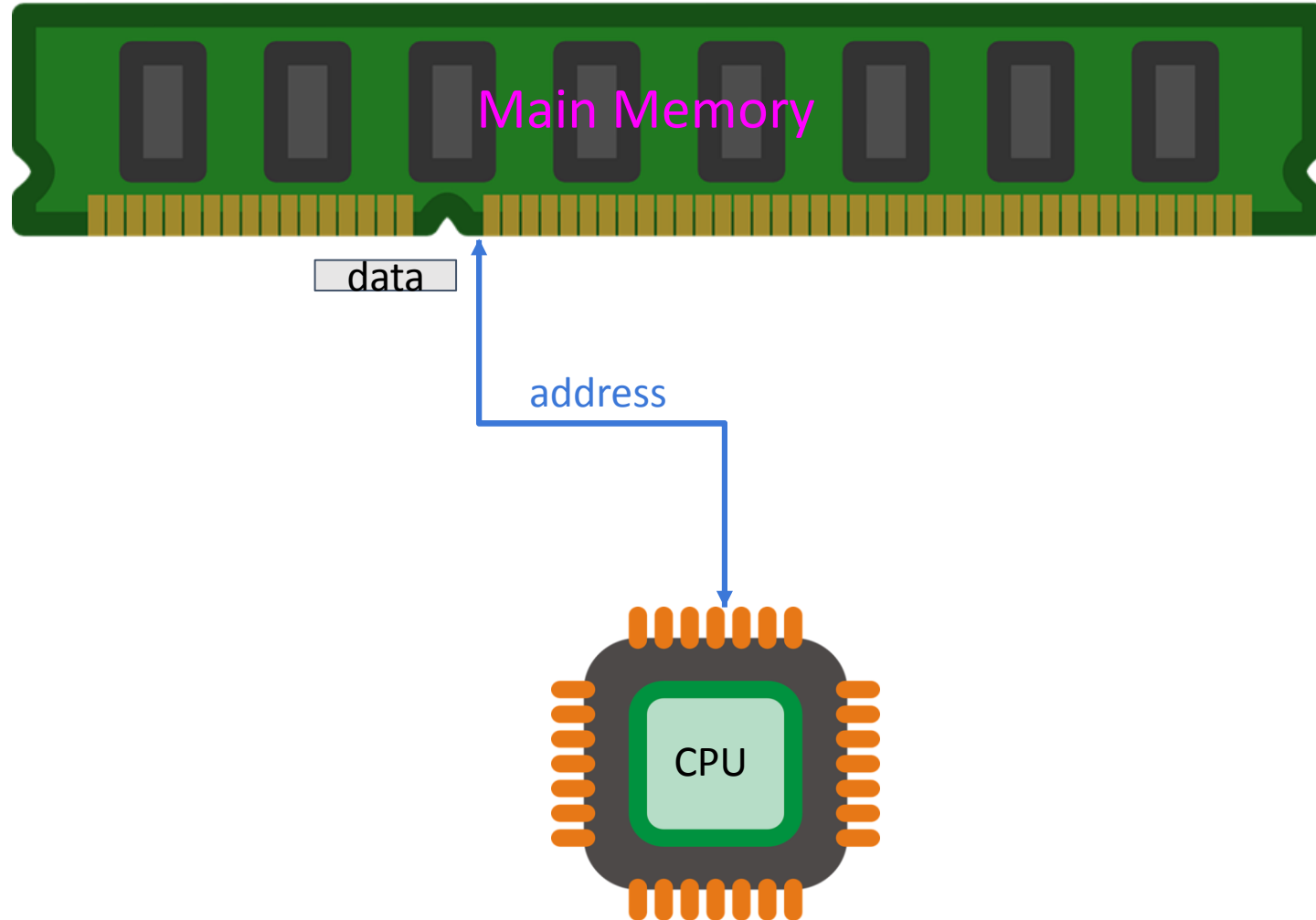


Cache Side-Channel Attacks (Brief Introduction)

Presenter: Aria Shahverdi

12/6/2017

How do we load data from Main Memory?

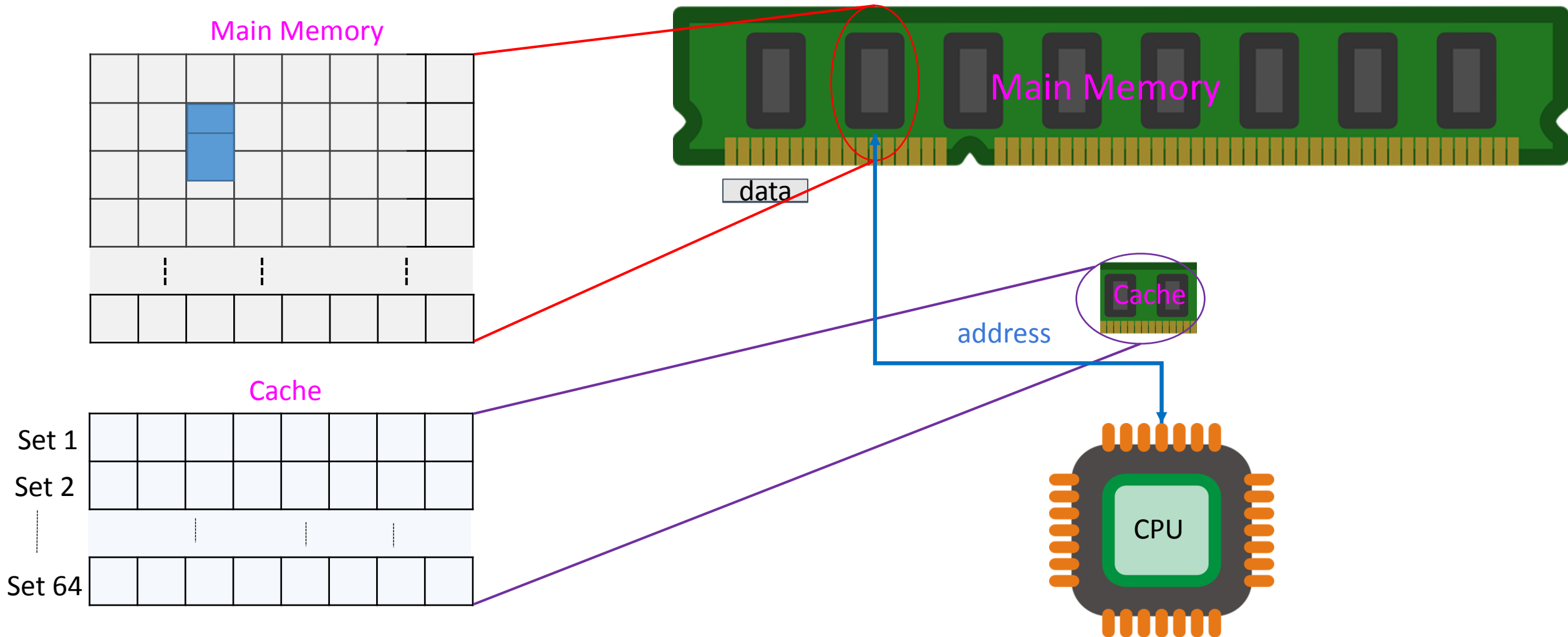


Memory Locality

- Future memory accesses are near past memory accesses
- Memories take advantages of two locality
 - **Temporal Locality**: near in time
 - We will often access the same data again very soon
 - **Spatial Locality**: near in space/distance
 - Our next access is often very close to our last access (or recent accesses)

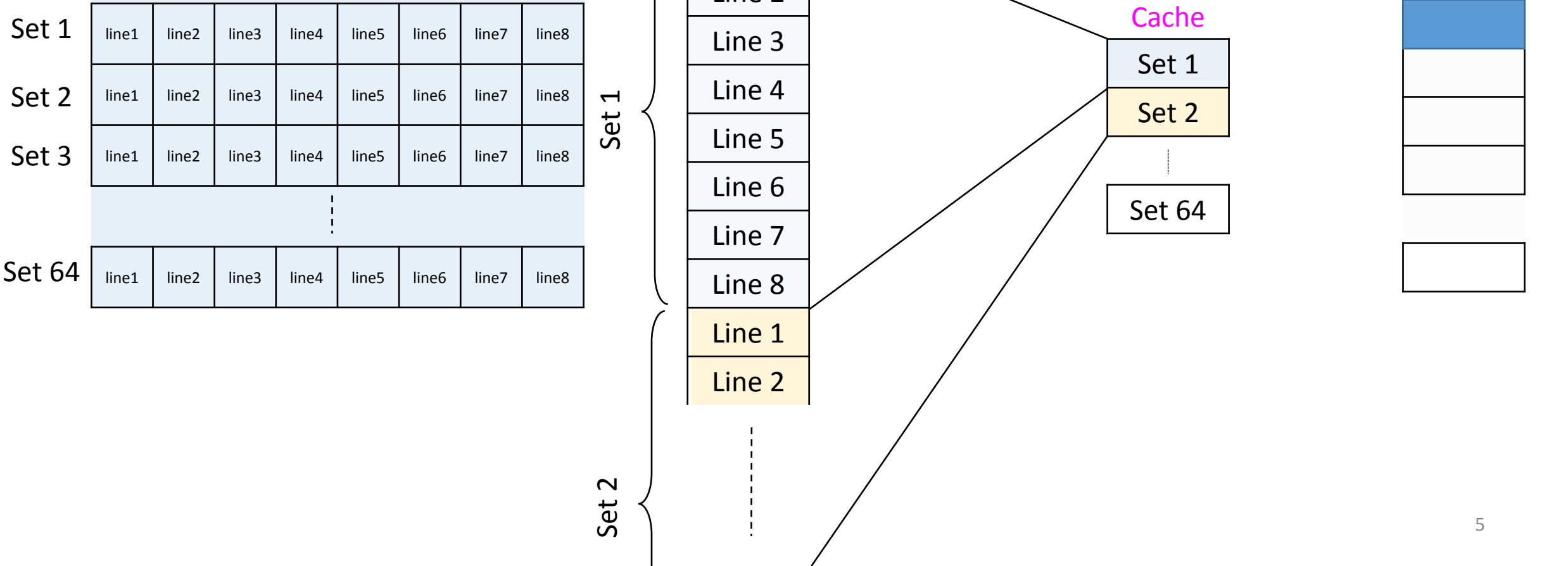
```
for (i = 0; i < 20; i++)           a[0]  
    a[i] = a[i]*2;                 a[1]  
                                       a[2]  
                                       ...
```

Cache Architecture High Level



Set Associative Cache Architecture

8-way set associative Cache



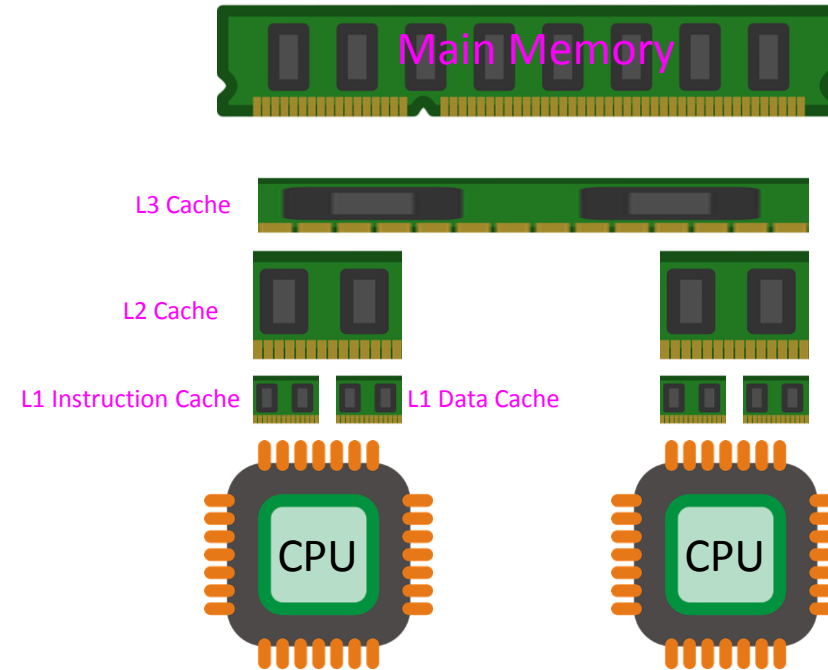
Introduction to Cache Architecture

Processor Name: Intel Core i5
Code Name: Skylake-U/Y Max TDP: 15.0 W
Package: Socket 1168 BGA
Technology: 14 nm Core VID: 0.885 V

Specification: Intel® Core™ i5-6200U CPU @ 2.30GHz
Family: 6 Model: E Stepping: 3
Ext. Family: 6 Ext. Model: 4E Revision: D0/K0/K1
Instructions: MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX, AVX2, FMA3

Clocks (Core #0)
Core Speed: 2295.51 MHz
Multiplier: x 23.0 (4 - 28)
Bus Speed: 99.81 MHz
Rated FSB:
Cache:
L1 Data: 2 x 32 KBytes 8-way
L1 Inst.: 2 x 32 KBytes 8-way
Level 2: 2 x 256 KBytes 4-way
Level 3: 3 MBytes 12-way

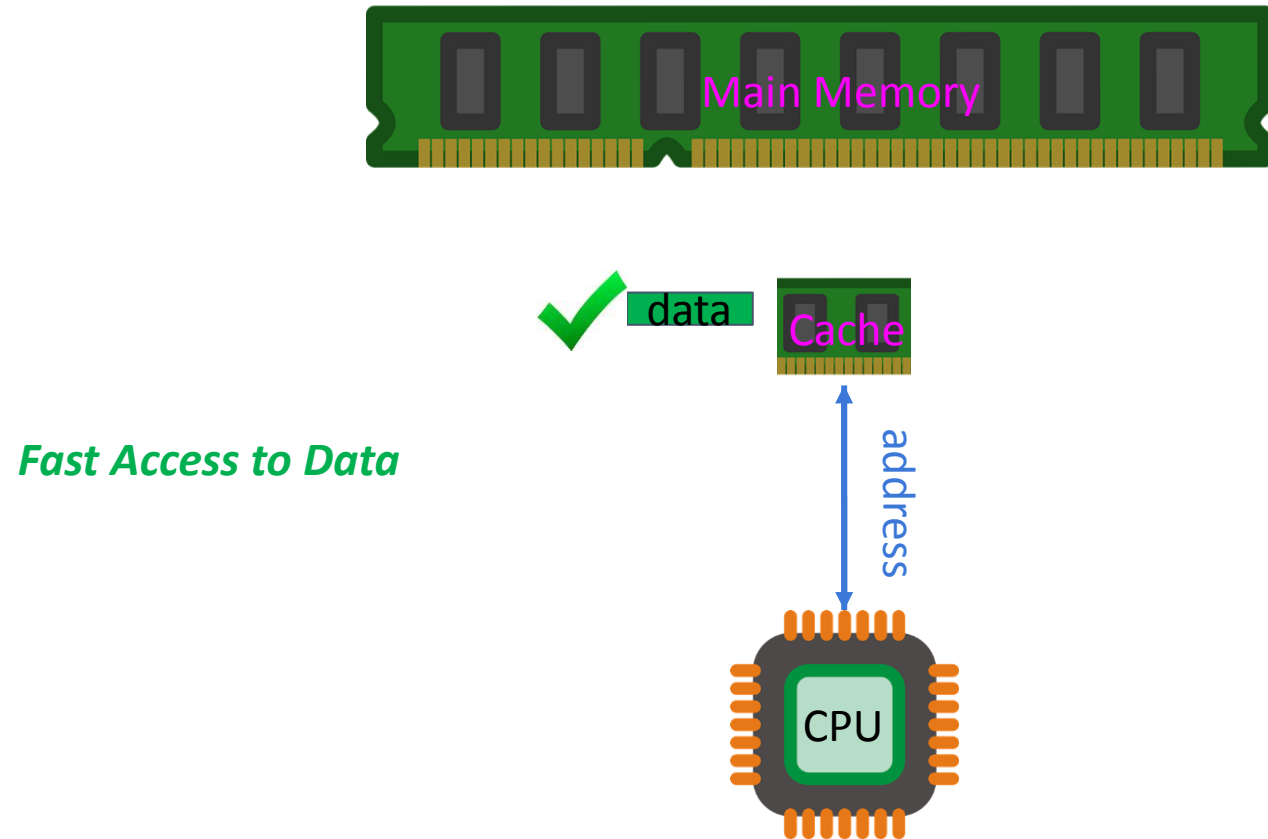
Selection: Socket #1 Cores: 2 Threads: 4



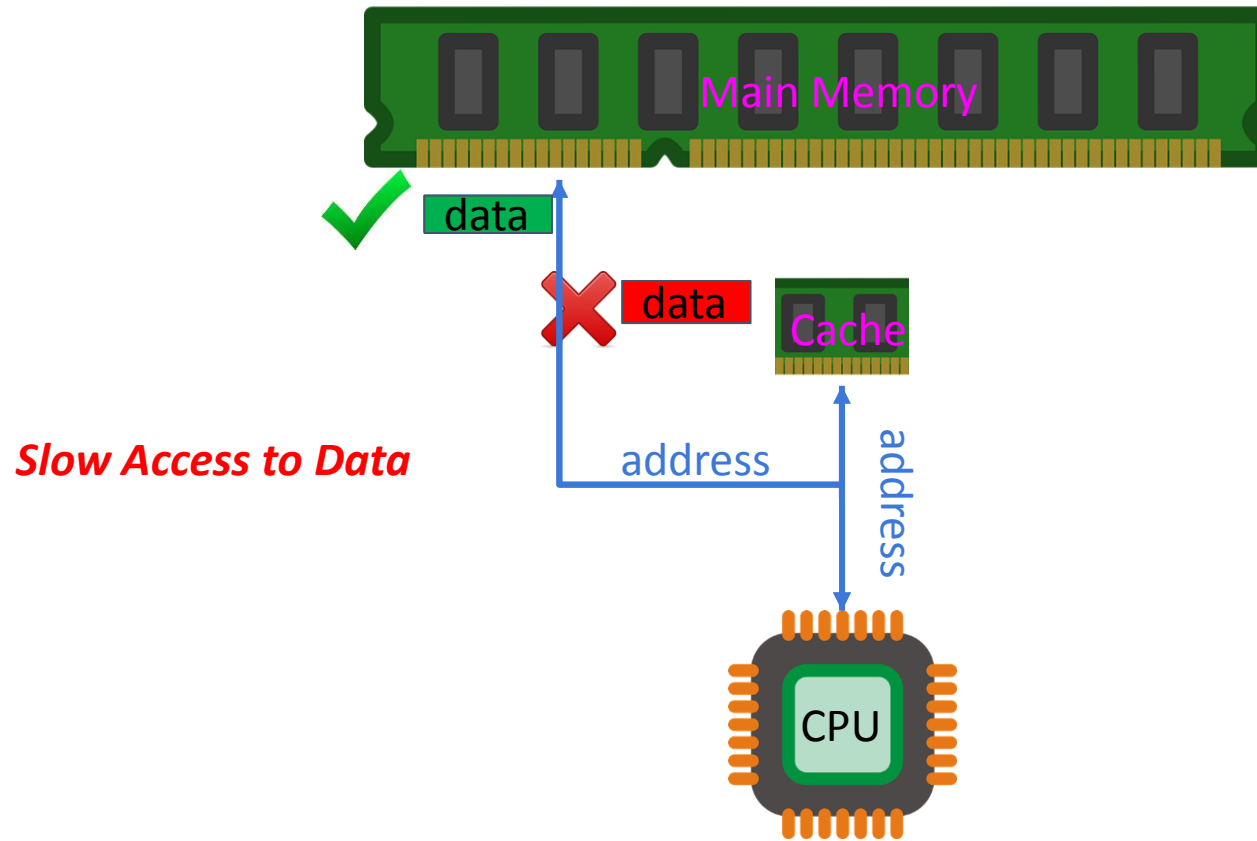
Cache Architecture (Summary)

- Unit of Memory in cache is a line
- A cache consists of multiple sets which stores fixed number of lines
- The number of lines in a set is called associativity
 - L1 is 8-way, L2 is 4-way, L3 is 12-way
- Last Level Cache (LLC) is inclusive
 - LLC contains copies of all of the data in the lower cache level
 - Evicting data from LLC remove that data from all other cache levels

Accessing Memory (Cache Hit)



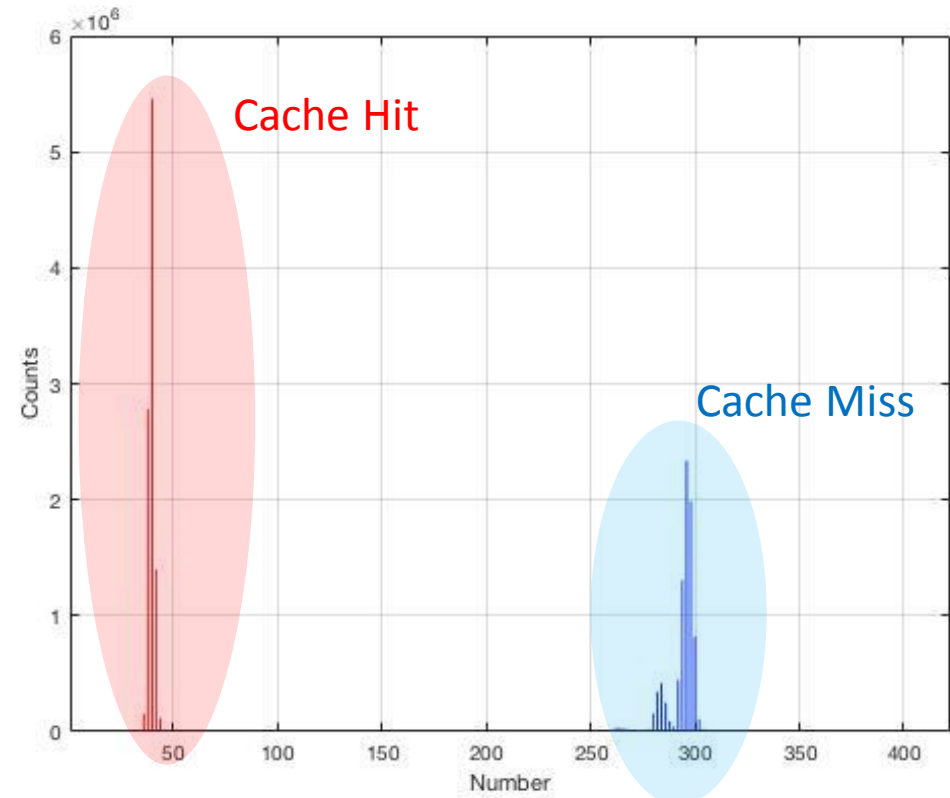
Accessing Memory (Cache Miss)



Cache Hit vs. Miss Time Difference

- ≈ 10 Million measurement

Number of Occurrence

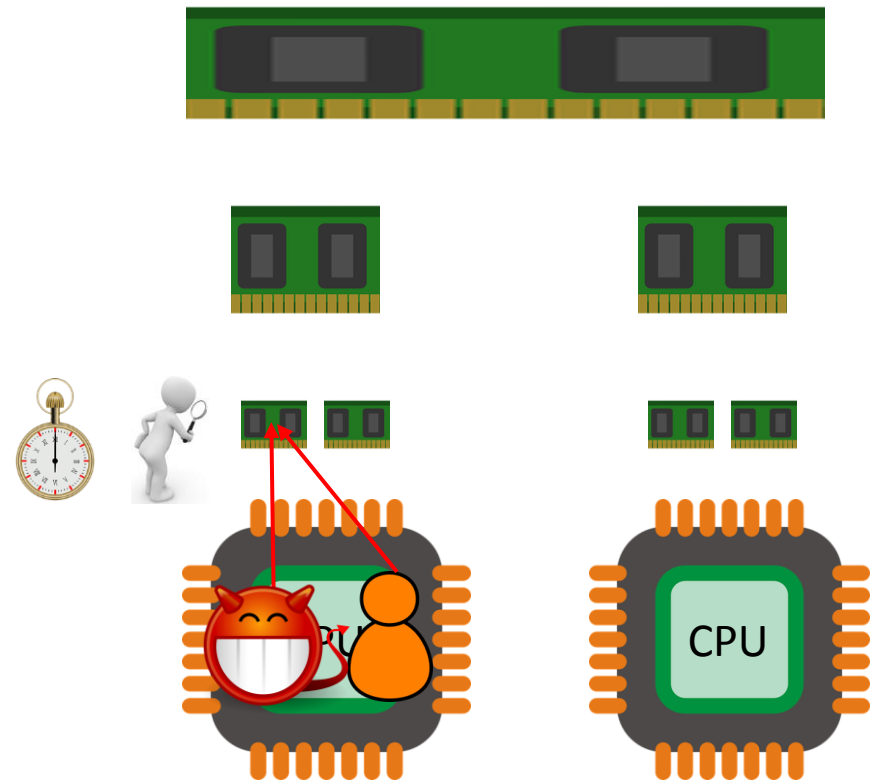
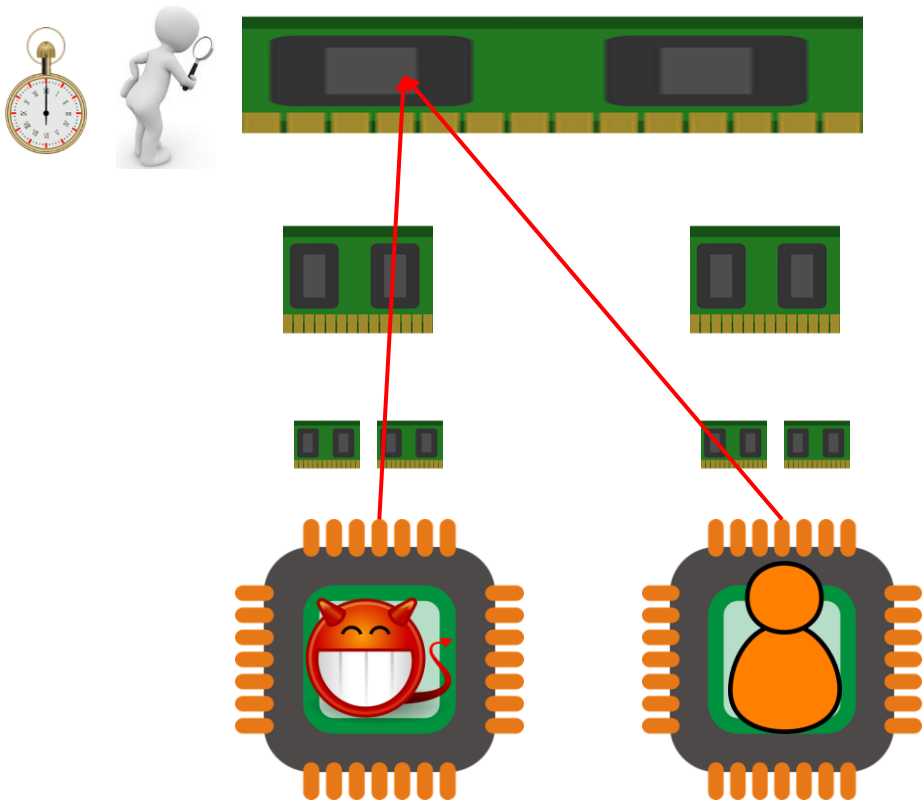
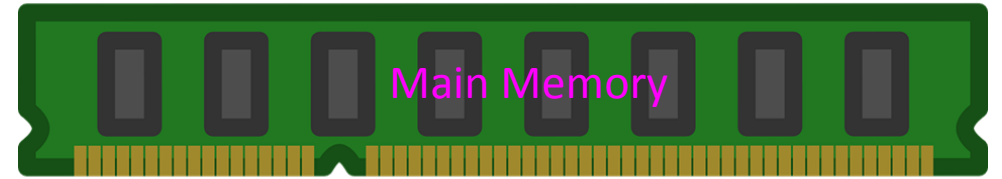
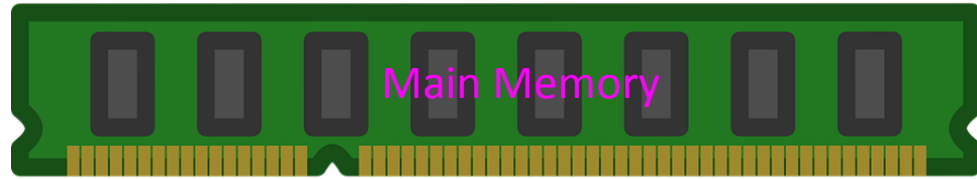


Clock Cycle

Cache Attack Model

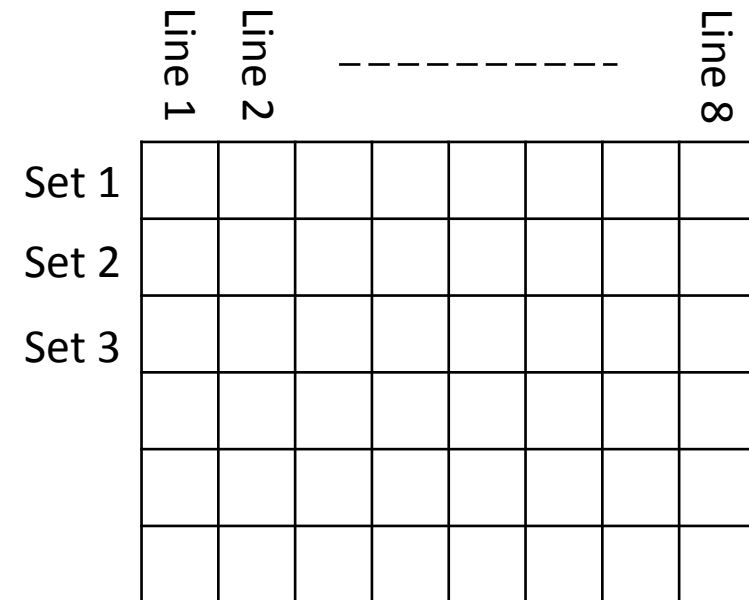
 : Attacker

 : Victim



Some Cache Attack Technique

- Evict and Time
- Flush and Reload
- Prime and Probe

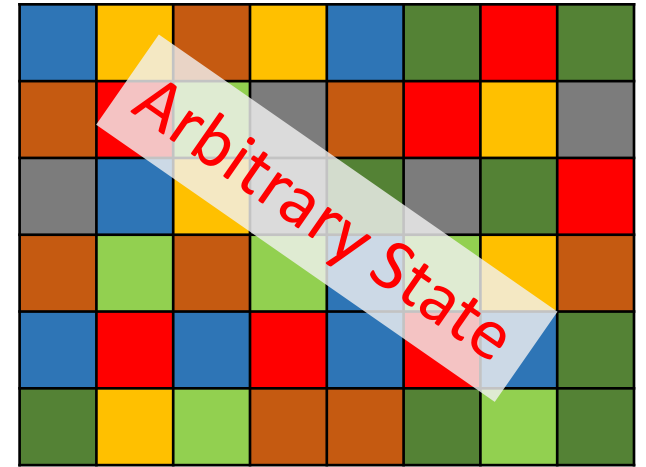


Some Cache Attack Technique

- Evict and Time
- Flush and Reload
- Prime and Probe

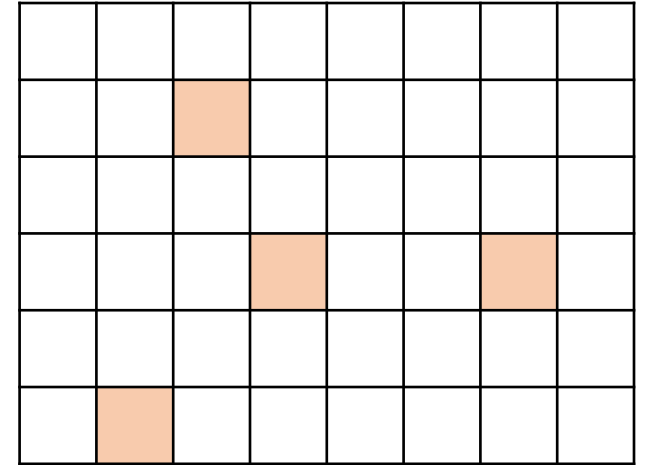
Evict and Time

1. Trigger encryption
2. Selectively manipulate the state of the cache (e.g. evict a full cache set)
3. Trigger encryption
4. Measure how long it took
5. Deduce what cache sets it accessed
6. Repeat step 1-4 to gain information on all the set the encryption accessed



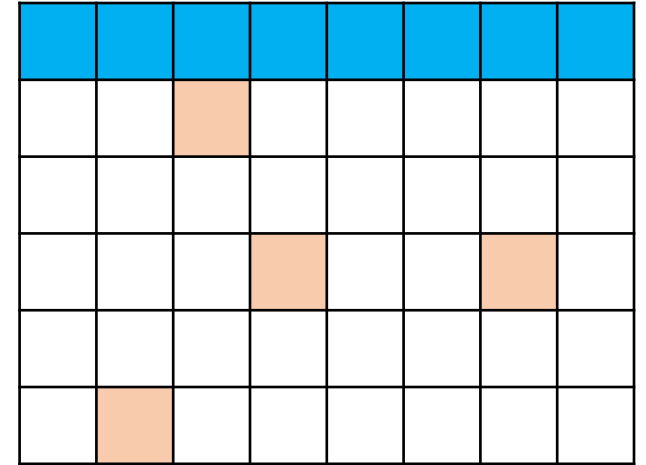
Evict and Time

1. **Trigger encryption**
2. Selectively manipulate the state of the cache (e.g. evict a full cache set)
3. Trigger encryption
4. Measure how long it took
5. Deduce what cache sets it accessed
6. Repeat step 1-4 to gain information on all the set the encryption accessed



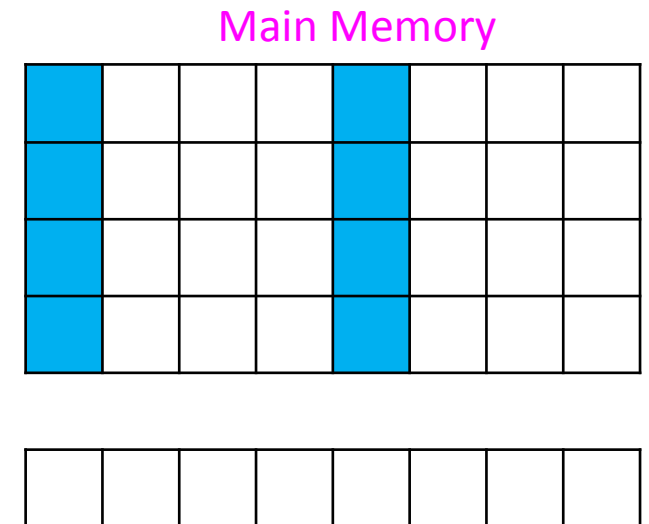
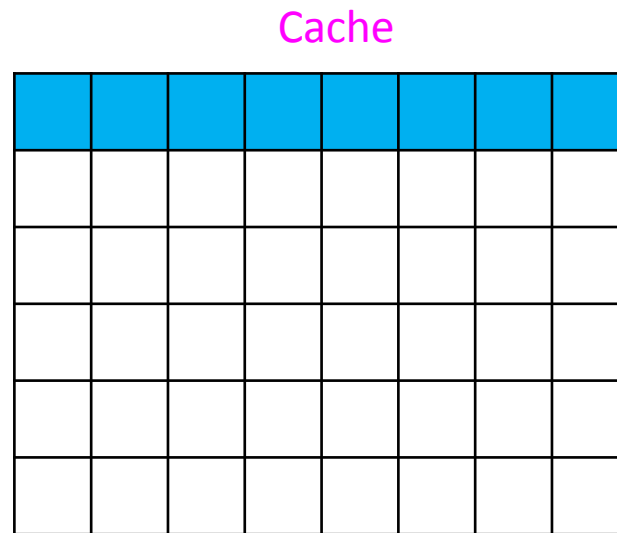
Evict and Time

1. Trigger encryption
2. **Selectively manipulate the state of the cache (e.g. evict a full cache set)**
3. Trigger encryption
4. Measure how long it took
5. Deduce what cache sets it accessed
6. Repeat step 1-4 to gain information on all the set the encryption accessed



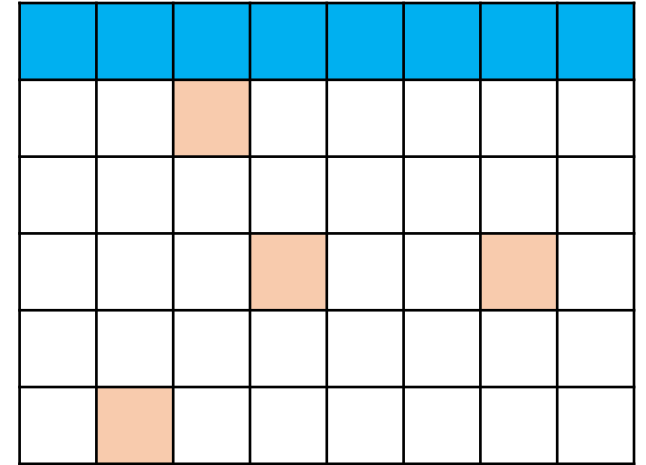
How do we fill a cache set?

- By Accessing some of the memory locations the corresponding locations in the cache is going to be filled.
- Main Challenge: which lines to access?



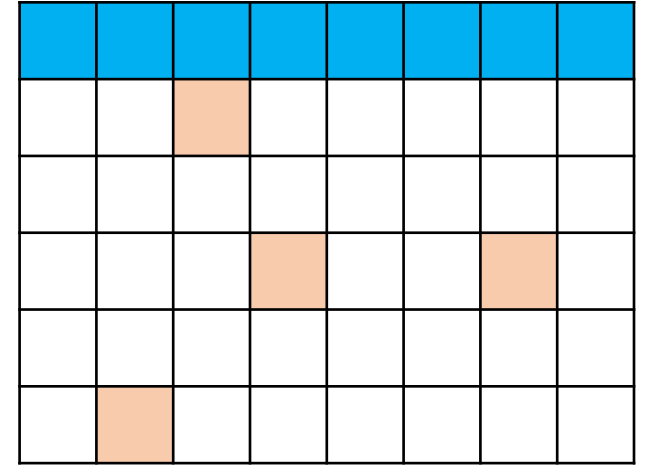
Evict and Time

1. Trigger encryption
2. Selectively manipulate the state of the cache (e.g. evict a full cache set)
3. Trigger encryption
4. Measure how long it took
5. Deduce what cache sets it accessed
6. Repeat step 1-4 to gain information on all the set the encryption accessed



Evict and Time

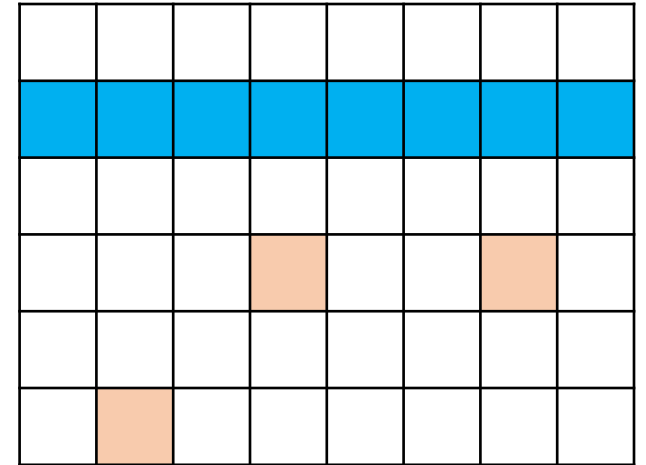
1. Trigger encryption
2. Selectively manipulate the state of the cache (e.g. evict a full cache set)
3. Trigger encryption
4. Measure how long it took
5. Deduce what cache sets it accessed
6. Repeat step 1-4 to gain information on all the set the encryption accessed



It took almost the same amount time

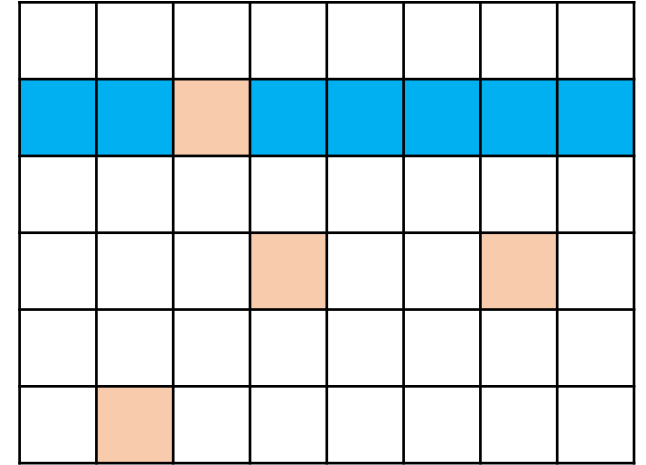
Evict and Time

1. Trigger encryption
2. **Selectively manipulate the state of the cache (e.g. evict a full cache set)**
3. Trigger encryption
4. Measure how long it took
5. Deduce what cache sets it accessed
6. Repeat step 1-4 to gain information on all the set the encryption accessed



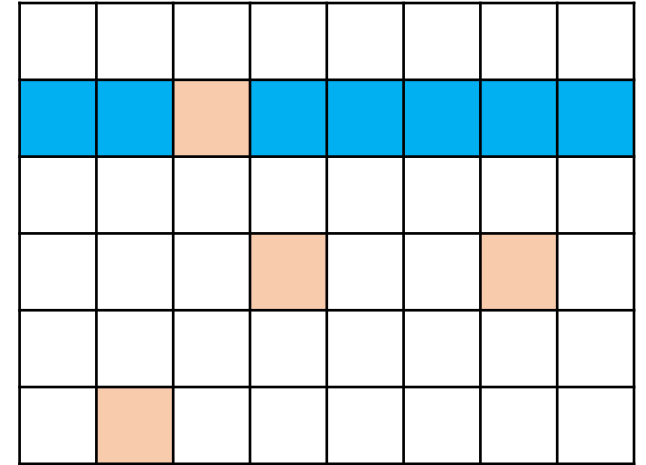
Evict and Time

1. Trigger encryption
2. Selectively manipulate the state of the cache (e.g. evict a full cache set)
3. Trigger encryption
4. Measure how long it took
5. Deduce what cache sets it accessed
6. Repeat step 1-4 to gain information on all the set the encryption accessed



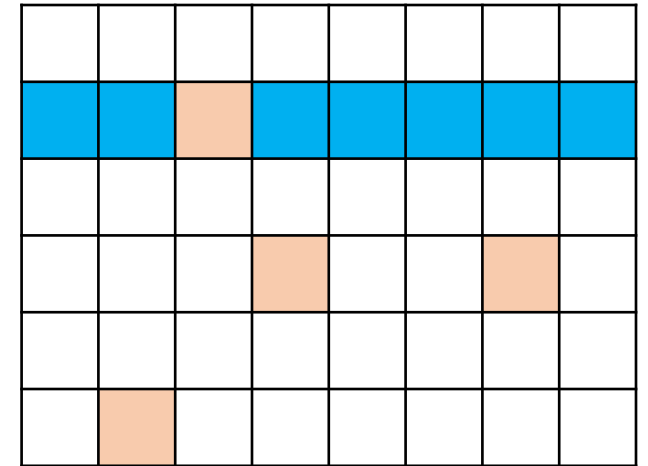
Evict and Time

1. Trigger encryption
2. Selectively manipulate the state of the cache (e.g. evict a full cache set)
3. Trigger encryption
4. Measure how long it took
5. Deduce what cache sets it accessed
6. Repeat step 1-4 to gain information on all the set the encryption accessed



Evict and Time

1. Trigger encryption
2. Selectively manipulate the state of the cache (e.g. evict a full cache set)
3. Trigger encryption
4. Measure how long it took
5. Deduce what cache sets it accessed
6. Repeat step 1-4 to gain information on all the set the encryption accessed



Set 2 was accessed!!

Some Cache Attack Technique

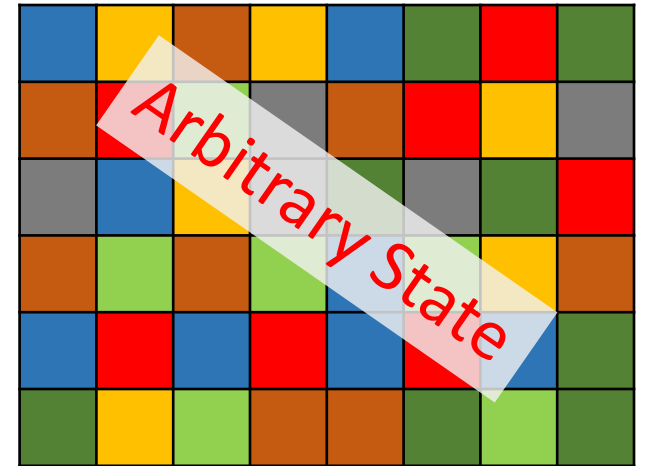
- Evict and Time
- **Flush and Reload**
- Prime and Probe

Flush and Reload

- Exploits cache behavior to leak information on victim access to shared memory.
 - Shared libraries
 - Memory de-duplication
- Spy monitors victim's access to shared code
 - Spy can determine what victim does
 - Spy can infer the data the victim operates on

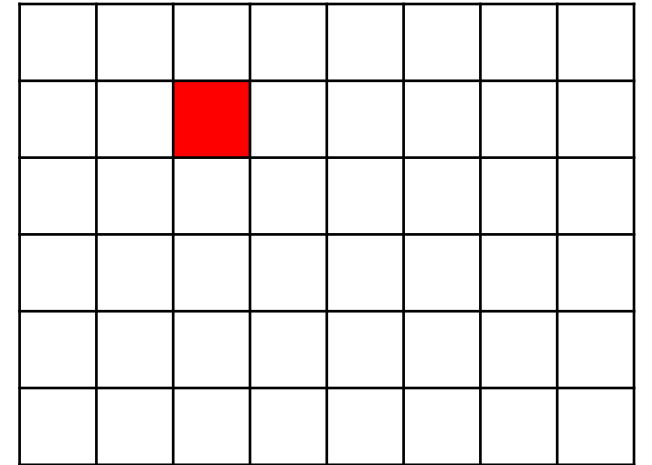
Flush and Reload

1. Flush memory line
2. Wait a bit
3. Measure time to Reload line
4. Repeat

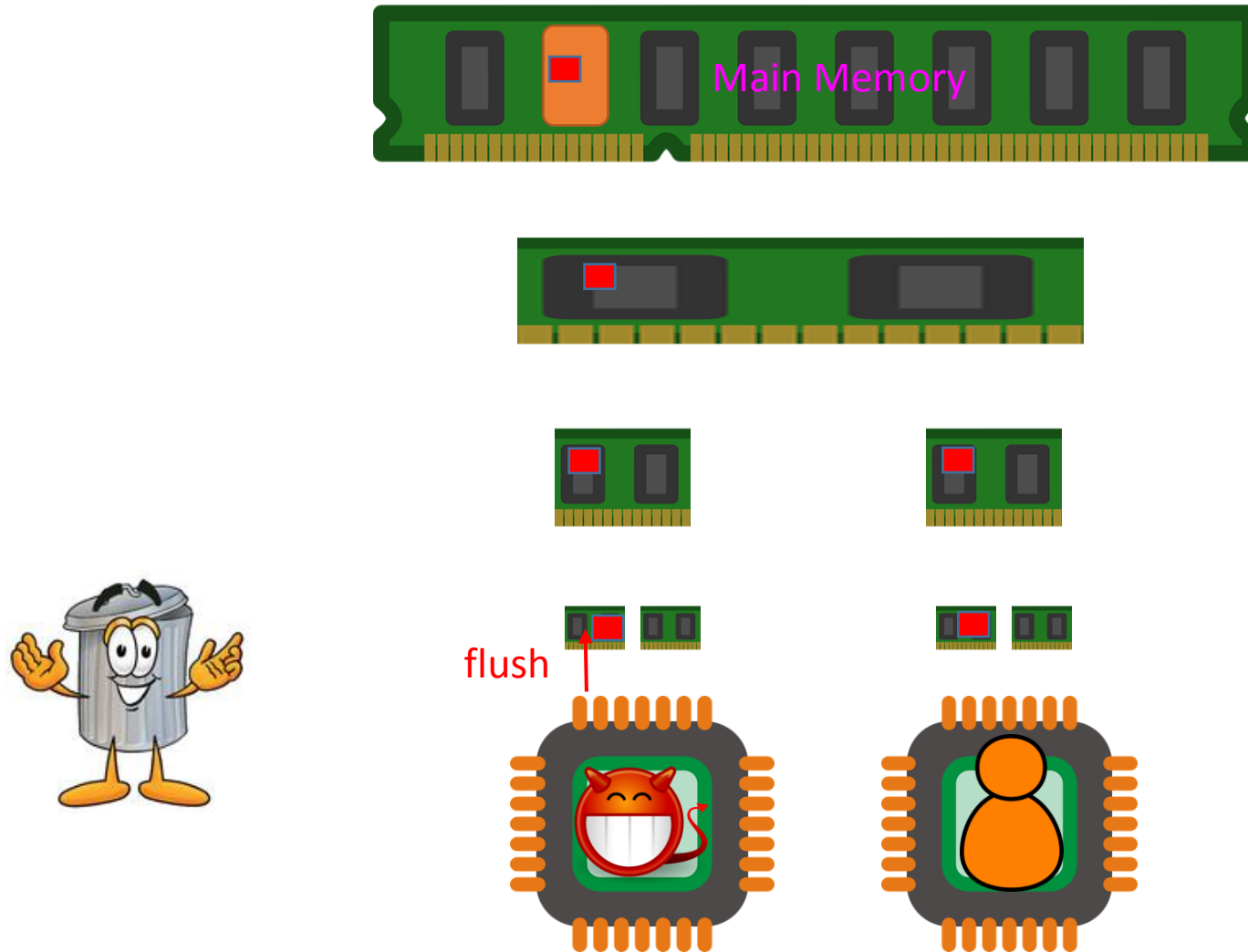


Flush and Reload

1. Flush memory line
2. Wait a bit
3. Measure time to Reload line
4. Repeat

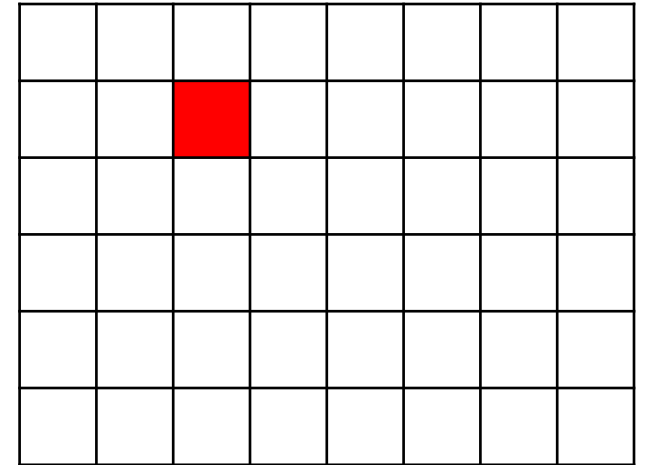


Flush a Line From Cache

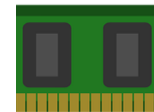
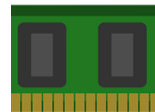
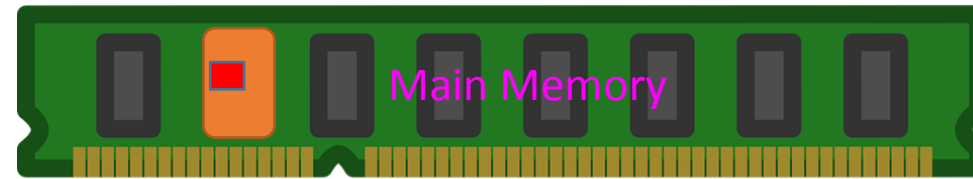


Flush and Reload

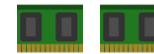
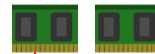
1. Flush memory line
2. **Wait a bit**
3. Measure time to Reload line
4. Repeat



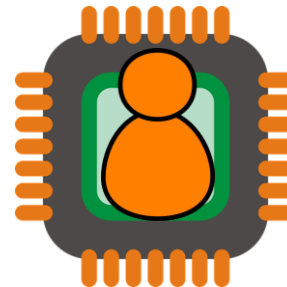
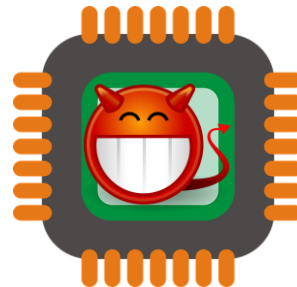
Reload a Line From Cache



No Access by Victim

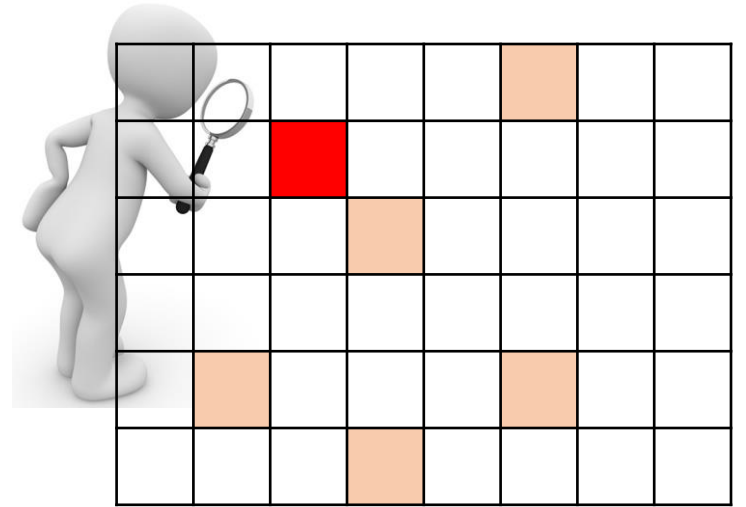


reload ↑



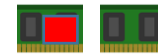
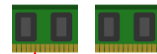
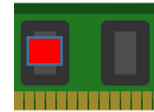
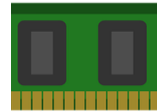
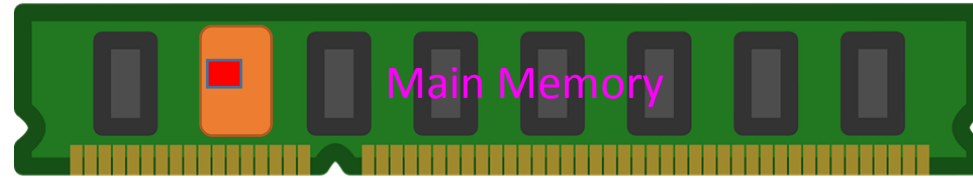
Flush and Reload

1. Flush memory line
2. Wait a bit
3. Measure time to Reload line
4. Repeat

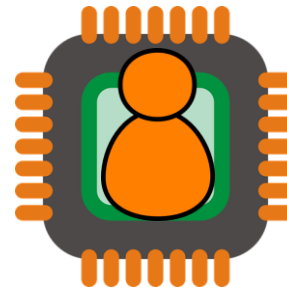
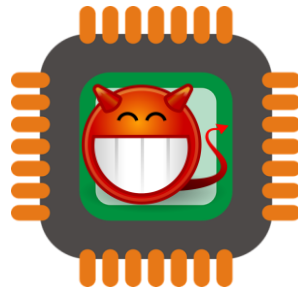


Slow means no access by victim

Reload a Line From Cache

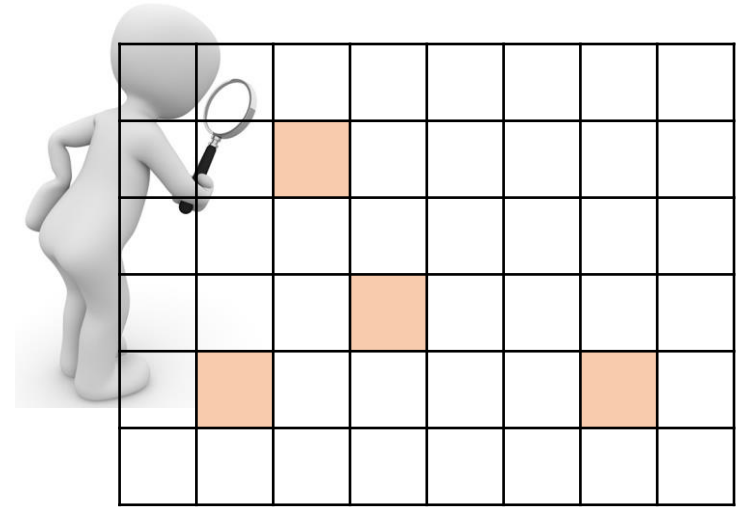


reload ↑



Flush and Reload

1. Flush memory line
2. Wait a bit
3. Measure time to Reload line
4. Repeat



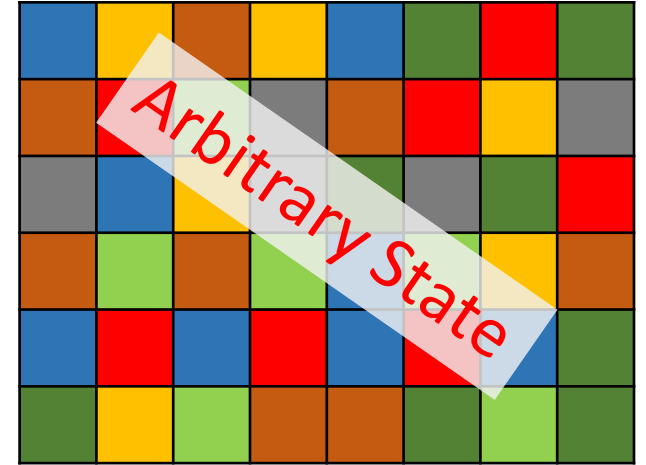
Fast means that victim accessed

Some Cache Attack Technique

- Evict and Time
- Flush and Reload
- Prime and Probe

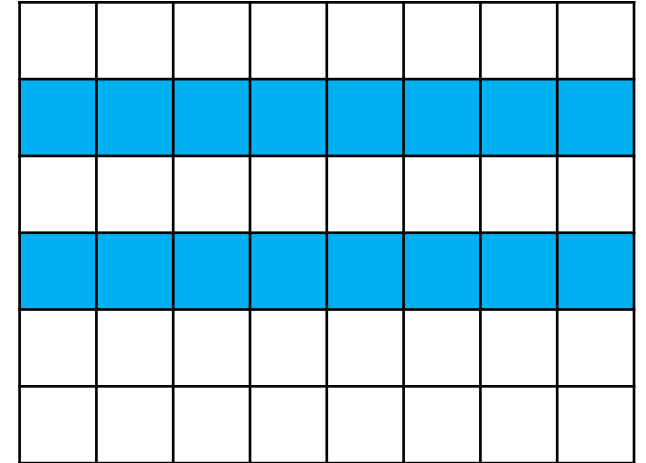
Prime and Probe

1. Attacker fills a set with its own data by accessing some locations in memory
2. Victim Executes and evicts some of the cache lines
3. Attacker accesses those cache line and measure time

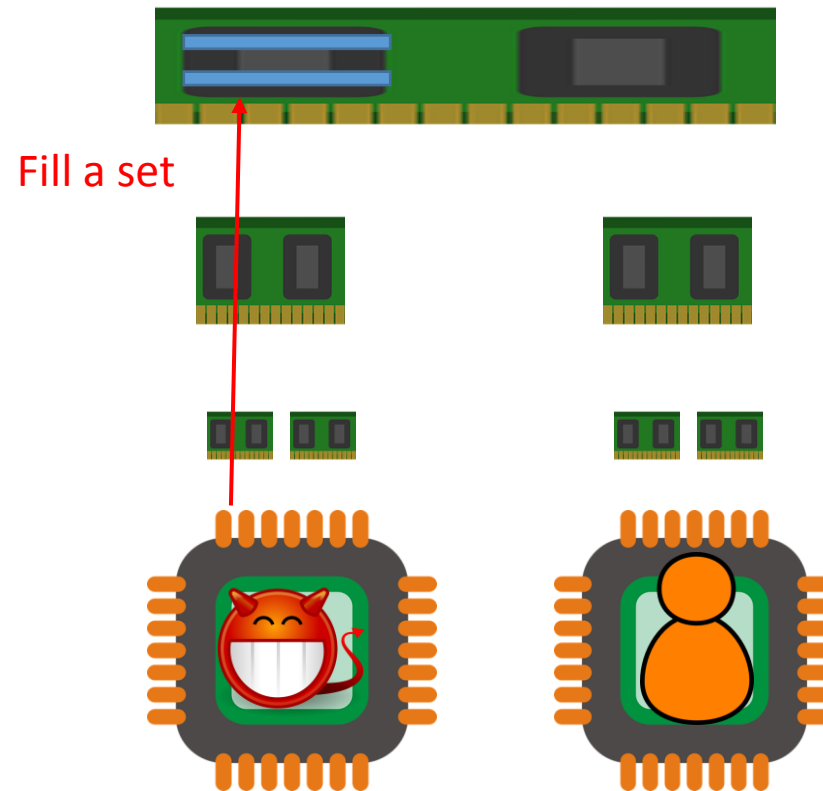
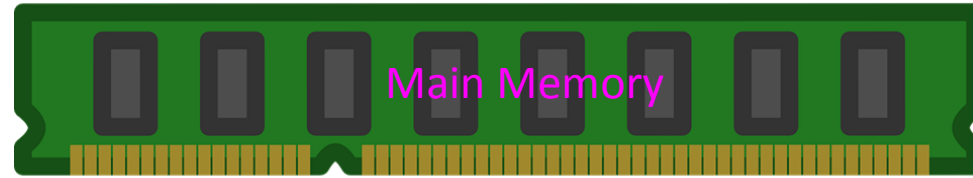


Prime and Probe

1. Attacker fills a set with its own data by accessing some locations in memory
2. Victim Executes and evicts some of the cache lines
3. Attacker accesses those cache line and measure time

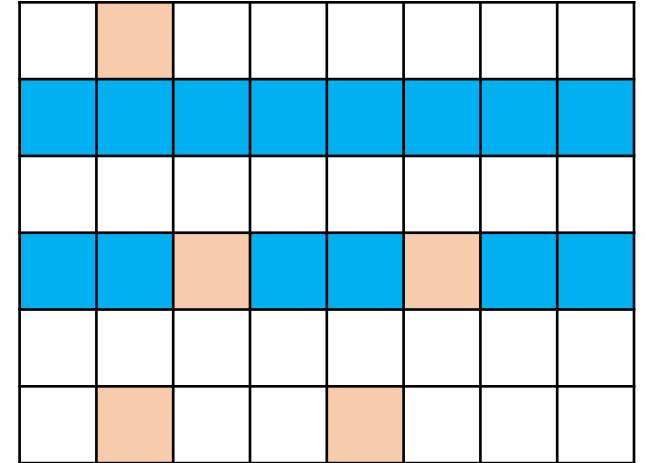


Fill a cache set (In this example 2 sets)

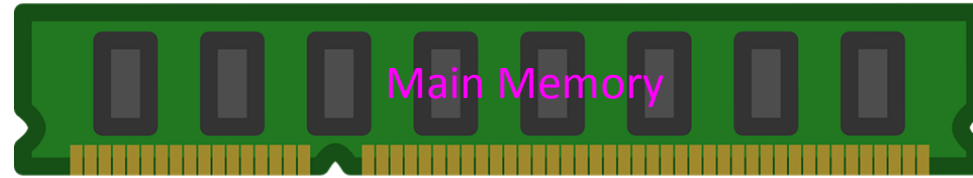


Prime and Probe

1. Attacker fills a set with its own data by accessing some locations in memory
2. Victim Executes and evicts some of the cache lines
3. Attacker accesses those cache line and measure time

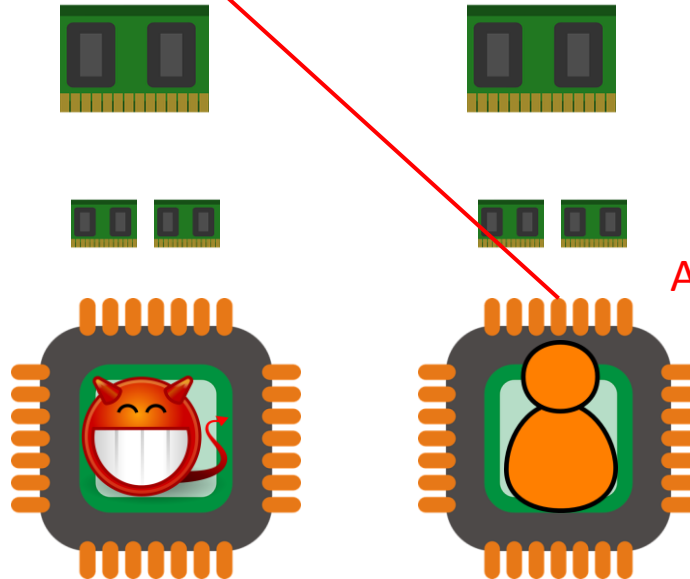


Victim Execution



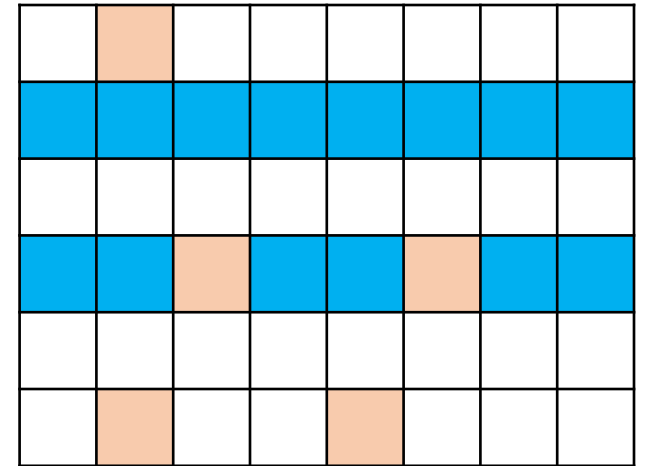
— : Attacker's data

— : Victim's data

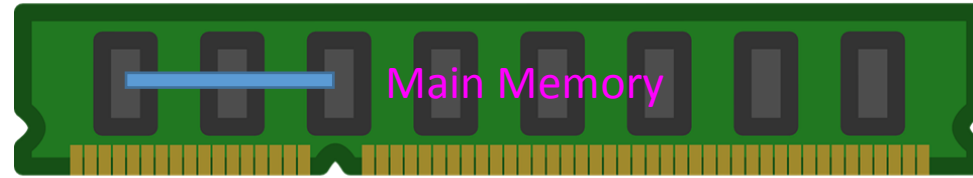


Prime and Probe

1. Attacker fills a set with its own data by accessing some locations in memory
2. Victim Executes and evicts some of the cache lines
3. Attacker accesses those cache line and measure time

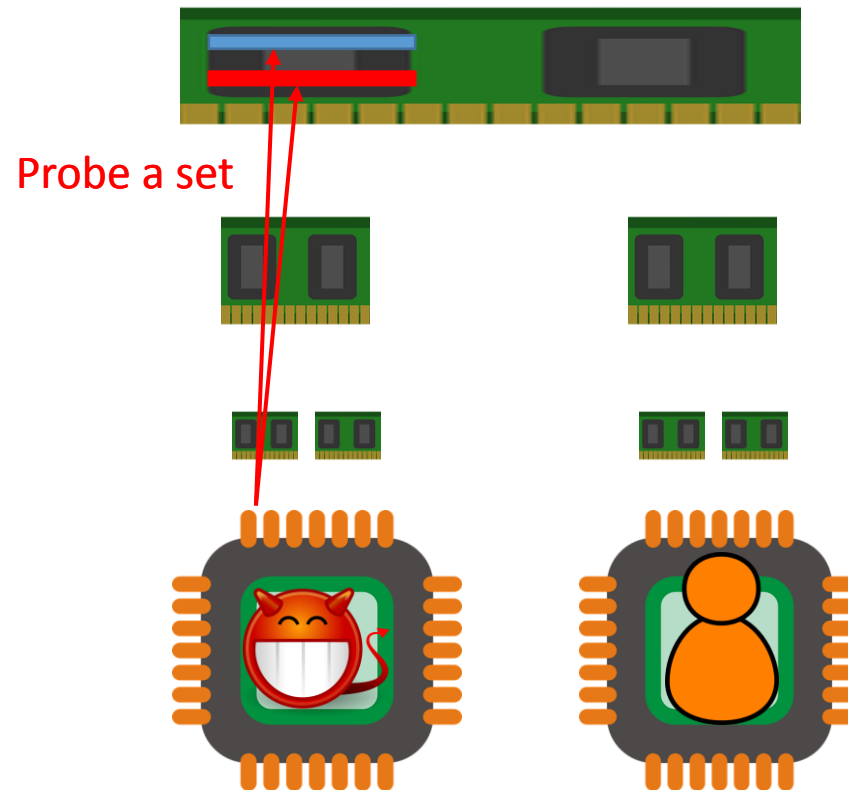


Probe



— : Attacker's data

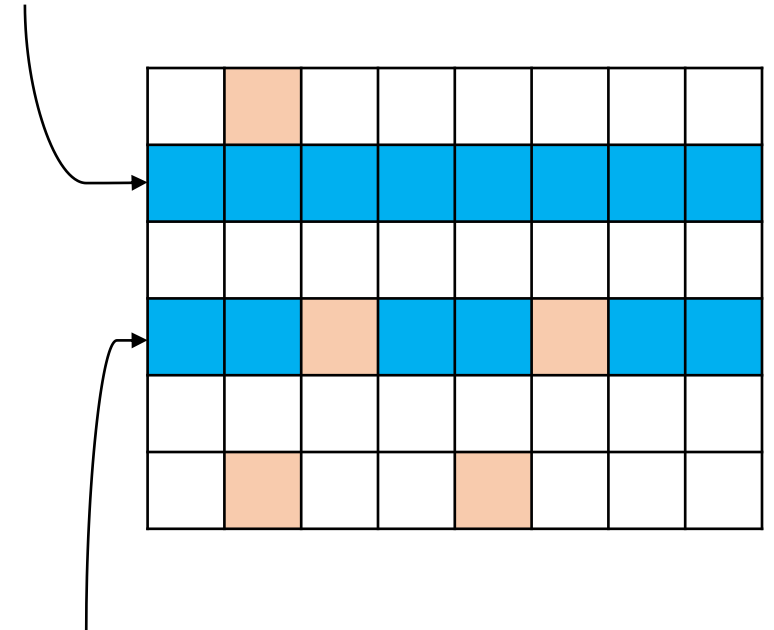
— : Victim's data



Prime and Probe

1. Attacker fills a set with its own data by accessing some locations in memory
2. Victim Executes and evicts some of the cache lines
3. Attacker accesses those cache line and measure time

Fast Access: Not accessed by the victim



Slow Access: Accessed by the victim

How To Recover Secret Key?

SMSS SM SMSS
1 0 0 1 1 0 0

- How do we compute $b^e \bmod n$?
 - Assume e is secret information we want to recover.
- Bit = 0 : Square
- Bit = 1 : Square + Multiply
- ✓ The Sequence of operation will reveal the secret information.

```
x ← 1
for i ← |e|-1 downto 0 do
  x ← x2 mod n
  if (ei = 1) then
    x = xb mod n
  endif
done
return x
```

A Sample Measurement (Flush and Reload)

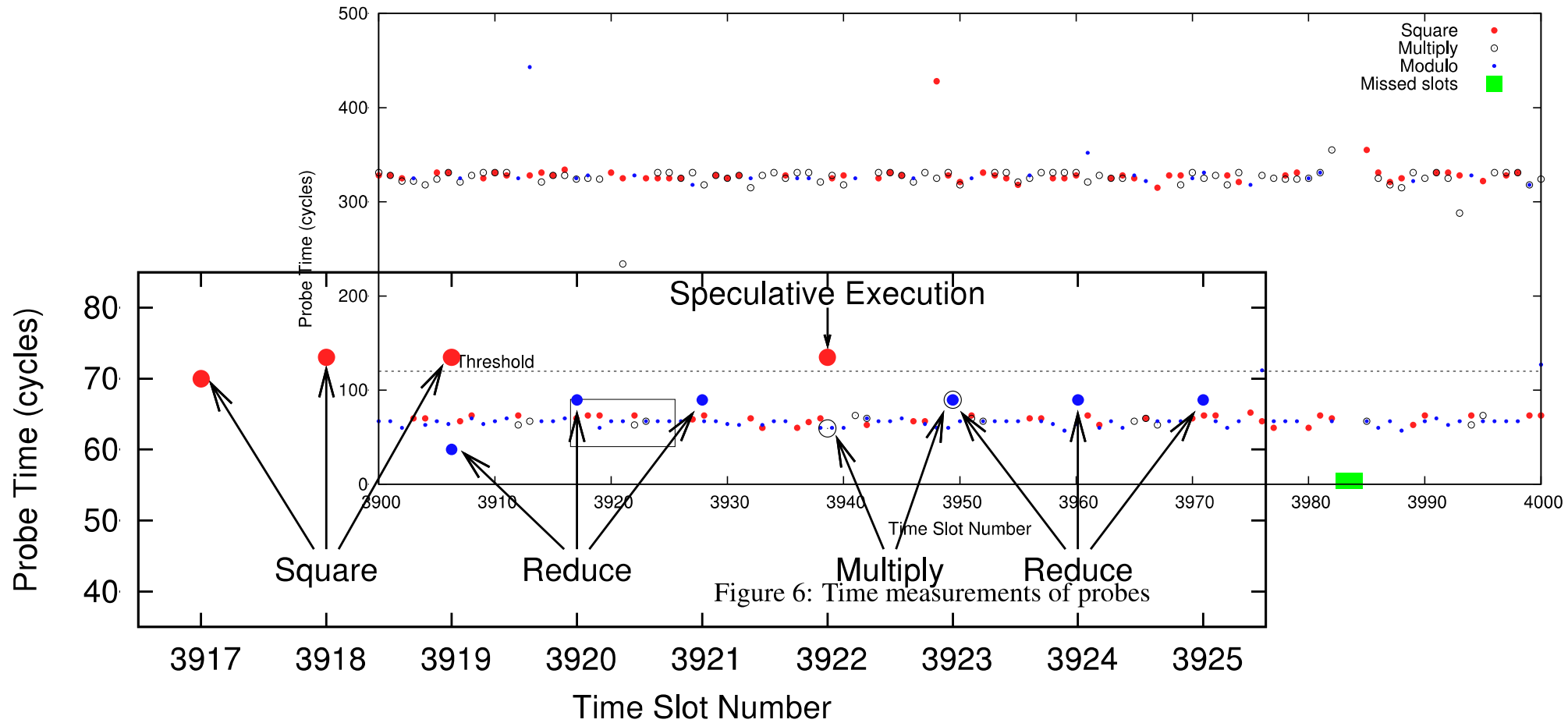
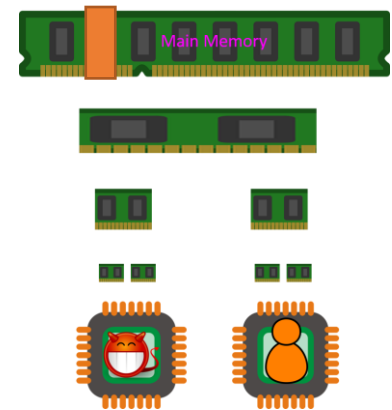
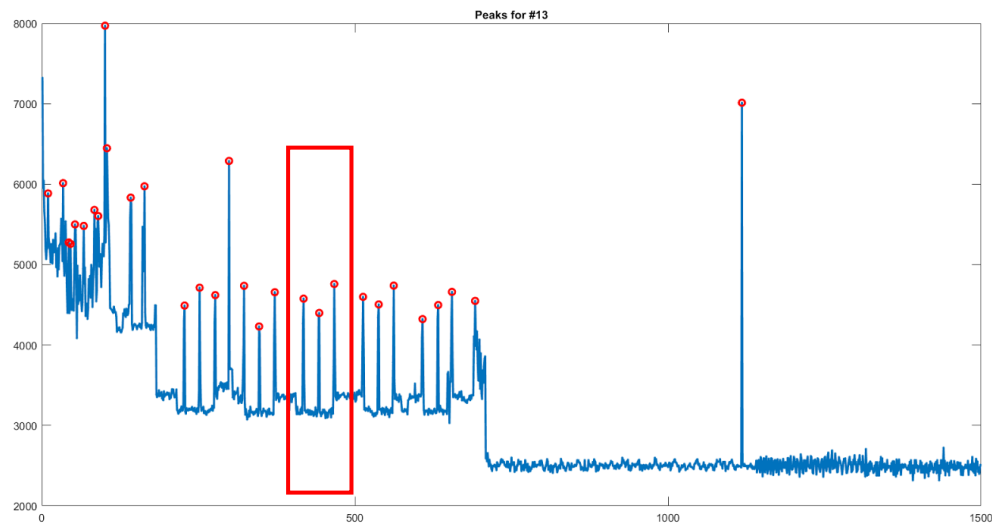


Figure 6: Time measurements of probes

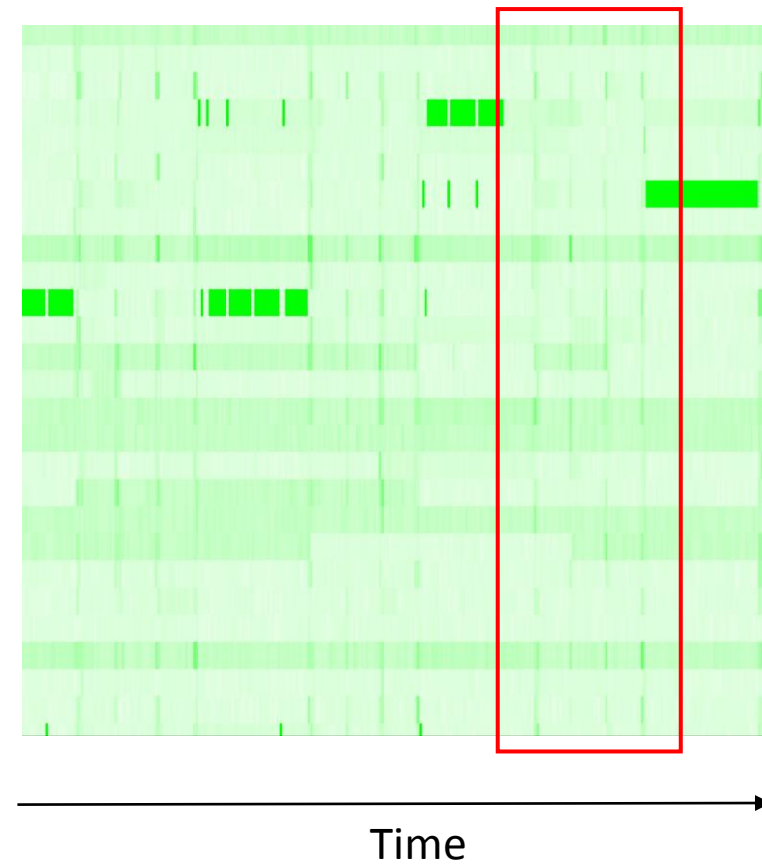


Cache Attack on Database

- Monitoring L1 Instruction cache
- Each Line represents an access to a function

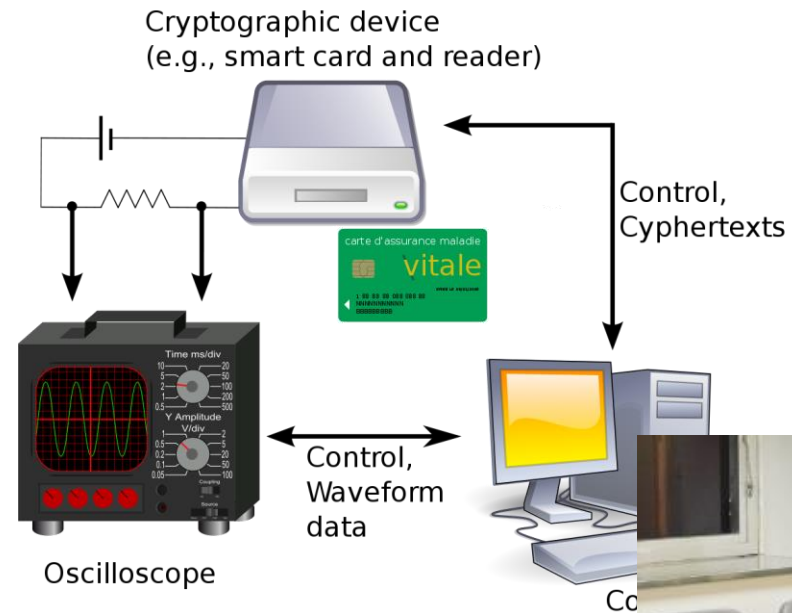


Set



Side Channel Attacks Examples

- Timing Attacks
 - Cache Attack
- Power Analysis Attack
- Electromagnetic Emissions
- Acoustic Emission
- Fault Attacks



Thank You