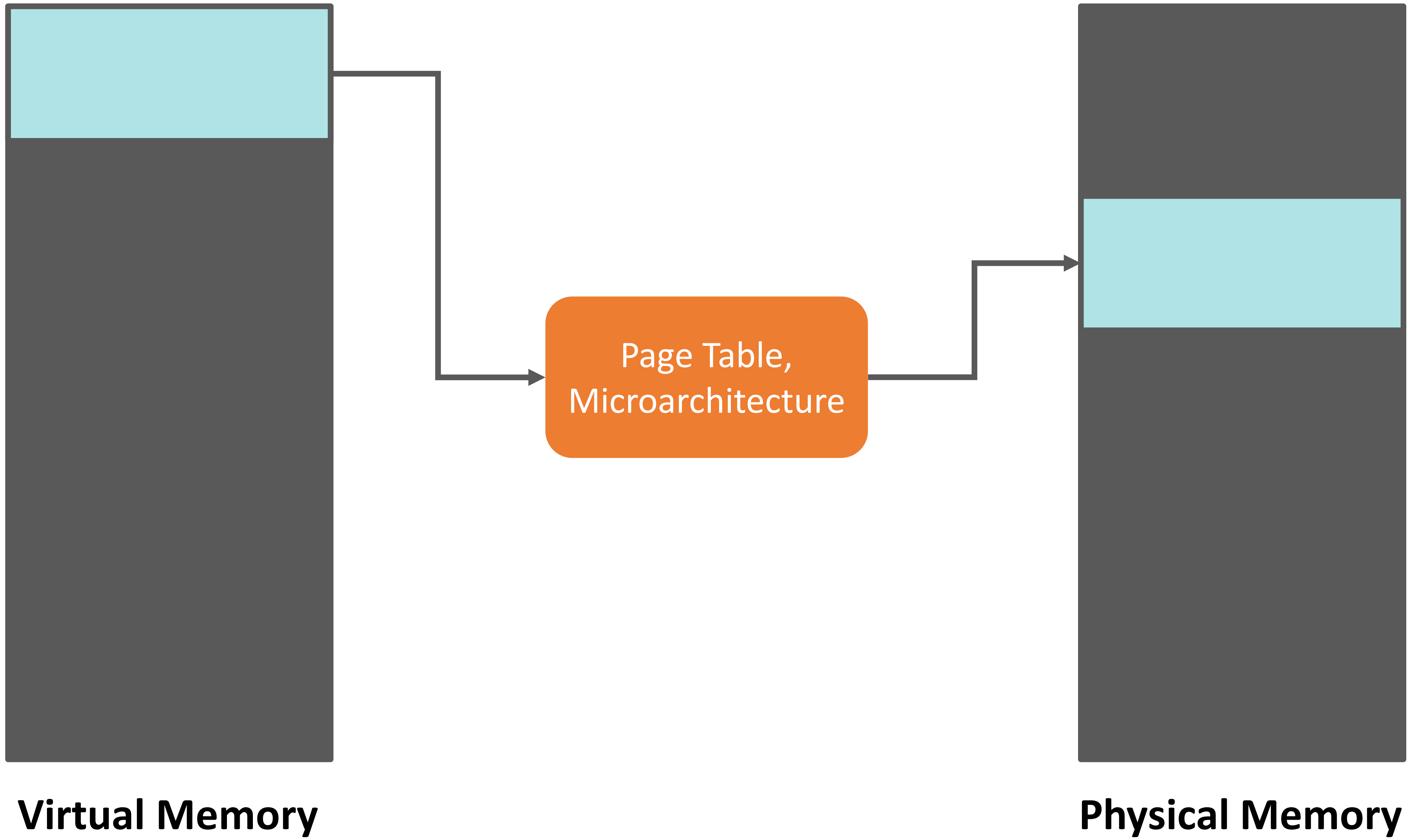
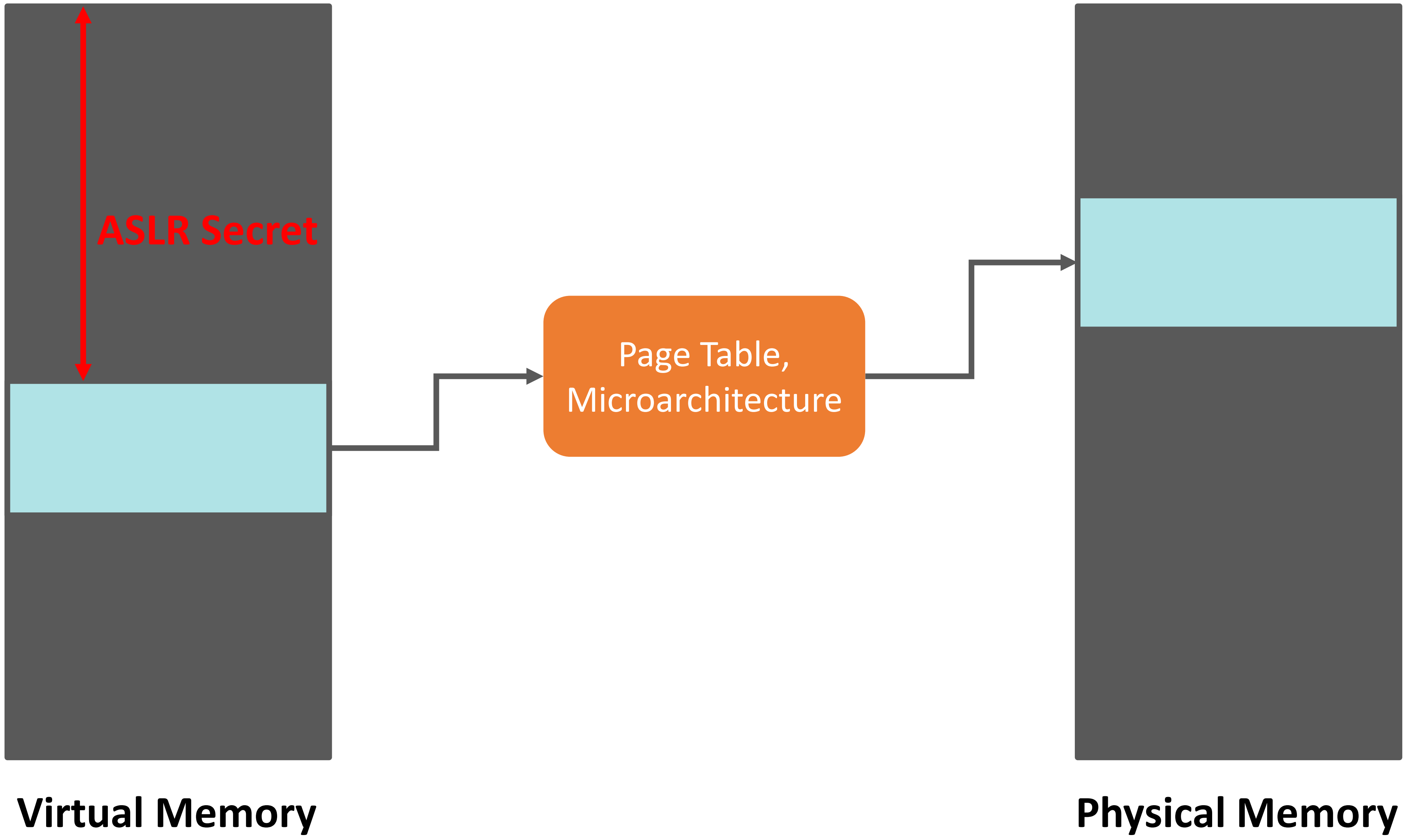
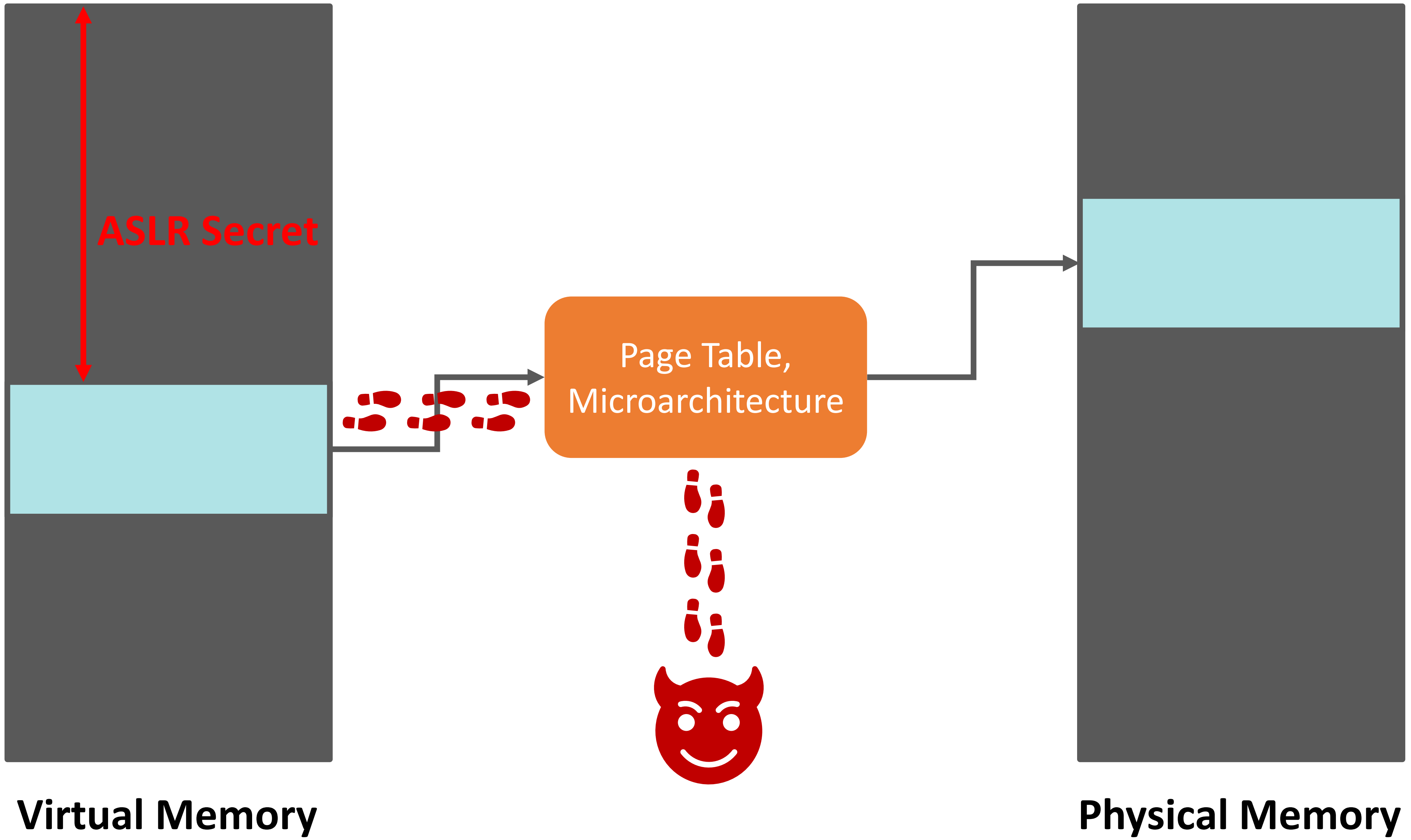
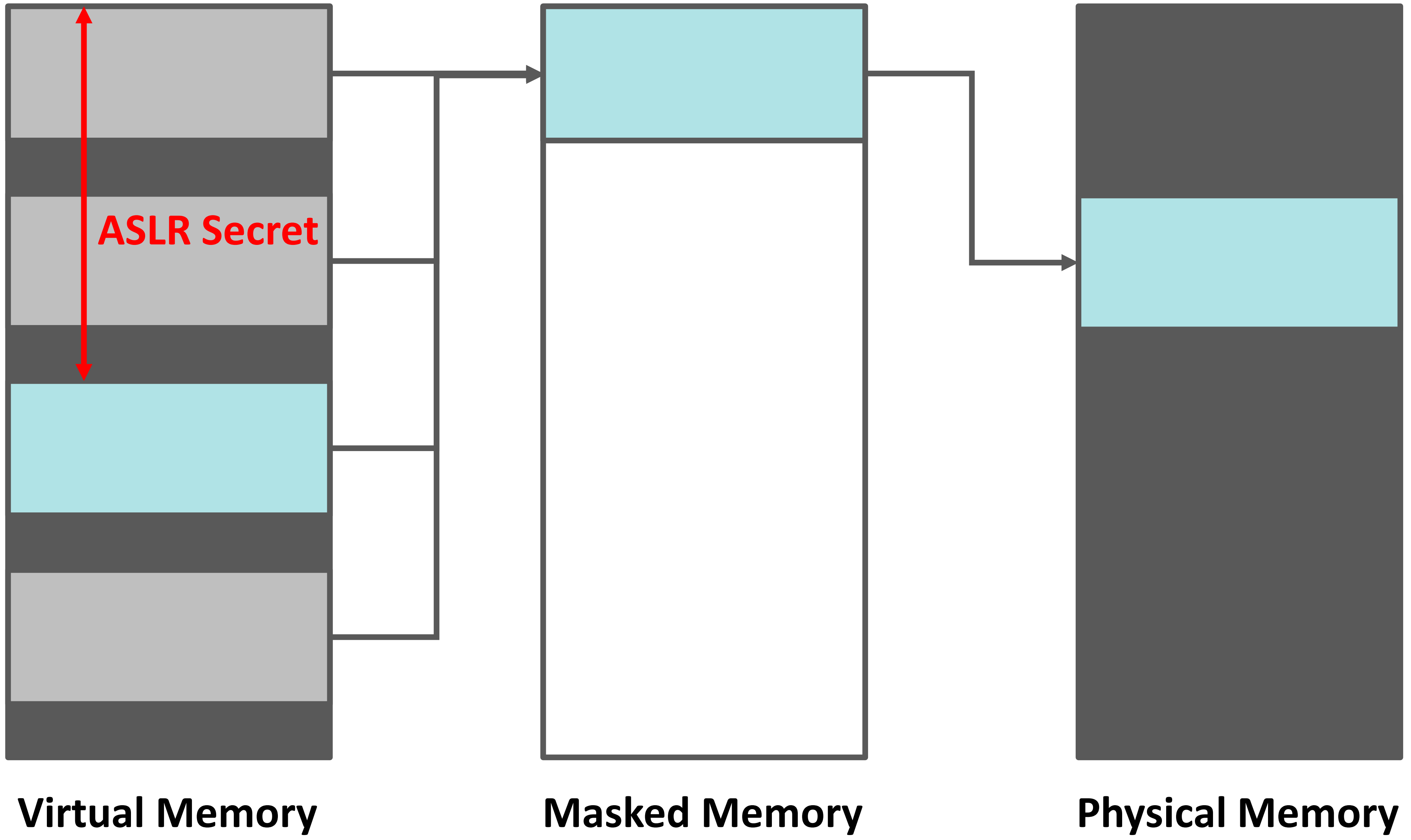
Oreo: Protecting ASLR Against Microarchitectural Attacks

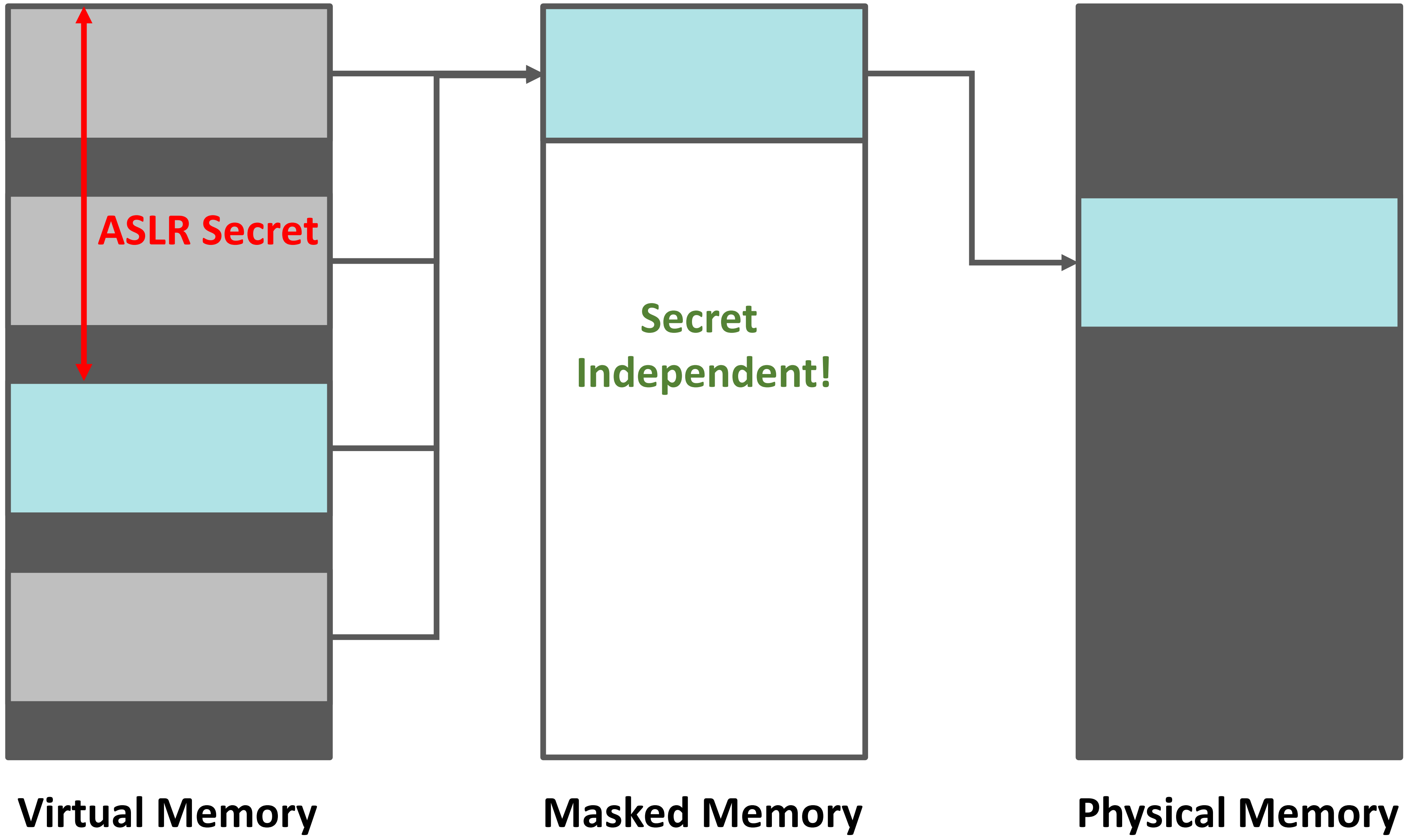
Shixin Song, Joseph Zhang, Mengjia Yan (MIT)


Page Table,
Microarchitecture
Virtual Memory
Physical Memory

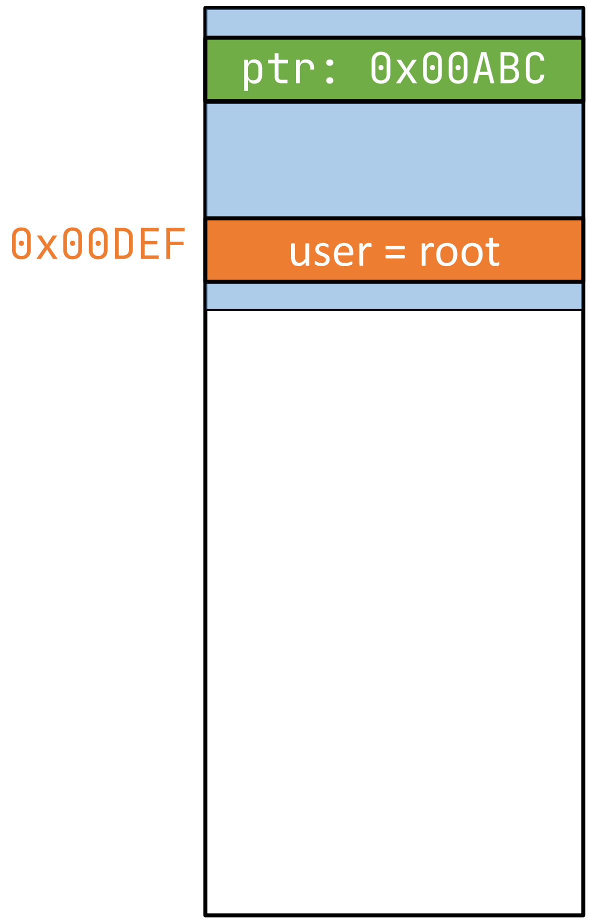

ASLR Secret
Page Table,
Microarchitecture
Virtual Memory
Physical Memory


ASLR Secret
Page Table, Microarchitecture
Virtual Memory
Physical Memory

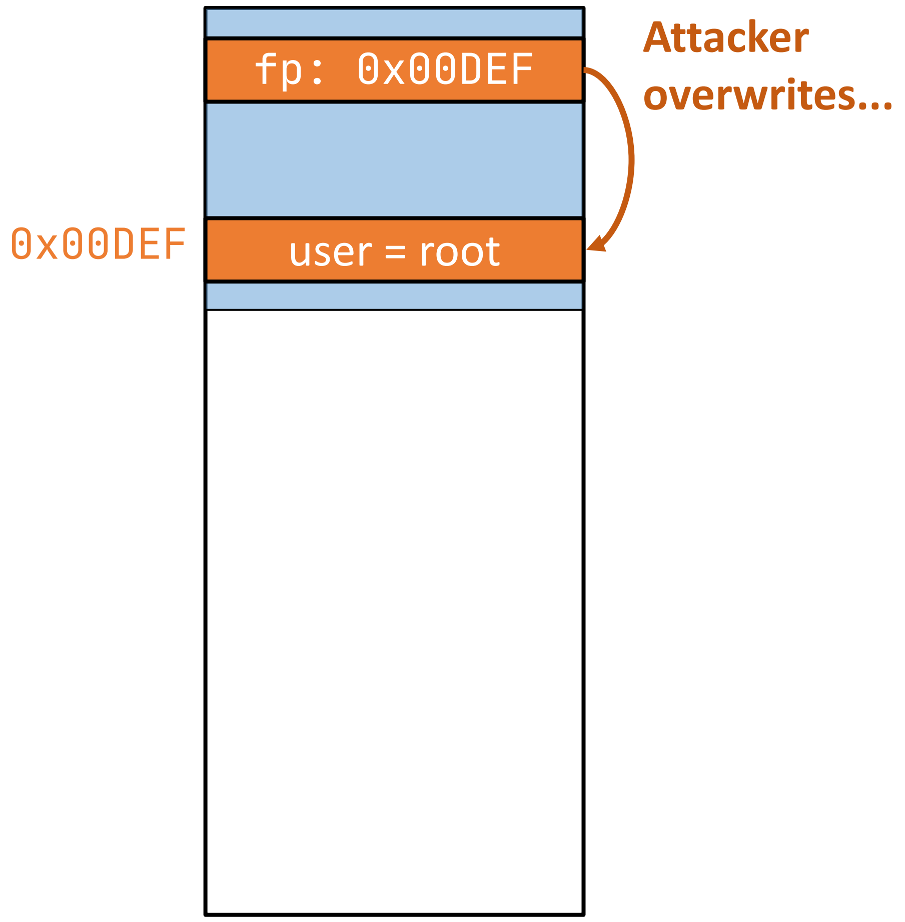

ASLR Secret
Virtual Memory
Masked Memory
Physical Memory


ASLR Secret
Secret Independent!
Virtual Memory
Masked Memory
Physical Memory

Address Space Layout Randomization (ASLR)

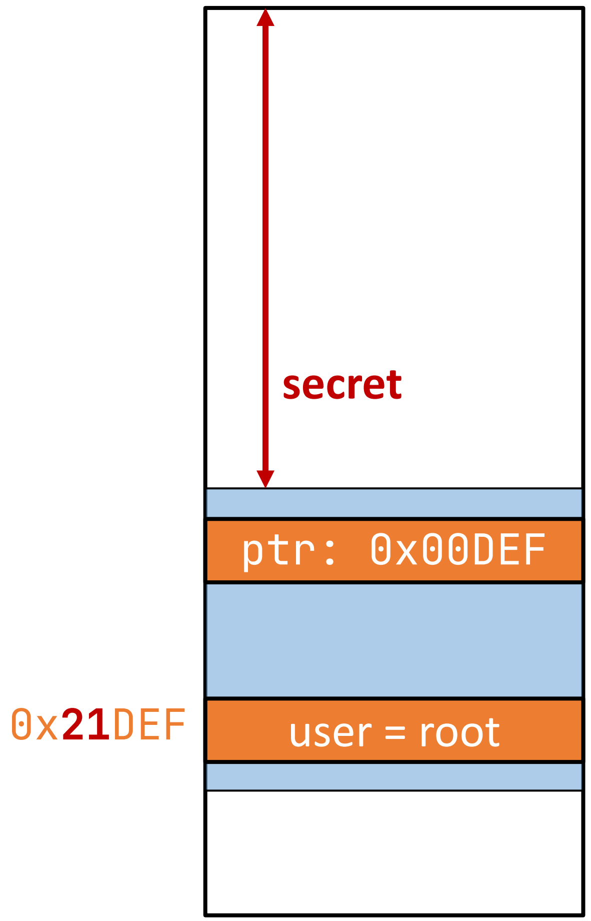


Address Space Layout Randomization (ASLR)



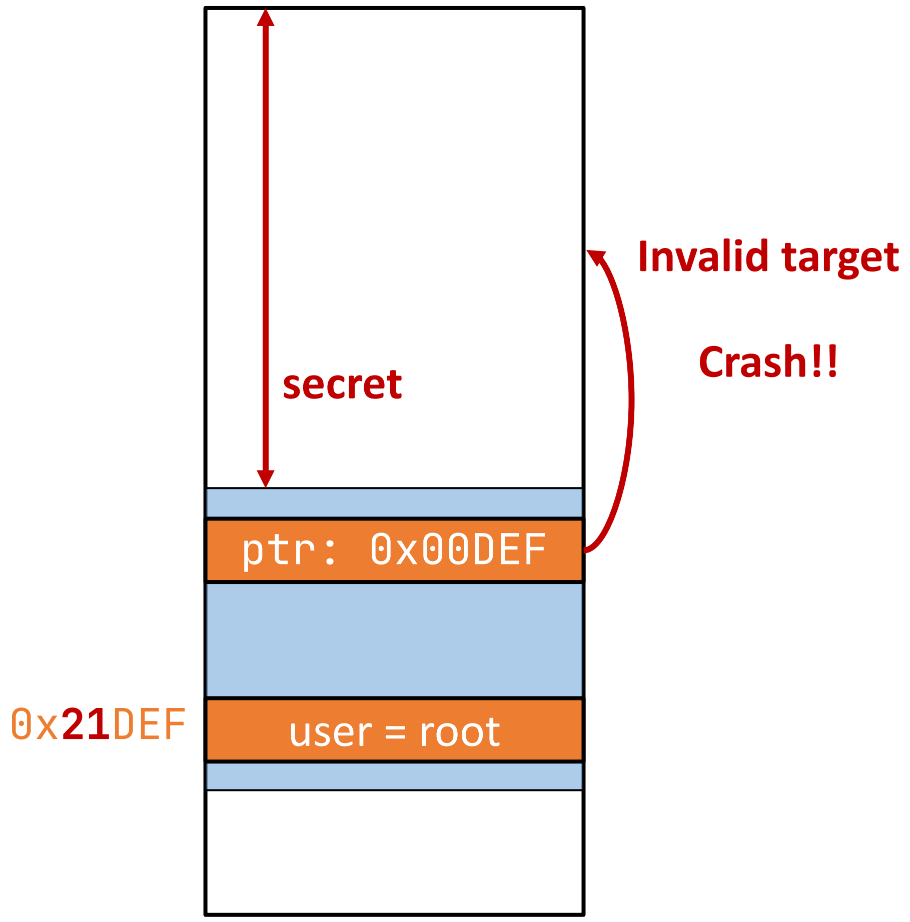
Address Space Layout Randomization (ASLR)

- ASLR is to relocate victim code with a randomized offset.



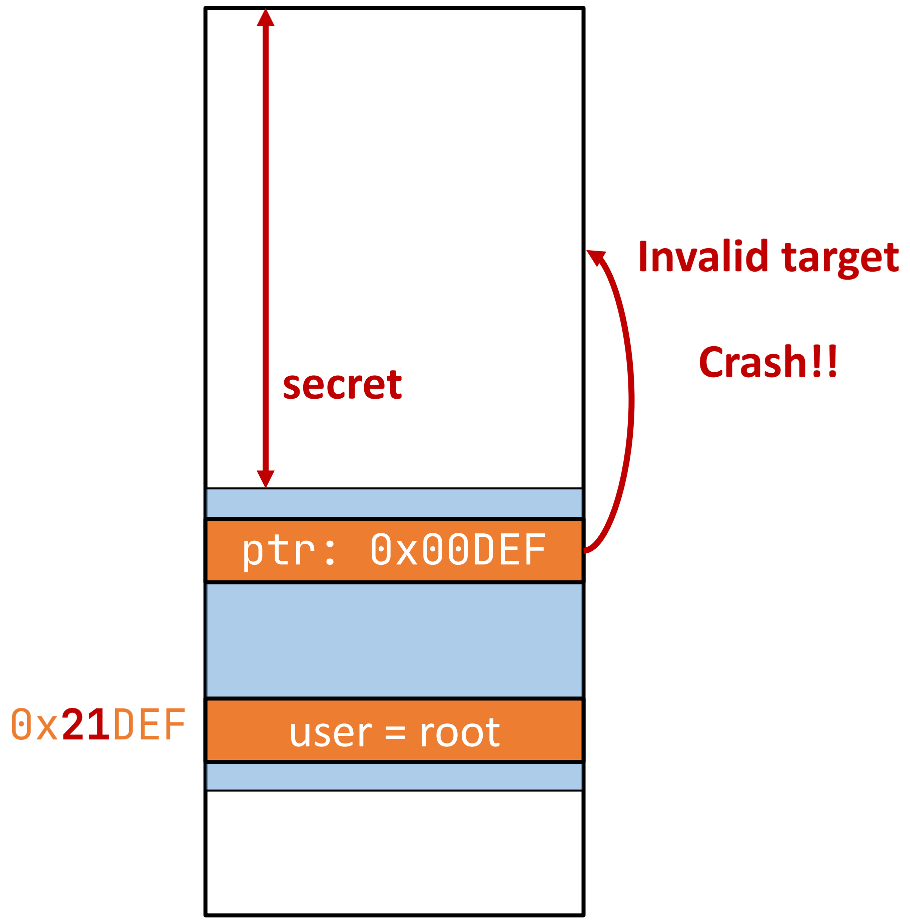
Address Space Layout Randomization (ASLR)

- ASLR is to relocate victim code with a randomized offset.



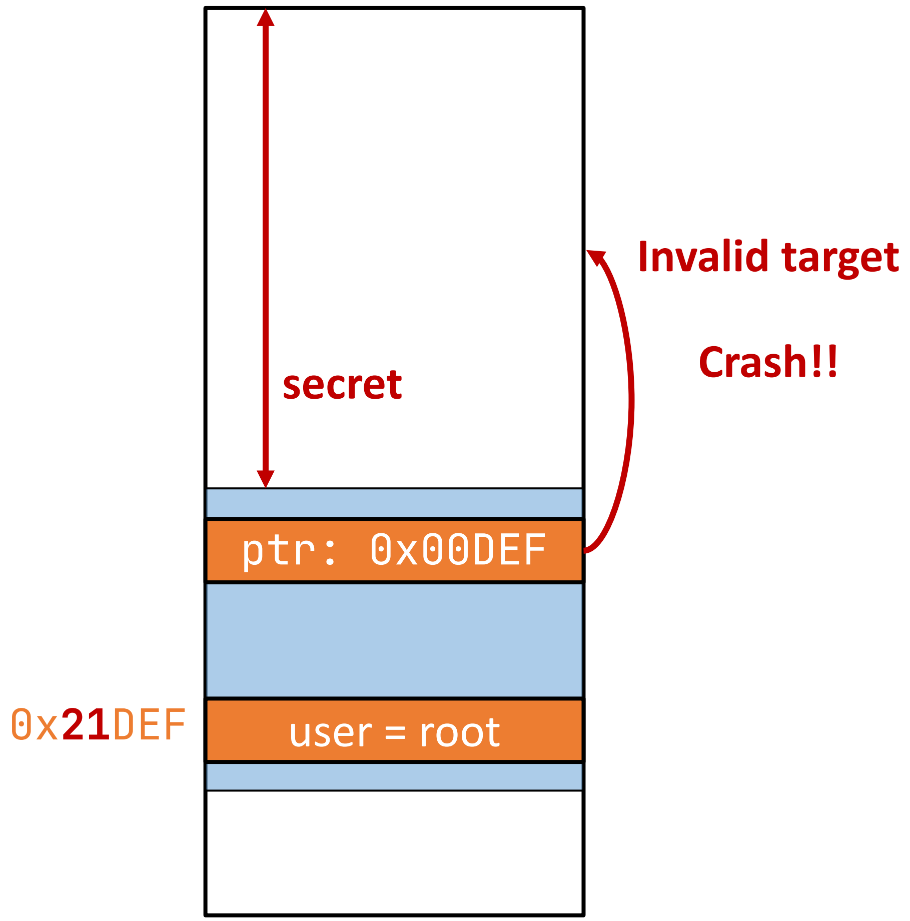
Address Space Layout Randomization (ASLR)

- ASLR is to relocate victim code with a randomized offset.
- Code reuse attacks need to perform an extra step to bypass ASLR.



Address Space Layout Randomization (ASLR)

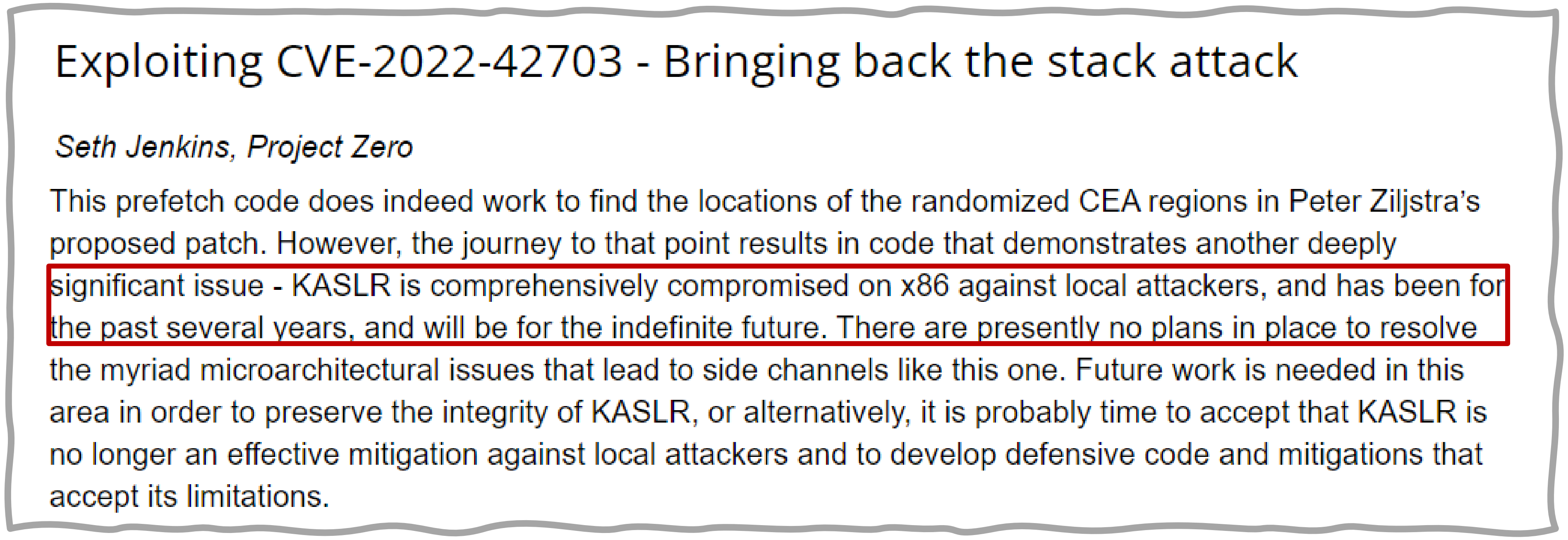
- ASLR is to relocate victim code with a randomized offset.
- Code reuse attacks need to perform an extra step to bypass ASLR.
- ASLR is widely deployed in modern systems:
  - e.g., Linux, Windows, macOS



However, bypassing ASLR becomes extremely easy with microarchitectural attacks.

Osiris
Jump Over ASLR
Spectre Probing
Code Region Probing
Binoculars
DrK
Phantom
AnC
Fallout
TLBleed
Double Page Fault
Take A Way
Data Bounce
EntryBleed
TagBleed
Prefetch Attack
EchoLoad

However, bypassing ASLR becomes extremely easy with microarchitectural attacks.

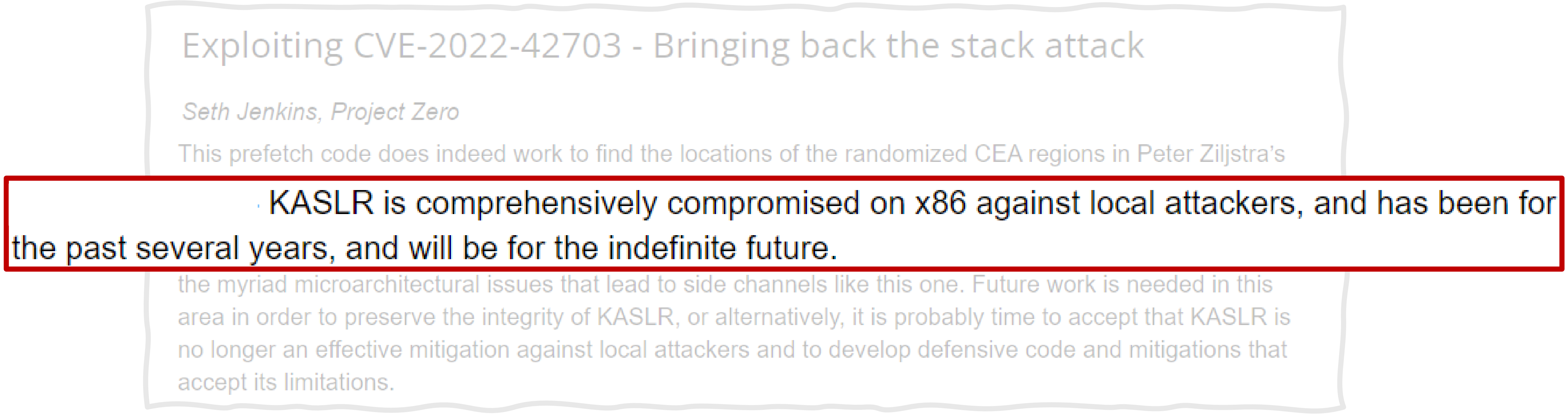


Exploiting CVE-2022-42703 - Bringing back the stack attack

Seth Jenkins, Project Zero

This prefetch code does indeed work to find the locations of the randomized CEA regions in Peter Ziljstra's proposed patch. However, the journey to that point results in code that demonstrates another deeply significant issue - KASLR is comprehensively compromised on x86 against local attackers, and has been for the past several years, and will be for the indefinite future. There are presently no plans in place to resolve the myriad microarchitectural issues that lead to side channels like this one. Future work is needed in this area in order to preserve the integrity of KASLR, or alternatively, it is probably time to accept that KASLR is no longer an effective mitigation against local attackers and to develop defensive code and mitigations that accept its limitations.

However, bypassing ASLR becomes extremely easy with microarchitectural attacks.



Exploiting CVE-2022-42703 - Bringing back the stack attack

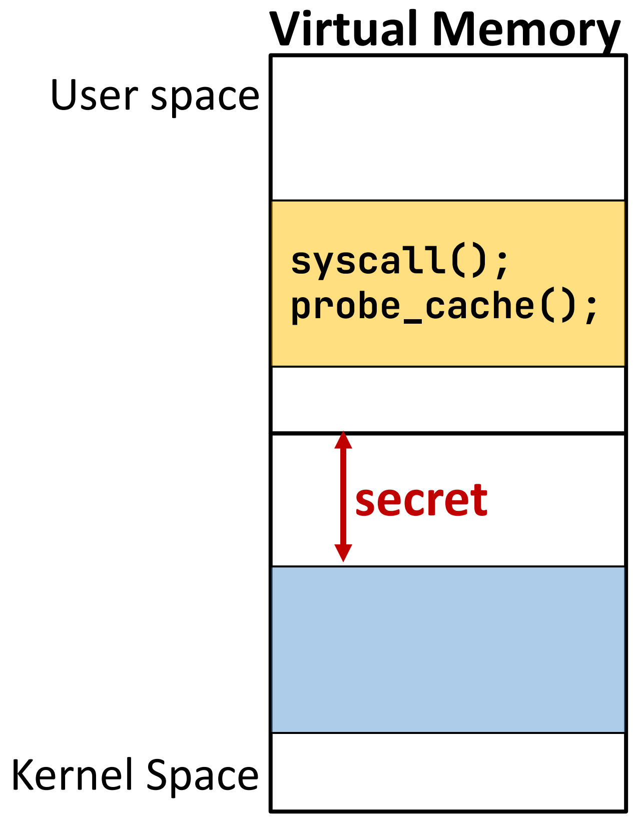
Seth Jenkins, Project Zero

This prefetch code does indeed work to find the locations of the randomized CEA regions in Peter Ziljstra's

KASLR is comprehensively compromised on x86 against local attackers, and has been for the past several years, and will be for the indefinite future.

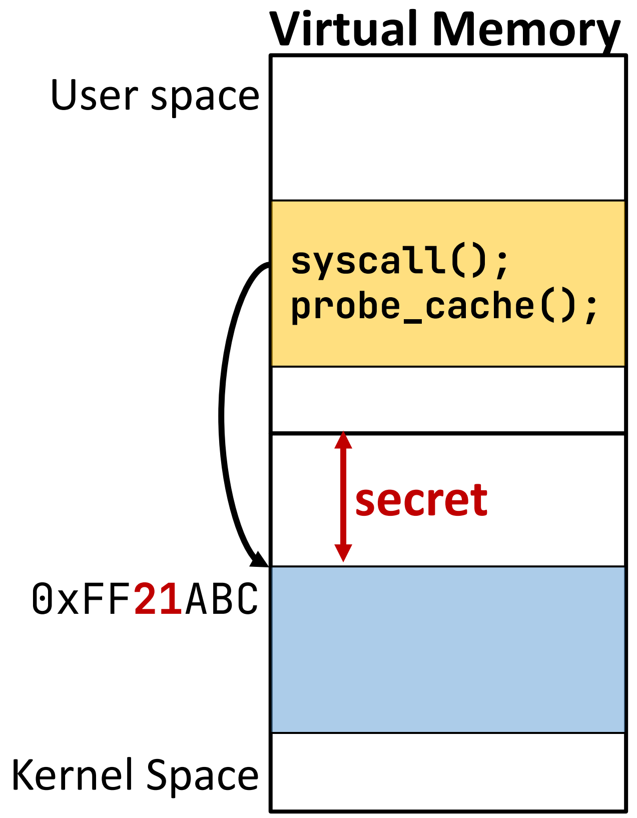
the myriad microarchitectural issues that lead to side channels like this one. Future work is needed in this area in order to preserve the integrity of KASLR, or alternatively, it is probably time to accept that KASLR is no longer an effective mitigation against local attackers and to develop defensive code and mitigations that accept its limitations.

Attack 1: ALSR Dependent Address Translation[1]



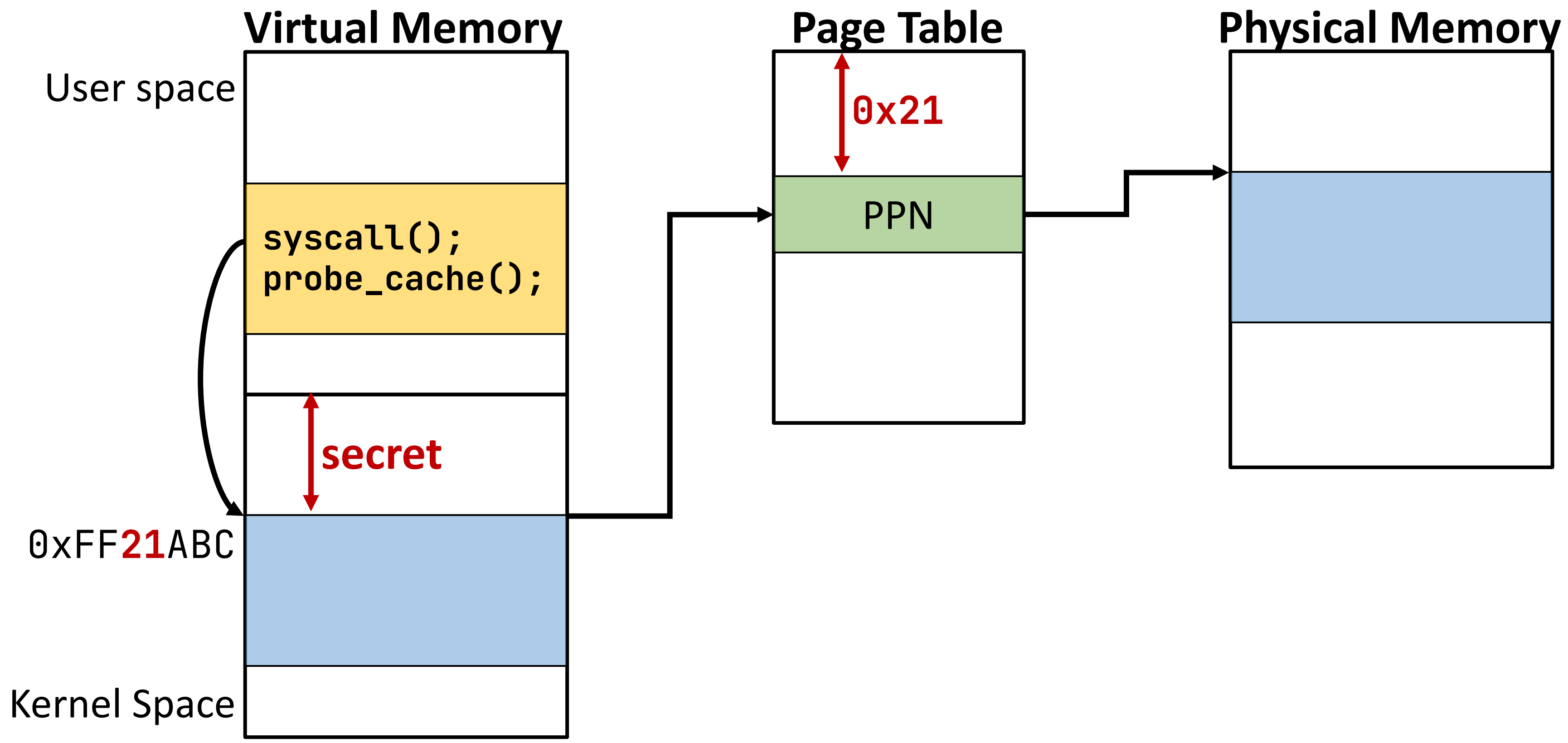
[1] Zhao, et al., “Binoculars: Contention-Based Side-Channel attacks exploiting the page walker,” in USENIX Security 2022.

Attack 1: ALSR Dependent Address Translation[1]



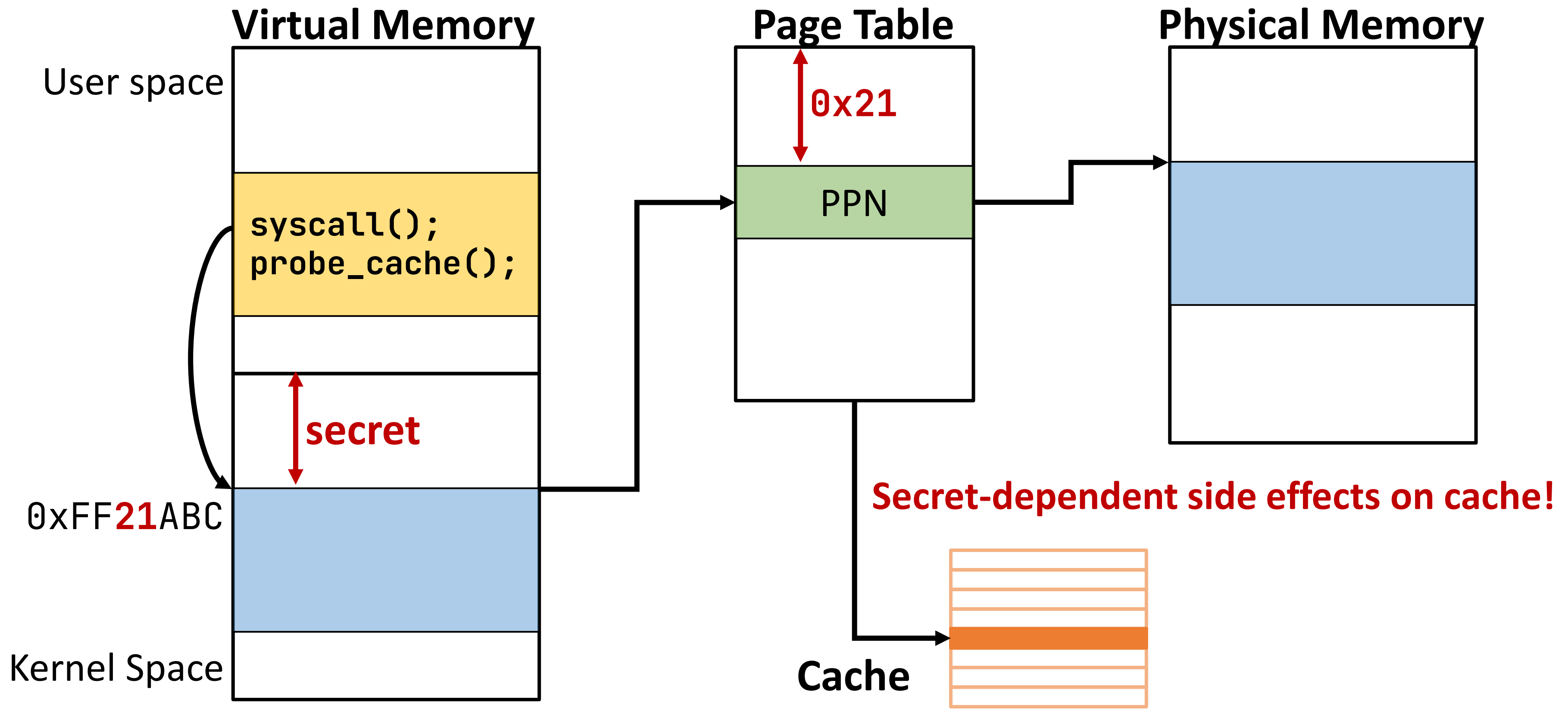
[1] Zhao, et al., “Binoculars: Contention-Based Side-Channel attacks exploiting the page walker,” in USENIX Security 2022.

Attack 1: ALSR Dependent Address Translation[1]



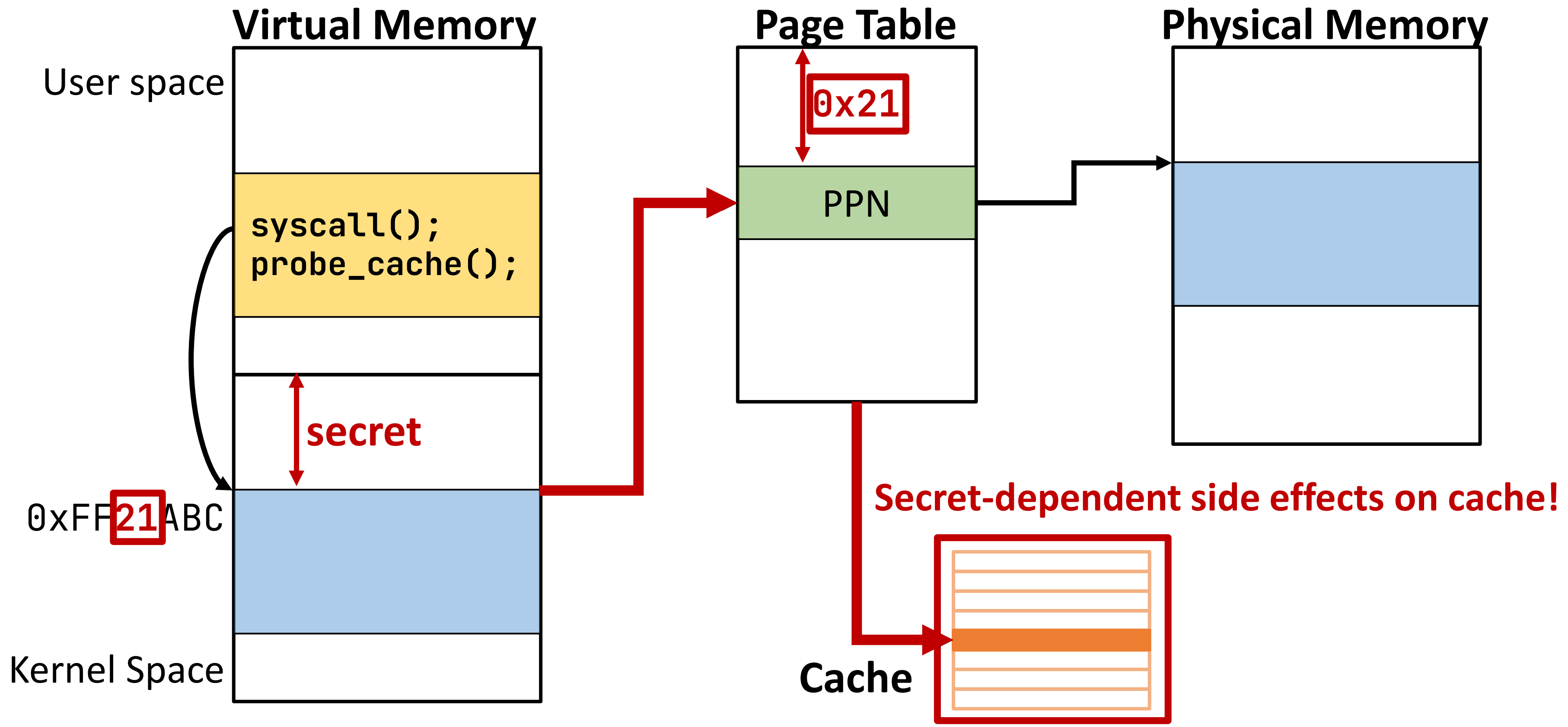
[1] Zhao, et al., “Binoculars: Contention-Based Side-Channel attacks exploiting the page walker,” in USENIX Security 2022.

Attack 1: ALSR Dependent Address Translation[1]



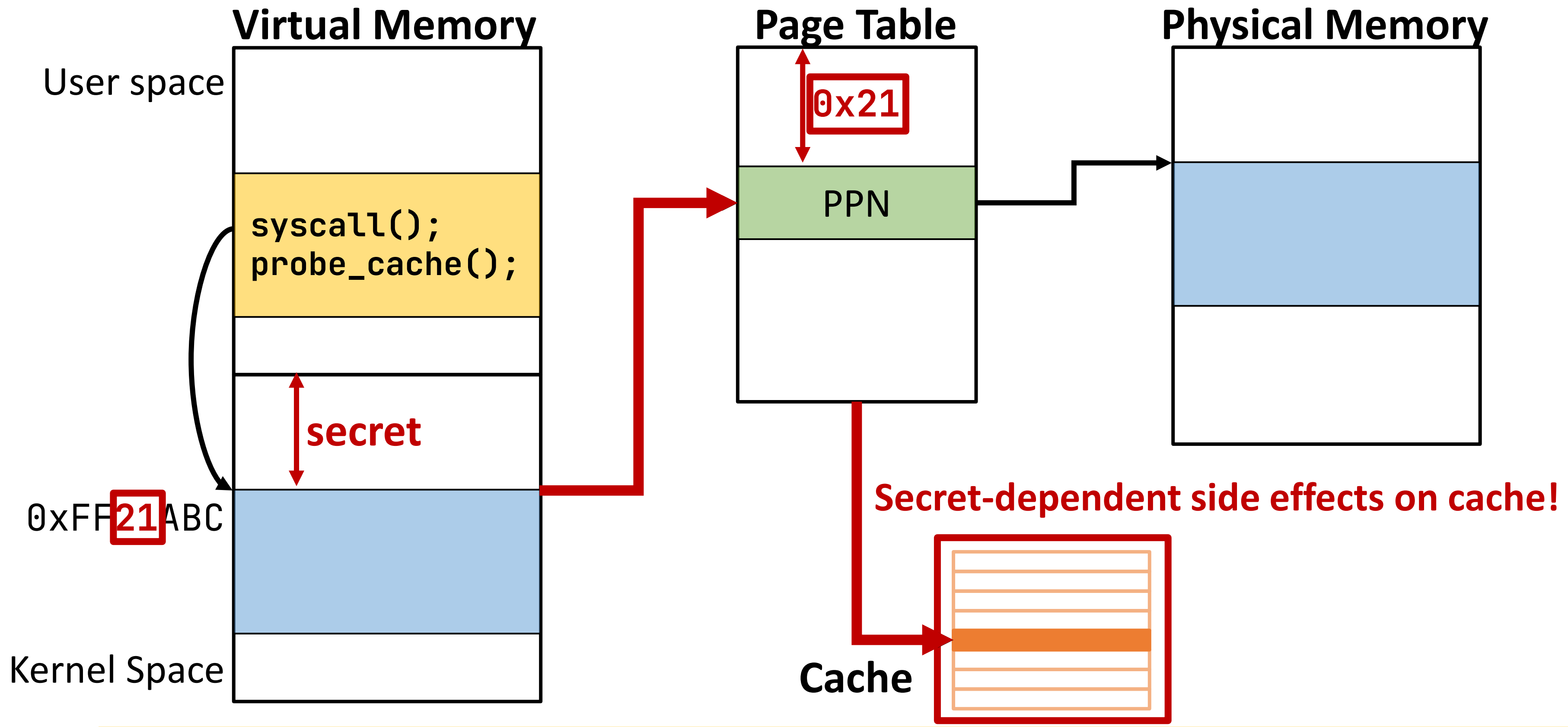
[1] Zhao, et al., “Binoculars: Contention-Based Side-Channel attacks exploiting the page walker,” in USENIX Security 2022.

Attack 1: ALSR Dependent Address Translation[1]



[1] Zhao, et al., “Binoculars: Contention-Based Side-Channel attacks exploiting the page walker,” in USENIX Security 2022.

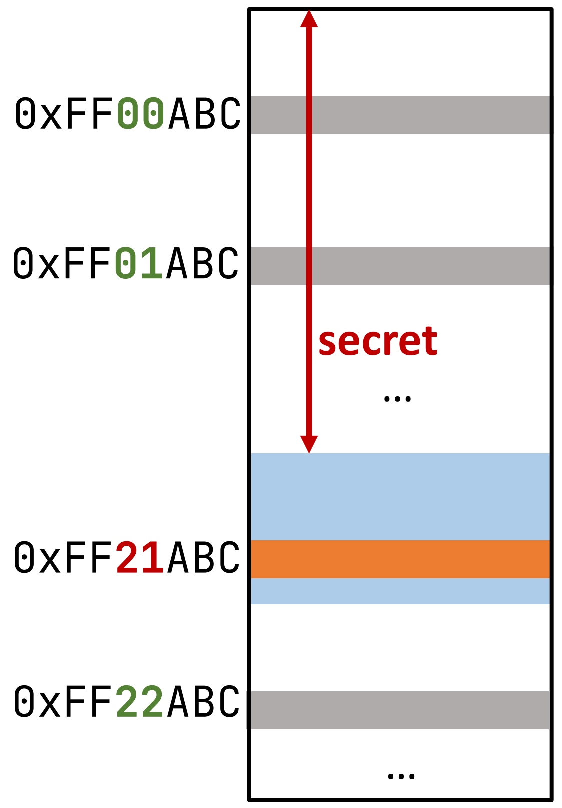
Attack 1: ALSR Dependent Address Translation[1]



ASLR secret is used to index into page tables and microarchitecture structures.

[1] Zhao, et al., “Binoculars: Contention-Based Side-Channel attacks exploiting the page walker,” in USENIX Security 2022.

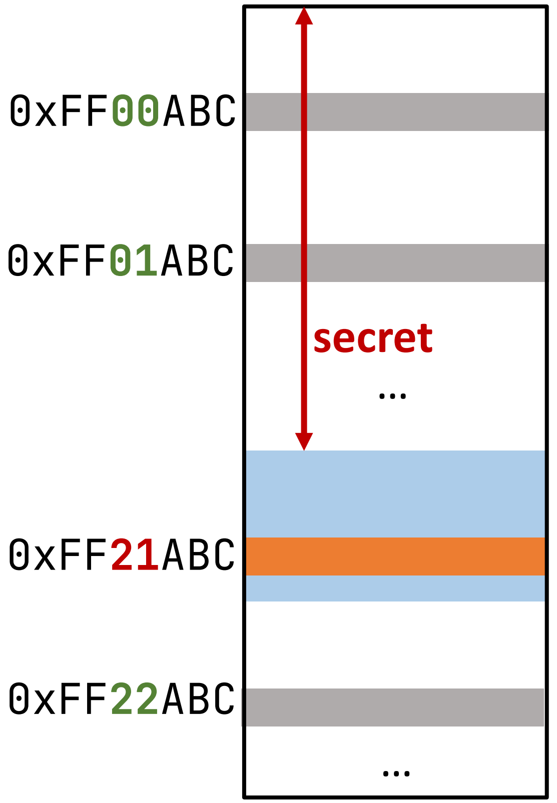
Attack 2: Distinguishing Valid/Invalid Addresses[1]



```
for guess_addr in
    [0xFF00ABC, 0xFF01ABC, ... 0xFF21ABC, ...]
{
    transient_probe(guess_addr);
    latency = transient_probe(guess_addr);
}
```

[1] Jang, et al., “Breaking kernel address space layout randomization with Intel TSX,” CCS 2016.

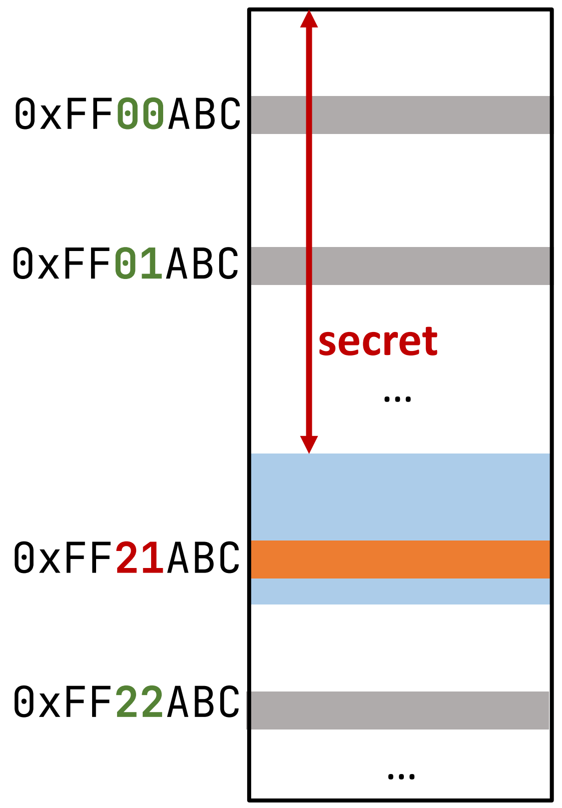
Attack 2: Distinguishing Valid/Invalid Addresses[1]



```
for guess_addr in
    [0xFF00ABC, 0xFF01ABC, ... 0xFF21ABC, ...]
{
    transient_probe(guess_addr);
    latency = transient_probe(guess_addr);
}
```

[1] Jang, et al., "Breaking kernel address space layout randomization with Intel TSX," CCS 2016.

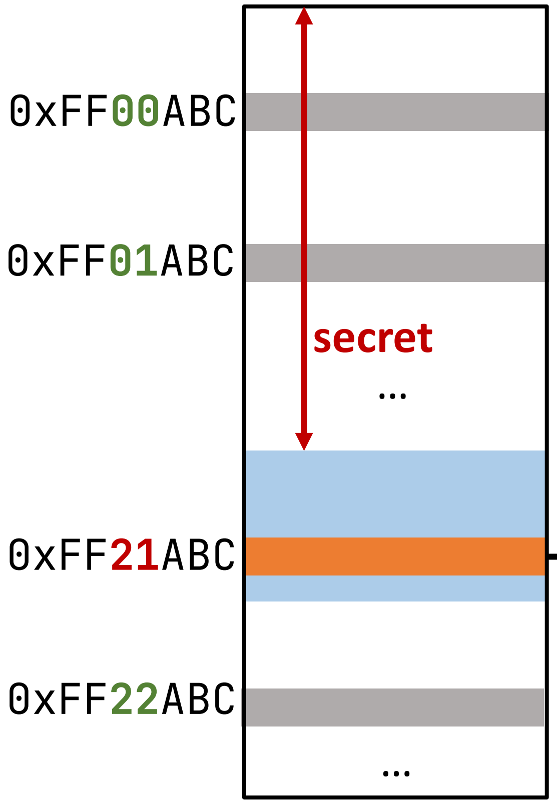
Attack 2: Distinguishing Valid/Invalid Addresses[1]



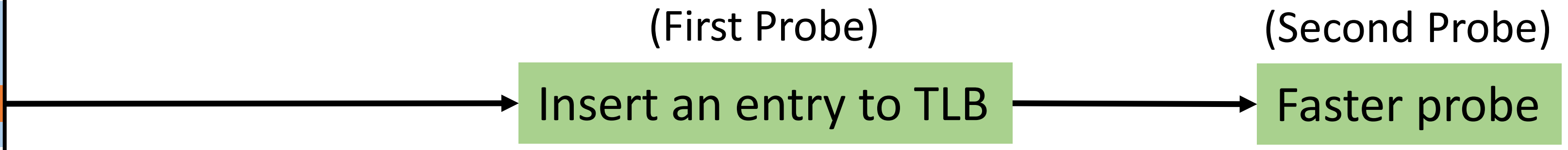
```
for guess_addr in
    [0xFF00ABC, 0xFF01ABC, ... 0xFF21ABC, ...]
{
    transient_probe(guess_addr);
    latency = transient_probe(guess_addr);
}
```

[1] Jang, et al., "Breaking kernel address space layout randomization with Intel TSX," CCS 2016.

Attack 2: Distinguishing Valid/Invalid Addresses[1]

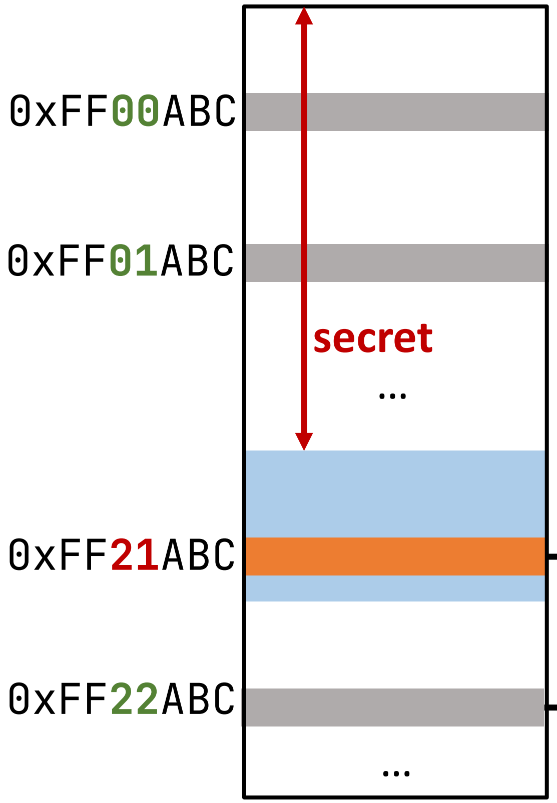


```
for guess_addr in
    [0xFF00ABC, 0xFF01ABC, ... 0xFF21ABC, ...]
{
    transient_probe(guess_addr);
    latency = transient_probe(guess_addr);
}
```



[1] Jang, et al., “Breaking kernel address space layout randomization with Intel TSX,” CCS 2016.

Attack 2: Distinguishing Valid/Invalid Addresses[1]



```
for guess_addr in
    [0xFF00ABC, 0xFF01ABC, ... 0xFF21ABC, ...]
{
    transient_probe(guess_addr);
    latency = transient_probe(guess_addr);
}
```

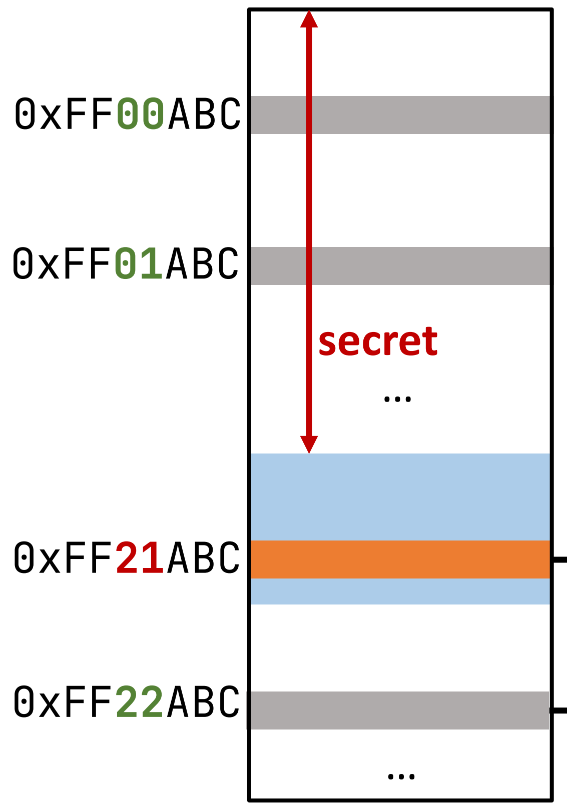
(First Probe) (Second Probe)

Insert an entry to TLB → Faster probe

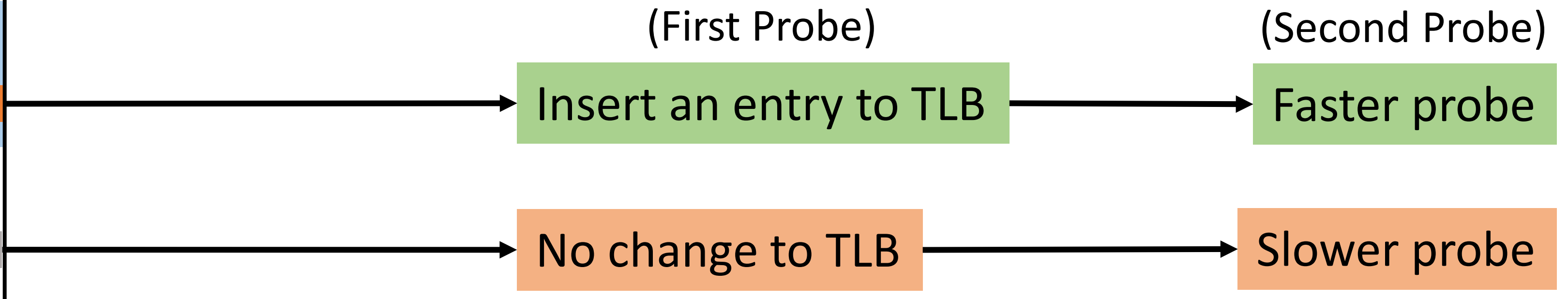
No change to TLB → Slower probe

[1] Jang, et al., "Breaking kernel address space layout randomization with Intel TSX," CCS 2016.

Attack 2: Distinguishing Valid/Invalid Addresses[1]

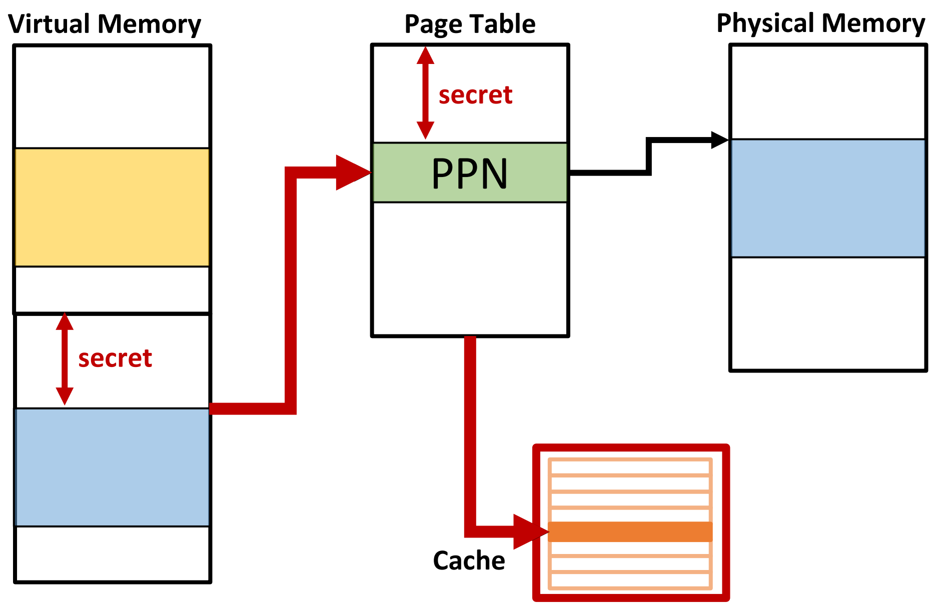


```
for guess_addr in
    [0xFF00ABC, 0xFF01ABC, ... 0xFF21ABC, ...]
{
    transient_probe(guess_addr);
    latency = transient_probe(guess_addr);
}
```

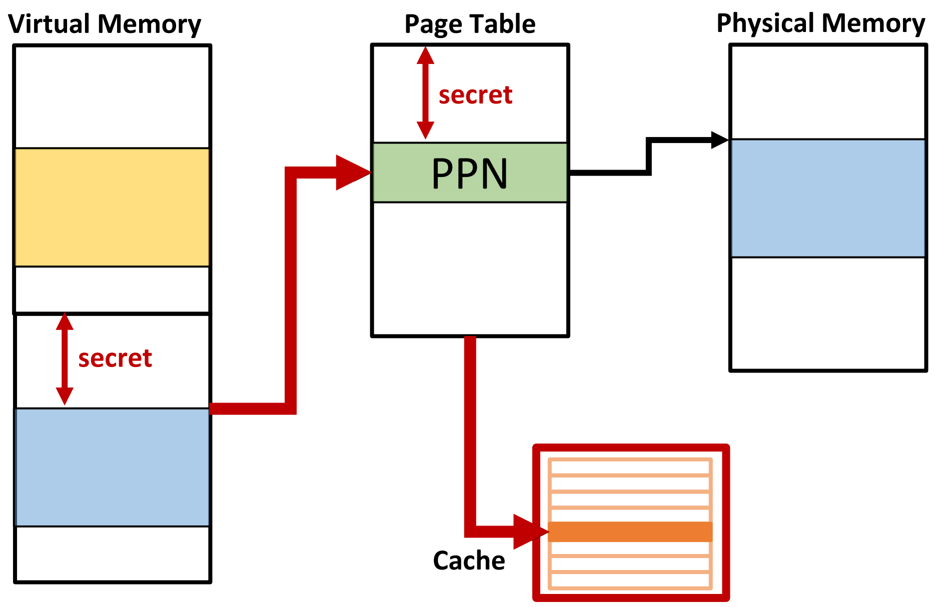


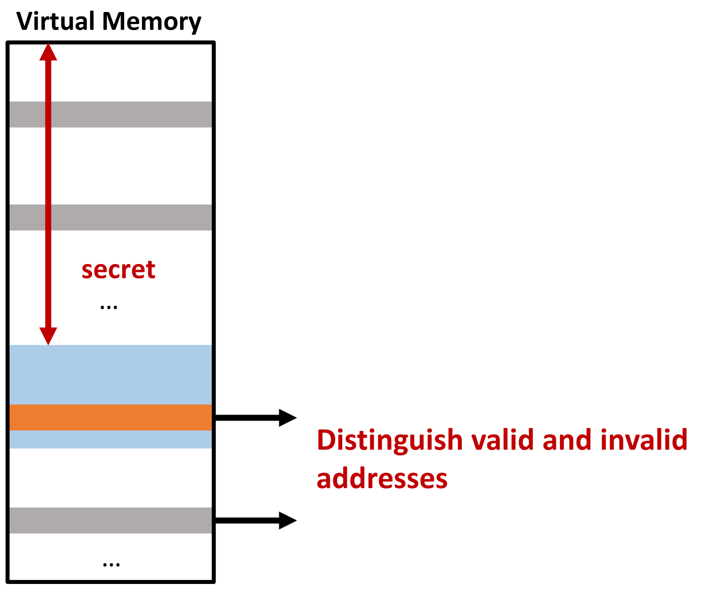
Accessing valid and invalid addresses has distinguishable microarchitectural side effects.

[1] Jang, et al., "Breaking kernel address space layout randomization with Intel TSX," CCS 2016.



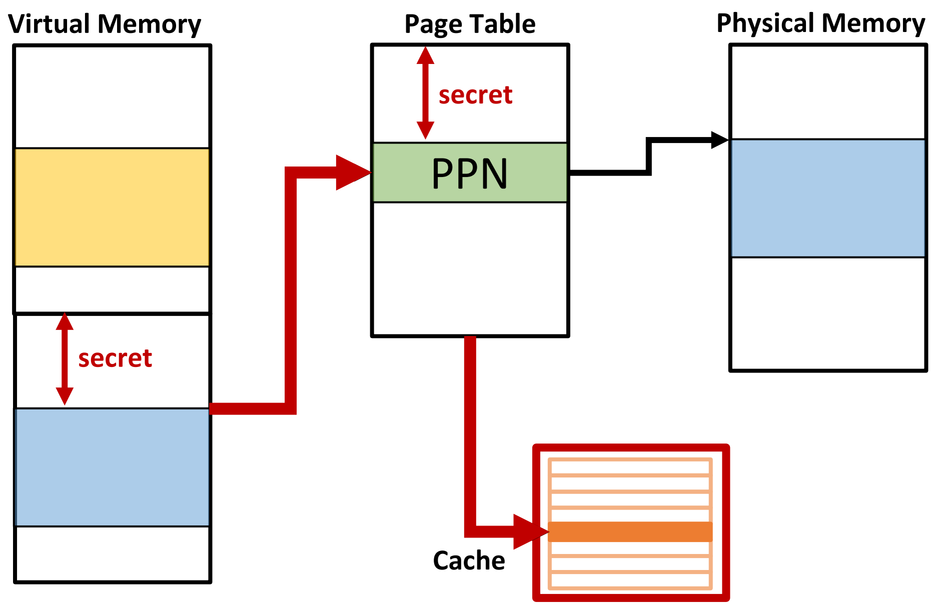
ASLR secret is used to index into page tables and microarchitecture structures.

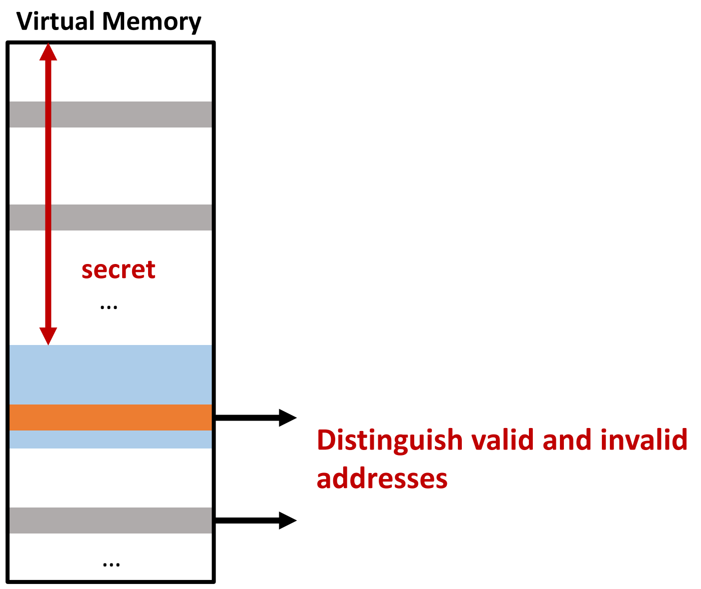




ASLR secret is used to index into page tables and microarchitecture structures.

Accessing valid and invalid addresses has distinguishable microarchitectural side effects.

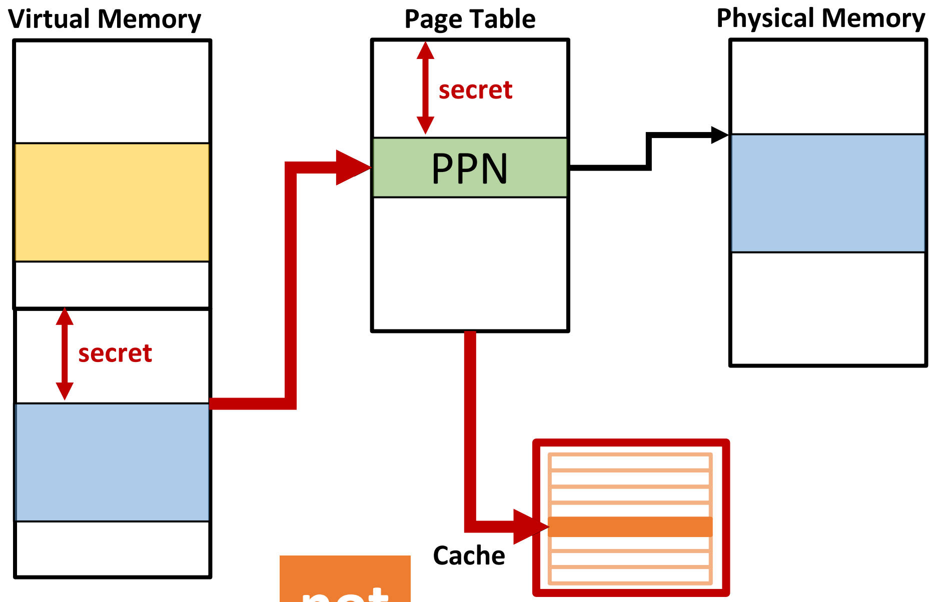




ASLR secret is used to index into page tables and microarchitecture structures.

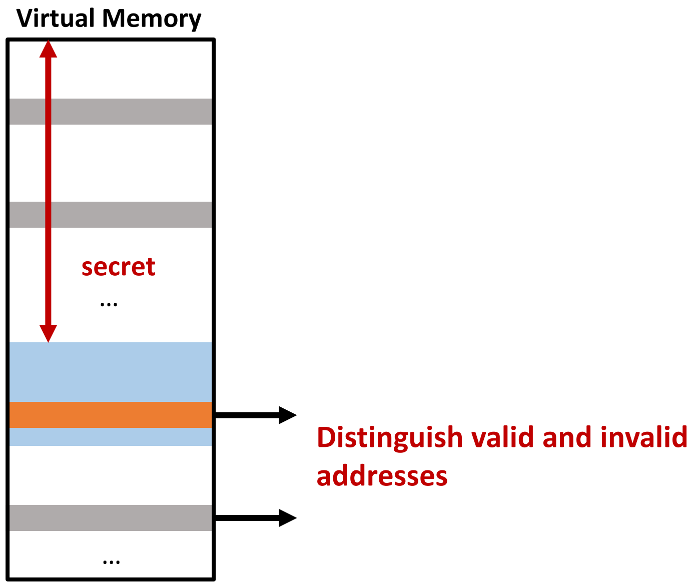
Accessing valid and invalid addresses has distinguishable microarchitectural side effects.

Our goal is to make ASLR bits microarchitectural independent.



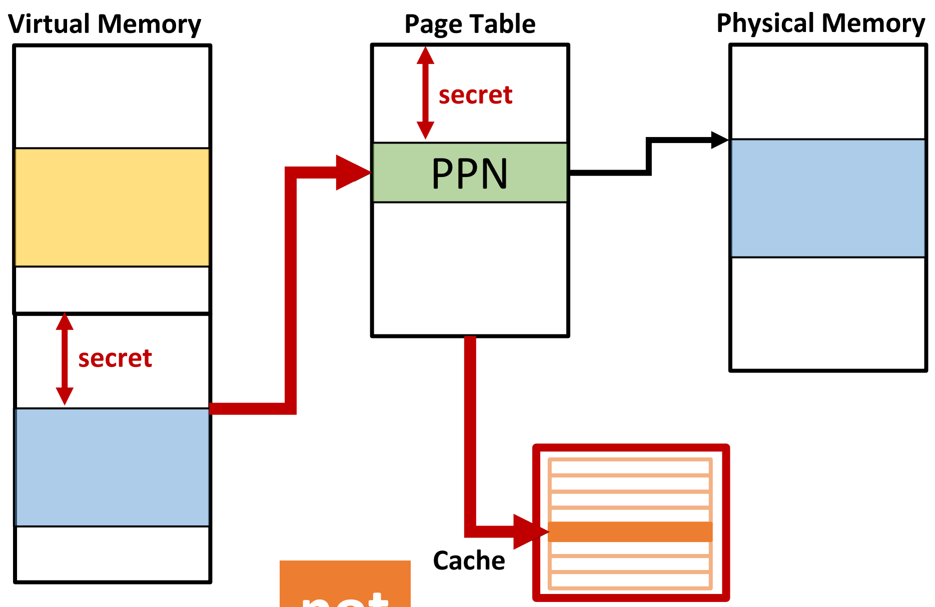
not

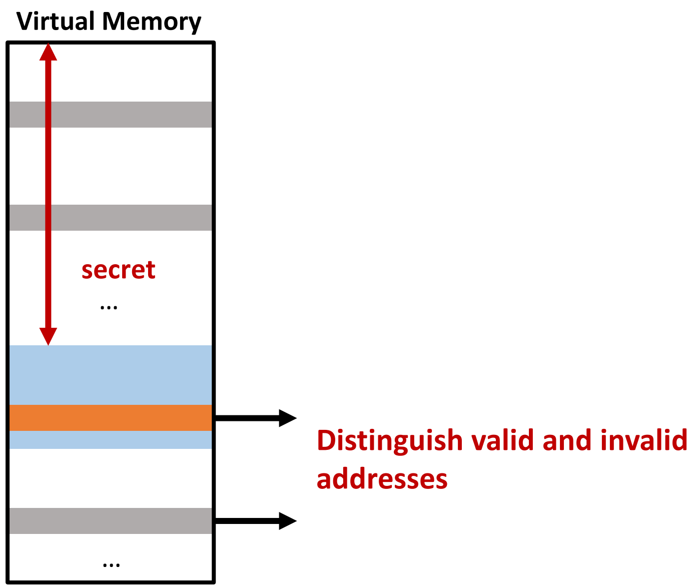
ASLR secret is used to index into page tables and microarchitecture structures.



Accessing valid and invalid addresses has distinguishable microarchitectural side effects.

Our goal is to make ASLR bits microarchitectural independent.





not

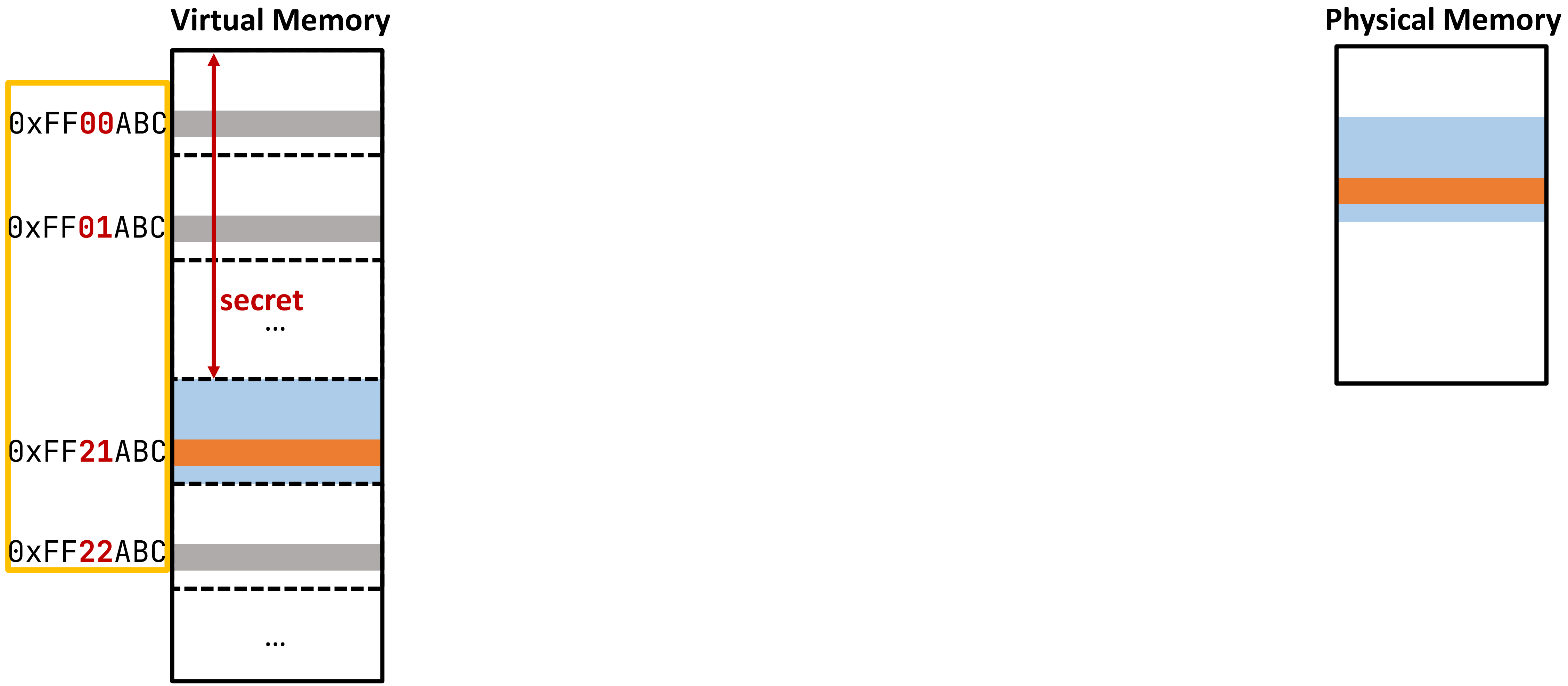
ASLR secret is used to index into page tables and microarchitecture structures.

Accessing valid and invalid addresses has ~~distinguishable~~ microarchitectural side effects.

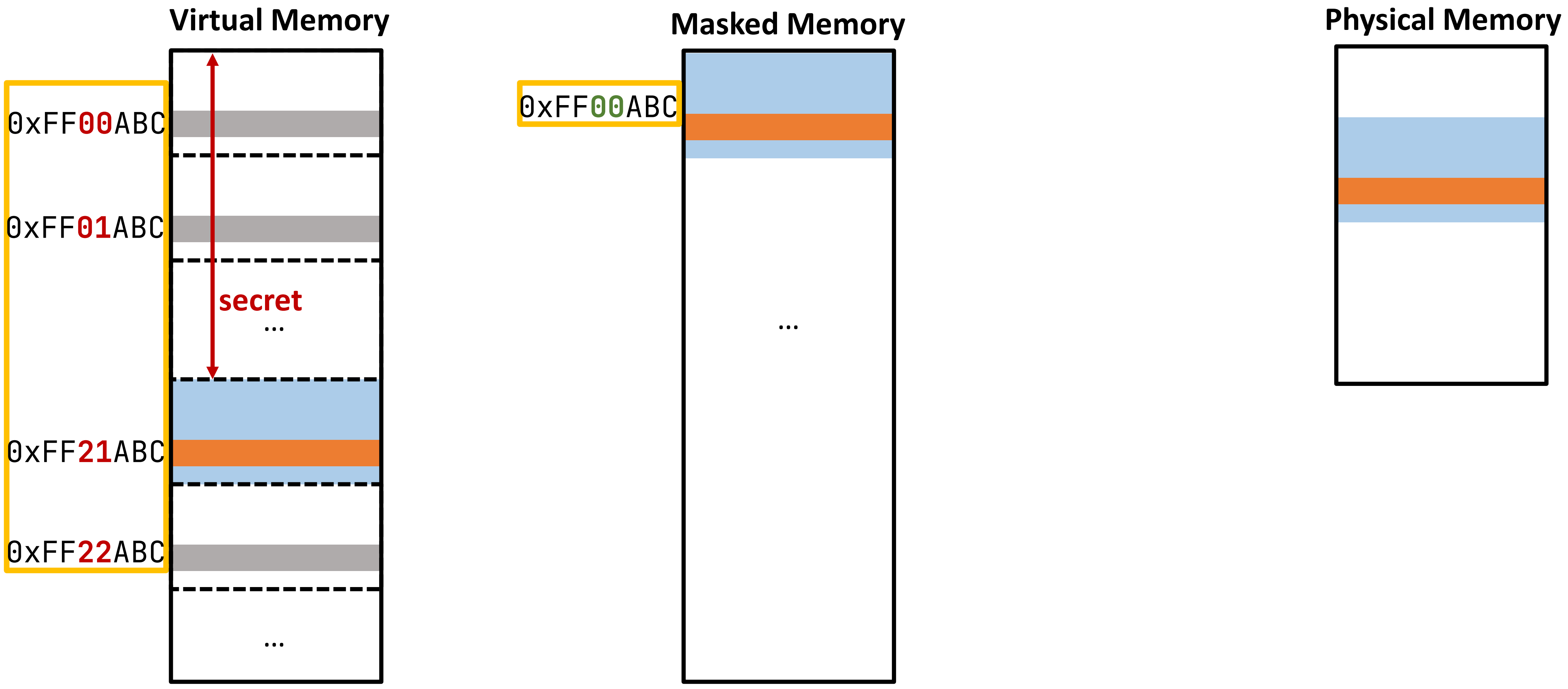
indistinguishable

Our goal is to make ASLR bits microarchitectural independent.

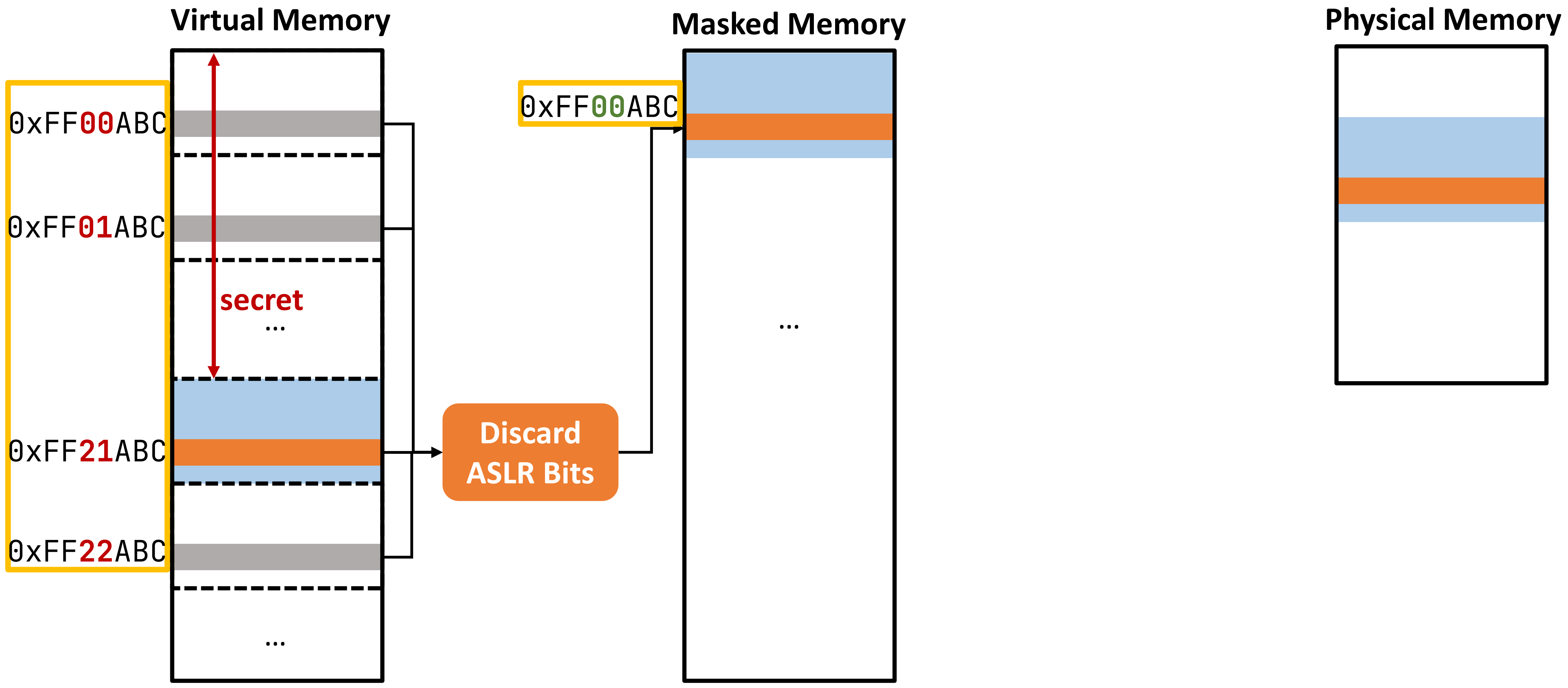
Oreo: New Memory Interface



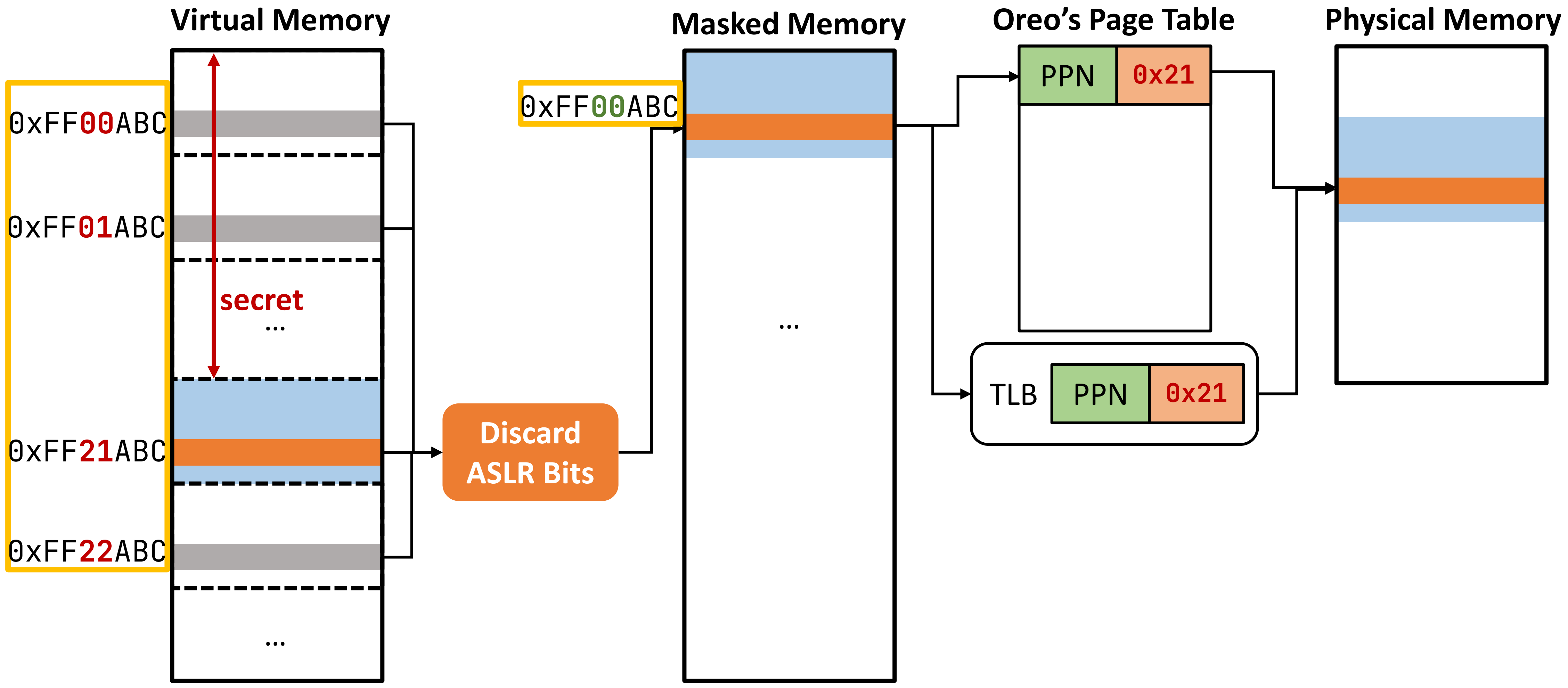
Oreo: New Memory Interface



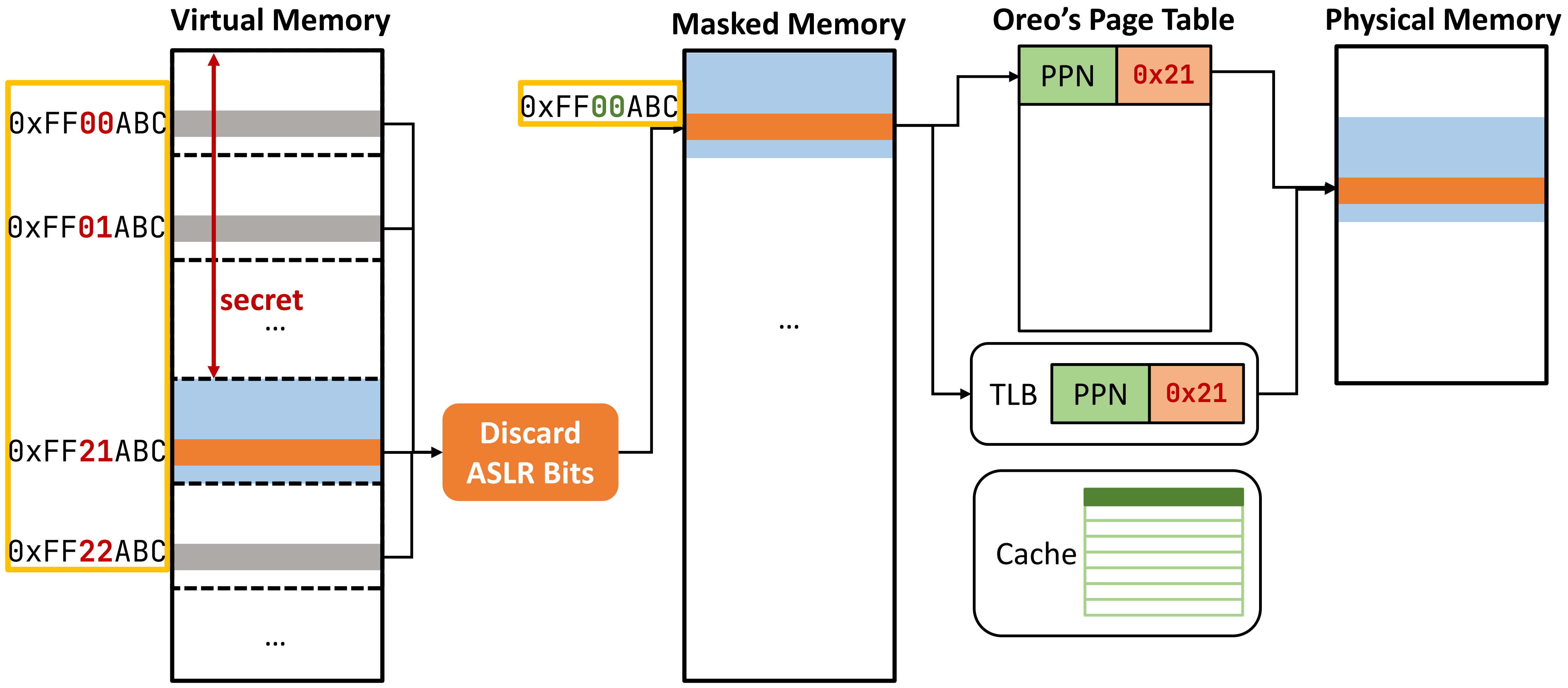
Oreo: New Memory Interface



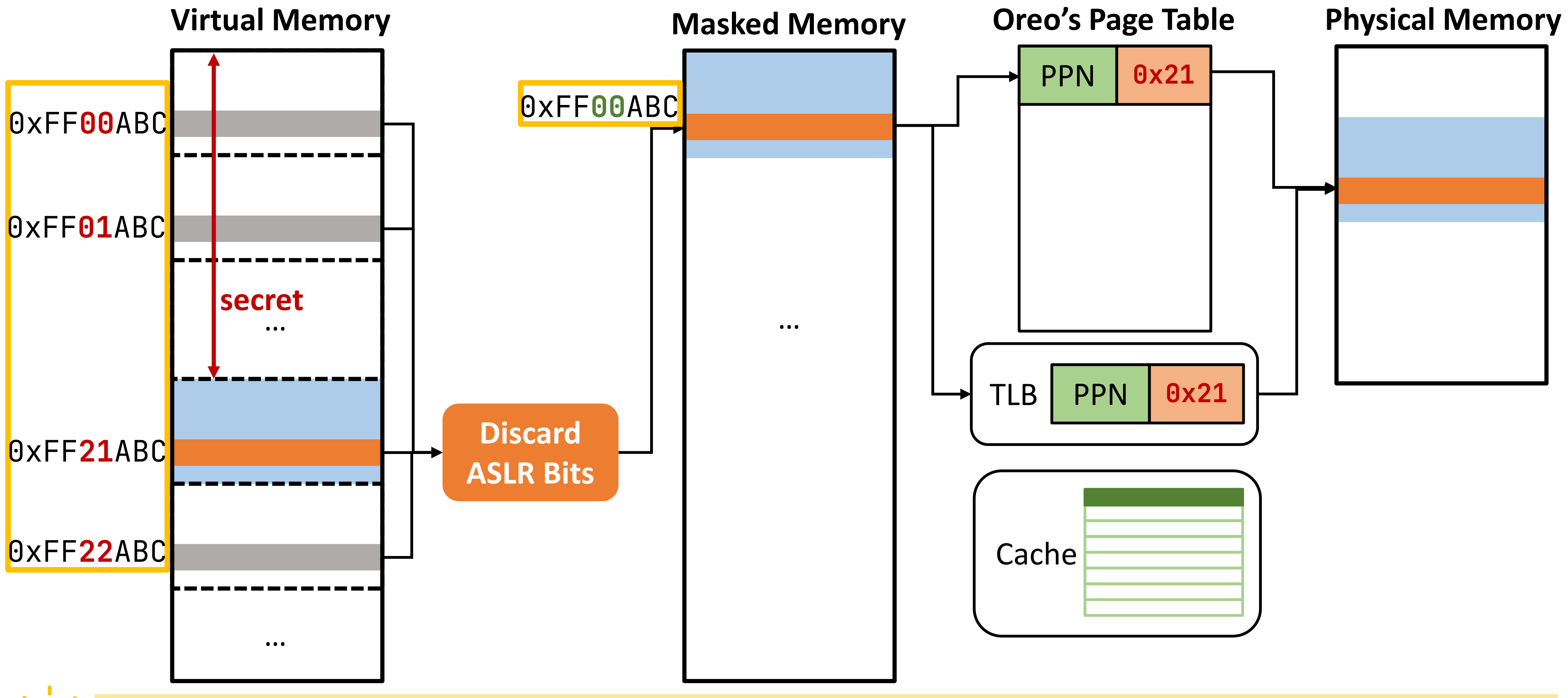
Oreo: New Memory Interface



Oreo: New Memory Interface

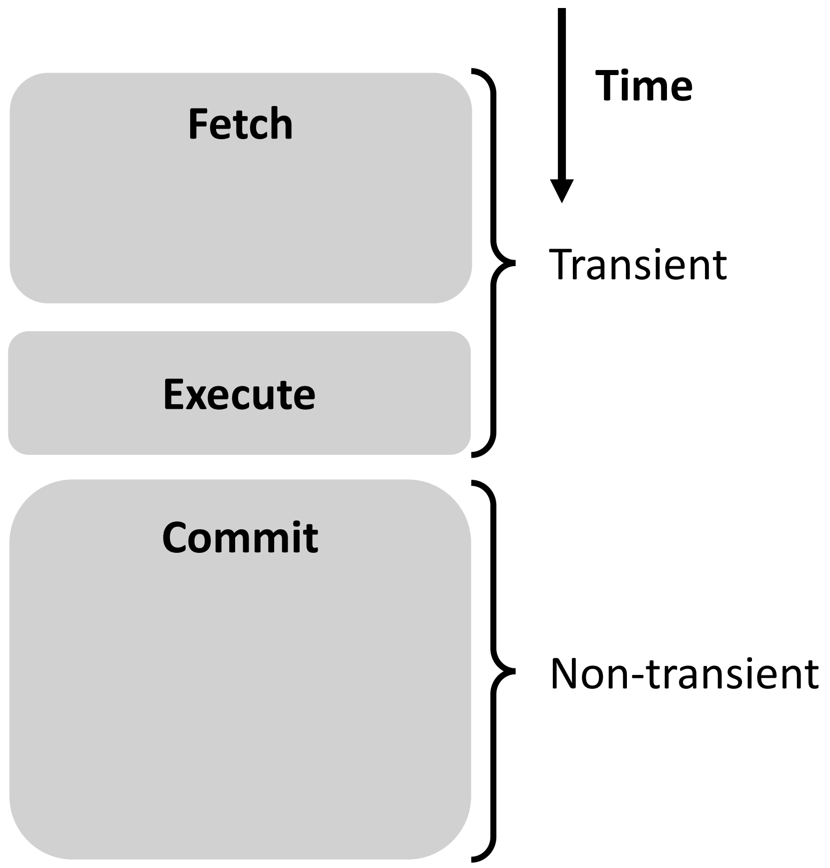


Oreo: New Memory Interface

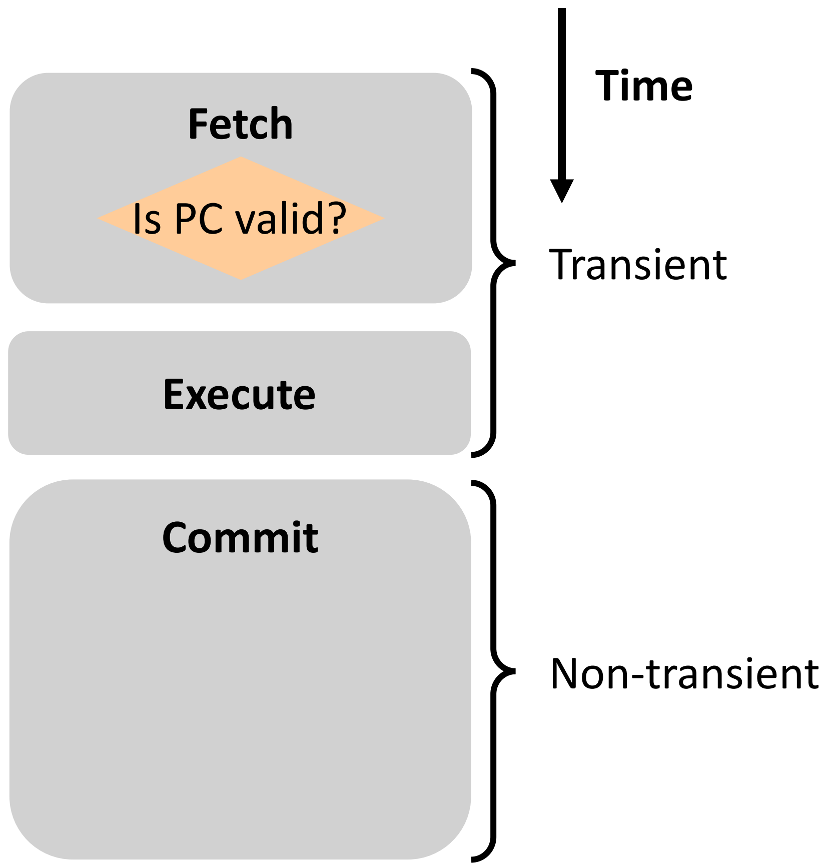


Oreo uses secret-independent masked addresses in page table walk.

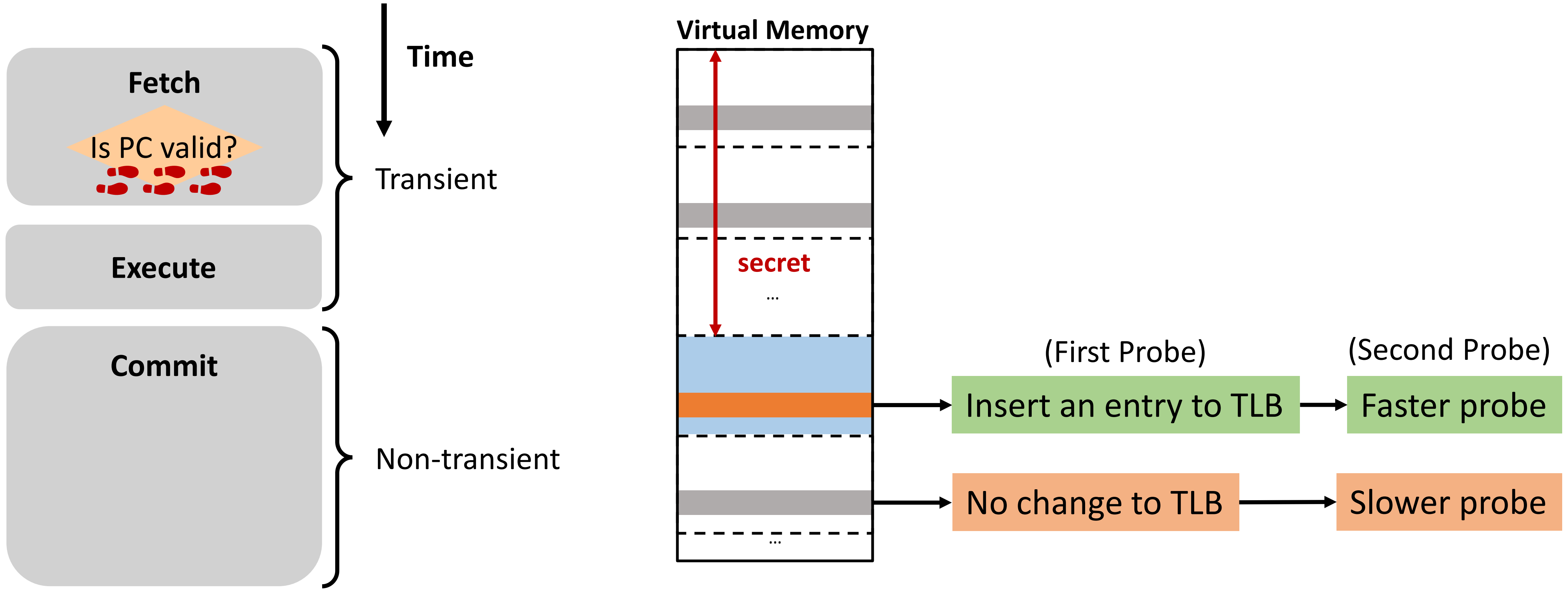
Oreo: ASLR Security Check



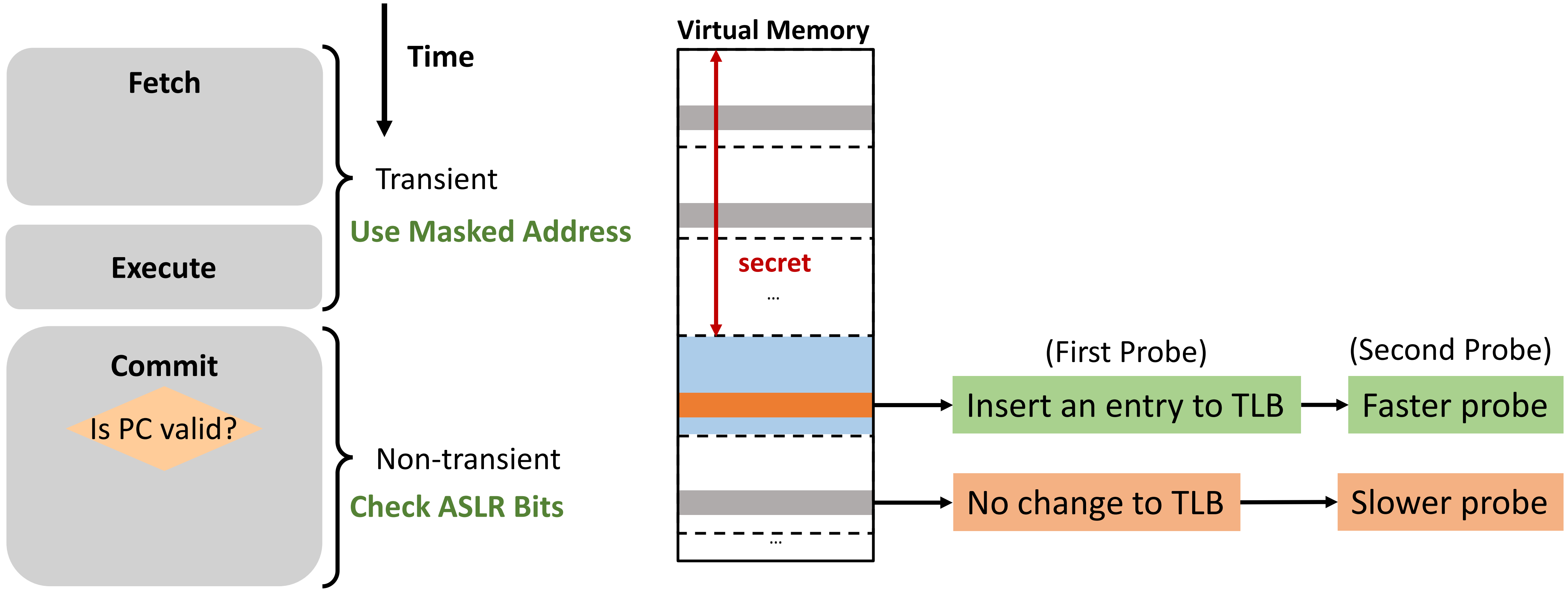
Oreo: ASLR Security Check



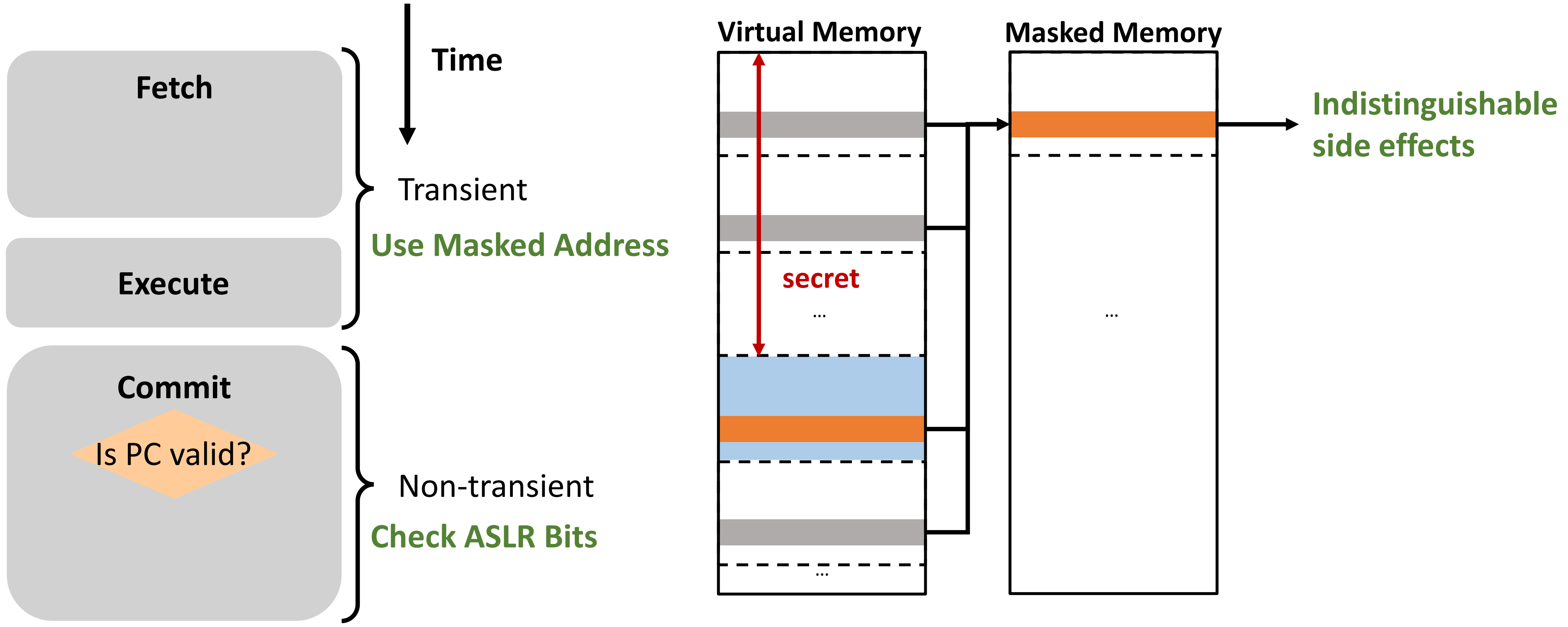
Oreo: ASLR Security Check



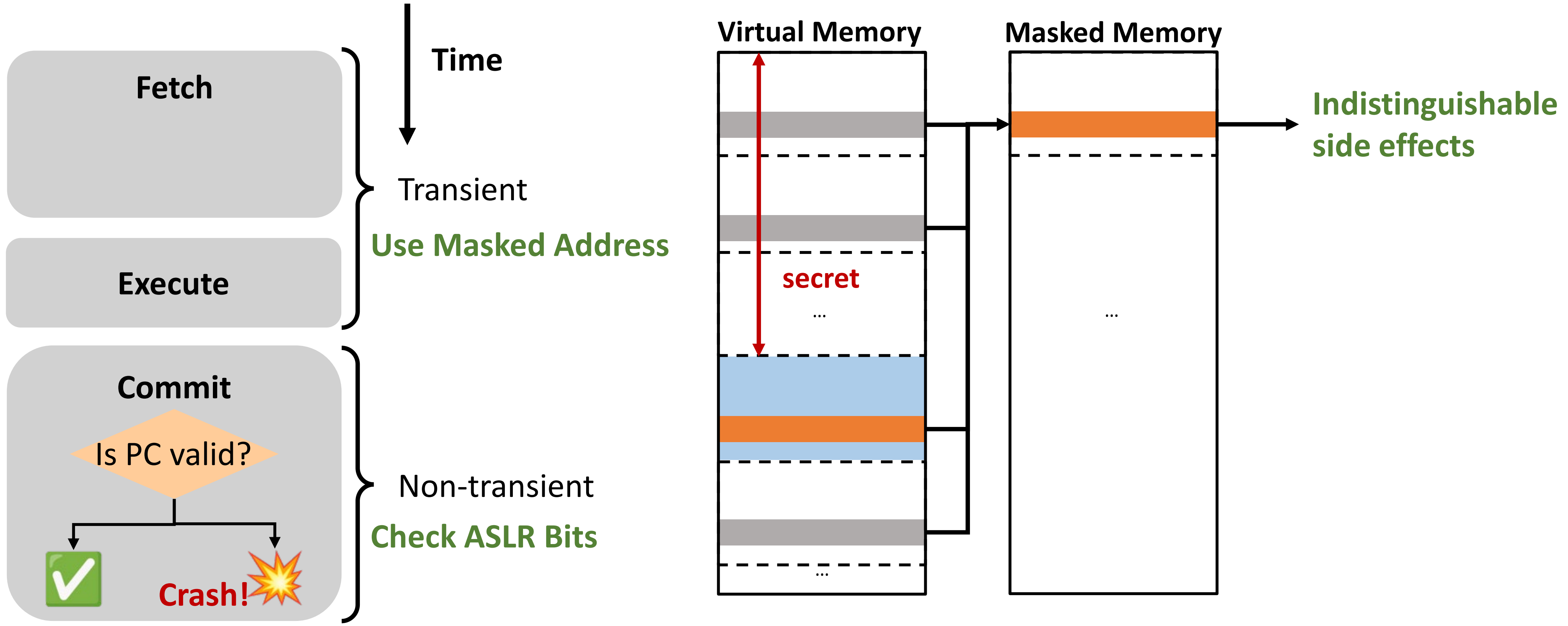
Oreo: ASLR Security Check



Oreo: ASLR Security Check

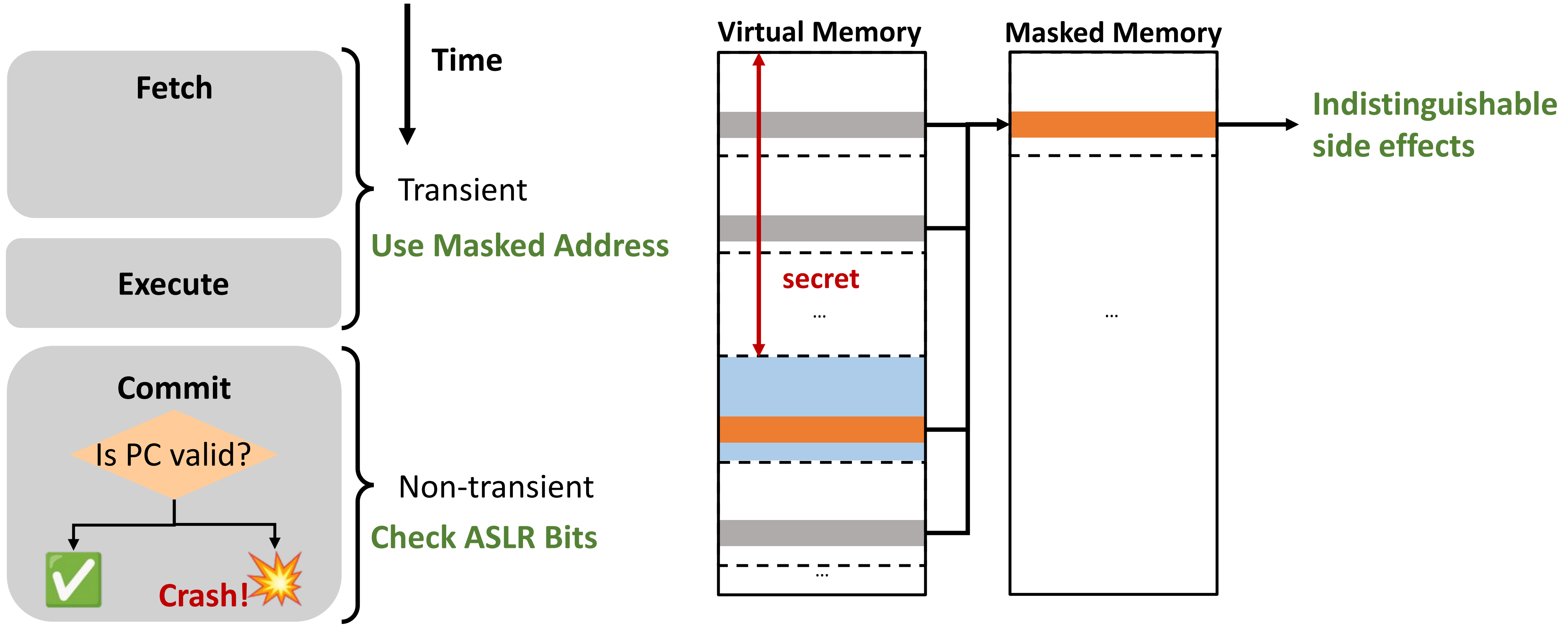


Oreo: ASLR Security Check



Attacker can only distinguish valid and invalid addresses through crashes.

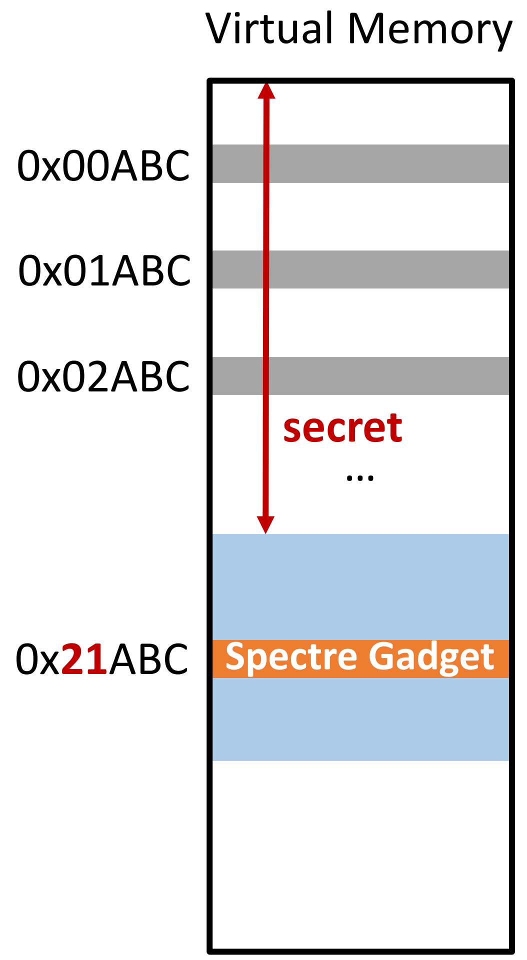
Oreo: ASLR Security Check



Attacker can only distinguish valid and invalid addresses through crashes.

Delaying security check does not affect pipeline performance.

Security Dilemma



Security Dilemma

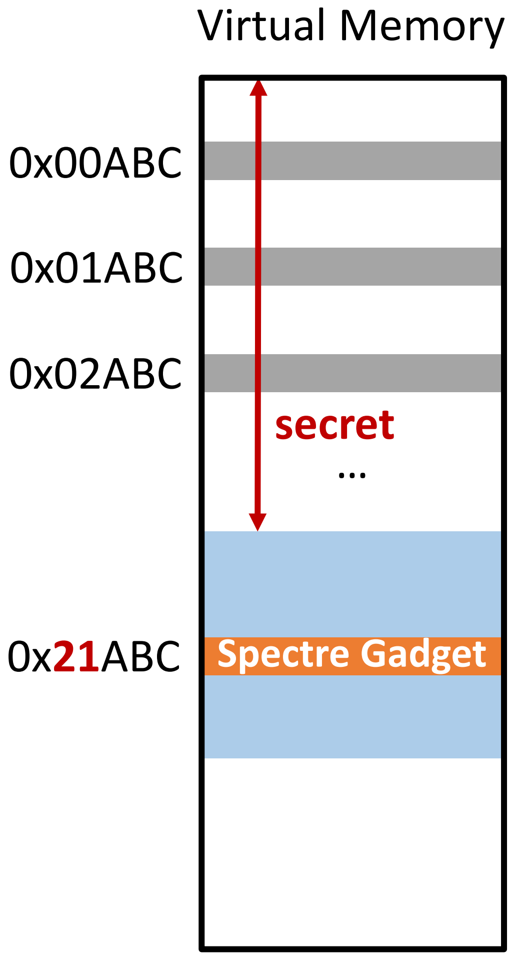
Without Oreo:

Spectre Attack needs:

Leak the secret **0x21**

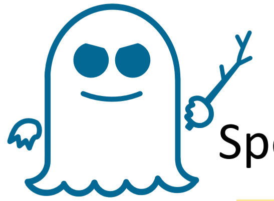
+

```
fp = 0x21ABC;
if (false) {
     jmp fp;
}
```



Security Dilemma

Without Oreo:



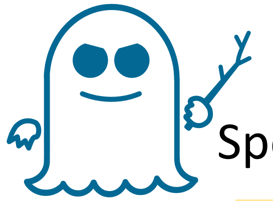
Spectre Attack needs:

Leak the secret **0x21**

+

```
fp = 0x21ABC;
if (false) {
      jmp fp;
}
```

With Oreo:

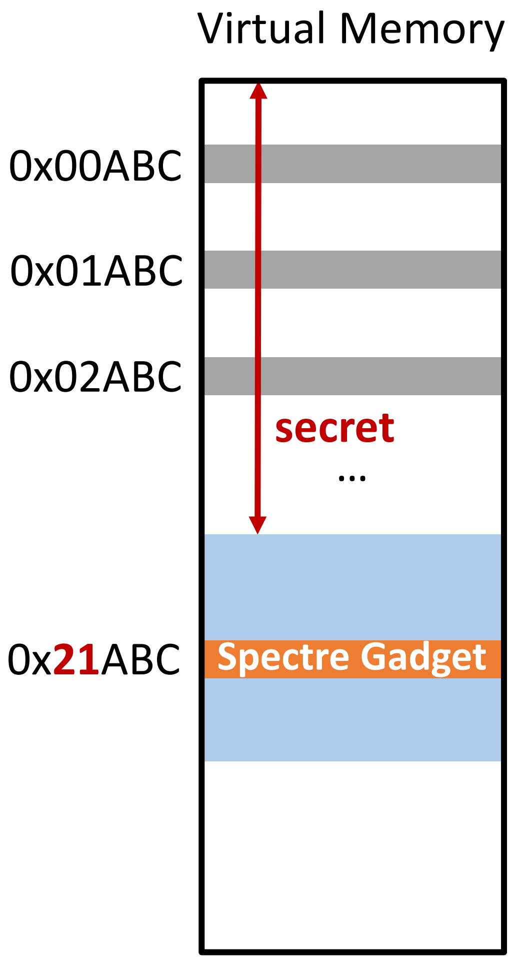


Spectre Attack needs:

~~Leak the secret **0x21**~~

+

```
fp = 0x00ABC;
if (false) {
      jmp fp;
}
```



Security Dilemma

Without Oreo:

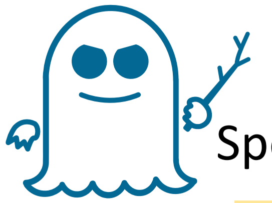
Spectre Attack needs:

Leak the secret **0x21**

+

```
fp = 0x21ABC;
if (false) {
     jmp fp;
}
```

With Oreo:

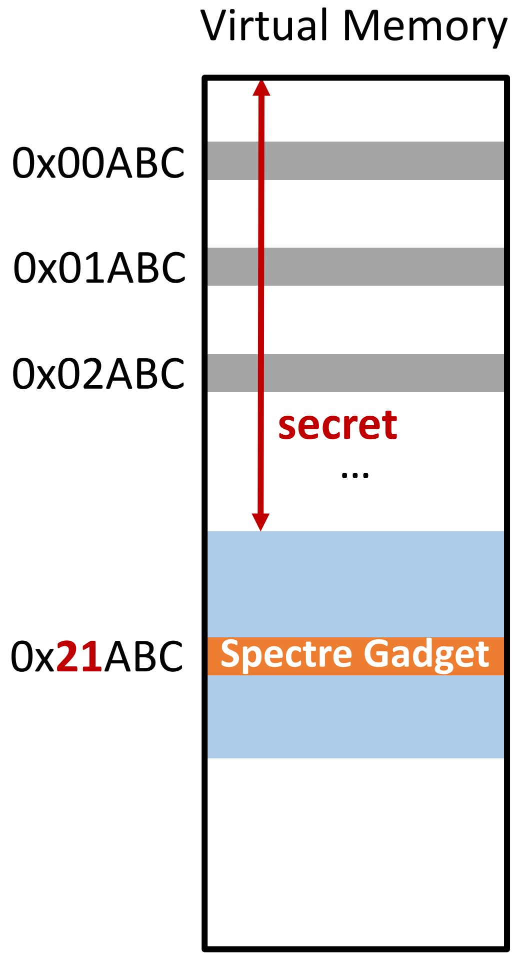


Spectre Attack needs:

~~Leak the secret **0x21**~~

+

```
fp = 0x00ABC;
if (false) {
     jmp fp;
}
```

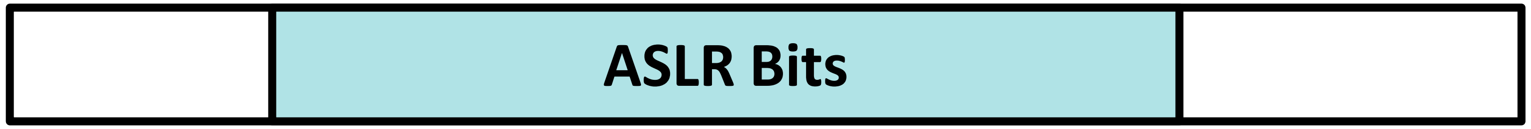


Making valid addresses indistinguishable however makes Spectre attacks easier.

Dilemma Analysis

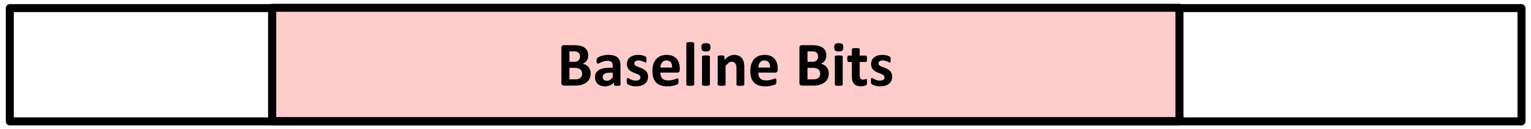
Virtual Address

Dilemma Analysis



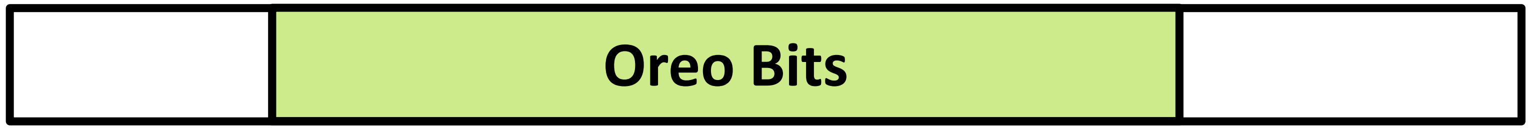
Baseline Bits	Oreo Bits	Security Outcome

Dilemma Analysis



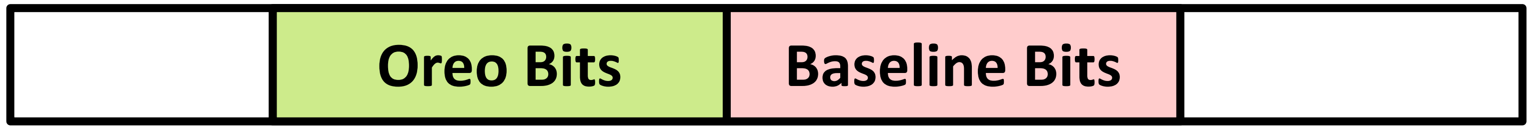
Baseline Bits	Oreo Bits	Security Outcome
All	None	❌ Vulnerable to ASLR bypasses

Dilemma Analysis



Baseline Bits	Oreo Bits	Security Outcome
All	None	❌ Vulnerable to ASLR bypasses
None	All	❌ Vulnerable to Spectre-like attacks

Dilemma Analysis

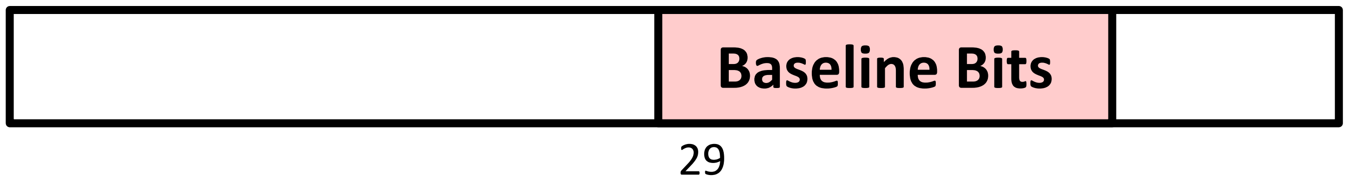


Baseline Bits	Oreo Bits	Security Outcome
All	None	❌ Vulnerable to ASLR bypasses
None	All	❌ Vulnerable to Spectre-like attacks
Some	Some	✅ Safe on both sides!

Oreo: Solution to Security Dilemma

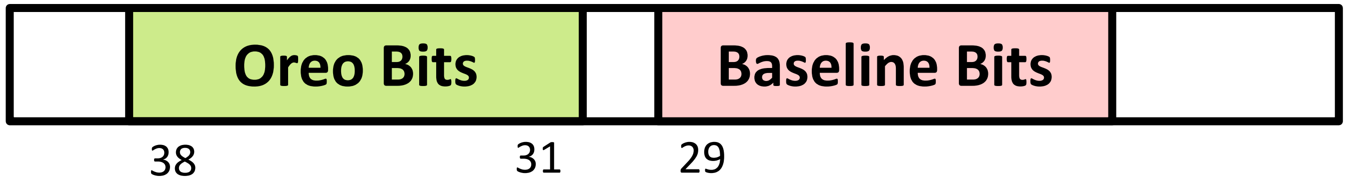
Oreo: Solution to Security Dilemma

Kernel ASLR



Oreo: Solution to Security Dilemma

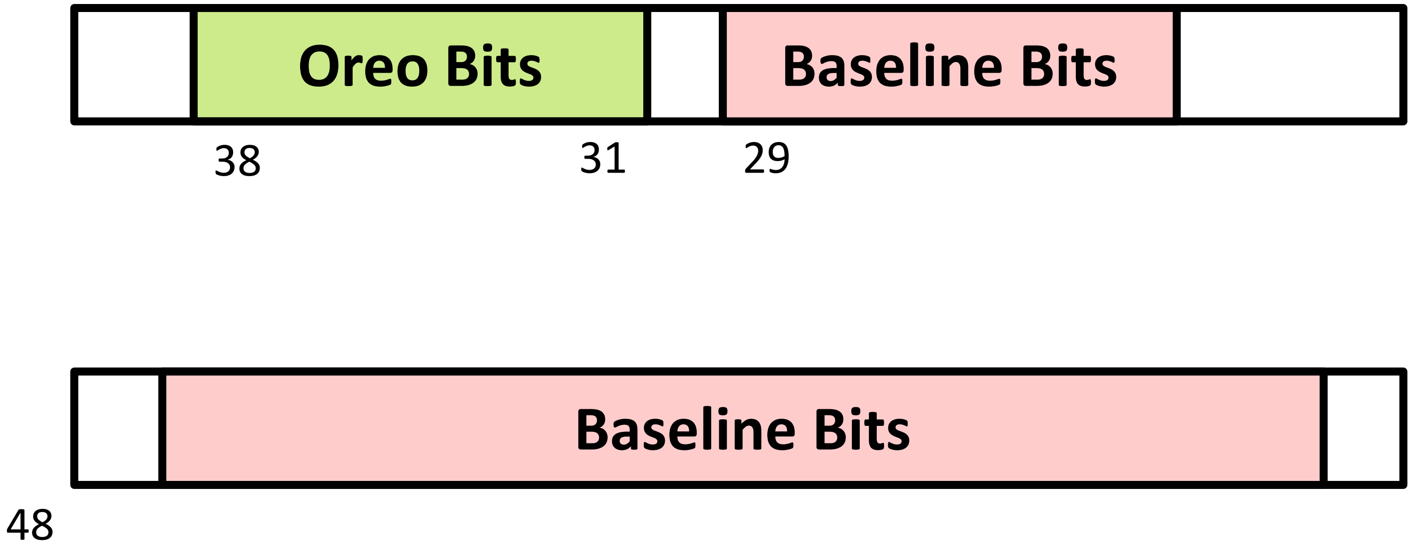
Kernel ASLR



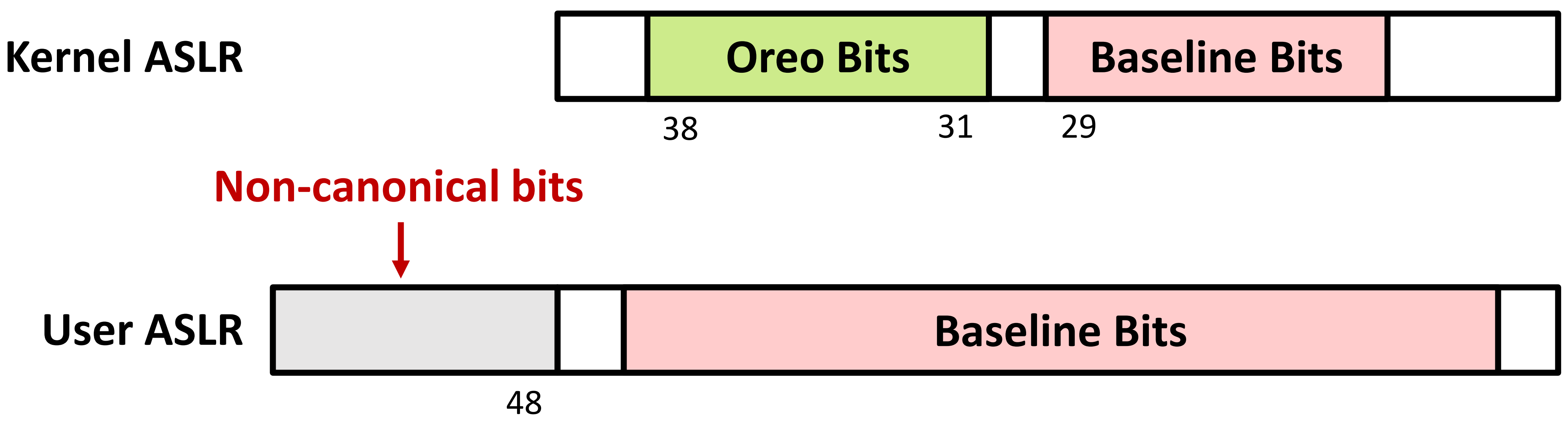
Oreo: Solution to Security Dilemma

Kernel ASLR

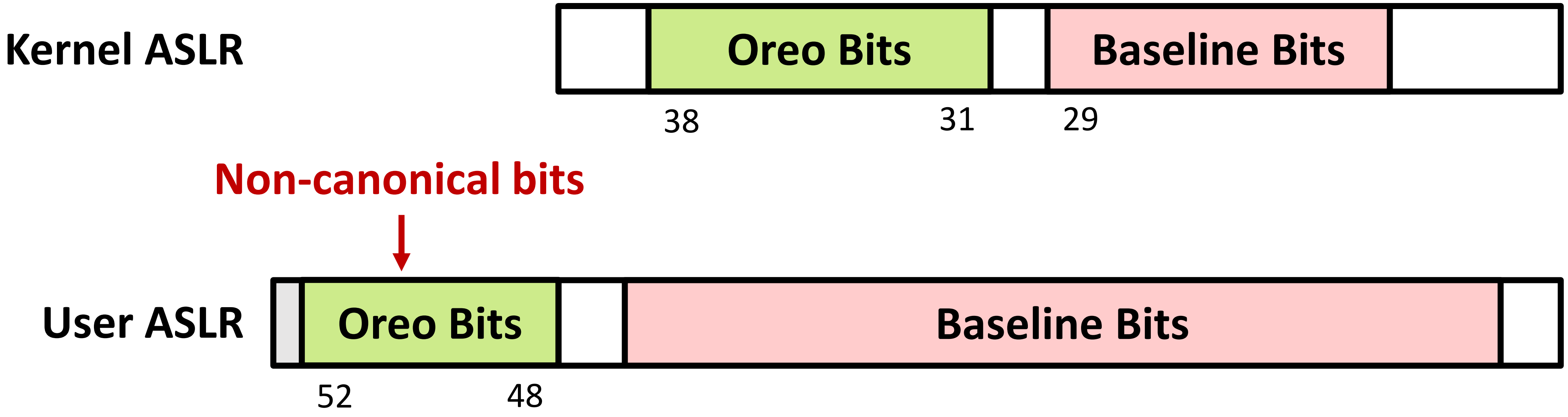
User ASLR



Oreo: Solution to Security Dilemma



Oreo: Solution to Security Dilemma



More in the Paper...

- Prototype
  - SW: Linux
  - HW: gem5 simulator

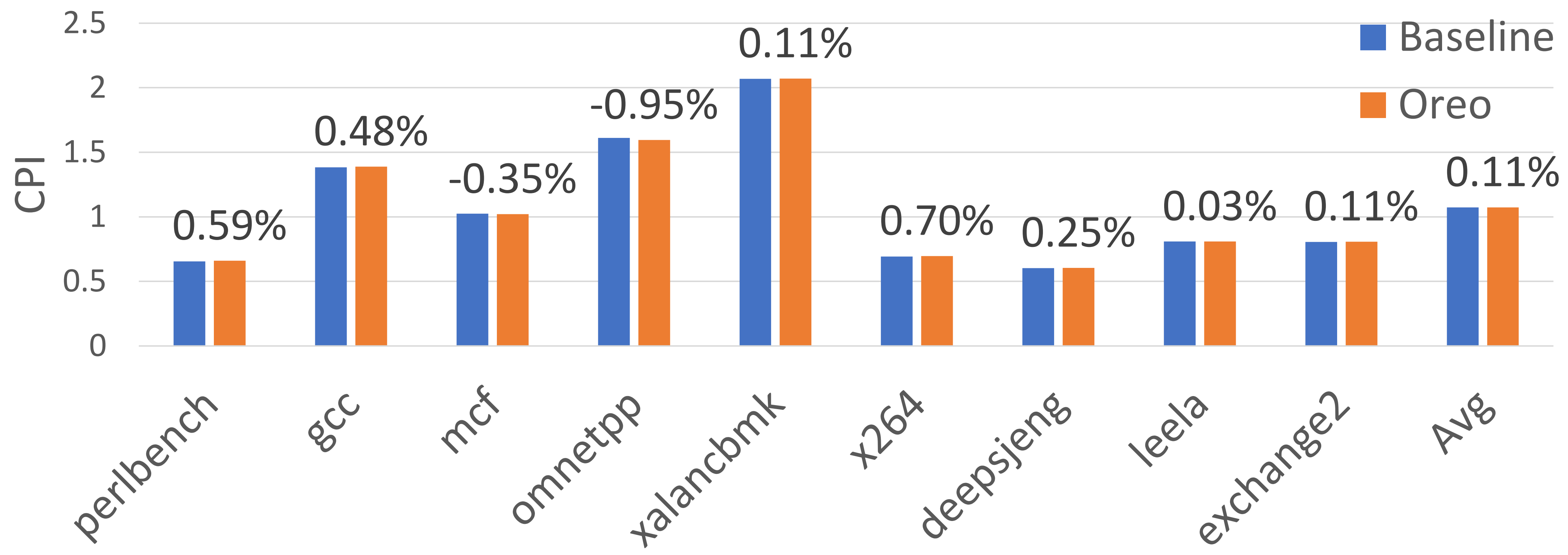
More in the Paper...

- Prototype
  - SW: Linux
  - HW: gem5 simulator
- Evaluation
  - Performance evaluation on SPEC and LEBench
  - Security evaluation on multiple leakage paths

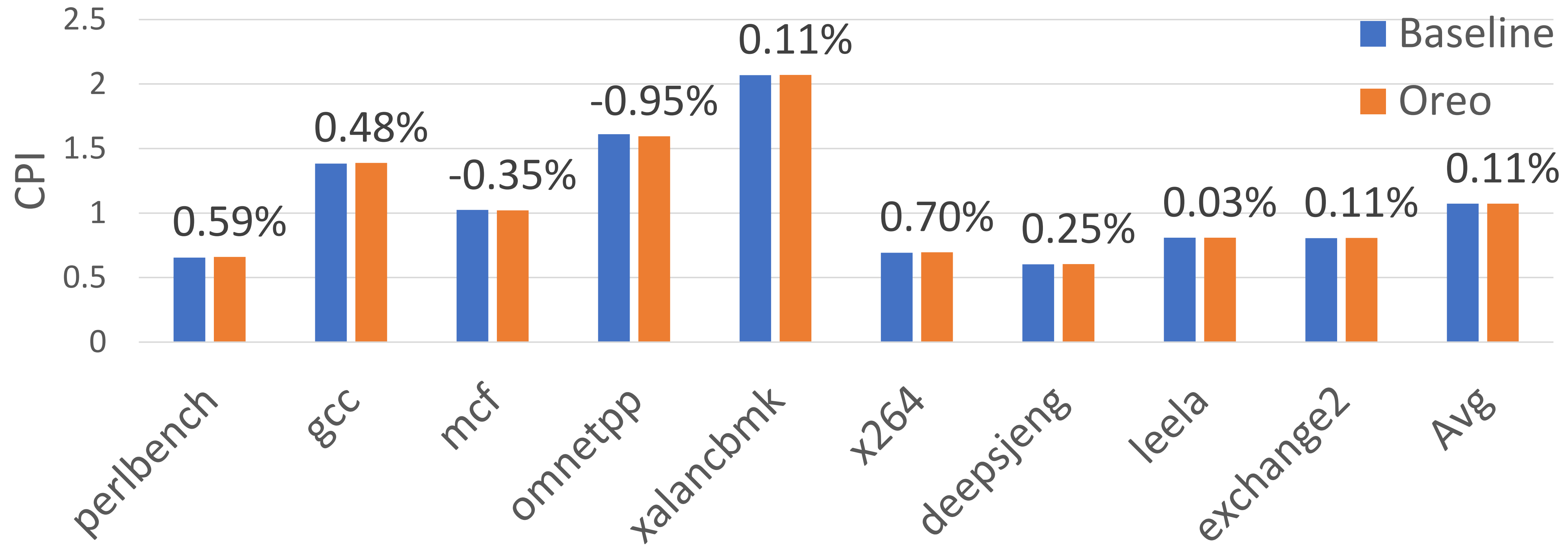
More in the Paper...

- Prototype
  - SW: Linux
  - HW: gem5 simulator
- Evaluation
  - Performance evaluation on SPEC and LEBench
  - Security evaluation on multiple leakage paths
- Formal reasoning of Oreo's security property (in extended version)

Performance Overhead: SPEC

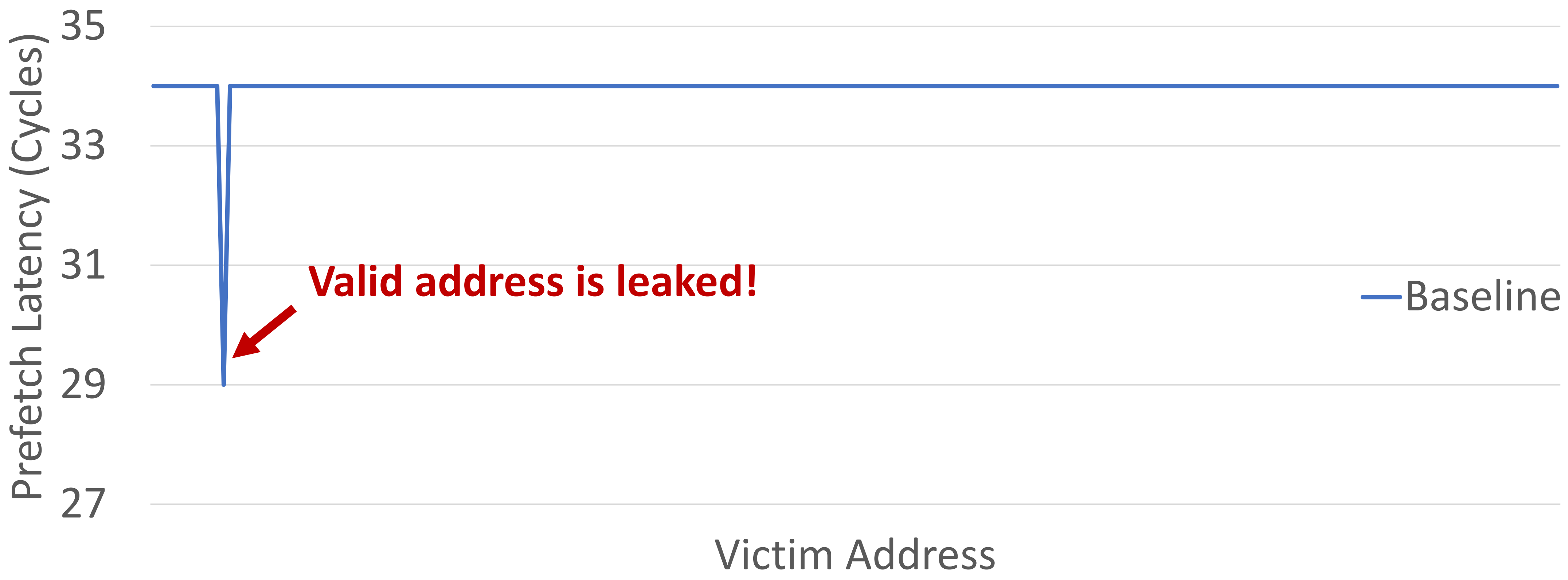


Performance Overhead: SPEC

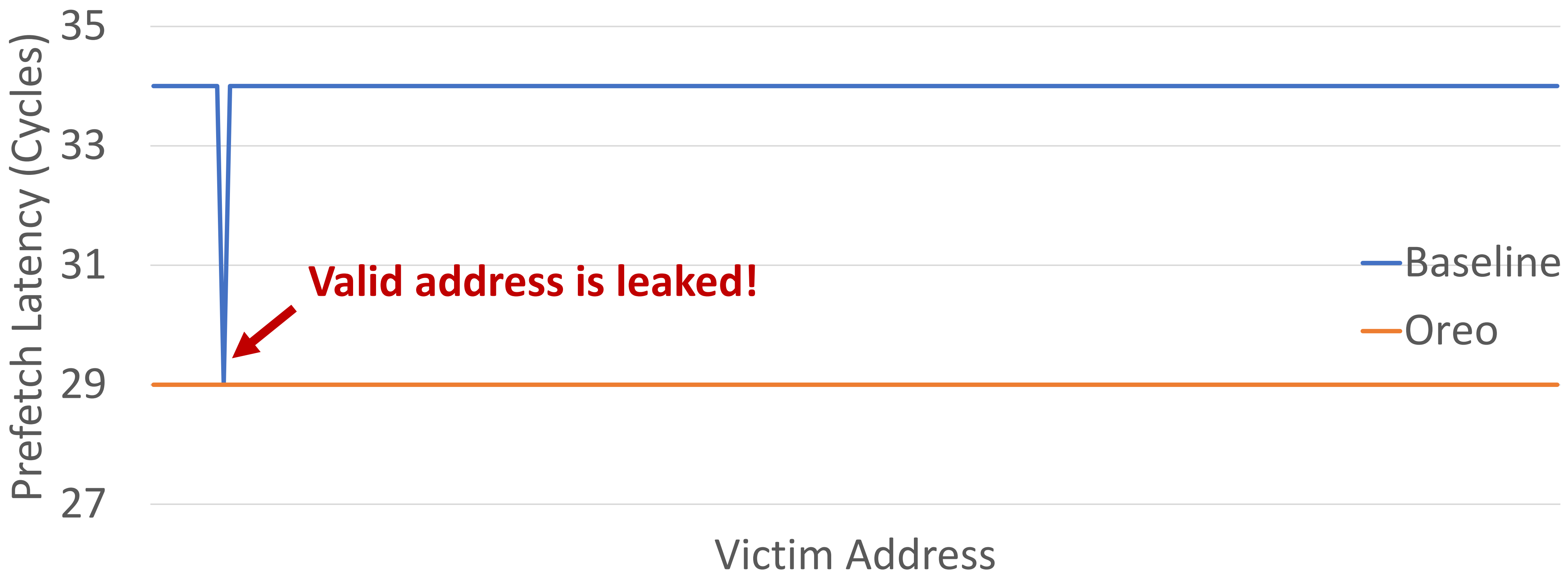


Oreo introduces negligible performance overhead.

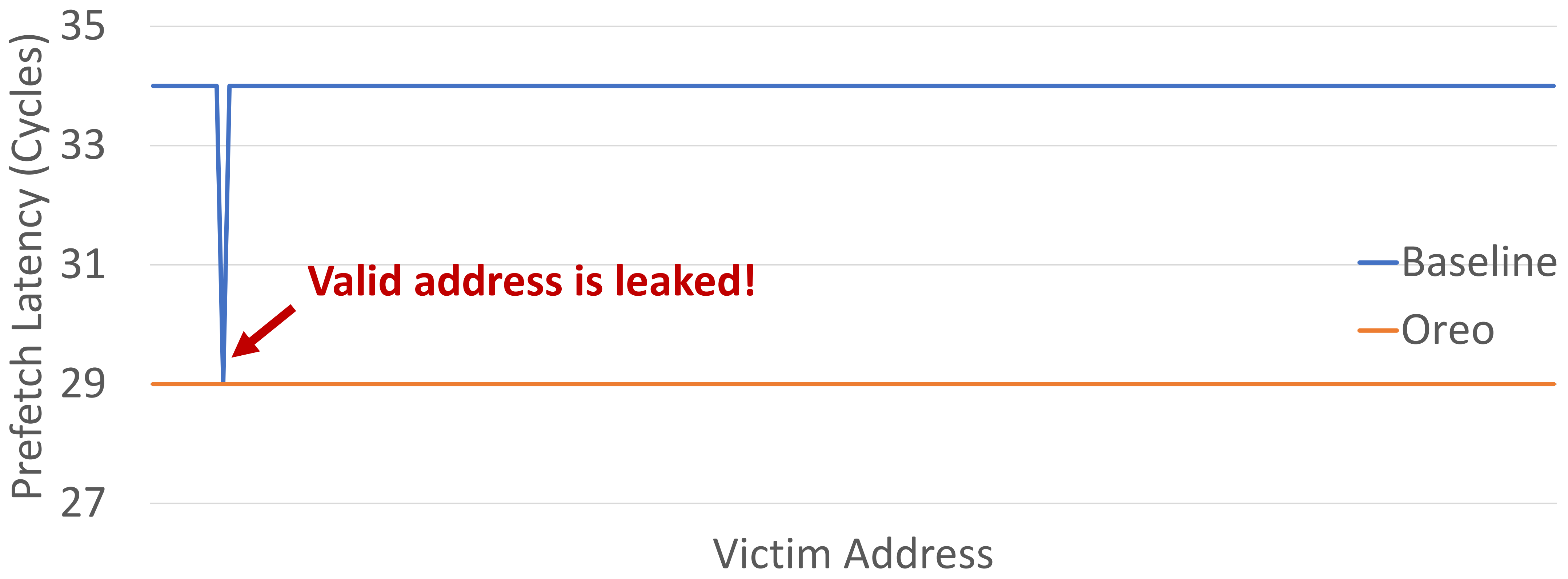
Security Evaluation: Prefetch Attack



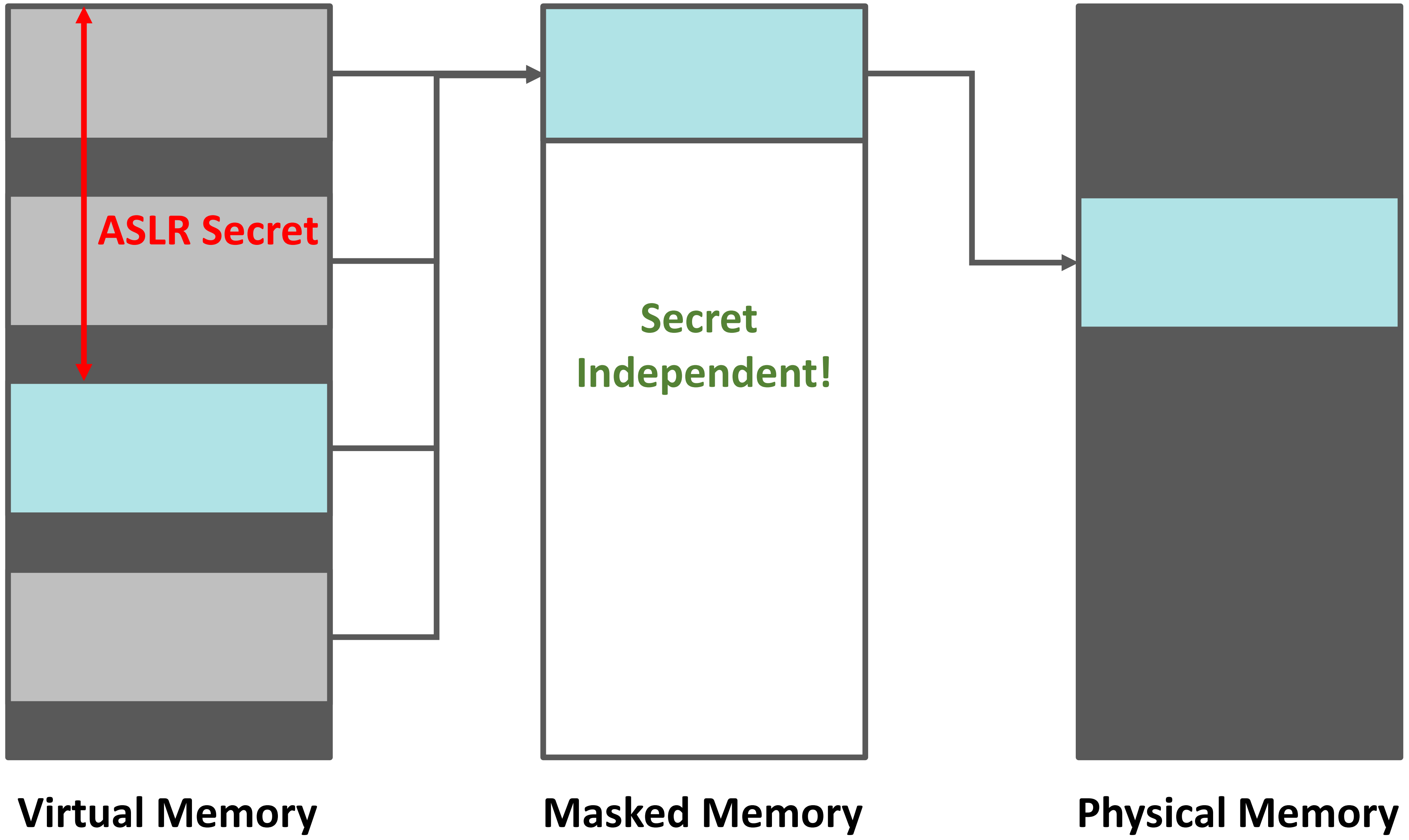
Security Evaluation: Prefetch Attack



Security Evaluation: Prefetch Attack



The prefetch attack no longer works on Oreo.


ASLR Secret
Secret Independent!
Virtual Memory
Masked Memory
Physical Memory