



LLC Attacks

Applicability & Countermeasures

Gorka Irazoqui

PhD student in WPI

girazoki@wpi.edu

Xiaofei (Rex) Guo

Security researcher at Intel

<http://rexfly.net>





“You must be kidding, cache attacks are not practical!”

ANDROID DEVICES VULNERABLE TO ARMAGEDDON CACHE ATTACK

SECURITY NEWS | AUGUST 15, 2016 | 0 | BY JOSEPH STEINBERG

The paper ARMageddon: Cache Attacks on Mobile Devices have been included in 25th USENIX Security Symposium. The

“You must be kidding, cache attacks are not practical!”

ANDROID DEVICES VULNERABLE TO ARMAGEDDON CACHE ATTACK

SECURITY NEWS | AUGUST 15, 2016 | 0 | BY JOSEPH STEINBERG

The paper ARMageddon: Cache Attacks on Mobile Devices have been included in 25th USENIX Security Symposium. The

“You must be kidding, cache attacks are not practical!”

RISK ASSESSMENT —

Storing secret crypto keys in the Amazon cloud? New attack can steal them

Technique allows full recovery of 2048-bit RSA key stored in Amazon's EC2 service.

DAN GOODIN - 9/28/2015, 2:55 PM

ANDROID DEVICES VULNERABLE TO ARMAGEDDON CACHE ATTACK

SECURITY NEWS | AUGUST 15, 2016 | 0 | BY JOSEPH STEINBERG

The paper ARMageddon: Cache Attacks on Mobile Devices have been included in 25th USENIX Security Symposium. The

“You must be kidding, cache attacks are not practical!”

RISK ASSESSMENT —

Storing secret crypto keys in the Amazon cloud? New attack can steal them

Technique allows full recovery of 2048-bit RSA key stored in Amazon's EC2 service.

DAN GOODIN - 9/28/2015, 2:55 PM

CacheBleed OpenSSL Vulnerability Affects Intel-Based Cloud Servers

Only Sandy Bridge (and earlier) Intel CPUs are affected

Mar 2, 2016 08:26 GMT · By Catalin Cimpanu · Share:   

Yesterday's OpenSSL updates (1.0.2g and 1.0.1s) not only brought a fix against the already infamous DROWN attack but also patched seven other security flaws, one labeled as high, one moderate, and five as low severity.

ANDROID DEVICES VULNERABLE TO ARMAGEDDON CACHE ATTACK

SECURITY NEWS | AUGUST 15, 2016 | 0 | BY JOSEPH STEINBERG

The paper ARMageddon: Cache Attacks on Mobile Devices have been included in 25th USENIX Security Symposium. The

“You must be kidding, cache attacks are not practical!”

RISK ASSESSMENT —

Storing secret crypto keys in the Amazon cloud? New attack can steal them

Technique allows full recovery of 2048-bit RSA key stored in Amazon's EC2 service.

DAN GOODIN - 9/28/2015, 2:55 PM

Security considerations and disallowing inter-Virtual Machine Transparent Page Sharing (2080735)

Purpose

This article acknowledges the recent academic research that leverages Transparent Page Sharing (TPS) to gain unauthorized access to data under certain highly controlled conditions and documents VMware's precautionary measure of restricting TPS to individual virtual machines by default in upcoming ESXi releases. At this time, VMware believes that the published information disclosure due to TPS between virtual machines is impractical in a real world deployment.

CacheBleed OpenSSL Vulnerability Affects Intel-Based Cloud Servers

Only Sandy Bridge (and earlier) Intel CPUs are affected

Mar 2, 2016 08:26 GMT · By Catalin Cimpanu · Share:   


Yesterday's OpenSSL updates (1.0.2g and 1.0.1s) not only brought a fix against the already infamous DROWN attack but also patched seven other security flaws, one labeled as high, one moderate, and five as low severity.


Last Level Cache Attacks: Feasibility Trend?


Last Level Cache Attacks: Feasibility Trend?

Cache Attacks and Countermeasures: The Case of AES

Article in Lecture Notes in Computer Science 2005 · January 2005 with 26 Reads
DOI: 10.1007/11605805_1 · Source: DBLP

1st  [Dag Arne Osvik](#)
i.d. 4.67 · Customs Solutions Group S.A.

2nd  [Adi Shamir](#)

3rd  [Eran Tromer](#)
i.d. 14.96 · Tel Aviv University


Intel, Spark, AMD | Linux | OpenSSL AES

Last Level Cache Attacks: Feasibility Trend?

Cache Attacks and Countermeasures: The Case of AES

Yet another MicroArchitectural Attack: : exploiting I-Cache.

Conference Paper (PDF Available) · January 2007 with 32 Reads
DOI: 10.1145/1314466.1314469 · Source: DBLP
Conference: Proceedings of the 2007 ACM workshop on Computer Security Architecture, CSAW 2007, Fairfax, VA, USA, November 2, 2007



1st

Onur Acıgmez

7.04 · Unknown



Last Level Cache Attacks: Feasibility Trend?

Cache Attacks and Countermeasures: The Case of AES

Yet another MicroArchitectural Attack: : exploiting I-Cache.

FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack


Intel, Spark, AMD | Linux | OpenSSL AES

Intel | Linux | OpenSSL RSA


Intel (Cross-core) | Linux (deduplication) | GnuPG RSA

Conference Paper · August 2014 with 13 Reads

Conference: USENIX Security Symposium








1st **Yuval Yarom**
1.71 · University of Adelaide






2nd **Katrina Falkner**
16.03 · University of Adelaide

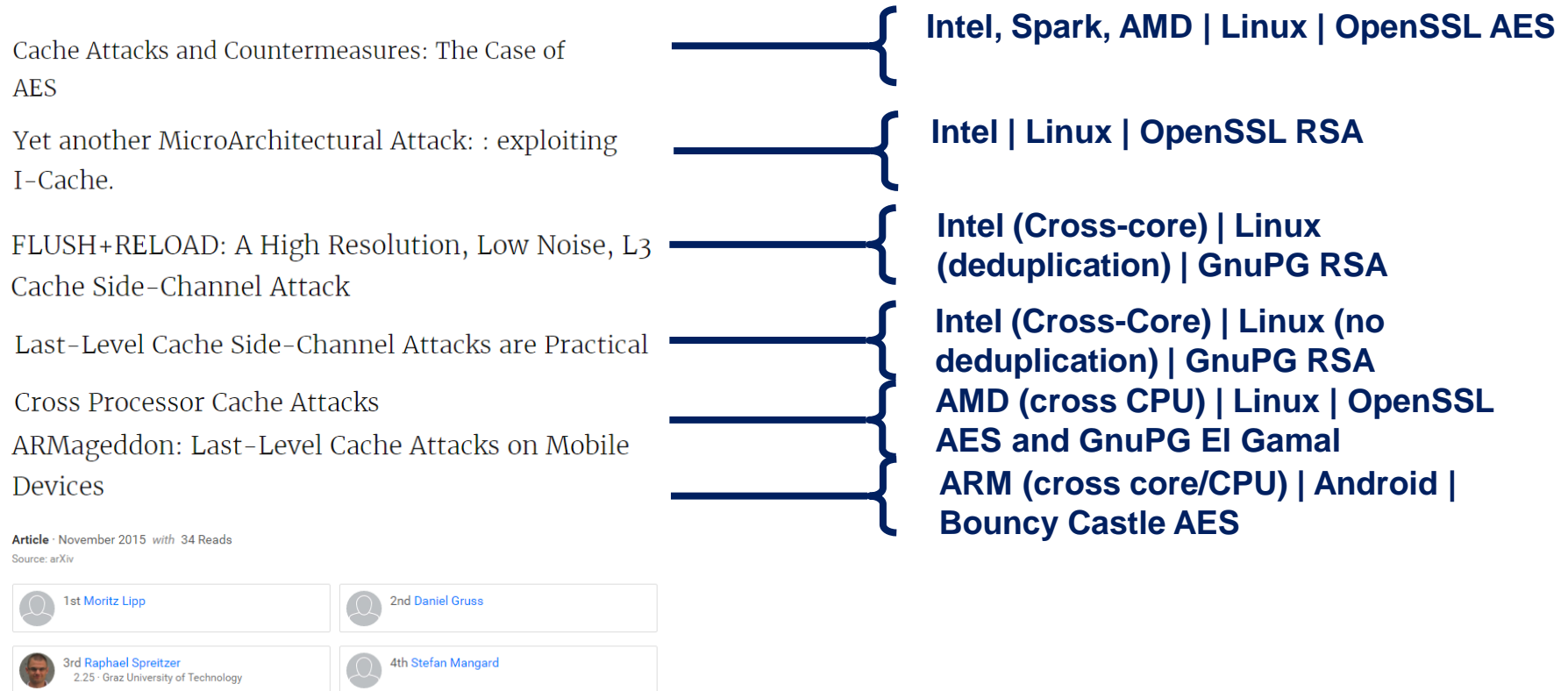
Last Level Cache Attacks: Feasibility Trend?

Cache Attacks and Countermeasures: The Case of AES		Intel, Spark, AMD Linux OpenSSL AES
Yet another MicroArchitectural Attack: : exploiting I-Cache.		Intel Linux OpenSSL RSA
FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack		Intel (Cross-core) Linux (deduplication) GnuPG RSA
Last-Level Cache Side-Channel Attacks are Practical		Intel (Cross-Core) Linux (no deduplication) GnuPG RSA
<p>Article · July 2015 with 30 Reads DOI: 10.1109/SP.2015.43</p> <div><div><div><div>1st</div><div>Fangfei Liu</div><div>il 1.47 · Princeton University</div></div></div><div><div><div>2nd</div><div>Yuval Yarom</div><div>il 1.71 · University of Adelaide</div></div></div><div><div><div>3rd</div><div>Qian ge</div><div>il 0.01 · UNSW Australia</div></div><div><div><div>Last</div><div>Ruby B. Lee</div><div>il 26.61 · Princeton University</div></div></div><div> +1</div></div></div>		

Last Level Cache Attacks: Feasibility Trend?

Cache Attacks and Countermeasures: The Case of AES		Intel, Spark, AMD Linux OpenSSL AES
Yet another MicroArchitectural Attack: : exploiting I-Cache.		Intel Linux OpenSSL RSA
FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack		Intel (Cross-core) Linux (deduplication) GnuPG RSA
Last-Level Cache Side-Channel Attacks are Practical		Intel (Cross-Core) Linux (no deduplication) GnuPG RSA
Cross Processor Cache Attacks		AMD (cross CPU) Linux OpenSSL AES and GnuPG El Gamal
<p>Conference Paper · January 2016 with 3 Reads DOI: 10.1145/2897845.2897867 Conference: the 11th ACM</p>		
<div><div><div>1st Gorka Irazoqui</div></div><div><div>2nd Thomas Eisenbarth</div></div></div>		
<div><div>3rd Berk Sunar 19.39 · Worcester Polytechnic Institute</div></div>		

Last Level Cache Attacks: Feasibility Trend?



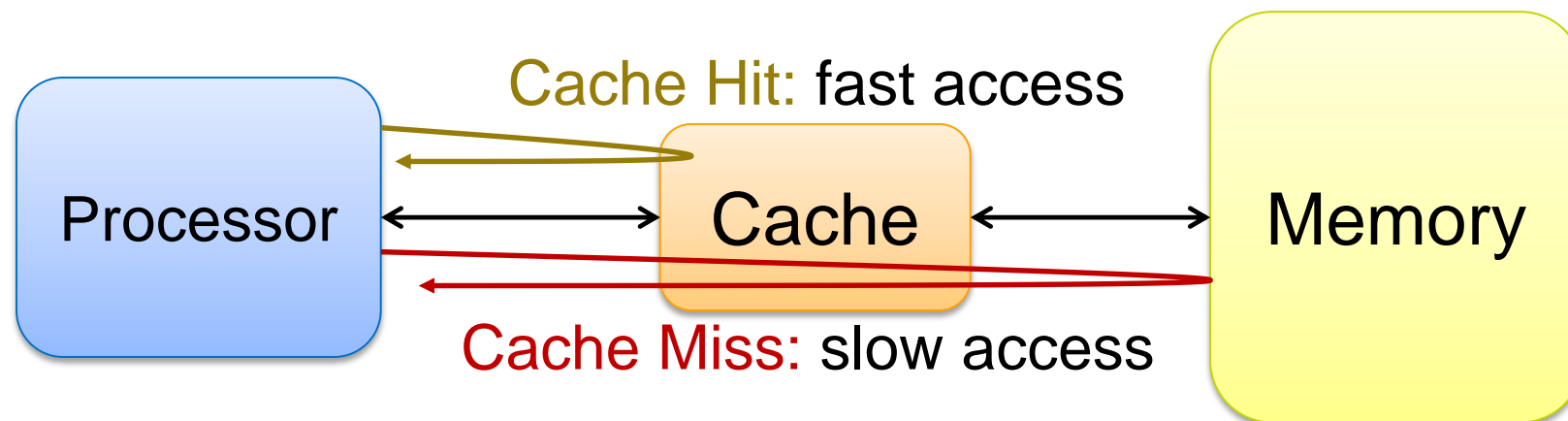
How do LLC attacks work?

Why Attack LLC?

Caches: fast access memories

Why Attack LLC?

Caches: fast access memories



Why Attack LLC?

Caches: fast access memories

Why Attack LLC?

Caches: fast access memories

Contention to guess recently used data

Why Attack LLC?

Caches: fast access memories

Contention to guess recently used data

Why LLC?

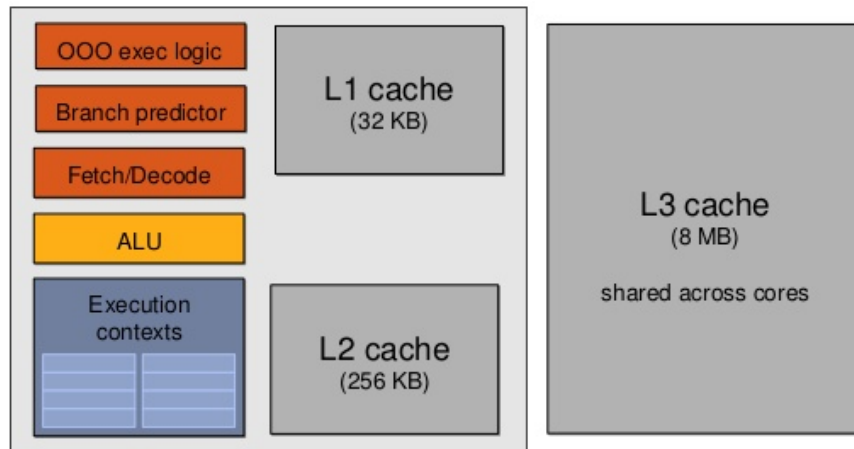
Why Attack LLC?

Caches: fast access memories

Contention to guess recently used data

Why LLC?

Intel i7-4770



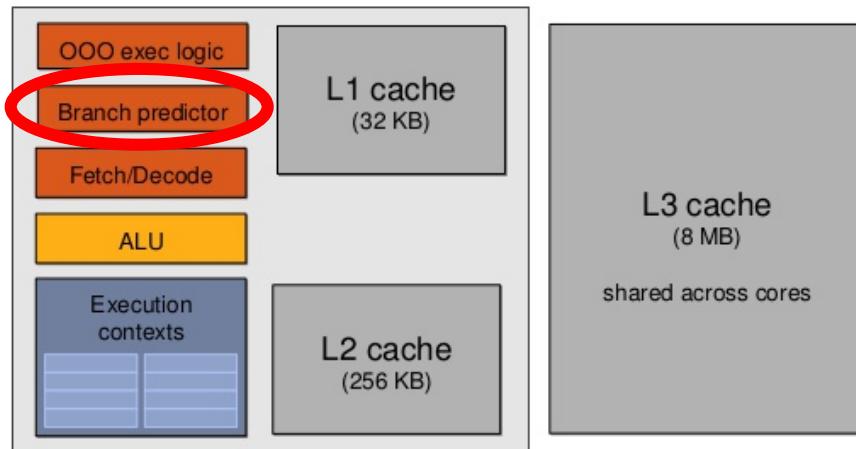
Why Attack LLC?

Caches: fast access memories

Contention to guess recently used data

Why LLC?

Intel i7-4770



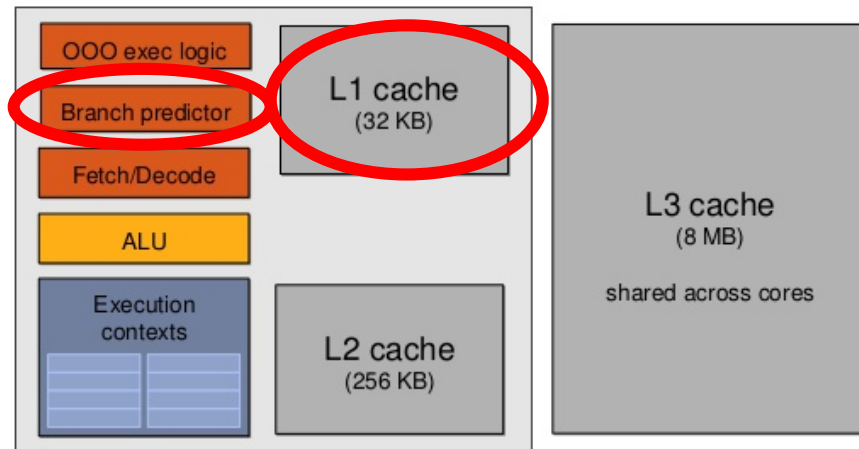
Why Attack LLC?

Caches: fast access memories

Contention to guess recently used data

Why LLC?

Intel i7-4770



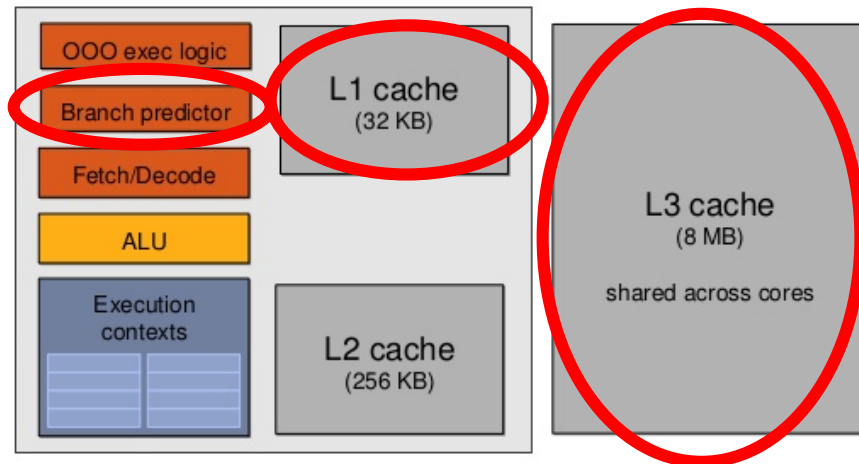
Why Attack LLC?

Caches: fast access memories

Contention to guess recently used data

Why LLC?

Intel i7-4770



Why Attack LLC?

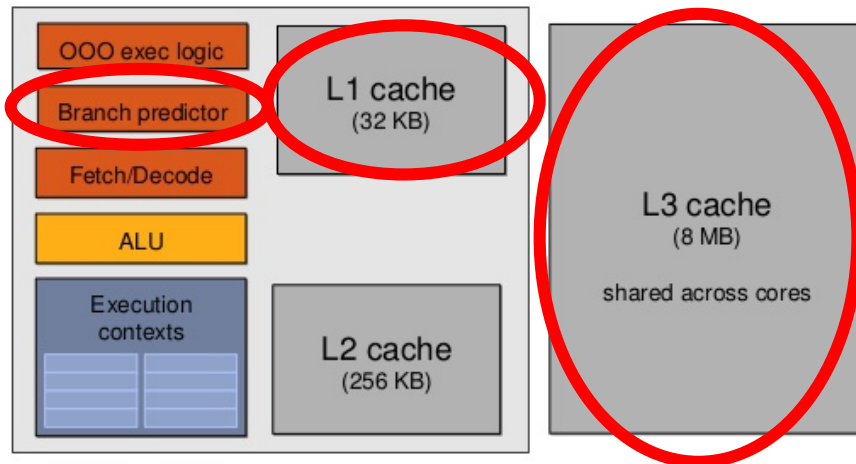
Caches: fast access memories

Contention to guess recently used data

Why LLC?

- Cross core, inclusiveness, high resolution..

Intel i7-4770



Why Attack LLC?

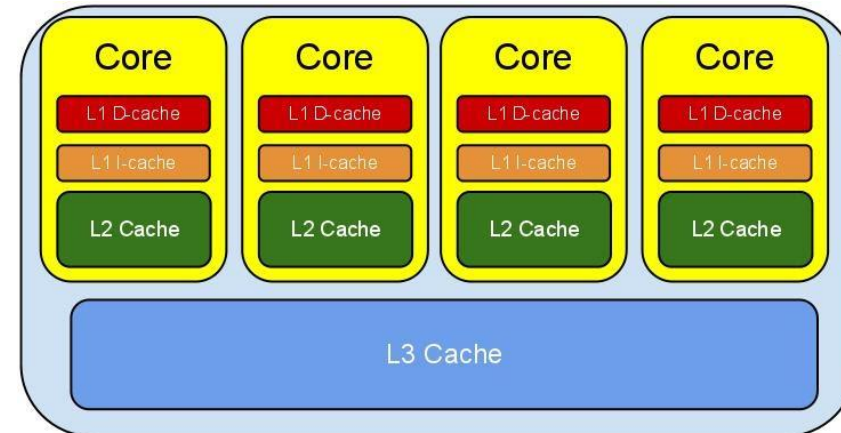
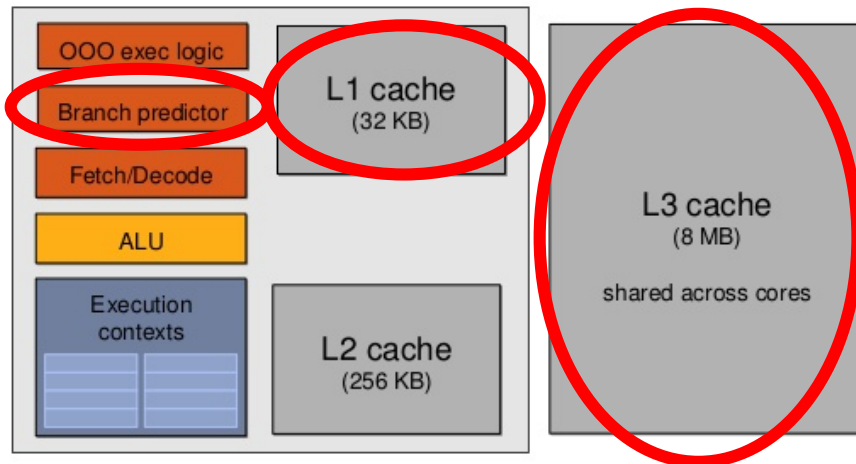
Caches: fast access memories

Contention to guess recently used data

Why LLC?

- Cross core, inclusiveness, high resolution..

Intel i7-4770



Why Attack LLC?

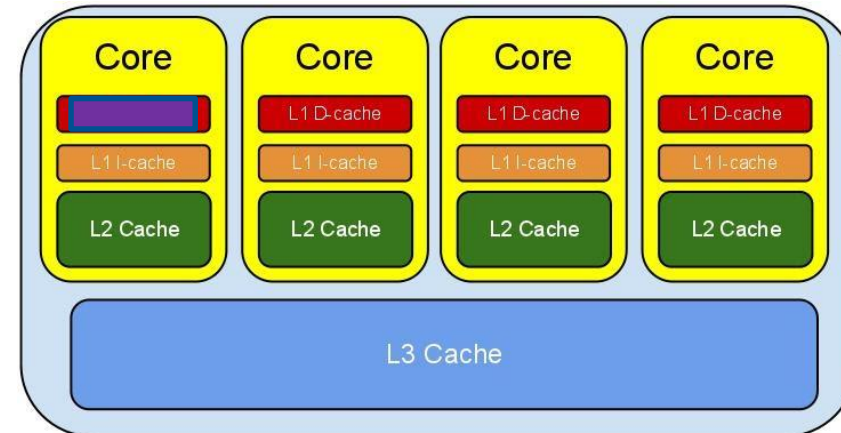
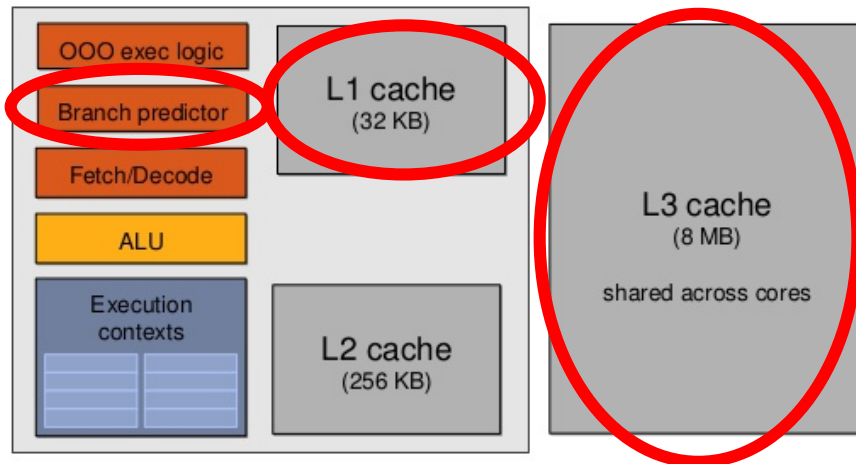
Caches: fast access memories

Contention to guess recently used data

Why LLC?

- Cross core, inclusiveness, high resolution..

Intel i7-4770



Why Attack LLC?

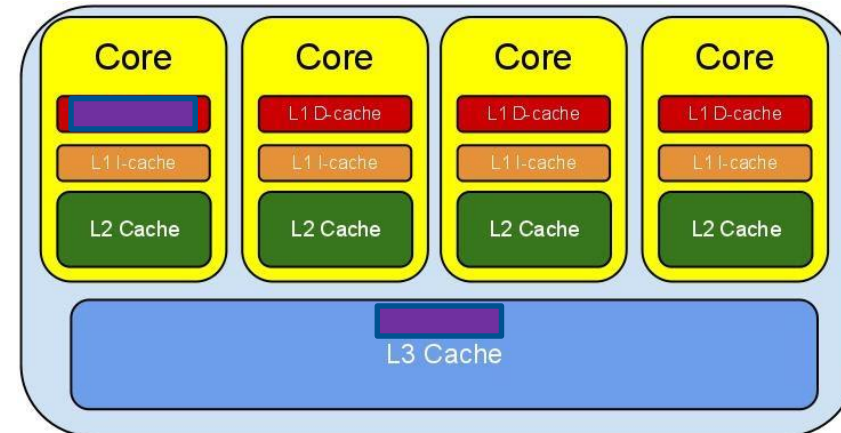
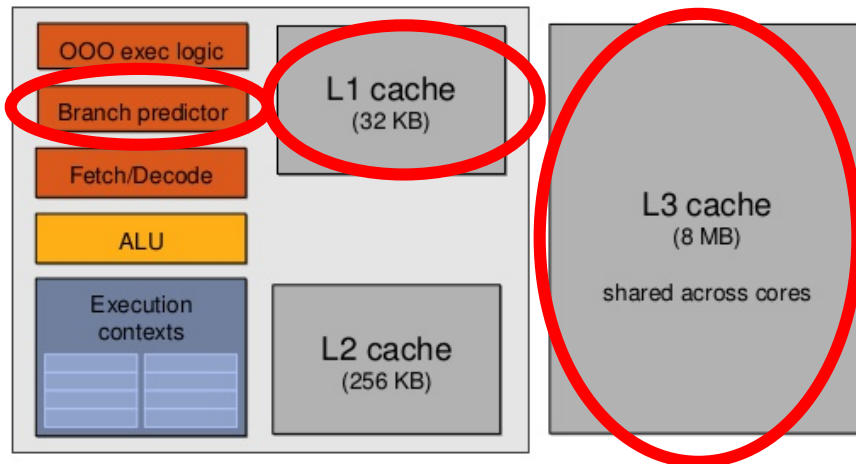
Caches: fast access memories

Contention to guess recently used data

Why LLC?

- Cross core, inclusiveness, high resolution..

Intel i7-4770



Why Attack LLC?

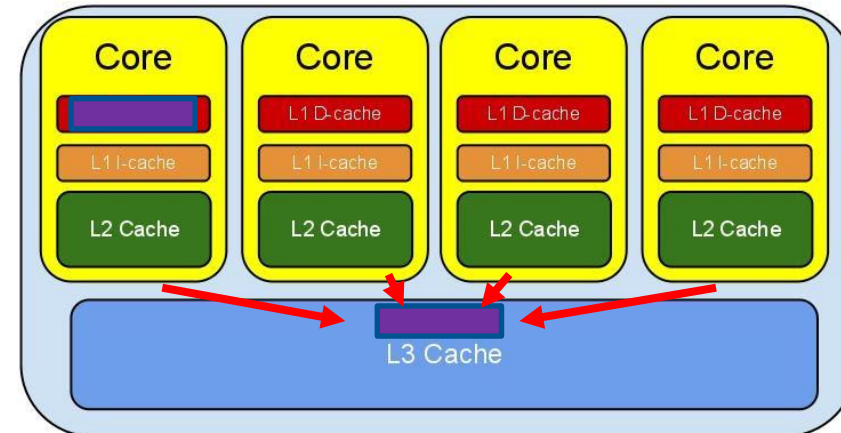
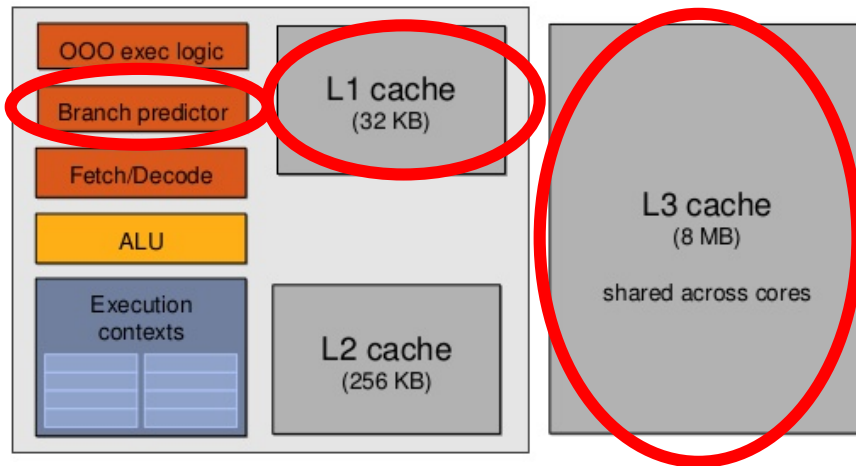
Caches: fast access memories

Contention to guess recently used data

Why LLC?

- Cross core, inclusiveness, high resolution..

Intel i7-4770



Why Attack LLC?

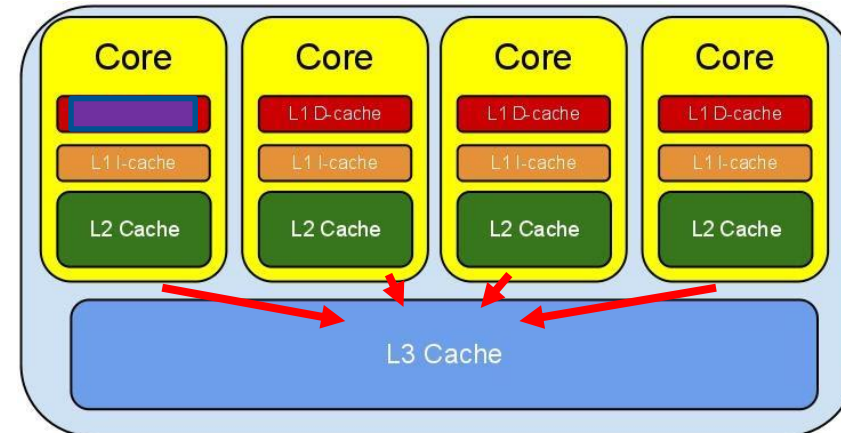
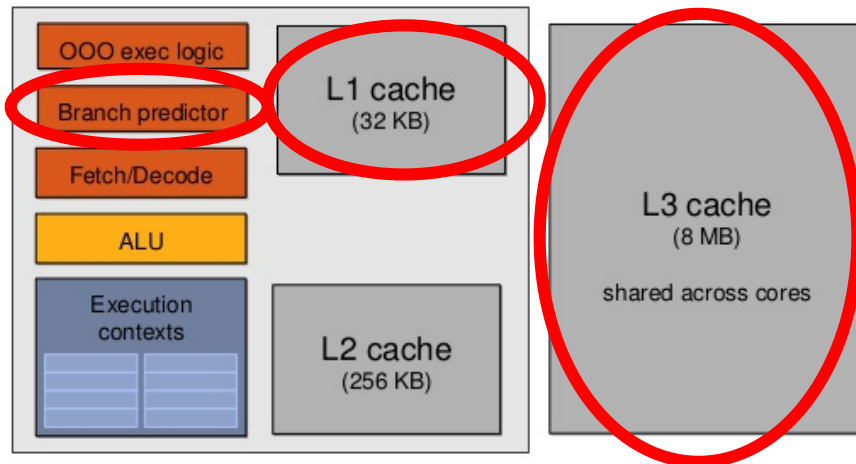
Caches: fast access memories

Contention to guess recently used data

Why LLC?

- Cross core, inclusiveness, high resolution..

Intel i7-4770



Why Attack LLC?

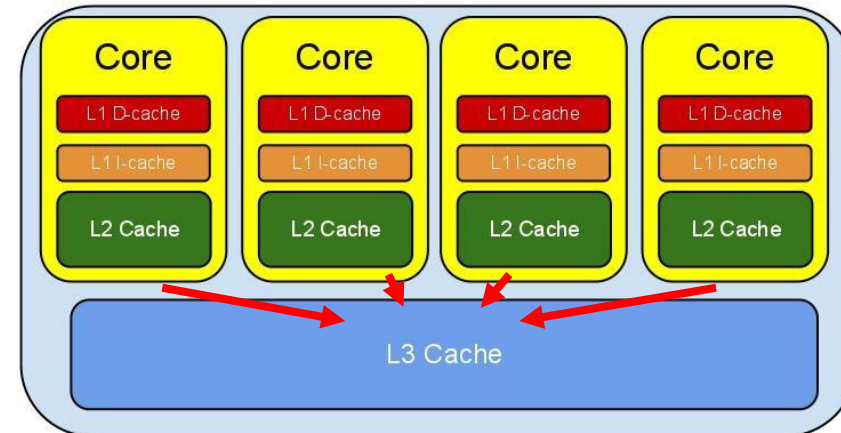
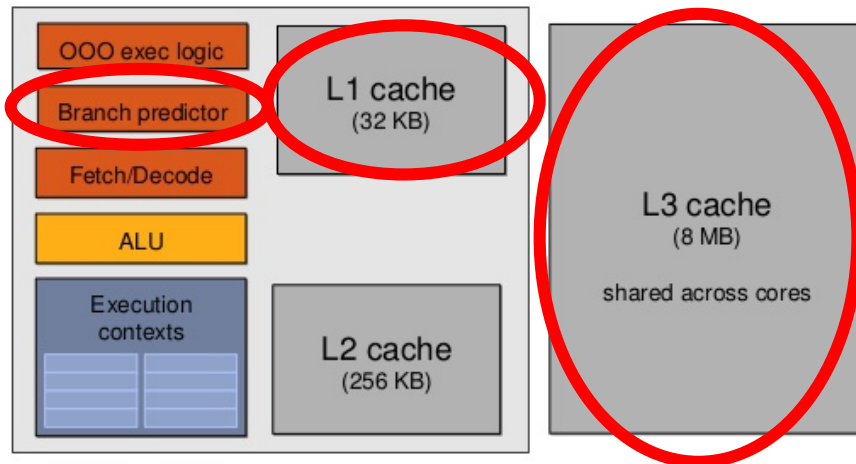
Caches: fast access memories

Contention to guess recently used data

Why LLC?

- Cross core, inclusiveness, high resolution..

Intel i7-4770



Cache Architecture

Cache architecture:

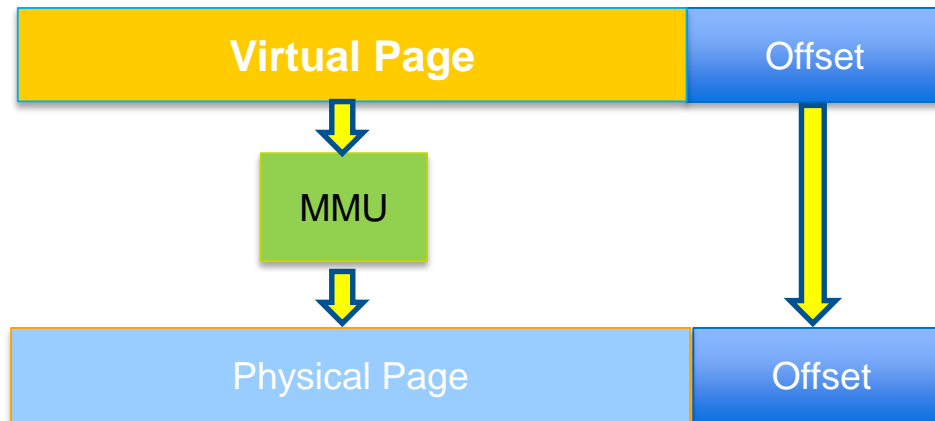
- Set associative: Cache divided in n-way sets
- Location in the cache determined by physical address



Cache Architecture

Cache architecture:

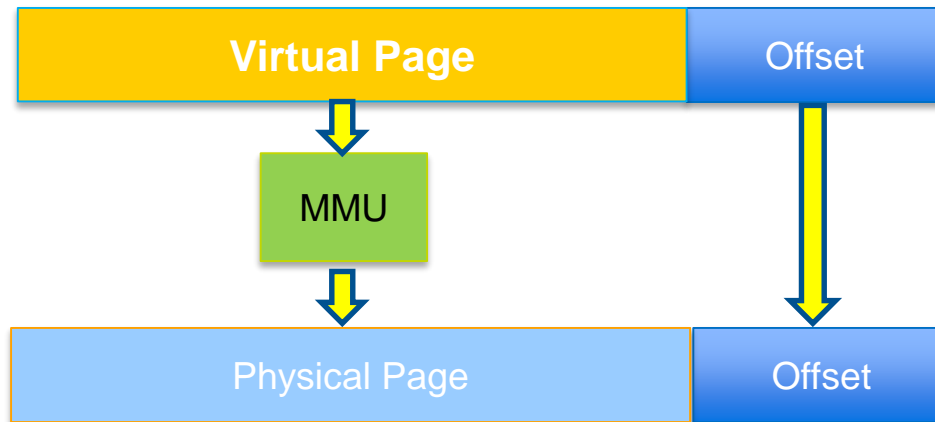
- Set associative: Cache divided in n-way sets
- Location in the cache determined by physical address



Cache Architecture

Cache architecture:

- Set associative: Cache divided in n-way sets
- Location in the cache determined by physical address

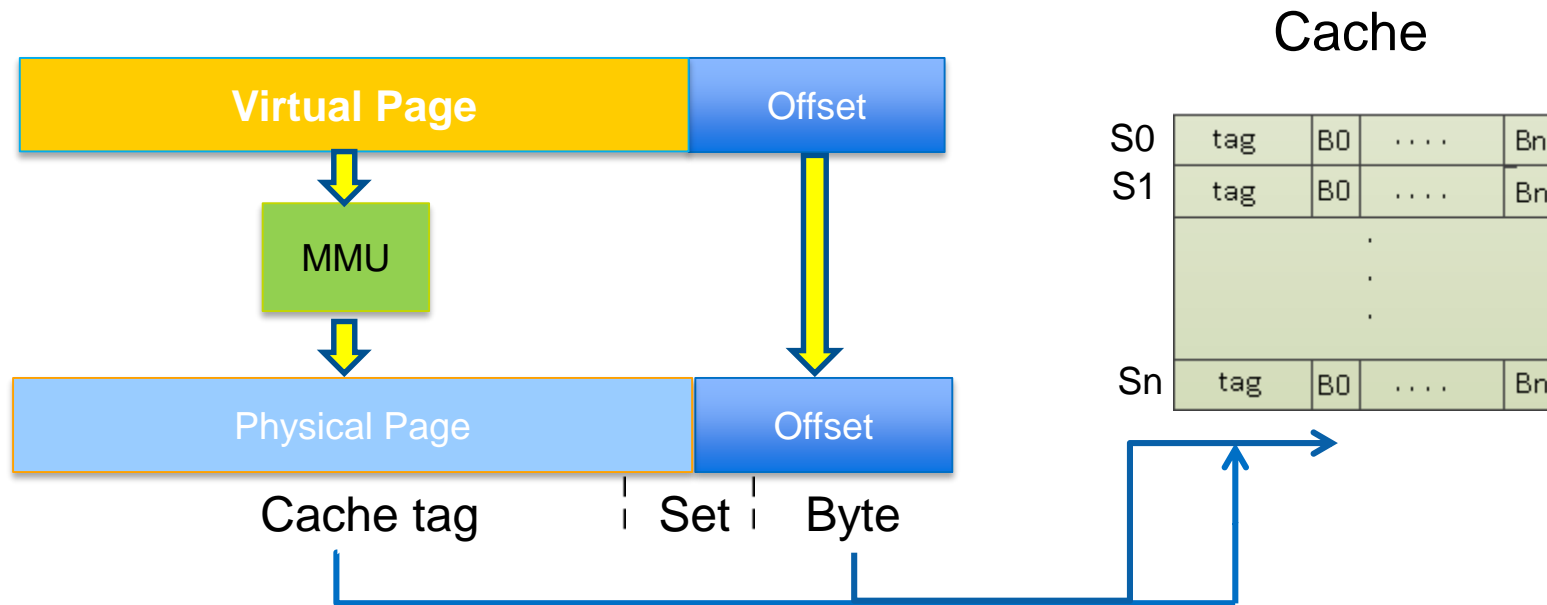


tag	B0	Bn
tag	B0	Bn
.			
.			
.			
tag	B0	Bn

Cache Architecture

Cache architecture:

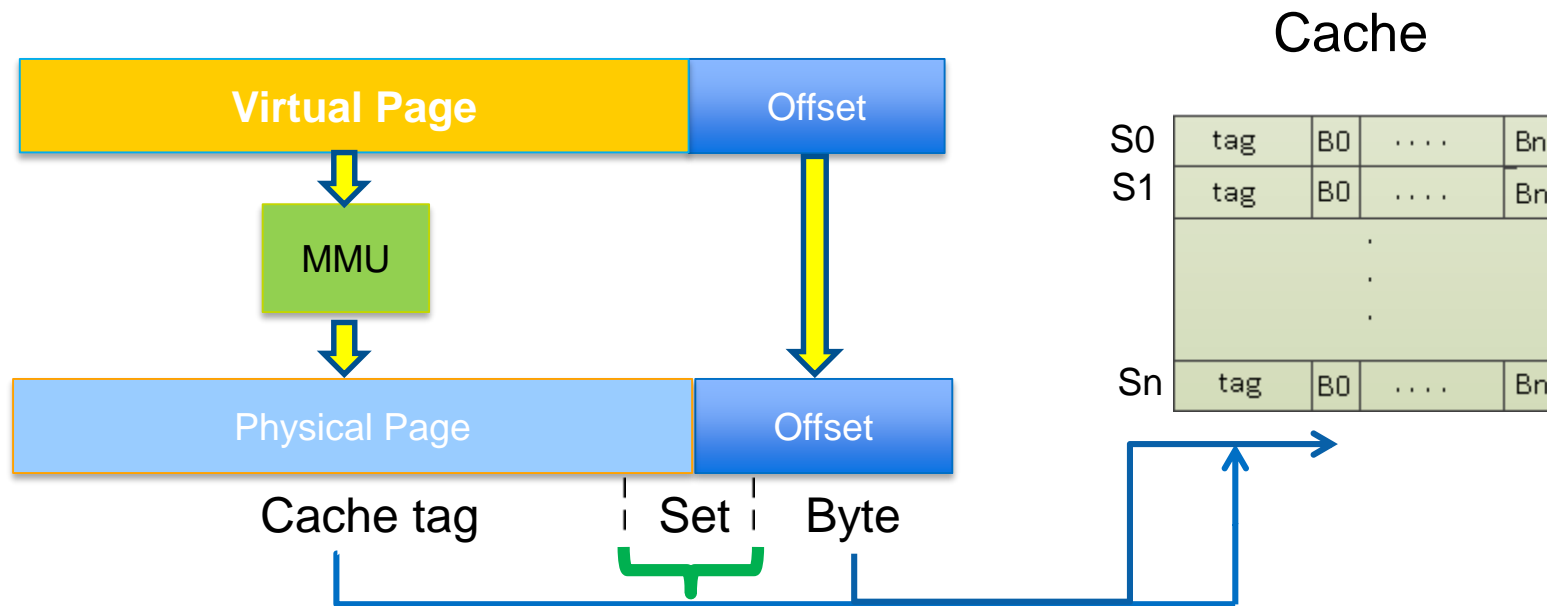
- Set associative: Cache divided in n-way sets
- Location in the cache determined by physical address



Cache Architecture

Cache architecture:

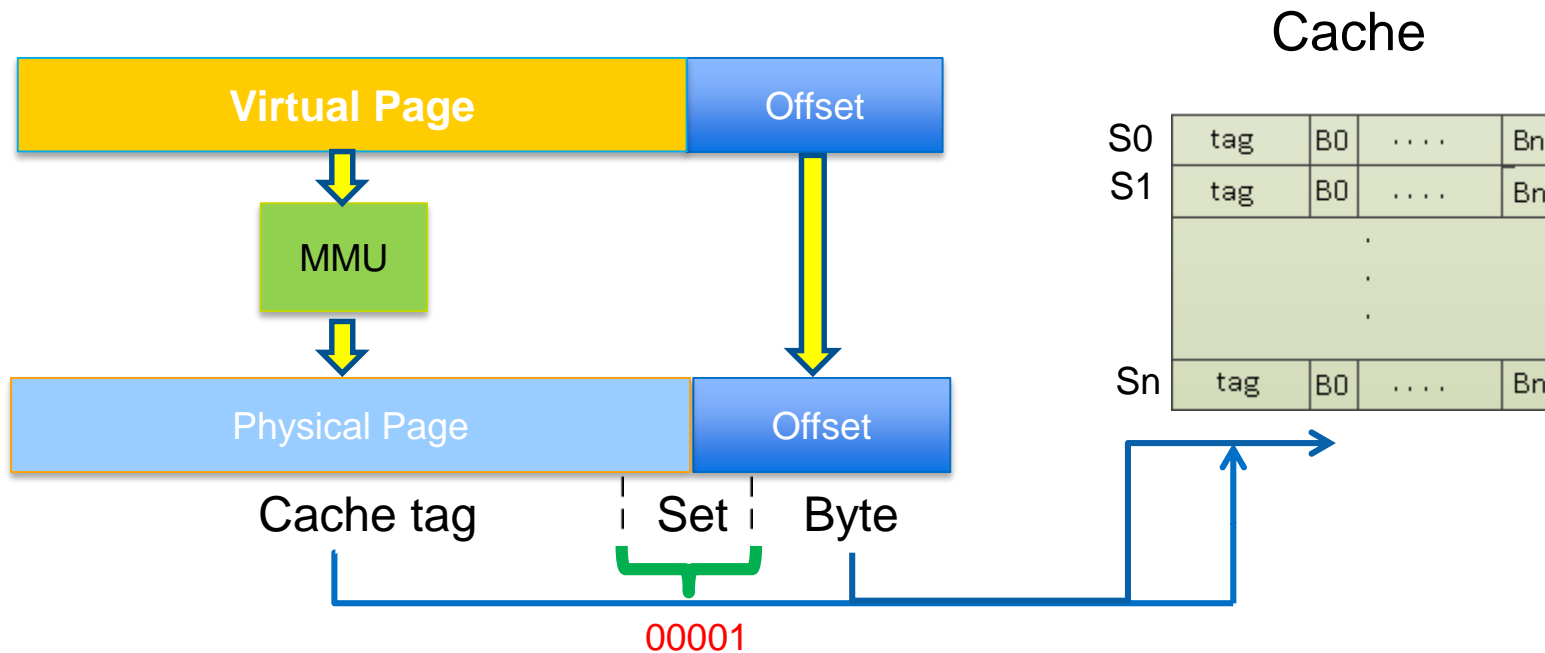
- Set associative: Cache divided in n-way sets
- Location in the cache determined by physical address



Cache Architecture

Cache architecture:

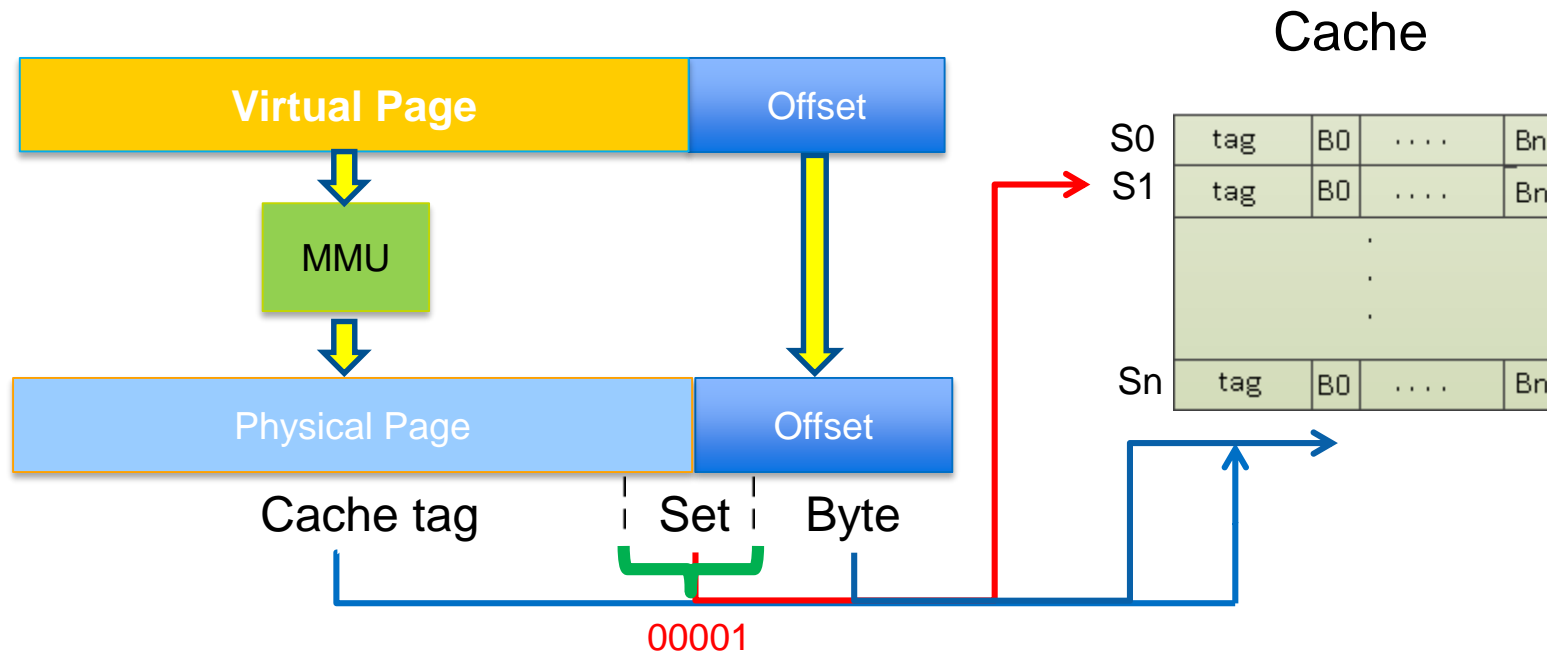
- Set associative: Cache divided in n-way sets
- Location in the cache determined by physical address



Cache Architecture

Cache architecture:

- Set associative: Cache divided in n-way sets
- Location in the cache determined by physical address



The Flush and Reload Attack

The first example: Flush and Reload

The Flush and Reload Attack

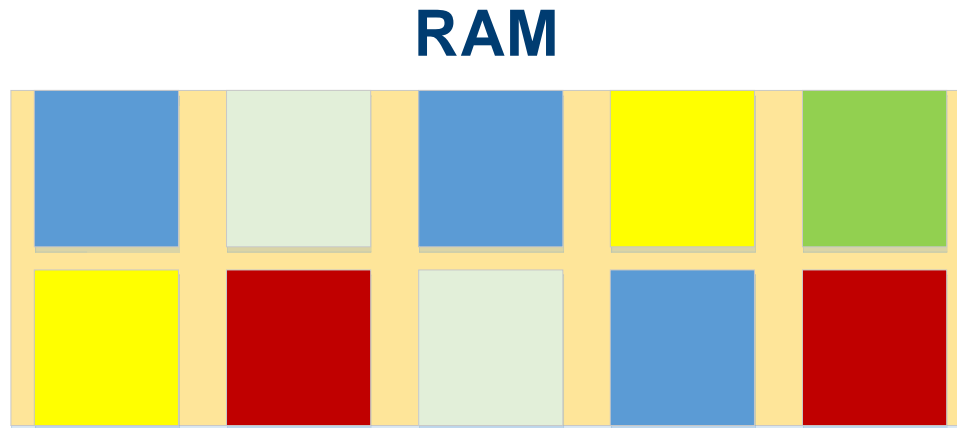
The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

The Flush and Reload Attack

The first example: Flush and Reload

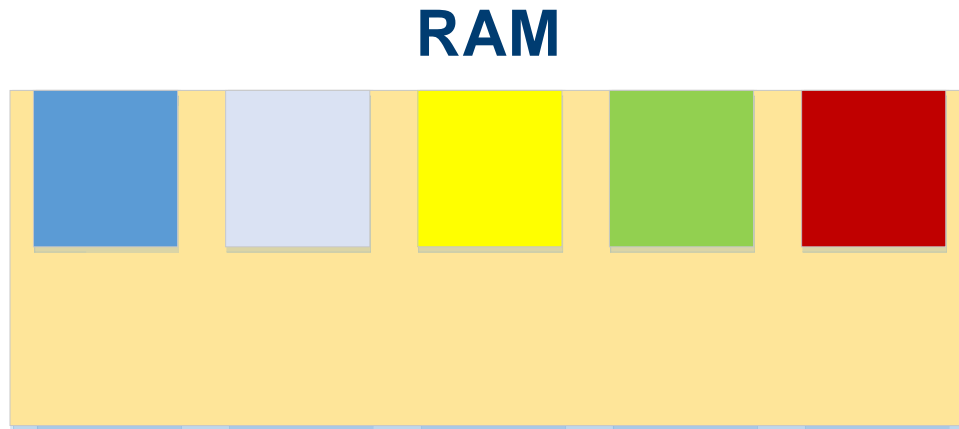
Assumptions: shared memory (e.g. KSM)



The Flush and Reload Attack

The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

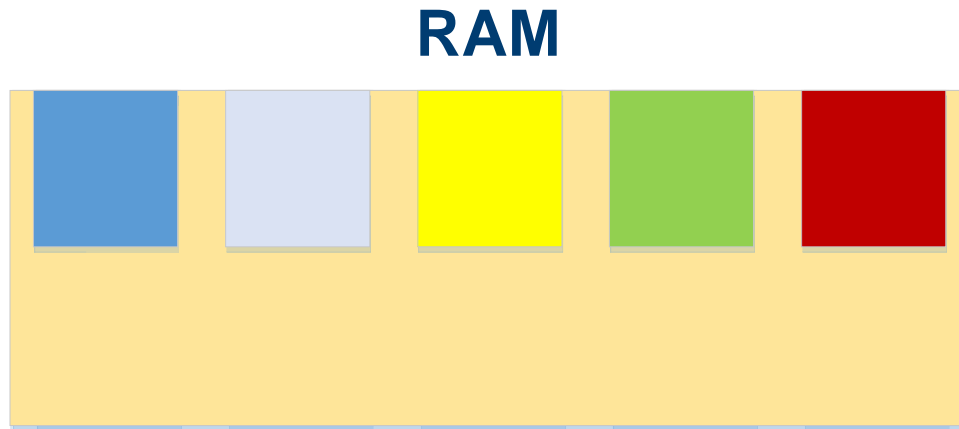


The Flush and Reload Attack

The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!



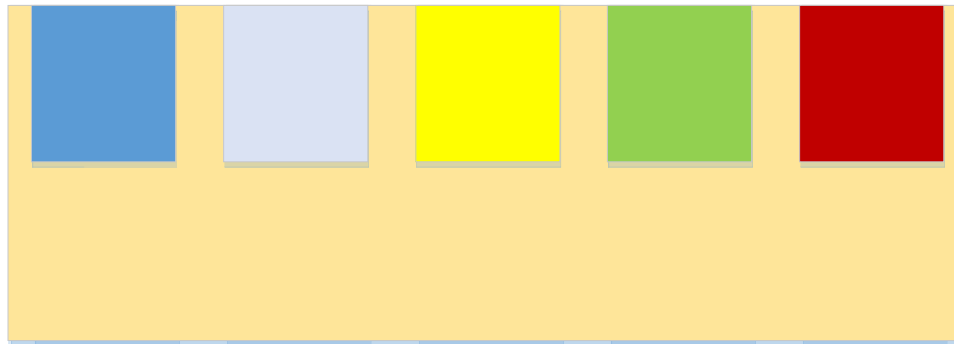
The Flush and Reload Attack

The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

RAM



Security considerations and disallowing inter-Virtual Machine Transparent Page Sharing (2080735)

Purpose

This article acknowledges the recent academic research that leverages Transparent Page Sharing (TPS) to gain unauthorized access to data under certain highly controlled conditions and documents VMware's precautionary measure of restricting TPS to individual virtual machines by default in upcoming ESXi releases. At this time, VMware believes that the published information disclosure due to TPS between virtual machines is impractical in a real world deployment.

The Flush and Reload Attack

The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

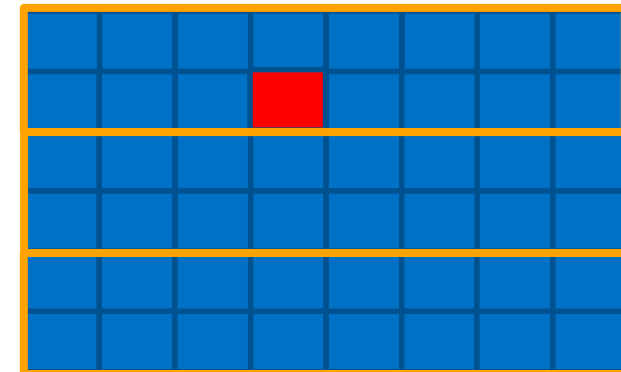
The Flush and Reload Attack

The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

Cache



The Flush and Reload Attack

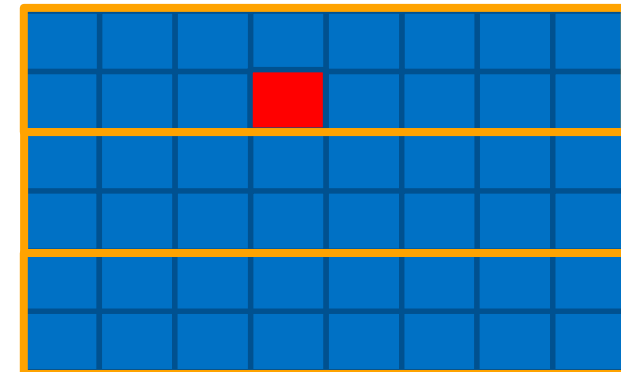
The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

Attacker flushes

Cache



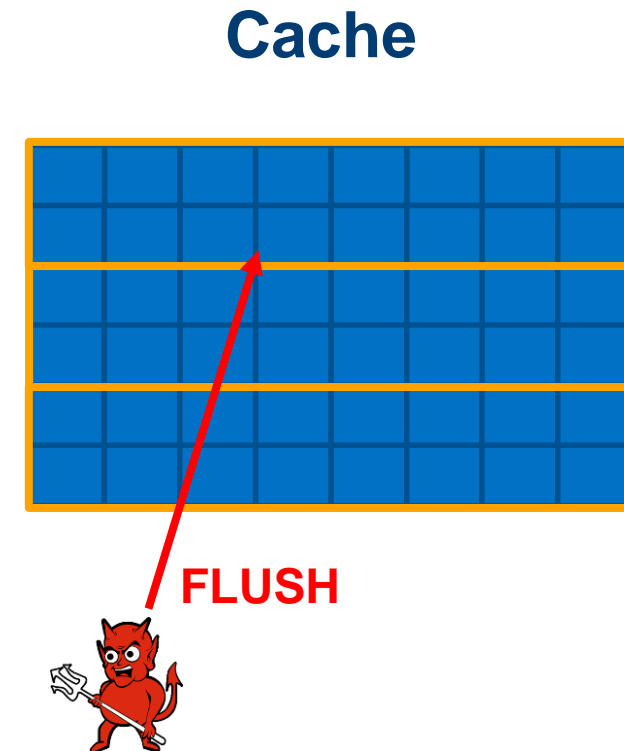
The Flush and Reload Attack

The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

Attacker flushes



The Flush and Reload Attack

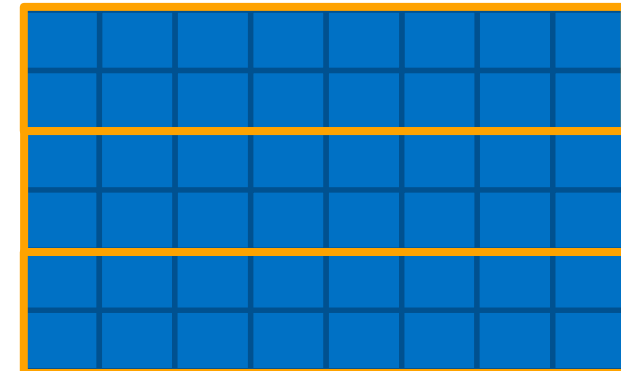
The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

Attacker flushes

Cache



The Flush and Reload Attack

The first example: Flush and Reload

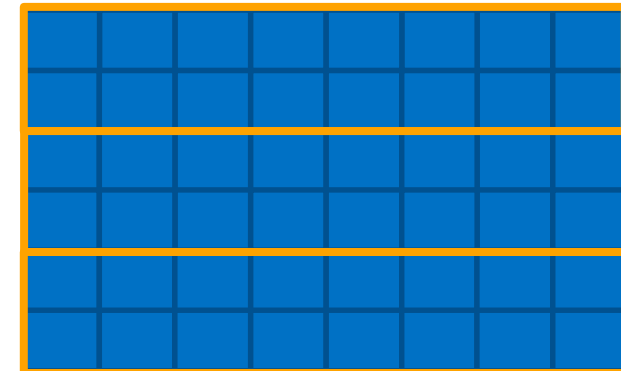
Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

Attacker flushes

Victim access/not access

Cache



The Flush and Reload Attack

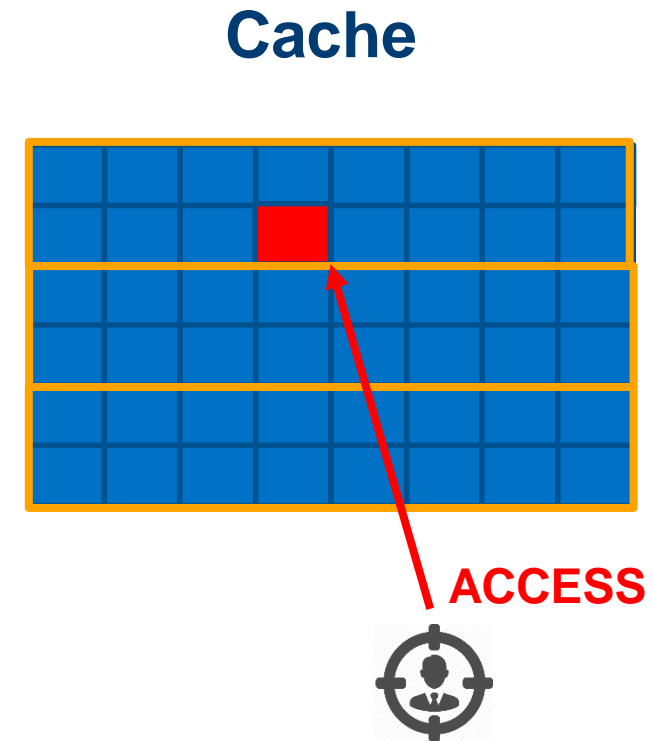
The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

Attacker flushes

Victim access/not access



The Flush and Reload Attack

The first example: Flush and Reload

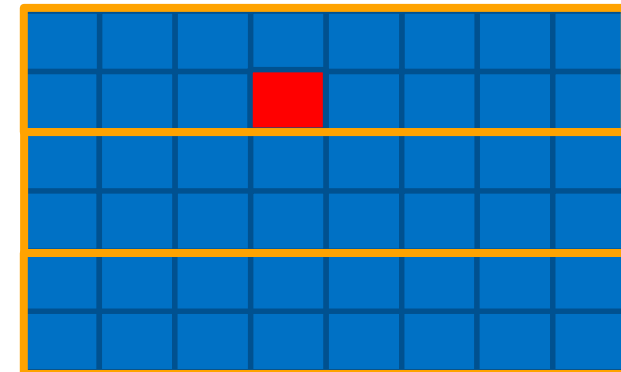
Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

Attacker flushes

Victim access/not access

Cache



The Flush and Reload Attack

The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

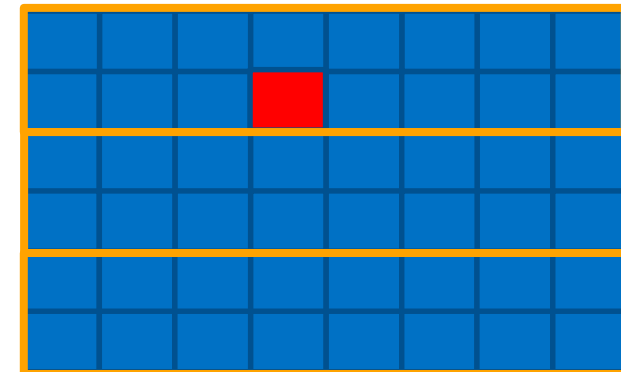
Deduplication poses big threats!

Attacker flushes

Victim access/not access

Attacker re-accesses

Cache



The Flush and Reload Attack

The first example: Flush and Reload

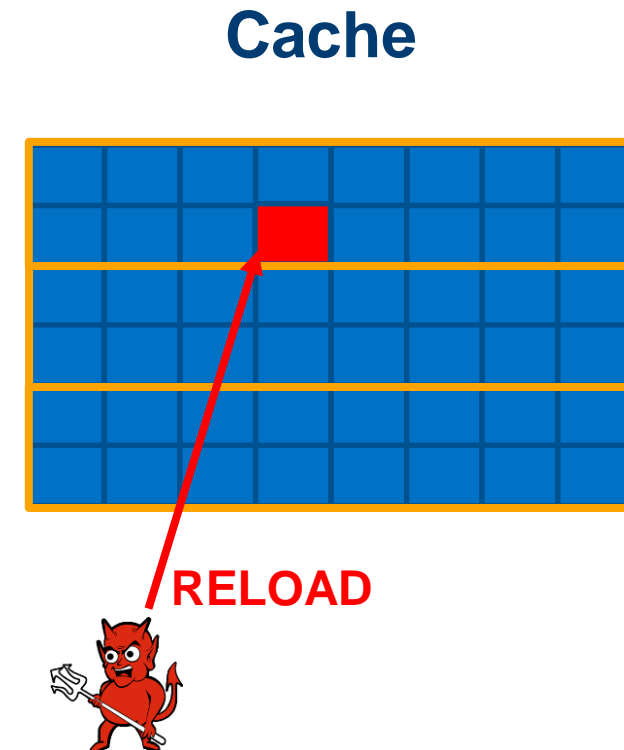
Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

Attacker flushes

Victim access/not access

Attacker re-accesses



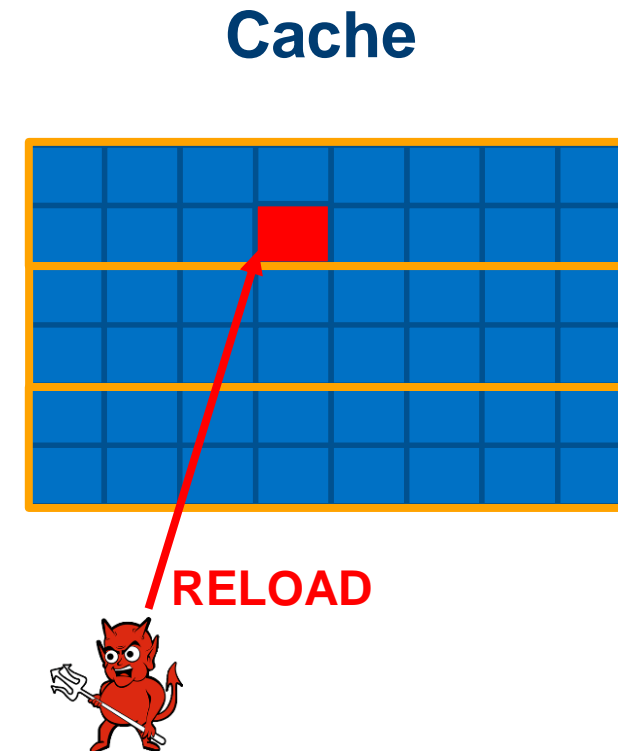
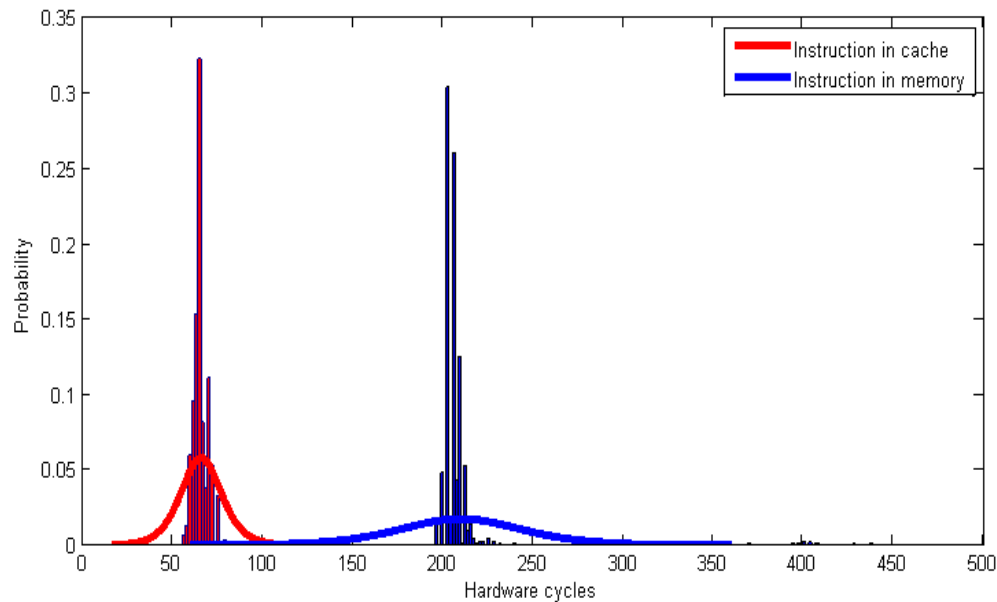
The Flush and Reload Attack

The first example: Flush and Reload

Assumptions: shared memory (e.g. KSM)

Deduplication poses big threats!

Attacker flushes

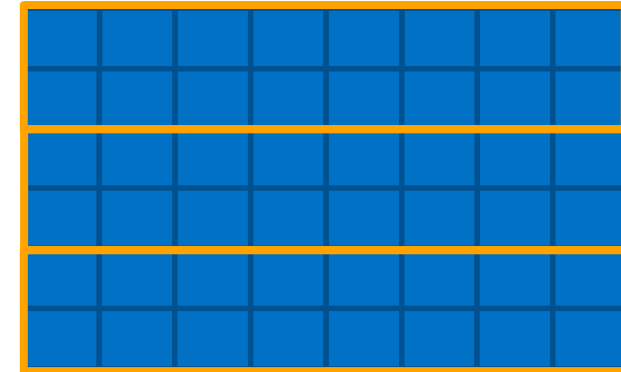


The Prime and Probe Attack

The second example: Prime and Probe

No special assumptions

Cache



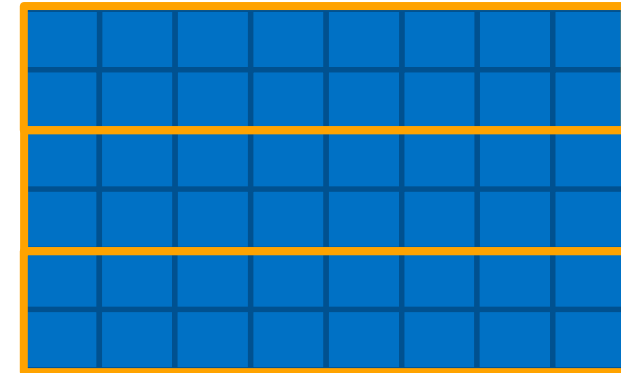
The Prime and Probe Attack

The second example: Prime and Probe

No special assumptions

Attacker Primes

Cache

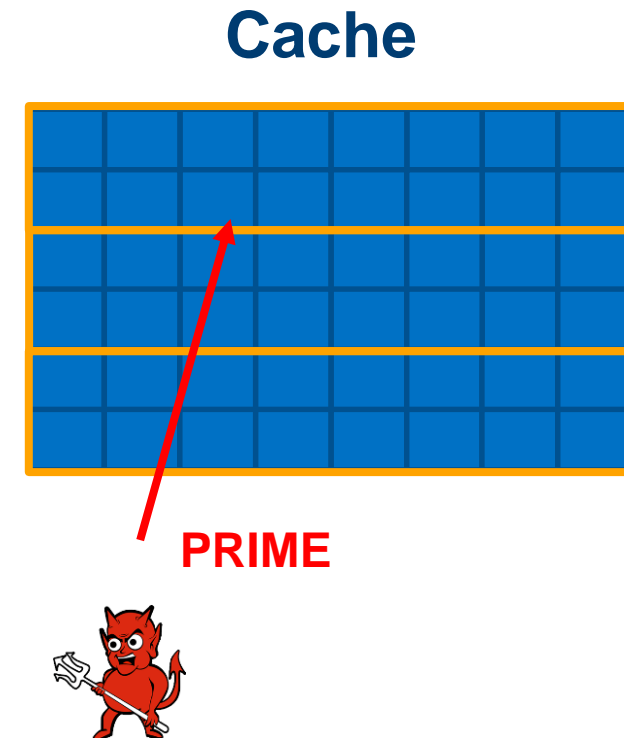


The Prime and Probe Attack

The second example: Prime and Probe

No special assumptions

Attacker Primes

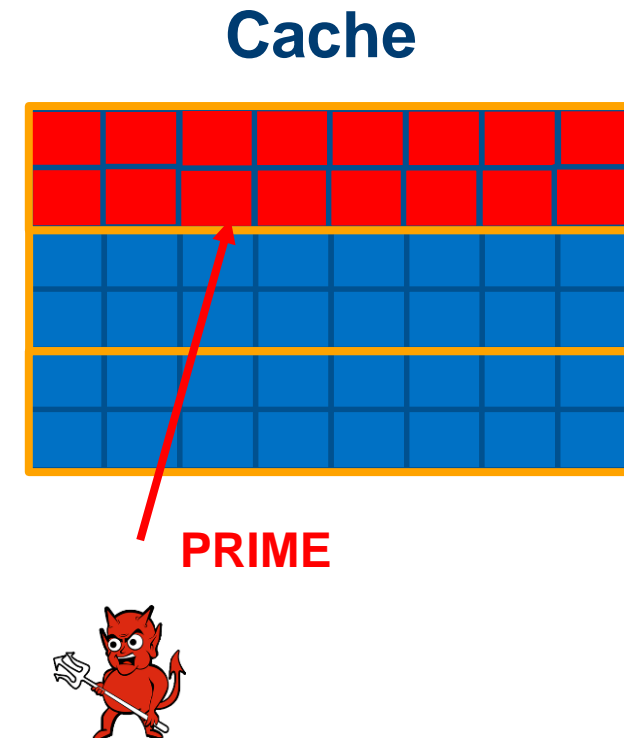


The Prime and Probe Attack

The second example: Prime and Probe

No special assumptions

Attacker Primes



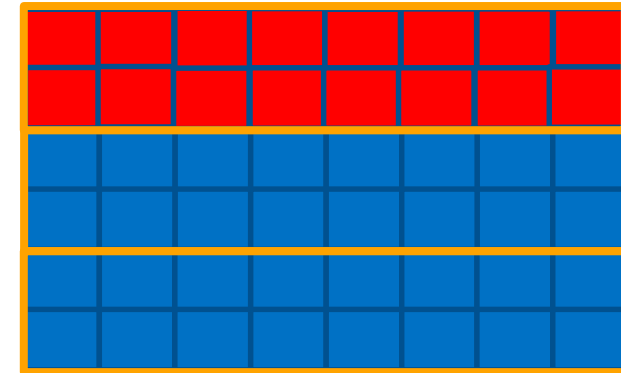
The Prime and Probe Attack

The second example: Prime and Probe

No special assumptions

Attacker Primes

Cache



The Prime and Probe Attack

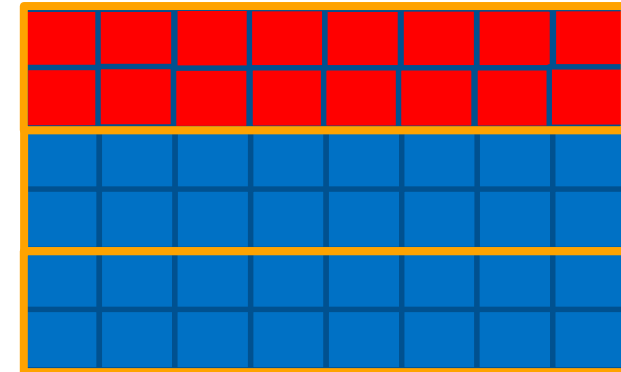
The second example: Prime and Probe

No special assumptions

Attacker Primes

Victim accesses/not accesses

Cache



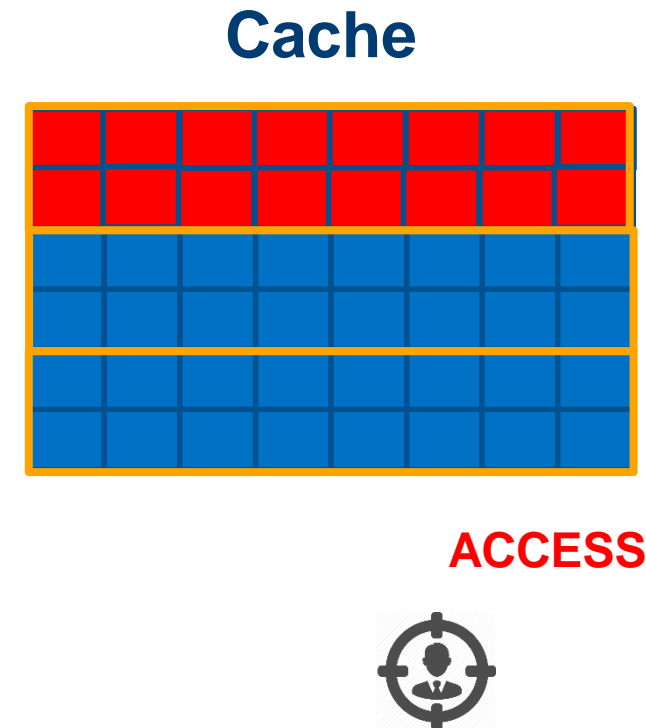
The Prime and Probe Attack

The second example: Prime and Probe

No special assumptions

Attacker Primes

Victim accesses/not accesses



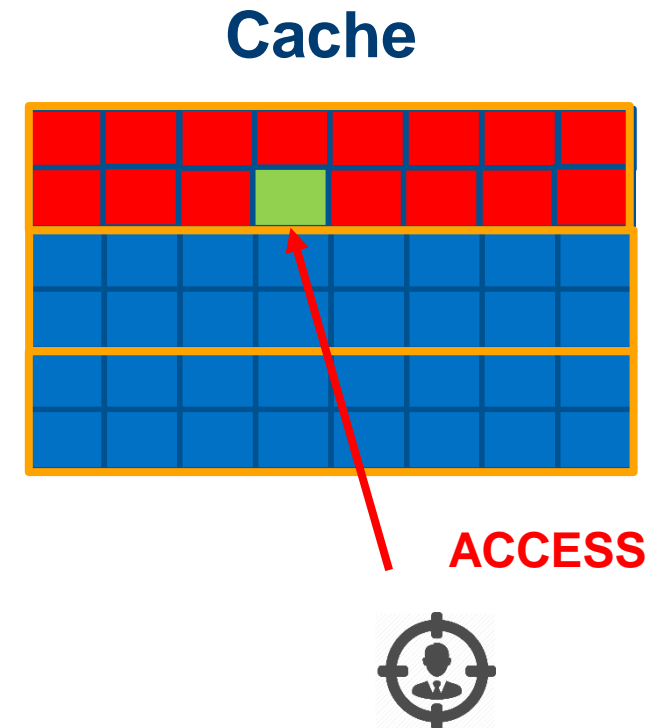
The Prime and Probe Attack

The second example: Prime and Probe

No special assumptions

Attacker Primes

Victim accesses/not accesses



The Prime and Probe Attack

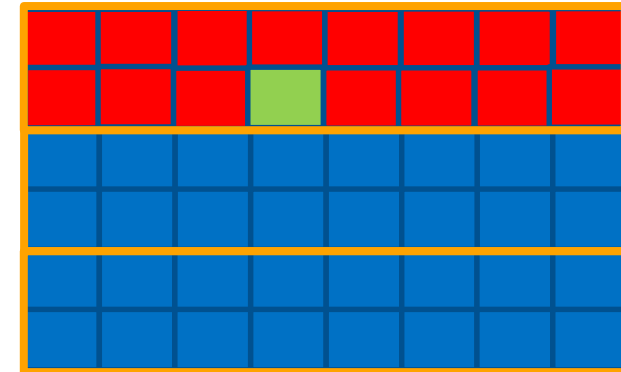
The second example: Prime and Probe

No special assumptions

Attacker Primes

Victim accesses/not accesses

Cache



The Prime and Probe Attack

The second example: Prime and Probe

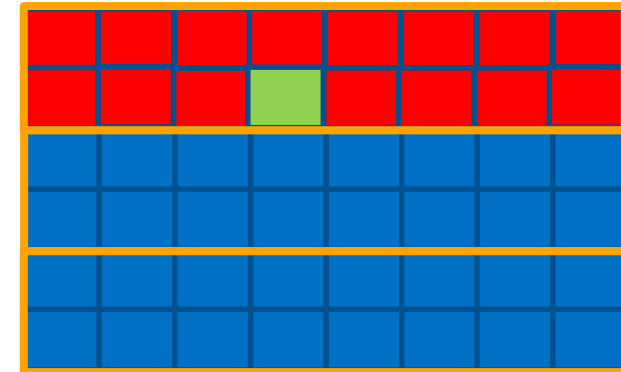
No special assumptions

Attacker Primes

Victim accesses/not accesses

Attacker re-accesses

Cache



The Prime and Probe Attack

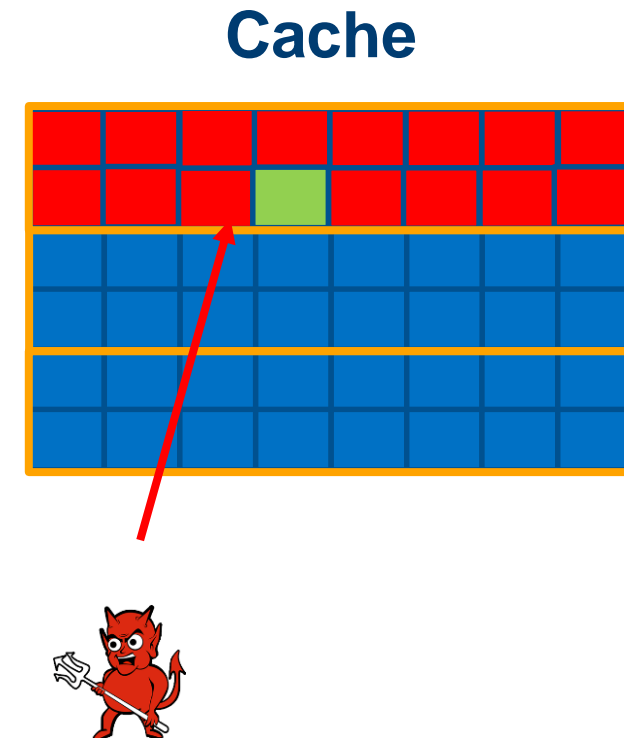
The second example: Prime and Probe

No special assumptions

Attacker Primes

Victim accesses/not accesses

Attacker re-accesses



The Prime and Probe Attack

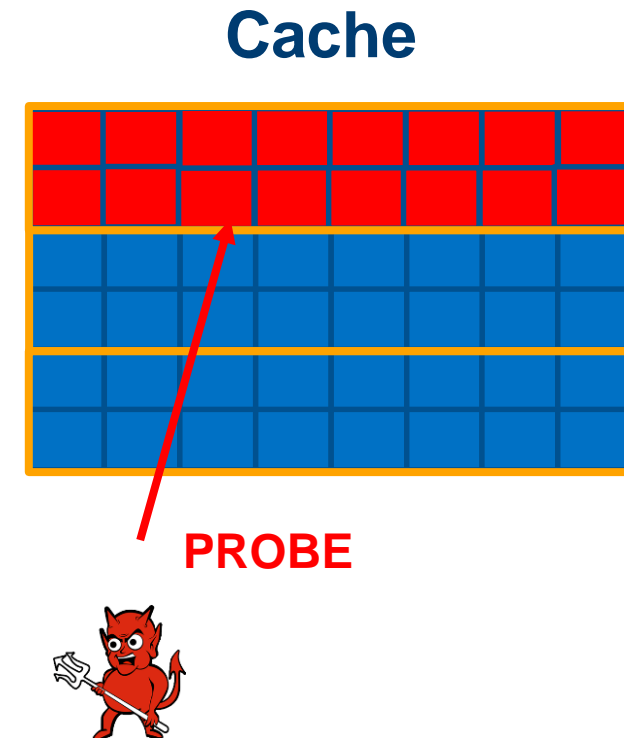
The second example: Prime and Probe

No special assumptions

Attacker Primes

Victim accesses/not accesses

Attacker re-accesses



The Prime and Probe Attack

The second example: Prime and Probe

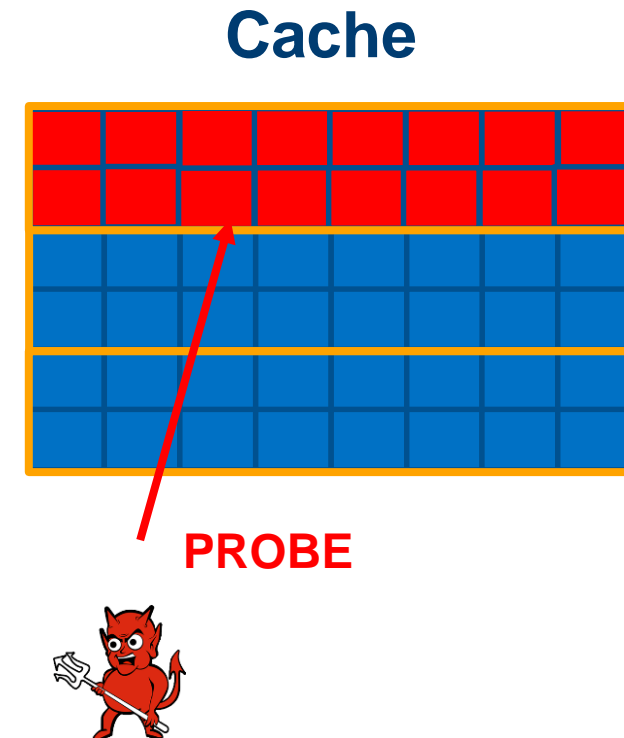
No special assumptions

Attacker Primes

Victim accesses/not accesses

Attacker re-accesses

Fast time: no access



The Prime and Probe Attack

The second example: Prime and Probe

No special assumptions

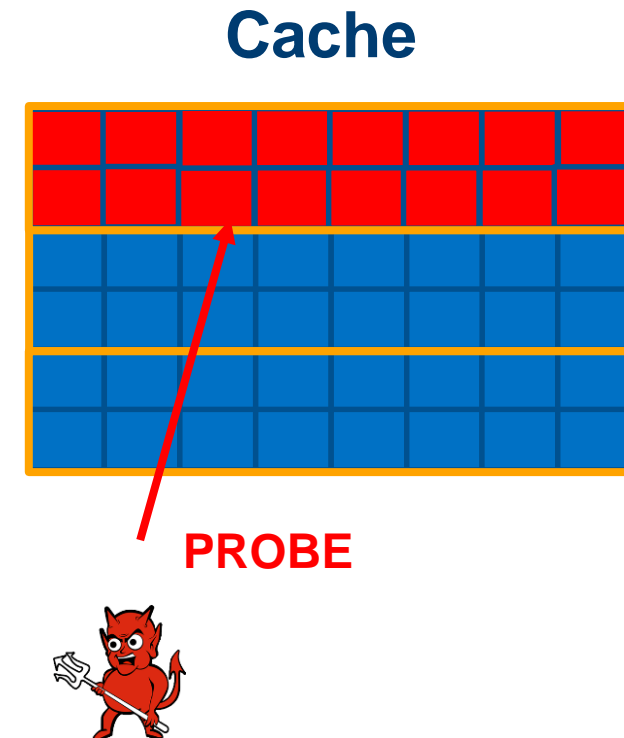
Attacker Primes

Victim accesses/not accesses

Attacker re-accesses

Fast time: no access

Slow time: access



How to Retrieve Information?

Montgomery ladder RSA

```
1 function modpow ( $a, b$ );  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   | if  $e_i = 0$  then  
5   |   |  $R_1 = R_0 * R_1 \bmod N;$   
6   |   |  $R_0 = R_0 * R_0 \bmod N;$   
7   | end  
8   | else  
9   |   |  $R_0 = R_0 * R_1 \bmod N;$   
10  |   |  $R_1 = R_1 * R_1 \bmod N;$   
11  | end  
12 end  
13 return  $R_0;$ 
```

How to Retrieve Information?

Montgomery ladder RSA

```
1 function modpow ( $a, b$ );  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

How to Retrieve Information?

Montgomery ladder RSA

```
1 function modpow (a, b);  
   Input  : base  $b$ , modulus  $N$ , secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

➡ $P=0x7ffc480$

Flush and Reload

How to Retrieve Information?

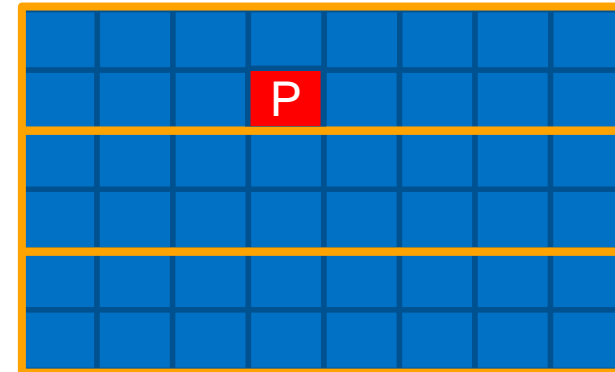
Montgomery ladder RSA

```
1 function modpow ( $a, b$ );  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

➔ $P=0x7ffc480$

Flush and Reload Cache



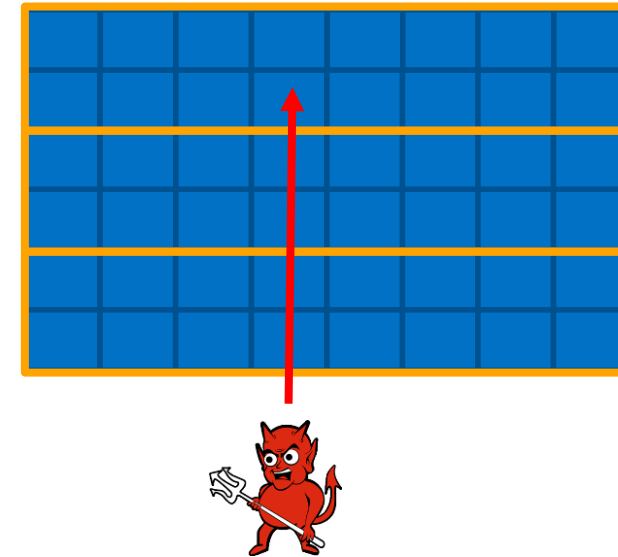
How to Retrieve Information?

Montgomery ladder RSA

```
1 function modpow (a, b);  
   Input  : base  $b$ , modulus  $N$ , secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1$ ;  $R_1 = b$ ;  
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N$ ;  
6      $R_0 = R_0 * R_0 \bmod N$ ;  
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N$ ;  
10     $R_1 = R_1 * R_1 \bmod N$ ;  
11  end  
12 end  
13 return  $R_0$ ;
```

Physical address $\rightarrow P=0x7ffc480$

Flush and Reload Cache



How to Retrieve Information?

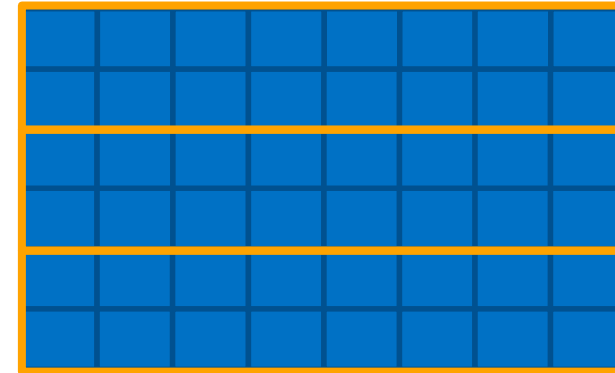
Montgomery ladder RSA

```
1 function modpow (a, b);  
   Input  : base  $b$ , modulus  $N$ , secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

➡ $P=0x7ffc480$

Flush and Reload Cache



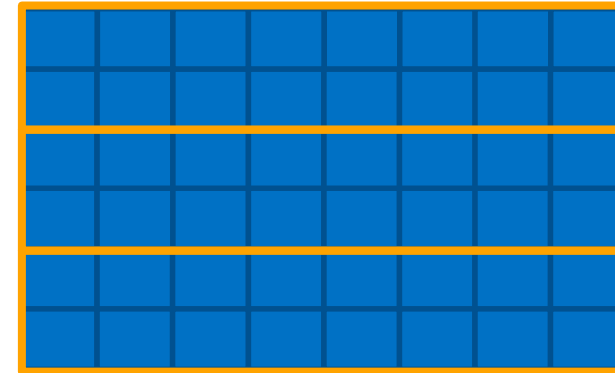
How to Retrieve Information?

Montgomery ladder RSA

```
1 function modpow (a, b);  
   Input  : base b, modulus N, secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address $\rightarrow P=0x7ffc480$

Flush and Reload Cache



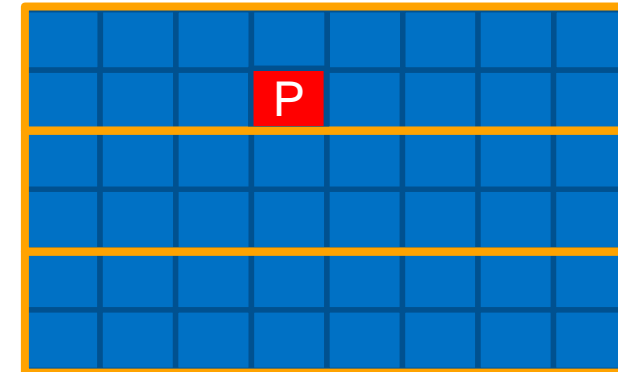
How to Retrieve Information?

Montgomery ladder RSA

```
1 function modpow (a, b);  
   Input  : base  $b$ , modulus  $N$ , secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address $\rightarrow P=0x7ffc480$

Flush and Reload Cache



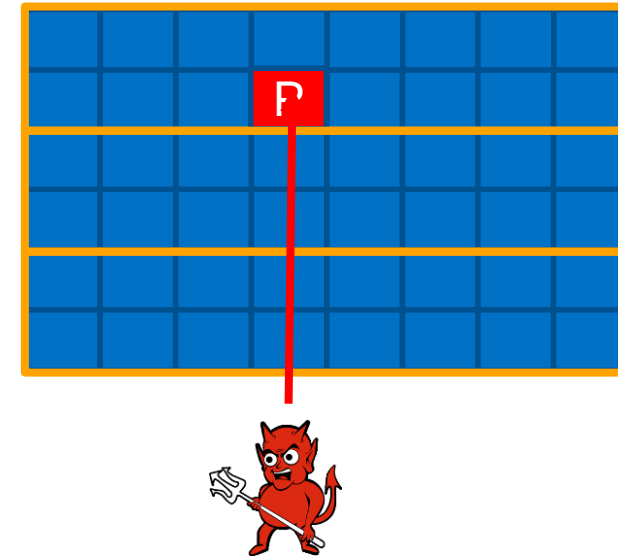
How to Retrieve Information?

Montgomery ladder RSA

```
1 function modpow ( $a, b$ );  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1$ ;  $R_1 = b$ ;  
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N$ ;  
6      $R_0 = R_0 * R_0 \bmod N$ ;  
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N$ ;  
10     $R_1 = R_1 * R_1 \bmod N$ ;  
11  end  
12 end  
13 return  $R_0$ ;
```

Physical address $\rightarrow P=0x7ffc480$

Flush and Reload Cache



How to Retrieve Information?

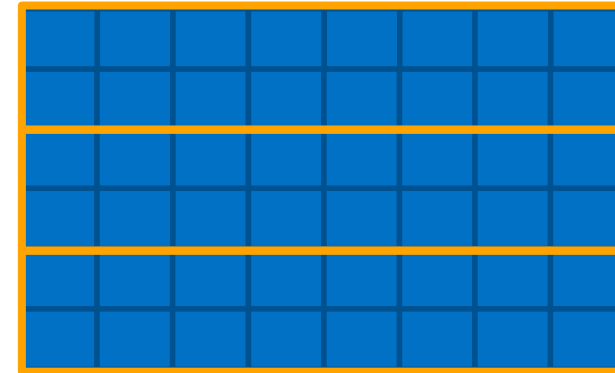
Montgomery ladder RSA

```
1 function modpow (a, b);  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

➔ P=0x7ffc480

Prime and Probe Cache



How to Retrieve Information?

Montgomery ladder RSA

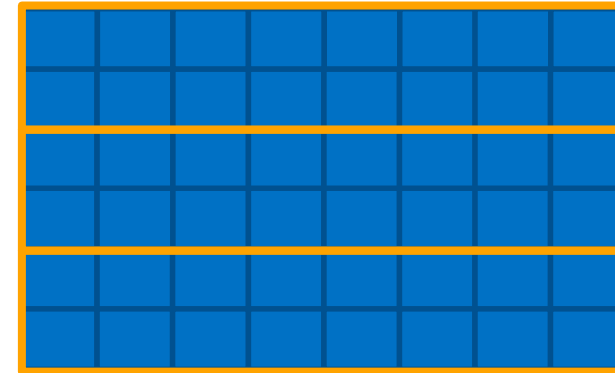
```
1 function modpow (a, b);  
   Input  : base  $b$ , modulus  $N$ , secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

$P=0x7ffc480$

SET=1554

Prime and Probe Cache



How to Retrieve Information?

Montgomery ladder RSA

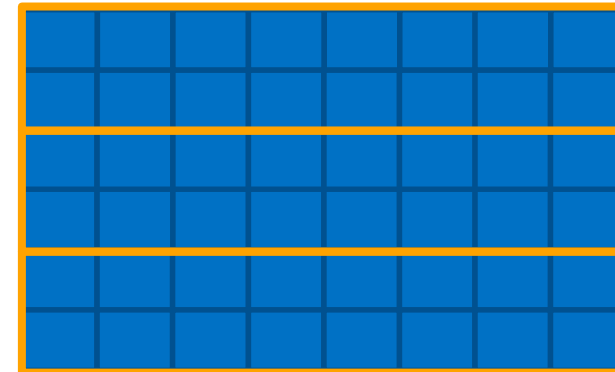
```
1 function modpow (a, b);  
   Input  : base b, modulus N, secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

$P=0x7ffc480$

SET=1554

Prime and Probe Cache



How to Retrieve Information?

Montgomery ladder RSA

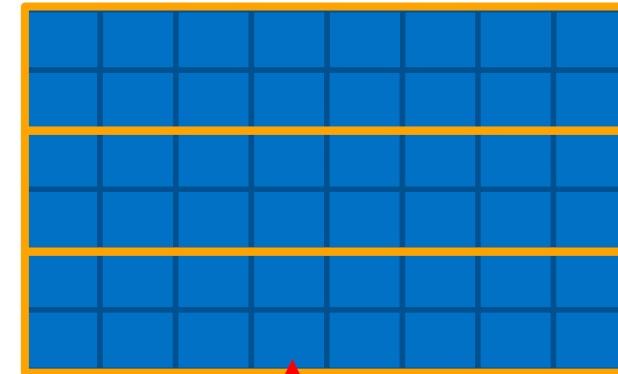
```
1 function modpow (a, b);  
   Input  : base  $b$ , modulus  $N$ , secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

$P=0x7ffc480$

SET=1554

Prime and Probe Cache



How to Retrieve Information?

Montgomery ladder RSA

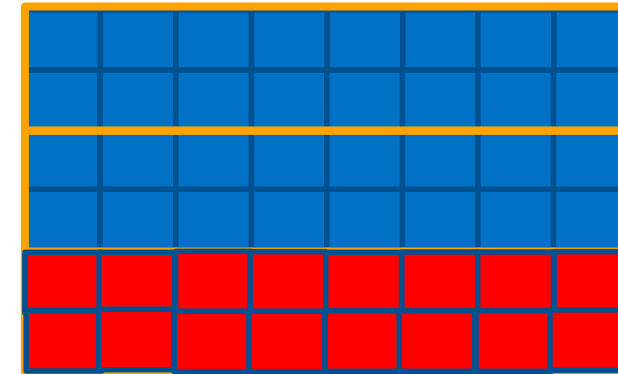
```
1 function modpow (a, b);  
   Input  : base b, modulus N, secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

$P=0x7ffc480$

SET=1554

Prime and Probe Cache



How to Retrieve Information?

Montgomery ladder RSA

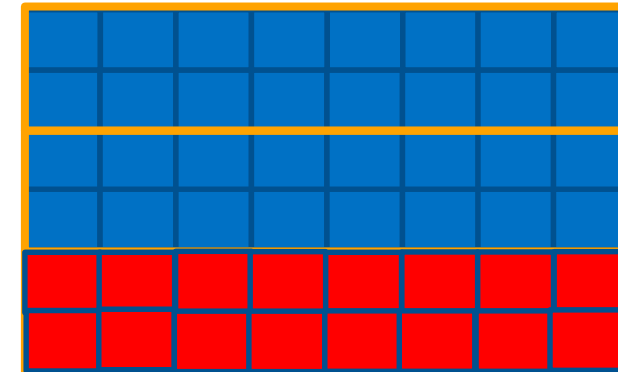
```
1 function modpow (a, b);  
   Input  : base b, modulus N, secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

$P=0x7ffc480$

SET=1554

Prime and Probe Cache



How to Retrieve Information?

Montgomery ladder RSA

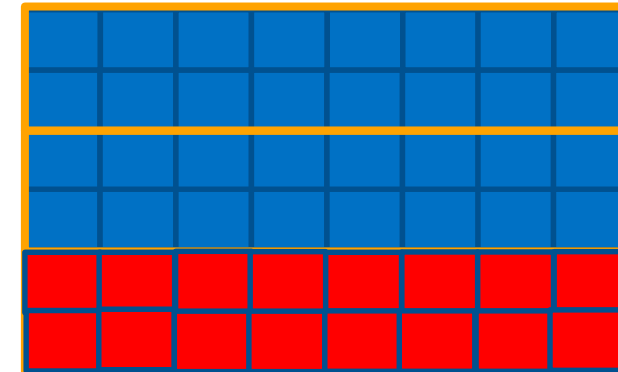
```
1 function modpow (a, b);  
   Input  : base b, modulus N, secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

$P = 0x7ffc480$

SET=1554

Prime and Probe Cache



How to Retrieve Information?

Montgomery ladder RSA

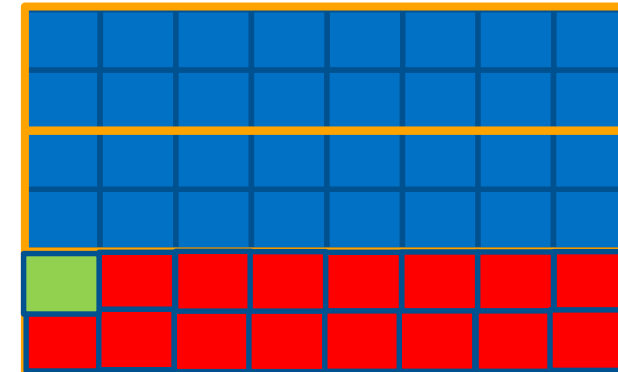
```
1 function modpow (a, b);  
   Input  : base b, modulus N, secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

$P=0x7ffc480$

SET=1554

Prime and Probe Cache



How to Retrieve Information?

Montgomery ladder RSA

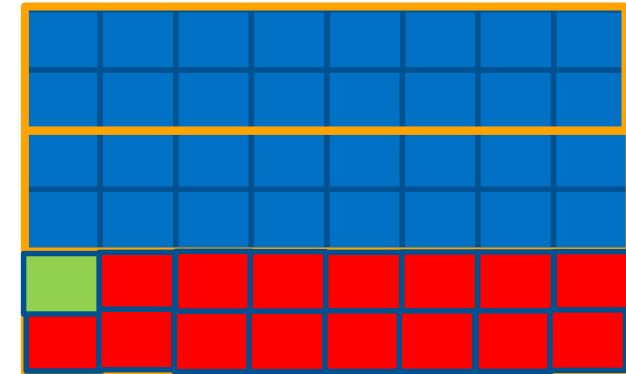
```
1 function modpow (a, b);  
   Input  : base b, modulus N, secret  
             $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R_0 = 1; R_1 = b;$   
3 for  $i = k - 1$  downto 0 do  
4   if  $e_i == 0$  then  
5      $R_1 = R_0 * R_1 \bmod N;$   
6      $R_0 = R_0 * R_0 \bmod N;$   
7   end  
8   else  
9      $R_0 = R_0 * R_1 \bmod N;$   
10     $R_1 = R_1 * R_1 \bmod N;$   
11  end  
12 end  
13 return  $R_0;$ 
```

Physical address

$P=0x7ffc480$

SET=1554

Prime and Probe Cache



Where are LLC attacks a threat?

Commercial IaaS/PaaS Cloud Infrastructures

VMs sharing underlying hardware

Commercial IaaS/PaaS Cloud Infrastructures

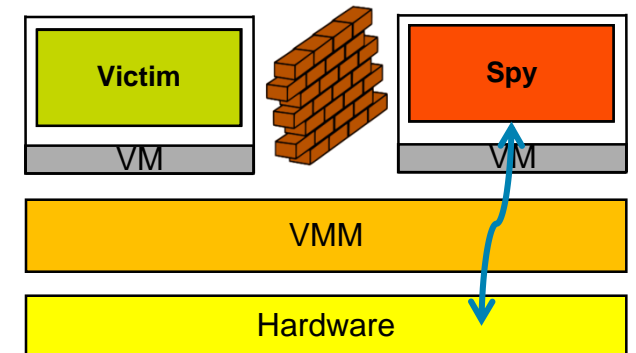
VMs sharing underlying hardware

Hardware isolation is usually not provided

Commercial IaaS/PaaS Cloud Infrastructures

VMs sharing underlying hardware

Hardware isolation is usually not provided

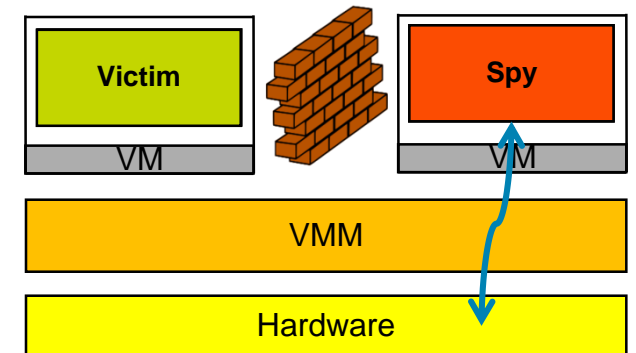


Commercial IaaS/PaaS Cloud Infrastructures

VMs sharing underlying hardware

Hardware isolation is usually not provided

Example: RSA key retrieved in Amazon EC2 [INCI16]



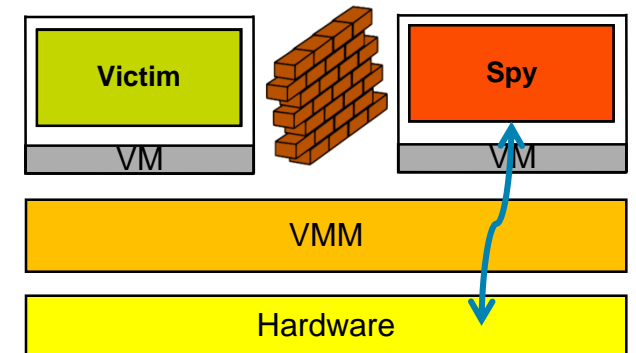
Commercial IaaS/PaaS Cloud Infrastructures

VMs sharing underlying hardware

Hardware isolation is usually not provided

Example: RSA key retrieved in Amazon EC2 [INCI16]

Pros:



Commercial IaaS/PaaS Cloud Infrastructures

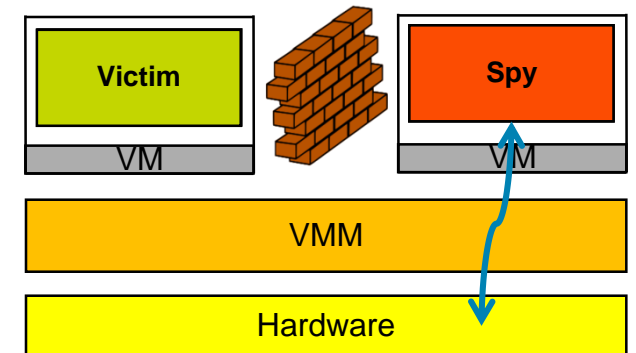
VMs sharing underlying hardware

Hardware isolation is usually not provided

Example: RSA key retrieved in Amazon EC2 [INCI16]

Pros:

- Own virtualized OS. Access to timers or huge pages



Commercial IaaS/PaaS Cloud Infrastructures

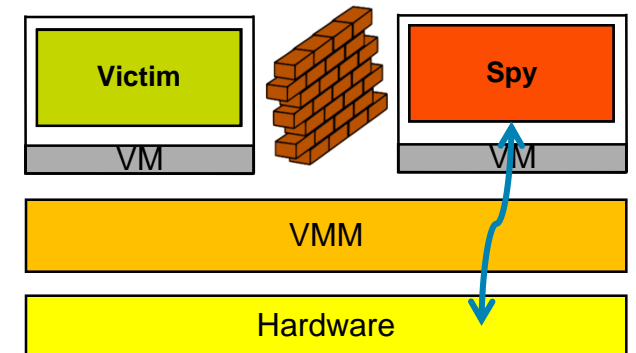
VMs sharing underlying hardware

Hardware isolation is usually not provided

Example: RSA key retrieved in Amazon EC2 [INCI16]

Pros:

- Own virtualized OS. Access to timers or huge pages
- If deduplication enabled, both attacks are applicable



Commercial IaaS/PaaS Cloud Infrastructures

VMs sharing underlying hardware

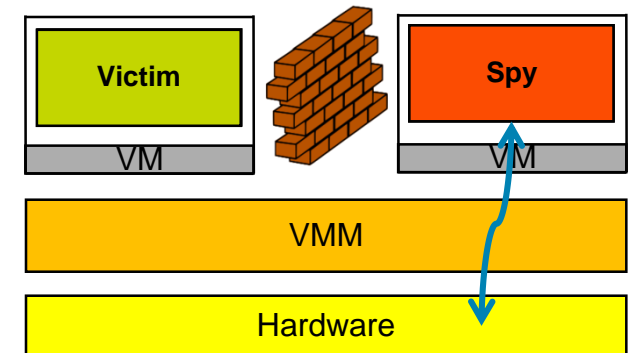
Hardware isolation is usually not provided

Example: RSA key retrieved in Amazon EC2 [INCI16]

Pros:

- Own virtualized OS. Access to timers or huge pages
- If deduplication enabled, both attacks are applicable

Cons:



Commercial IaaS/PaaS Cloud Infrastructures

VMs sharing underlying hardware

Hardware isolation is usually not provided

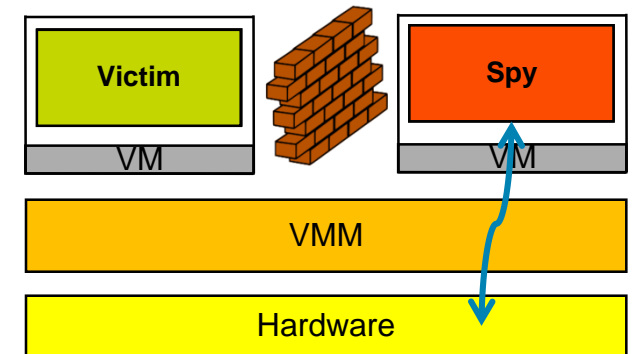
Example: RSA key retrieved in Amazon EC2 [INCI16]

Pros:

- Own virtualized OS. Access to timers or huge pages
- If deduplication enabled, both attacks are applicable

Cons:

- Co-residency can be hard to achieve



Commercial IaaS/PaaS Cloud Infrastructures

VMs sharing underlying hardware

Hardware isolation is usually not provided

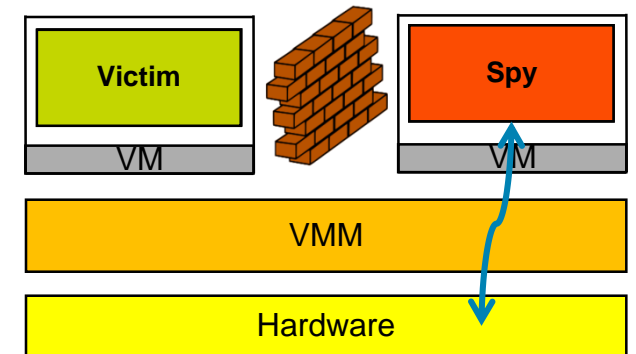
Example: RSA key retrieved in Amazon EC2 [INCI16]

Pros:

- Own virtualized OS. Access to timers or huge pages
- If deduplication enabled, both attacks are applicable

Cons:

- Co-residency can be hard to achieve
- High amount of noise



Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines

Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines



Hardware

Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines

www.yyyyyy.com

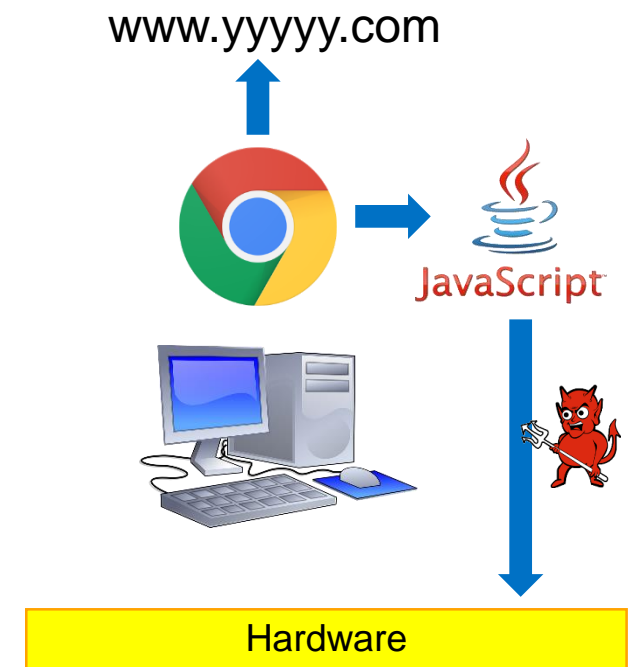


Hardware

Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines

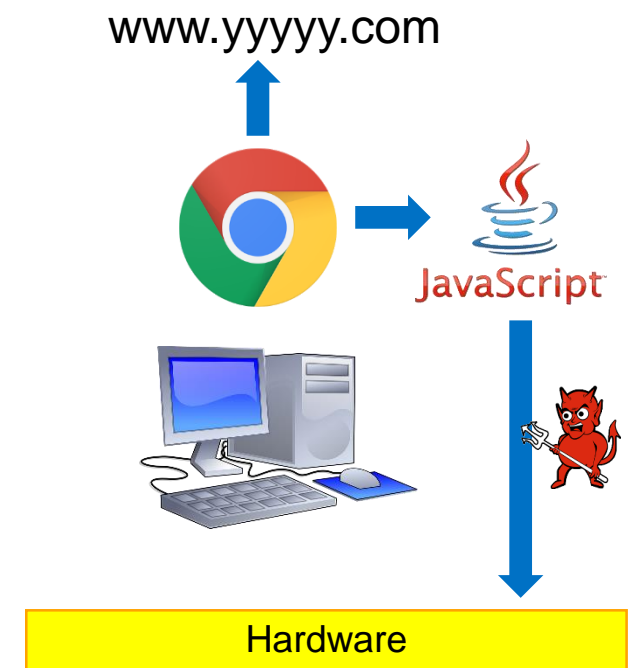


Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines

Example: Incognito browsing profiling [OREN15]



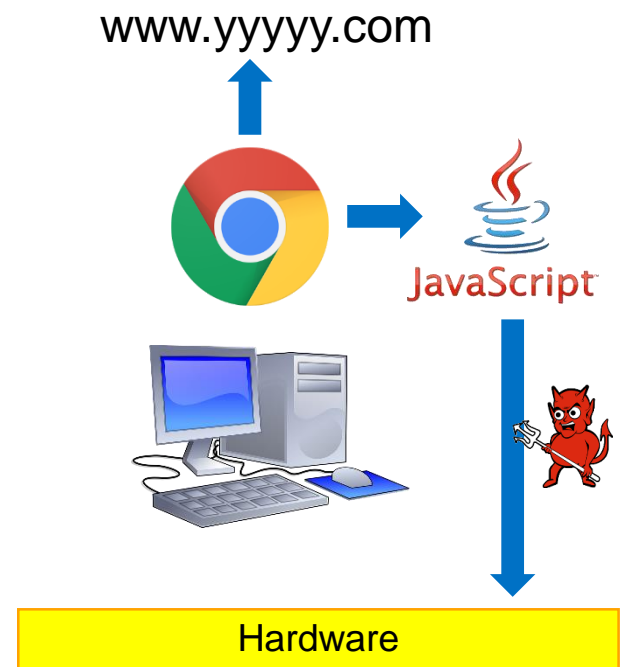
Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines

Example: Incognito browsing profiling [OREN15]

Pros:



Malicious Javascript Execution

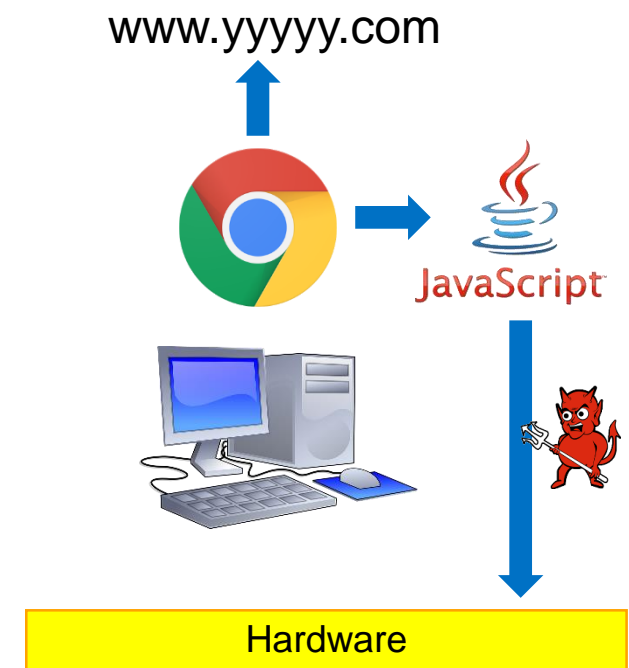
Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines

Example: Incognito browsing profiling [OREN15]

Pros:

- No need to find co-resident target



Malicious Javascript Execution

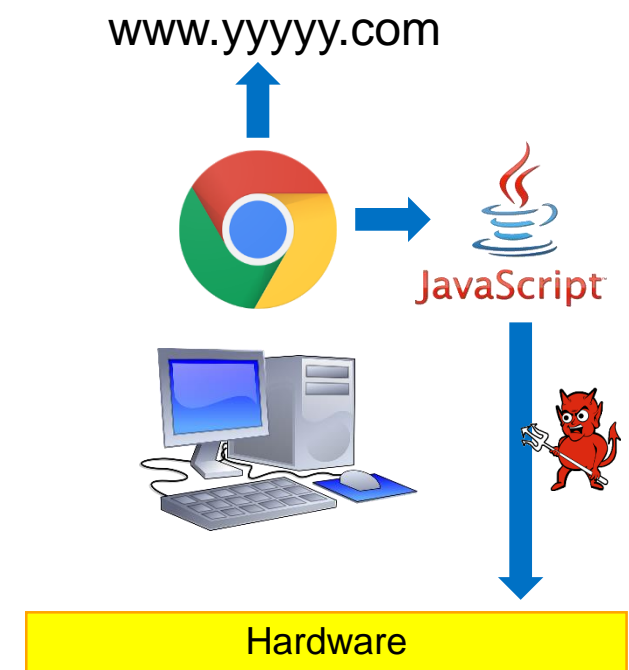
Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines

Example: Incognito browsing profiling [OREN15]

Pros:

- No need to find co-resident target
- Attack executed in local machine



Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

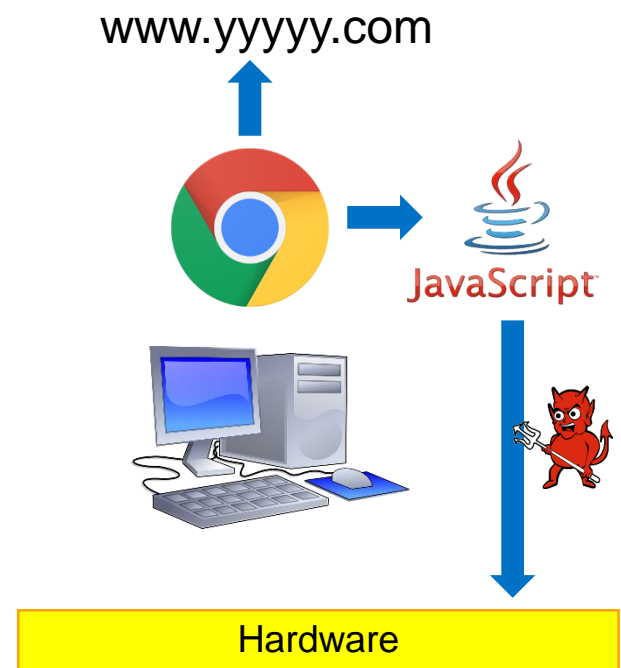
Victim access the target website, and her browser executes the javascript code in the local machines

Example: Incognito browsing profiling [OREN15]

Pros:

- No need to find co-resident target
- Attack executed in local machine

Cons:



Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines

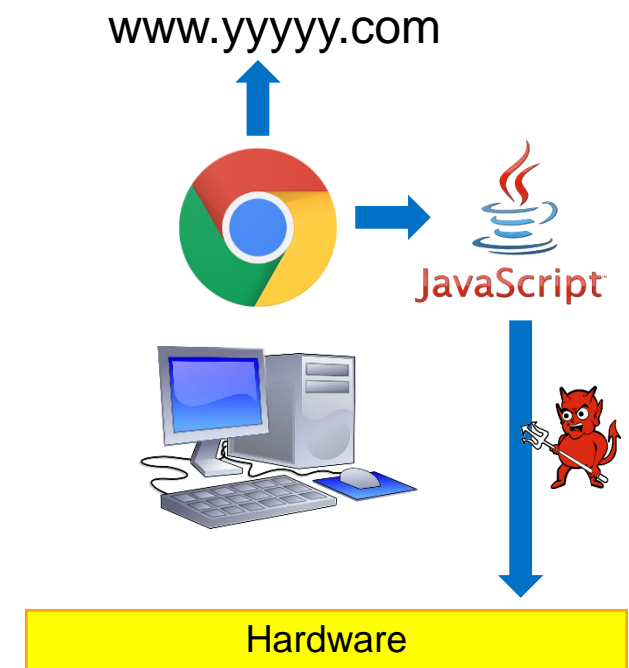
Example: Incognito browsing profiling [OREN15]

Pros:

- No need to find co-resident target
- Attack executed in local machine

Cons:

- Flush and Reload can not be applied



Malicious Javascript Execution

Attacker introduces cache attack containing javascript code into target website

Victim access the target website, and her browser executes the javascript code in the local machines

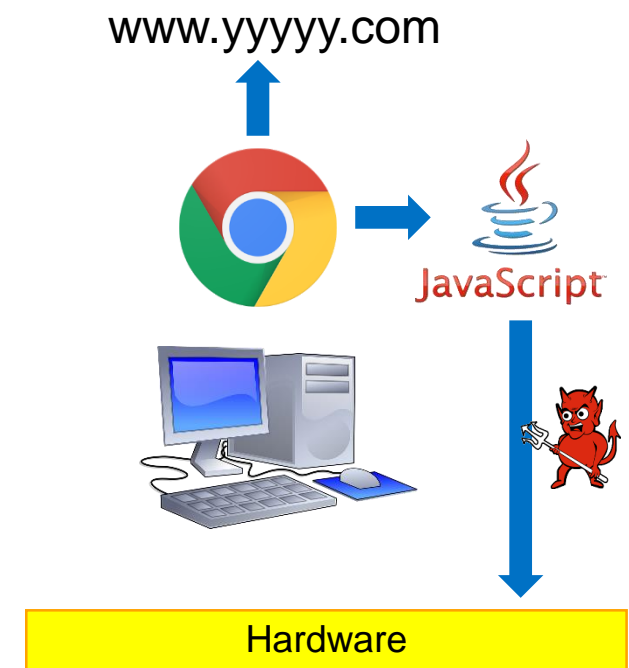
Example: Incognito browsing profiling [OREN15]

Pros:

- No need to find co-resident target
- Attack executed in local machine

Cons:

- Flush and Reload can not be applied
- Fine grain timers hard to achieve



Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

But both trusted and untrusted environments access same hardware caches!

Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

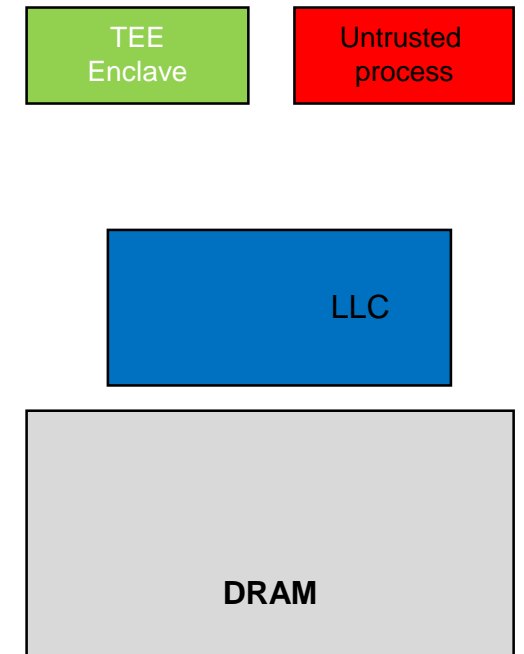
But both trusted and untrusted environments access same hardware caches!



Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

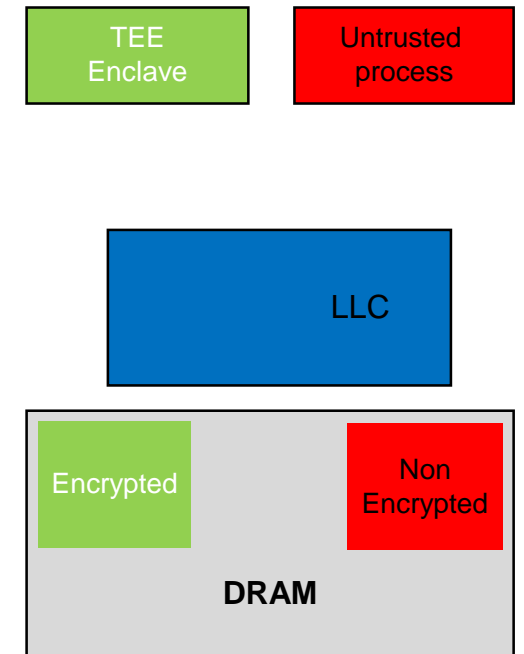
But both trusted and untrusted environments access same hardware caches!



Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

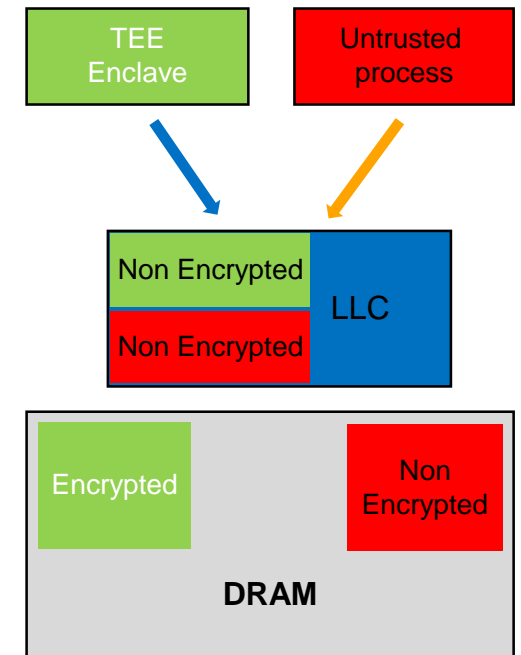
But both trusted and untrusted environments access same hardware caches!



Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

But both trusted and untrusted environments access same hardware caches!

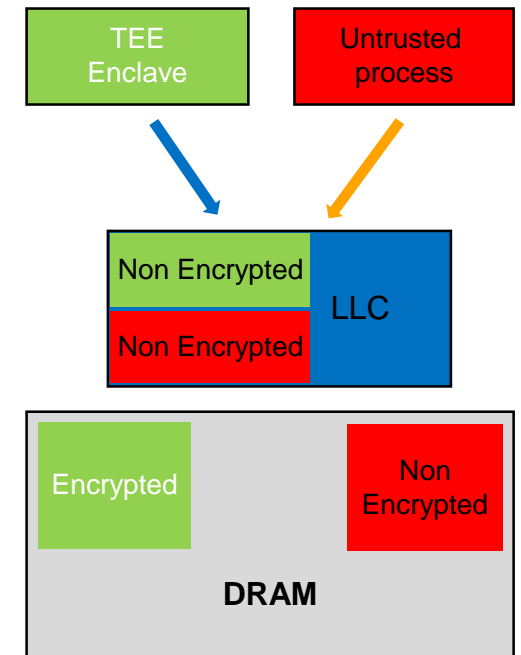


Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

But both trusted and untrusted environments access same hardware caches!

Example: TrustZone AES key steal [BRM15]



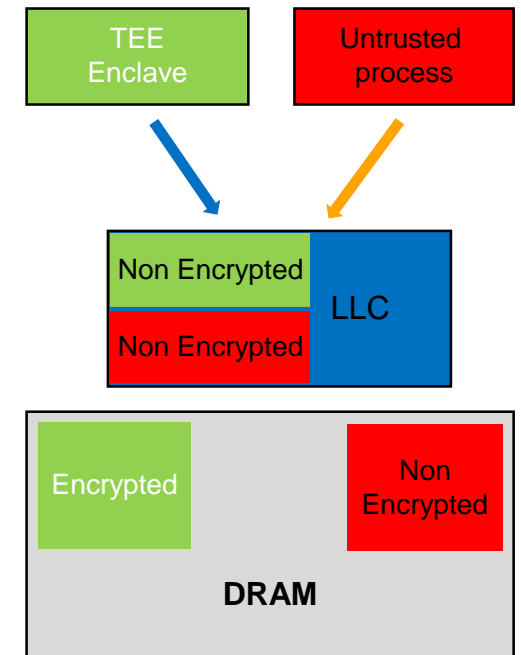
Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

But both trusted and untrusted environments access same hardware caches!

Example: TrustZone AES key steal [BRM15]

Pros:



Trusted Execution Environments

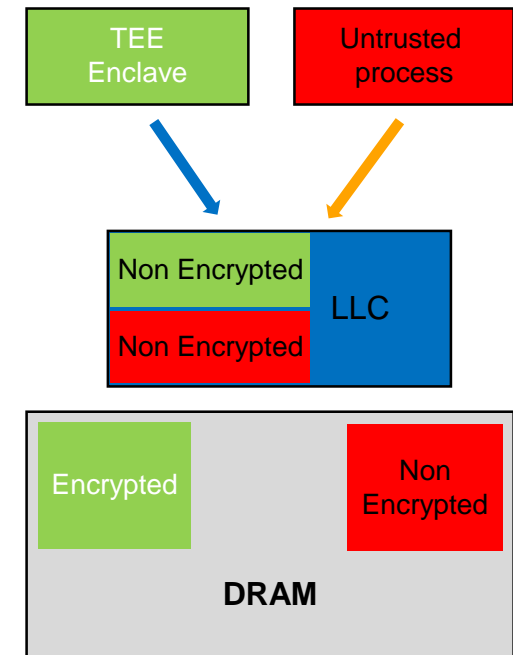
Trusted execution environments designed to achieve isolation from untrusted processes

But both trusted and untrusted environments access same hardware caches!

Example: TrustZone AES key steal [BRM15]

Pros:

- Higher resolution: access to OS fine grain resources (including scheduling)



Trusted Execution Environments

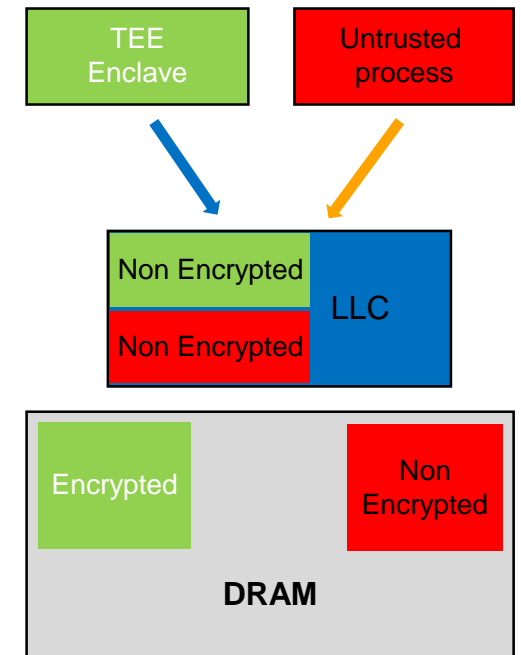
Trusted execution environments designed to achieve isolation from untrusted processes

But both trusted and untrusted environments access same hardware caches!

Example: TrustZone AES key steal [BRM15]

Pros:

- Higher resolution: access to OS fine grain resources (including scheduling)
- No need to find co-resident target



Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

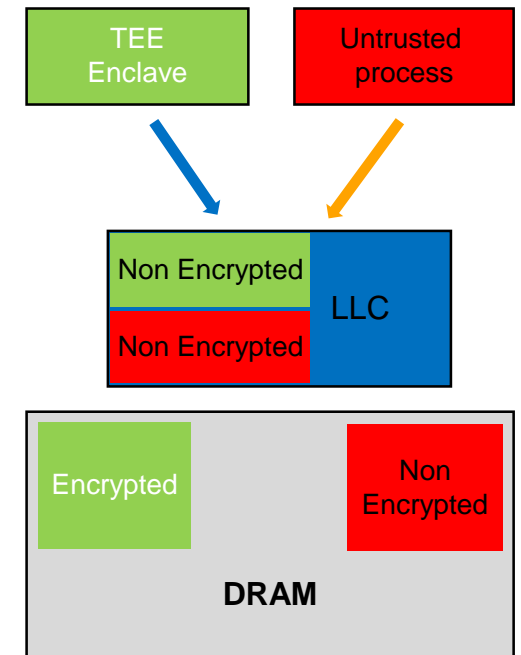
But both trusted and untrusted environments access same hardware caches!

Example: TrustZone AES key steal [BRM15]

Pros:

- Higher resolution: access to OS fine grain resources (including scheduling)
- No need to find co-resident target

Cons:



Trusted Execution Environments

Trusted execution environments designed to achieve isolation from untrusted processes

But both trusted and untrusted environments access same hardware caches!

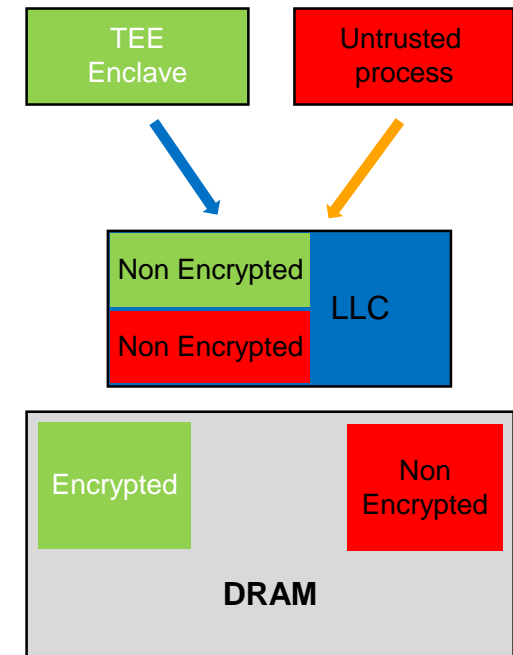
Example: TrustZone AES key steal [BRM15]

Pros:

- Higher resolution: access to OS fine grain resources (including scheduling)
- No need to find co-resident target

Cons:

- Flush and Reload not applicable (deduplication disabled)



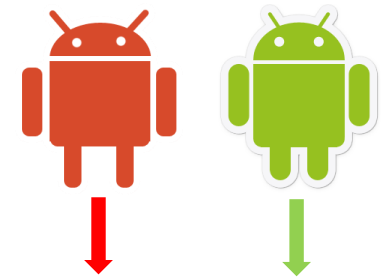
Malicious Smartphones Applications

Smartphone applications are properly isolated by the OS in the software side

Malicious Smartphones Applications

Smartphone applications are properly isolated by the OS in the software side

However, as with TEEs, all applications utilize the hardware caches



Malicious Smartphones Applications

Smartphone applications are properly isolated by the OS in the software side

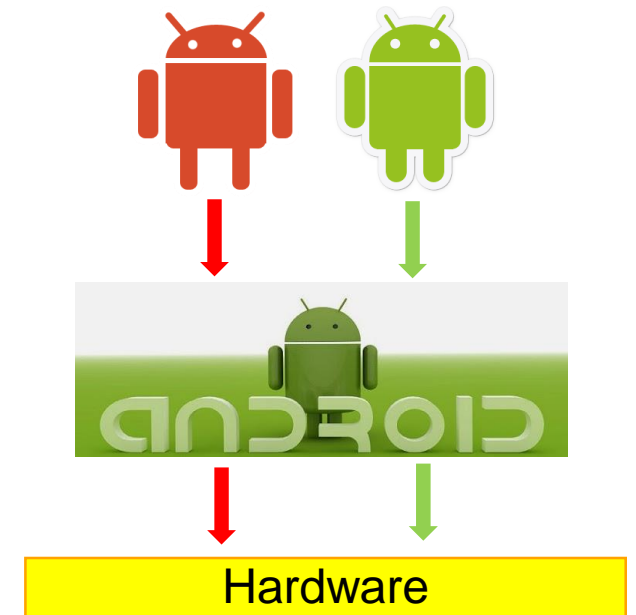
However, as with TEEs, all applications utilize the hardware caches



Malicious Smartphones Applications

Smartphone applications are properly isolated by the OS in the software side

However, as with TEEs, all applications utilize the hardware caches

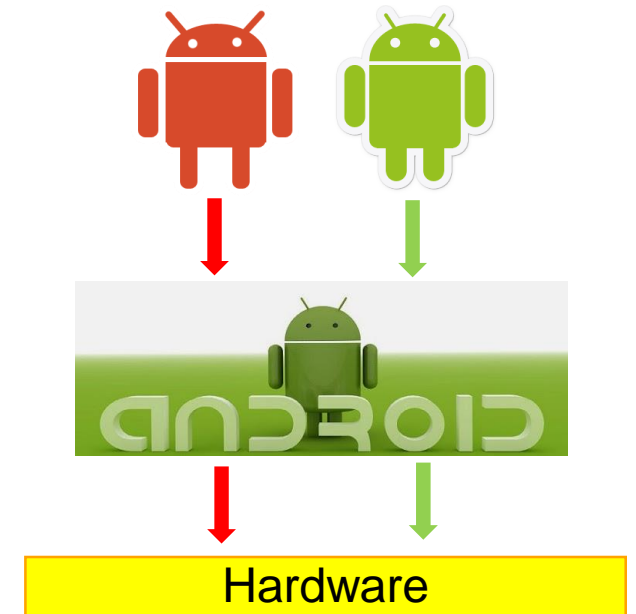


Malicious Smartphones Applications

Smartphone applications are properly isolated by the OS in the software side

However, as with TEEs, all applications utilize the hardware caches

Example: AES key steal across apps [LIPP16]



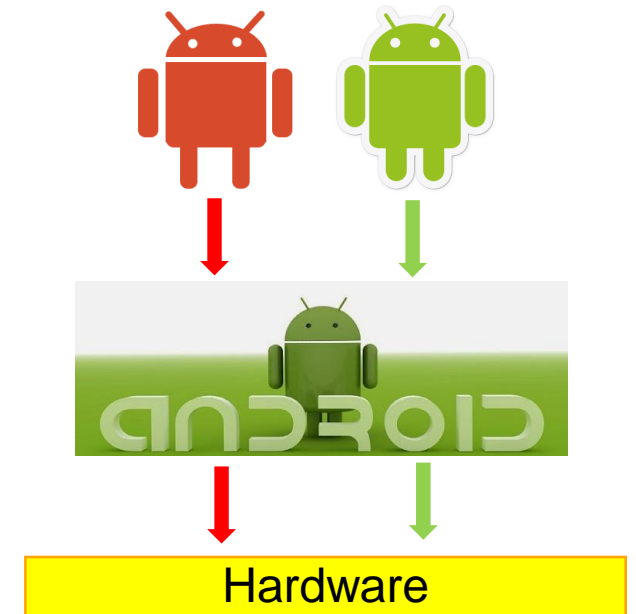
Malicious Smartphones Applications

Smartphone applications are properly isolated by the OS in the software side

However, as with TEEs, all applications utilize the hardware caches

Example: AES key steal across apps [LIPP16]

Pros:



Malicious Smartphones Applications

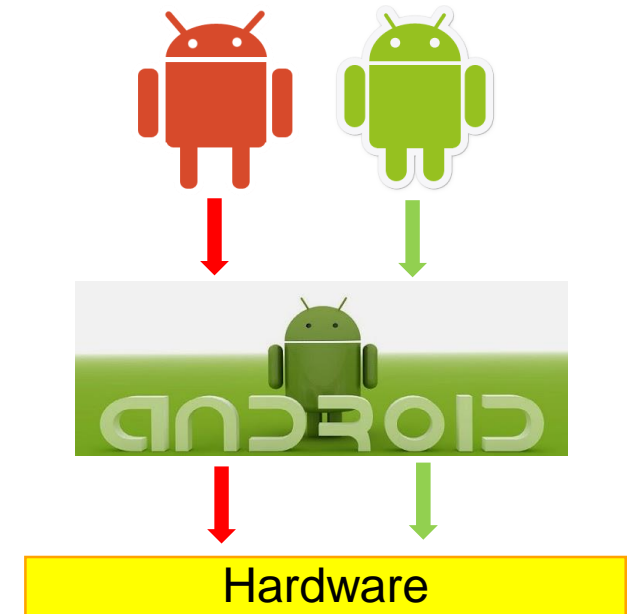
Smartphone applications are properly isolated by the OS in the software side

However, as with TEEs, all applications utilize the hardware caches

Example: AES key steal across apps [LIPP16]

Pros:

- Deduplication is generally used (e.g. Android)



Malicious Smartphones Applications

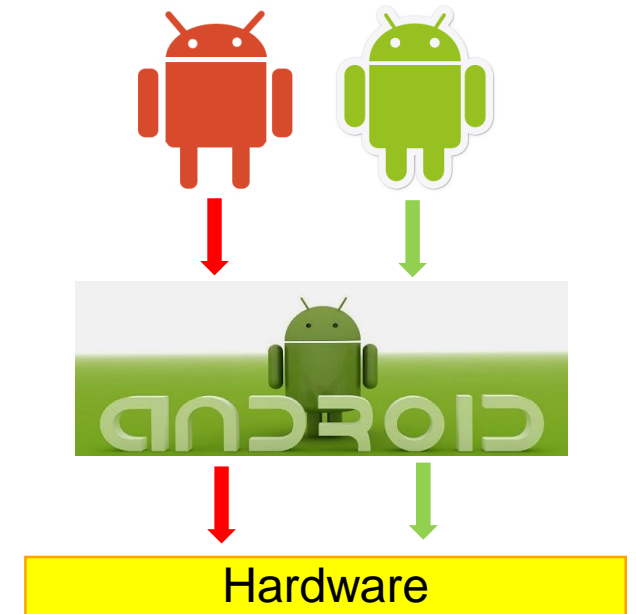
Smartphone applications are properly isolated by the OS in the software side

However, as with TEEs, all applications utilize the hardware caches

Example: AES key steal across apps [LIPP16]

Pros:

- Deduplication is generally used (e.g. Android)
- Easy deployment



Malicious Smartphones Applications

Smartphone applications are properly isolated by the OS in the software side

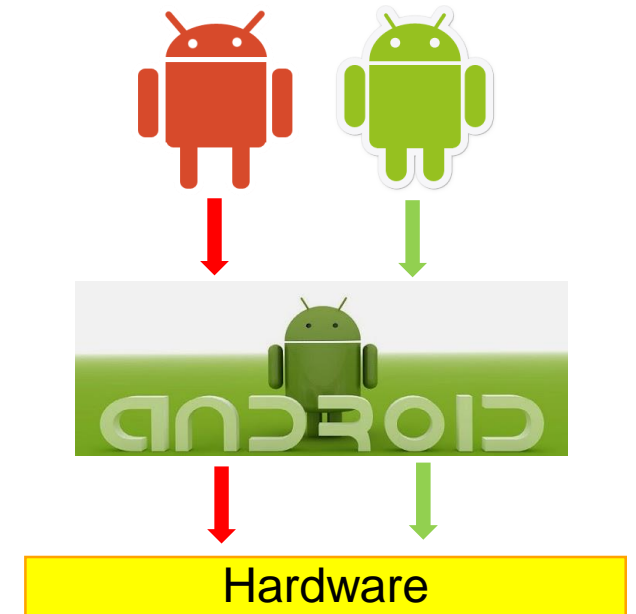
However, as with TEEs, all applications utilize the hardware caches

Example: AES key steal across apps [LIPP16]

Pros:

- Deduplication is generally used (e.g. Android)
- Easy deployment

Cons:



Malicious Smartphones Applications

Smartphone applications are properly isolated by the OS in the software side

However, as with TEEs, all applications utilize the hardware caches

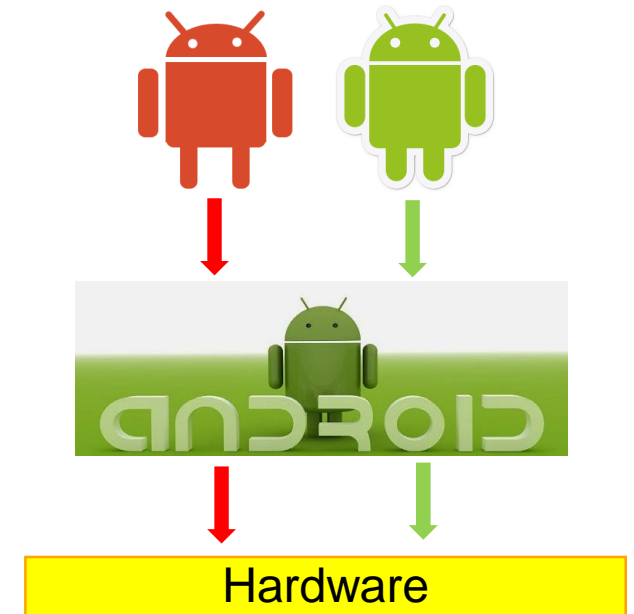
Example: AES key steal across apps [LIPP16]

Pros:

- Deduplication is generally used (e.g. Android)
- Easy deployment

Cons:

- Device dependent (e.g., non-inclusive cache)



How can we mitigate cache attacks?

Cache Leakage Free Code Design

Goals:

```
1 function modpow ( $a, b$ );  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R[0] = 1; R[1] = b;$   
3 for  $i = k - 1$  downto 0 do  
4    $R[0] * e_i + R[1] * \hat{e}_i = R[0] * R[1] \bmod N;$   
5    $R[1] * e_i + R[0] * \hat{e}_i =$   
      $R[1] * R[1] * e_i + R[0] * R[0] * \hat{e}_i \bmod N;$   
6 end  
7 return  $R[0];$ 
```

Cache Leakage Free Code Design

Goals:

- Secret independent execution flow

```
1 function modpow (a, b);  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R[0] = 1; R[1] = b;$   
3 for  $i = k - 1$  downto 0 do  
4    $R[0] * e_i + R[1] * \hat{e}_i = R[0] * R[1] \bmod N;$   
5    $R[1] * e_i + R[0] * \hat{e}_i =$   
      $R[1] * R[1] * e_i + R[0] * R[0] * \hat{e}_i \bmod N;$   
6 end  
7 return  $R[0];$ 
```

Cache Leakage Free Code Design

Goals:

- Secret independent execution flow

```
1 function modpow (a, b);  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R[0] = 1; R[1] = b;$   
3 for  $i = k - 1$  downto 0 do  
4    $R[0] * e_i + R[1] * \hat{e}_i = R[0] * R[1] \bmod N;$   
5    $R[1] * e_i + R[0] * \hat{e}_i =$   
6      $R[1] * R[1] * e_i + R[0] * R[0] * \hat{e}_i \bmod N;$   
7 end  
8 return  $R[0];$ 
```

Cache Leakage Free Code Design

Goals:

- Secret independent execution flow
- Secret independent memory accesses

```
1 function modpow (a, b);  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R[0] = 1; R[1] = b;$   
3 for  $i = k - 1$  downto 0 do  
4    $R[0] * e_i + R[1] * \hat{e}_i = R[0] * R[1] \bmod N;$   
5    $R[1] * e_i + R[0] * \hat{e}_i =$   
6      $R[1] * R[1] * e_i + R[0] * R[0] * \hat{e}_i \bmod N;$   
7 end  
8 return  $R[0];$ 
```

Cache Leakage Free Code Design

Goals:

- Secret independent execution flow
- Secret independent memory accesses

```
1 function modpow (a, b);  
   Input : base  $b$ , modulus  $N$ , secret  
            $E = (e_{k-1}, \dots, e_1, e_0)$   
   Output:  $b^E \bmod N$   
2  $R[0] = 1; R[1] = b;$   
3 for  $i = k - 1$  downto 0 do  
4   |  $R[0] * e_i + R[1] * \hat{e}_i = R[0] * R[1] \bmod N;$   
5   |  $R[1] * e_i + R[0] * \hat{e}_i =$   
   |  $R[1] * R[1] * e_i + R[0] * R[0] * \hat{e}_i \bmod N;$   
6 end  
7 return  $R[0];$ 
```

Avoiding Collisions in the LLC

Avoiding Collisions in the LLC

Approaches:

Avoiding Collisions in the LLC

Approaches:

- Page coloring

Avoiding Collisions in the LLC

Approaches:

- Page coloring

Page Coloring

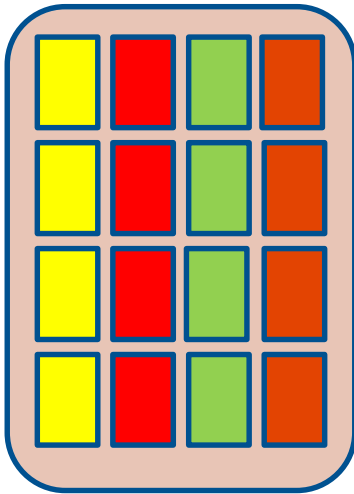
Avoiding Collisions in the LLC

Approaches:

- Page coloring

Page Coloring

DRAM

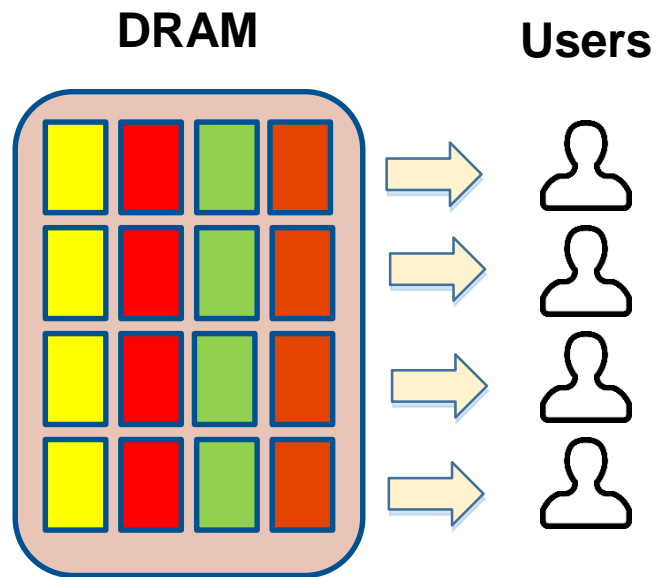


Avoiding Collisions in the LLC

Approaches:

- Page coloring

Page Coloring

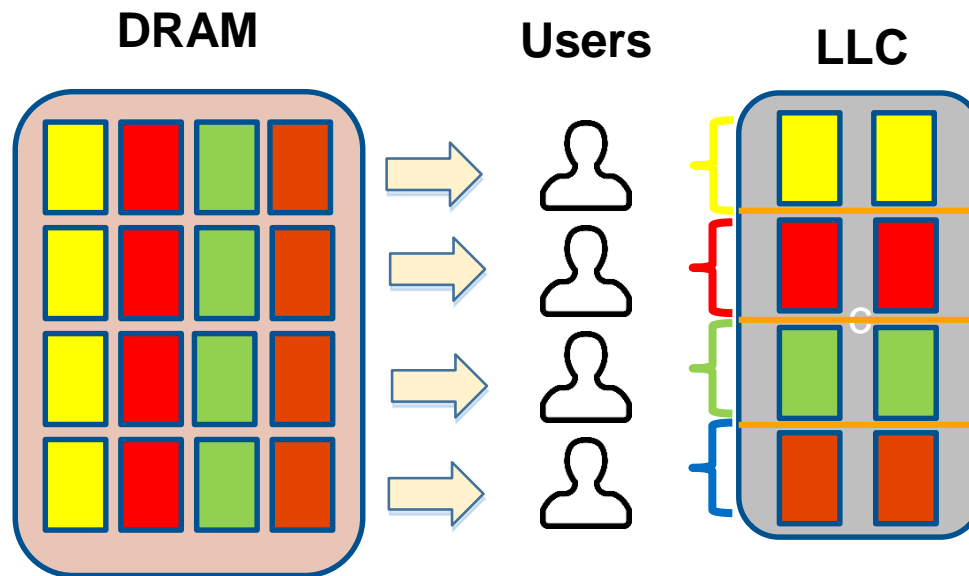


Avoiding Collisions in the LLC

Approaches:

- Page coloring

Page Coloring

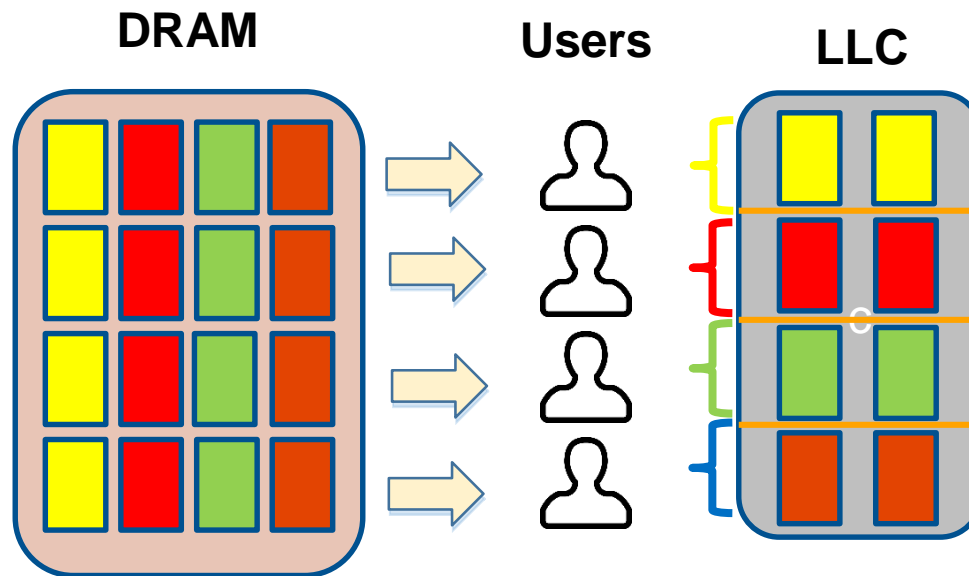


Avoiding Collisions in the LLC

Approaches:

- Page coloring
- Intel CAT technology

Page Coloring

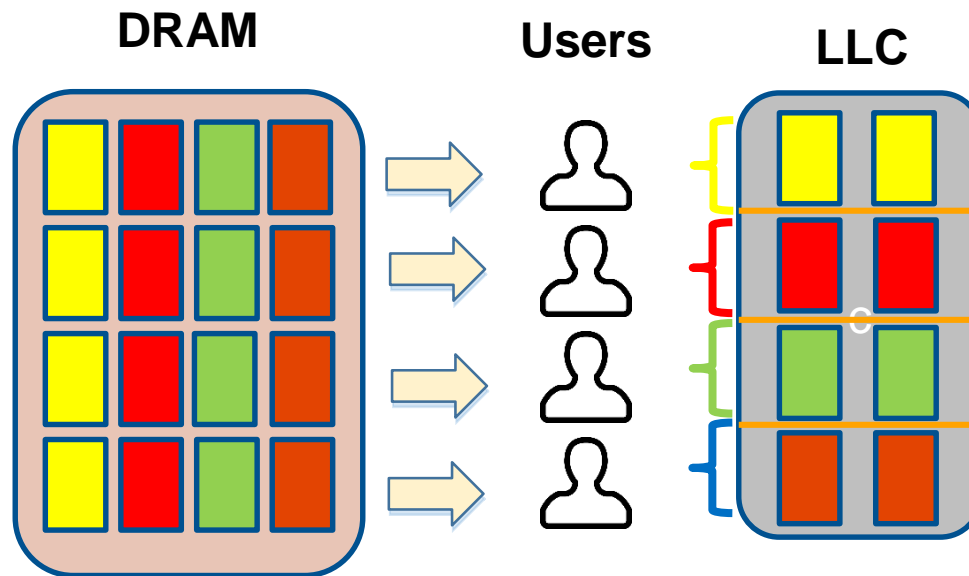


Avoiding Collisions in the LLC

Approaches:

- Page coloring
- Intel CAT technology

Page Coloring



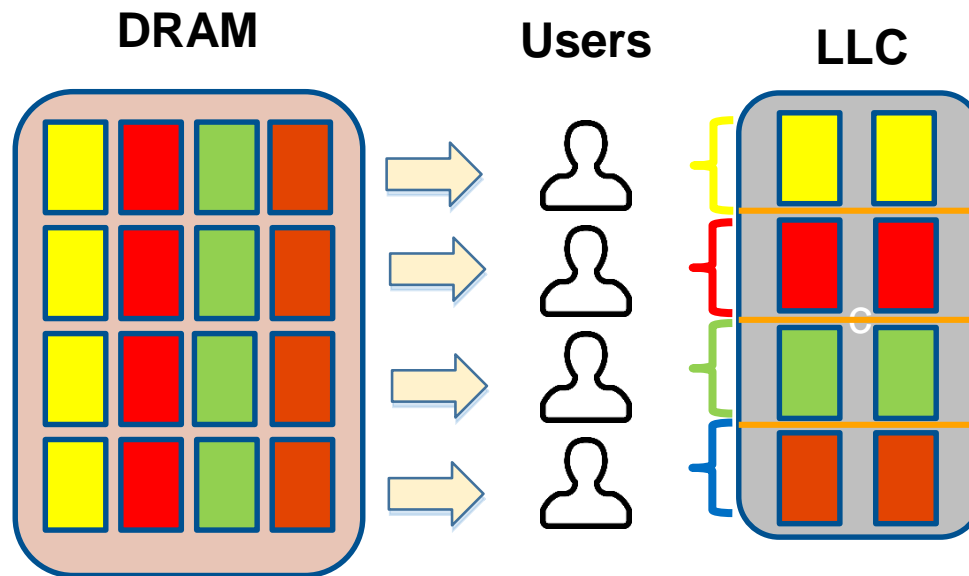
Intel CAT technology

Avoiding Collisions in the LLC

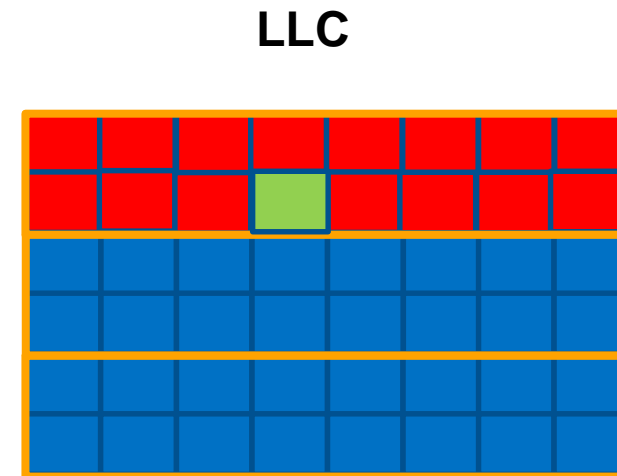
Approaches:

- Page coloring
- Intel CAT technology

Page Coloring



Intel CAT technology

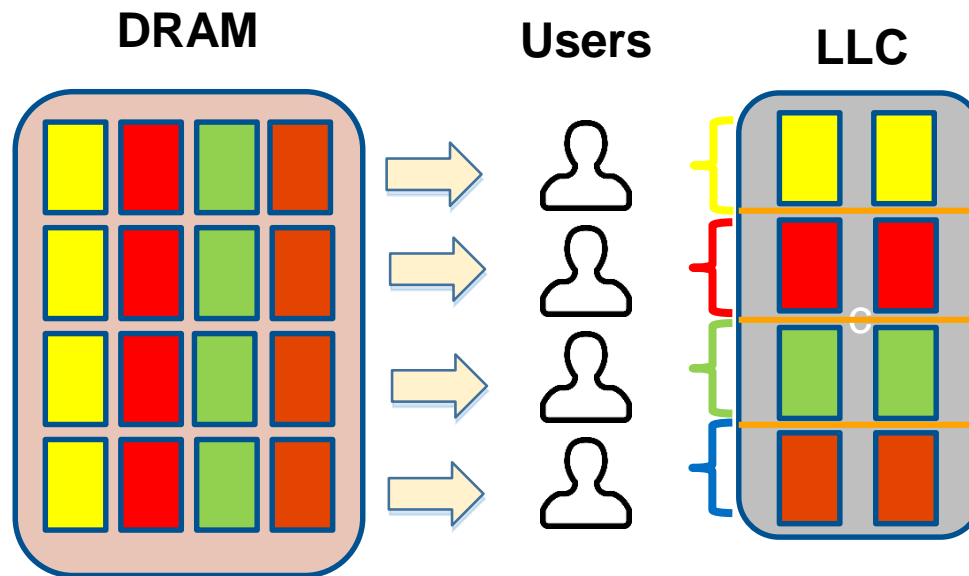


Avoiding Collisions in the LLC

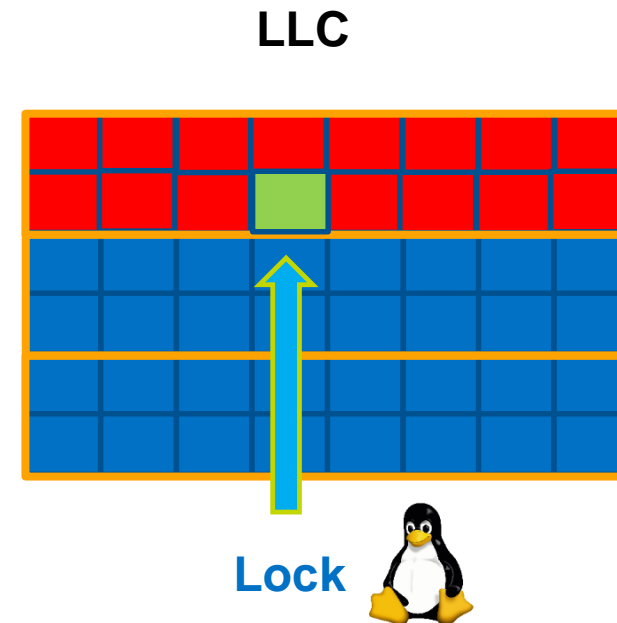
Approaches:

- Page coloring
- Intel CAT technology

Page Coloring



Intel CAT technology

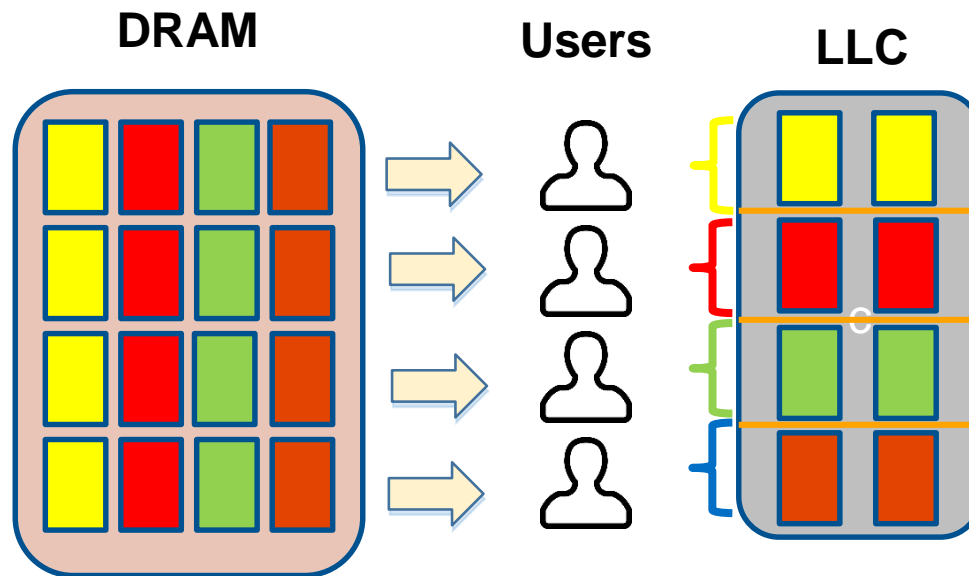


Avoiding Collisions in the LLC

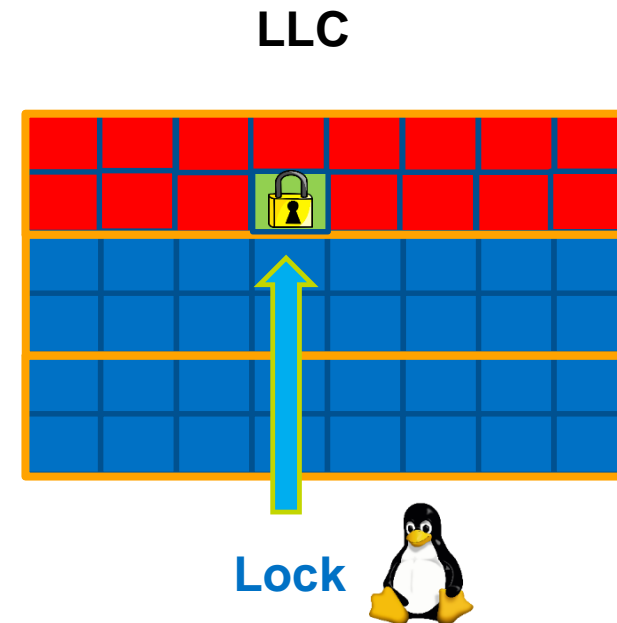
Approaches:

- Page coloring
- Intel CAT technology

Page Coloring



Intel CAT technology



Key Takeaways

Cache attacks are already practical!

Key Takeaways

Cache attacks are already practical!

IaaS/PaaS, web browsers, smartphones.. What else?

Key Takeaways

Cache attacks are already practical!

IaaS/PaaS, web browsers, smartphones.. What else?

Catching attention from many researchers: trend shows more practicality and applicability expected

Key Takeaways

Cache attacks are already practical!

IaaS/PaaS, web browsers, smartphones.. What else?

Catching attention from many researchers: trend shows more practicality and applicability expected

CALL TO ACTION:

Key Takeaways

Cache attacks are already practical!

IaaS/PaaS, web browsers, smartphones.. What else?

Catching attention from many researchers: trend shows more practicality and applicability expected

CALL TO ACTION:

- For software designers: introduce cache leakage free code design habits!

Key Takeaways

Cache attacks are already practical!

IaaS/PaaS, web browsers, smartphones.. What else?

Catching attention from many researchers: trend shows more practicality and applicability expected

CALL TO ACTION:

- For software designers: introduce cache leakage free code design habits!
- For hypervisor/OS designers: software countermeasures and hardware framework ready to use. Use it!