplus s(x_9^9) \oplus s(x_{10}^9) \oplus 2s(x_{11}^9) = s^{-1}(c_{15} \oplus k_{175}) \oplus k_{167} \oplus k_{171}$$

$\rightarrow$ 24 bits, $(k_{162}, k_{164}, k_{165}, k_{166}, k_{167}, k_{168}, k_{169}, k_{170}, k_{171}, k_{175})$

G4

$$2s(x_{12}^9) \oplus 3s(x_{13}^9) \oplus s(x_{14}^9) \oplus s(x_{15}^9) = s^{-1}(c_{12} \oplus k_{172}) \oplus k_{168} \oplus k_{172}$$
$$s(x_{12}^9) \oplus 2s(x_{13}^9) \oplus 3s(x_{14}^9) \oplus s(x_{15}^9) = s^{-1}(c_9 \oplus k_{169}) \oplus k_{169} \oplus k_{173}$$
$$s(x_{12}^9) \oplus s(x_{13}^9) \oplus 2s(x_{14}^9) \oplus 3s(x_{15}^9) = s^{-1}(c_6 \oplus k_{166}) \oplus k_{170} \oplus k_{174}$$
$$3s(x_{12}^9) \oplus s(x_{13}^9) \oplus s(x_{14}^9) \oplus 2s(x_{15}^9) = s^{-1}(c_3 \oplus k_{163}) \oplus k_{171} \oplus k_{175}$$

$\rightarrow$ 24 bits, $(k_{163}, k_{166}, k_{168}, k_{169}, k_{170}, k_{171}, k_{172}, k_{173}, k_{174}, k_{175})$

AES T-TABLE: 9TH+LAST ROUND EQUATIONS

G1

$$2s(x_0^9) \oplus 3s(x_1^9) \oplus s(x_2^9) \oplus s(x_3^9) = s^{-1}(c_0 \oplus k_{160}) \oplus k_{160} \oplus s(k_{173} \oplus k_{169}) \oplus 0x36$$

$$s(x_0^9) \oplus 2s(x_1^9) \oplus 3s(x_2^9) \oplus s(x_3^9) = s^{-1}(c_{13} \oplus k_{173}) \oplus k_{161} \oplus s(k_{174} \oplus k_{170})$$

$$s(x_0^9) \oplus s(x_1^9) \oplus 2s(x_2^9) \oplus 3s(x_3^9) = s^{-1}(c_{10} \oplus k_{170}) \oplus k_{162} \oplus s(k_{175} \oplus k_{171})$$

$$3s(x_0^9) \oplus s(x_1^9) \oplus s(x_2^9) \oplus 2s(x_3^9) = s^{-1}(c_7 \oplus k_{167}) \oplus k_{163} \oplus s(k_{172} \oplus k_{168})$$

G2 → 24 bits, $(k_{160}, k_{161}, k_{162}, k_{163}, k_{164}, k_{165}, k_{166}, k_{167}, k_{171}, k_{174})$

G3 → 24 bits, $(k_{162}, k_{164}, k_{165}, k_{166}, k_{167}, k_{168}, k_{169}, k_{170}, k_{171}, k_{175})$

G4 → 24 bits, $(k_{163}, k_{166}, k_{168}, k_{169}, k_{170}, k_{171}, k_{172}, k_{173}, k_{174}, k_{175})$

} 16 bits, (*All the key byte*

↓

Verify the valid key by G1 Equations

↓

Valid Key

AES S-BOX: KCA

- $s^{-1}(c_i \otimes k_i) \gg 6 = set\ number$
- **Expected Matrix A:** Rows correspond to ciphertexts and columns shows the excepted accessed cache lines.

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

- **Leackage Matrix L:** Rows correspond to ciphertexts and columns shows the number of accesses/cache lines observed from the attack.

$$L = \begin{bmatrix} 6 & 3 & 2 & 4 \\ 2 & 10 & 4 & 3 \\ 6 & 6 & 4 & 2 \\ 5 & 1 & 4 & 3 \end{bmatrix}$$

- 0.25 correlation between L and A → alidates our approach

COMPARISON OF SIDE CHANNELS ON SGX

Channel	CPC[1]	LLC[2]	BP[3]	PF[4]	TLB[5]	RH[6]
Possible	Yes	Yes	Yes	Yes	No	No
Resolution	64 byte	64 byte	Branches	4 kB	N/A	N/A
Noise	Local	Global	Local	N/A	N/A	N/A
Target	Data+Code	Data+Code	Code	Data+Code	N/A	N/A

[1] Core-Private Cache; [2] Last-Level Cache; [3] Branch Predictor Cache;
[4] Page Fault; [5] TLB Cache; [6] Rowhammeringt