GoFetch: Breaking Constant-Time Cryptographic Implementations Using Data Memory-Dependent Prefetchers

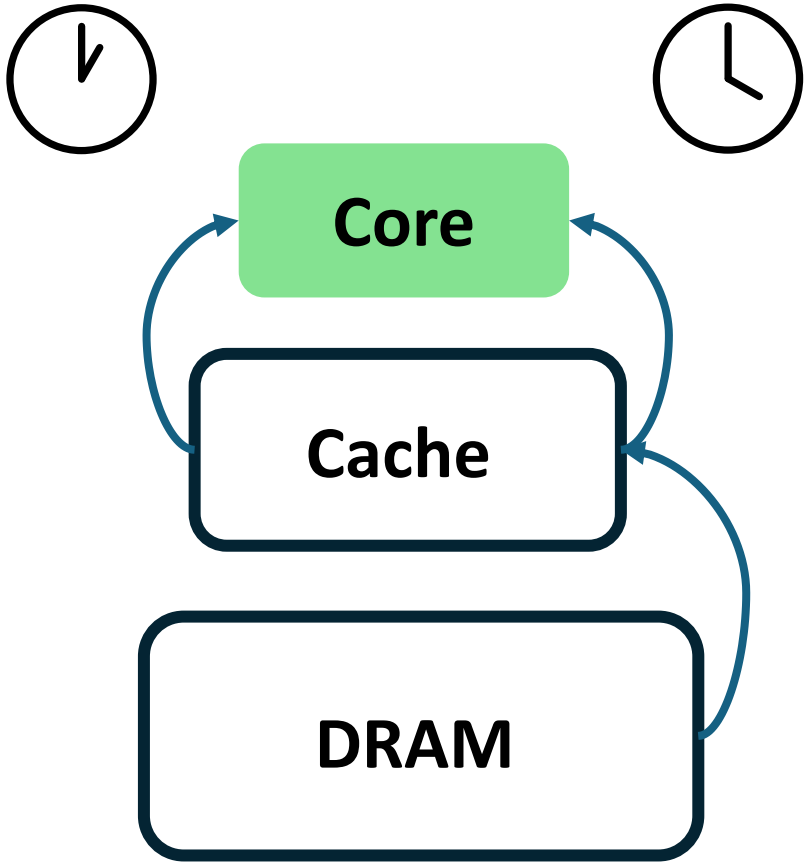
Boru Chen, Yingchen Wang, Pradyumna Shome,
Christopher Fletcher, David Kohlbrenner, Riccardo Paccagnella,
Daniel Genkin



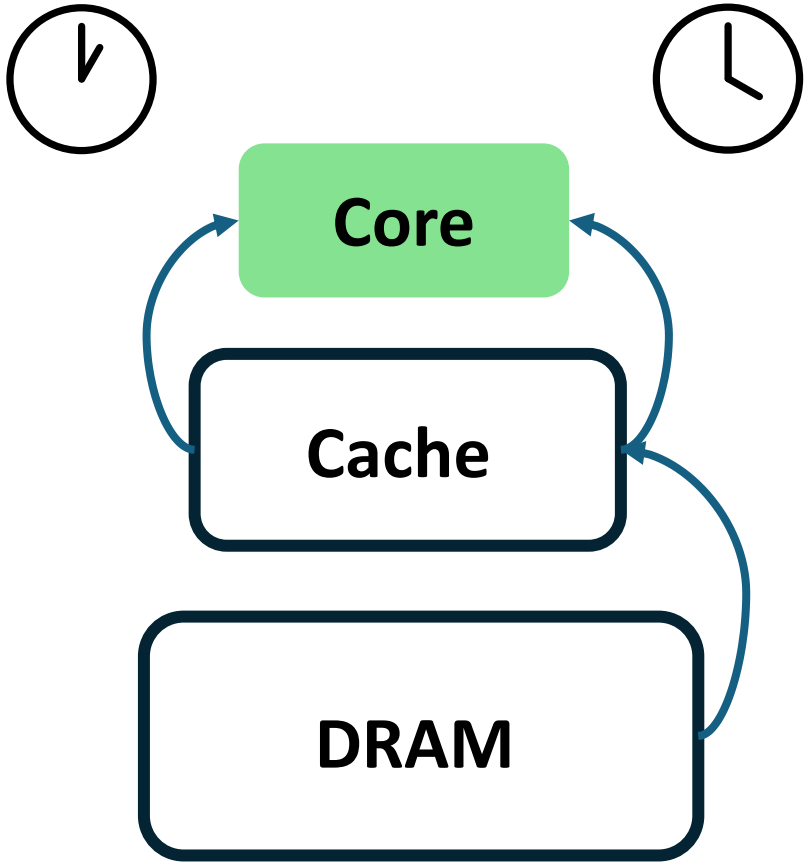

Carnegie Mellon University

Timing Attacks

Timing Attacks

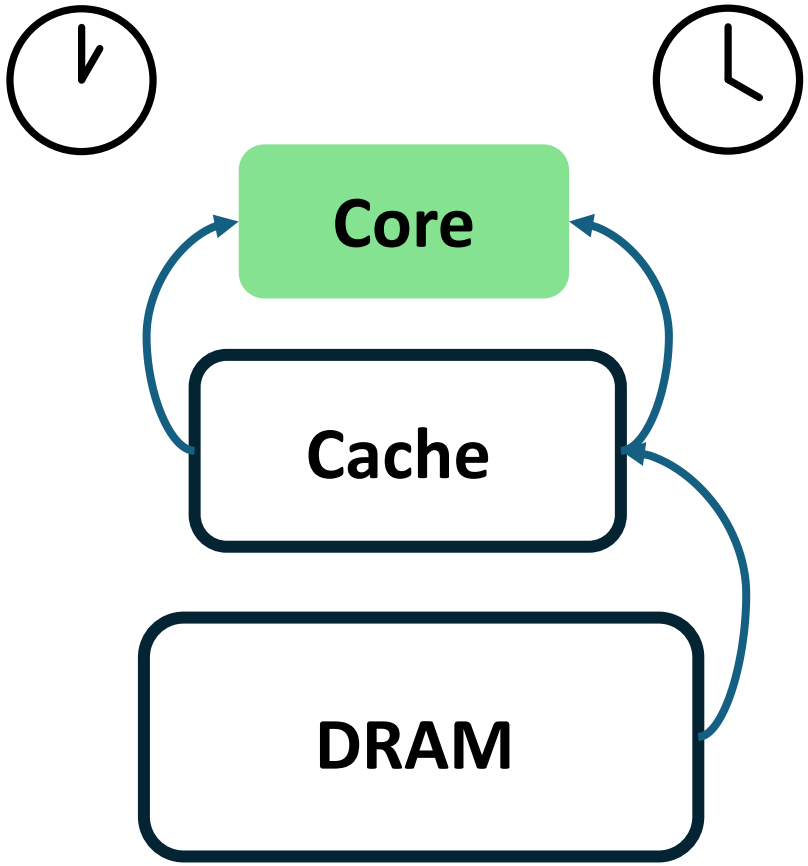


Timing Attacks




```
// secret = 1 or 0
if (secret)
{
  trash = *addr_A
}
```

Timing Attacks

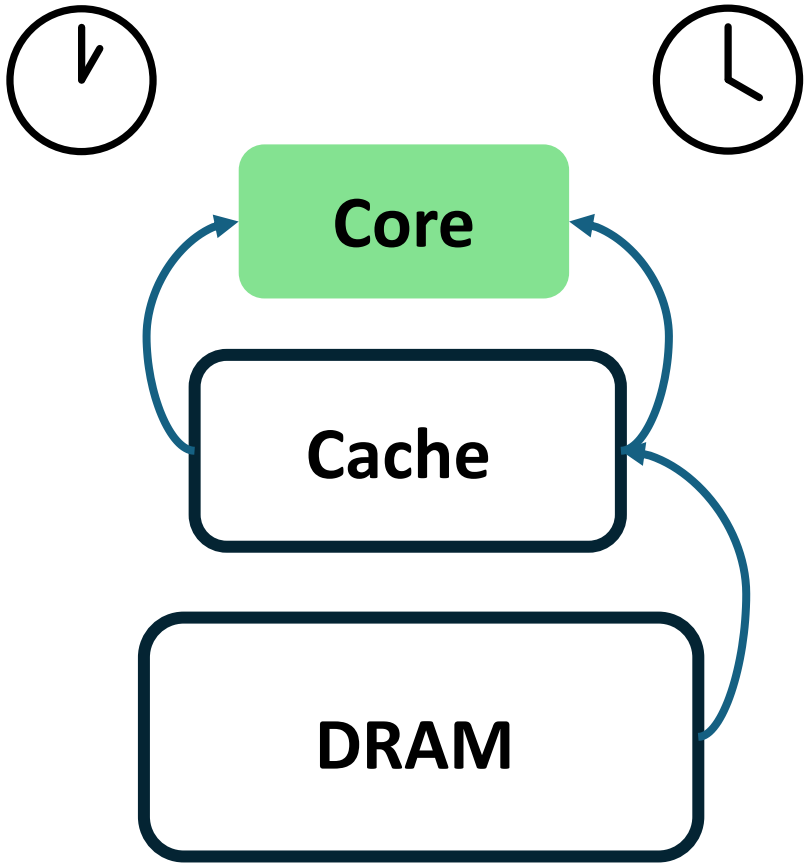



```
// secret = 1 or 0
if (secret)
{
  trash = *addr_A
}
```

Flush
Cache

Load
addr_A

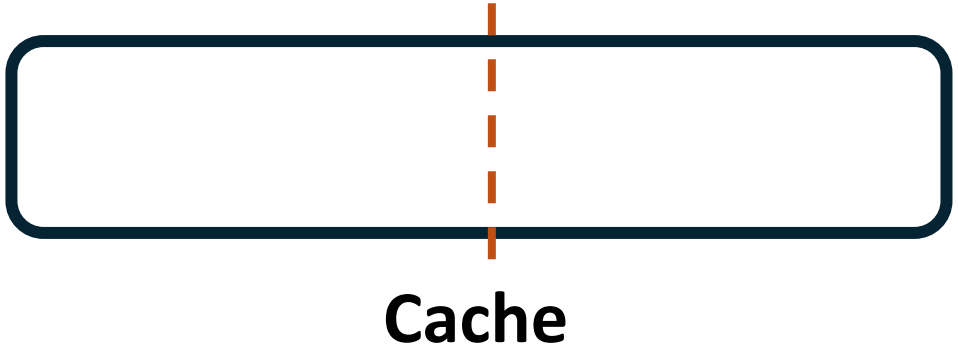
Timing Attacks



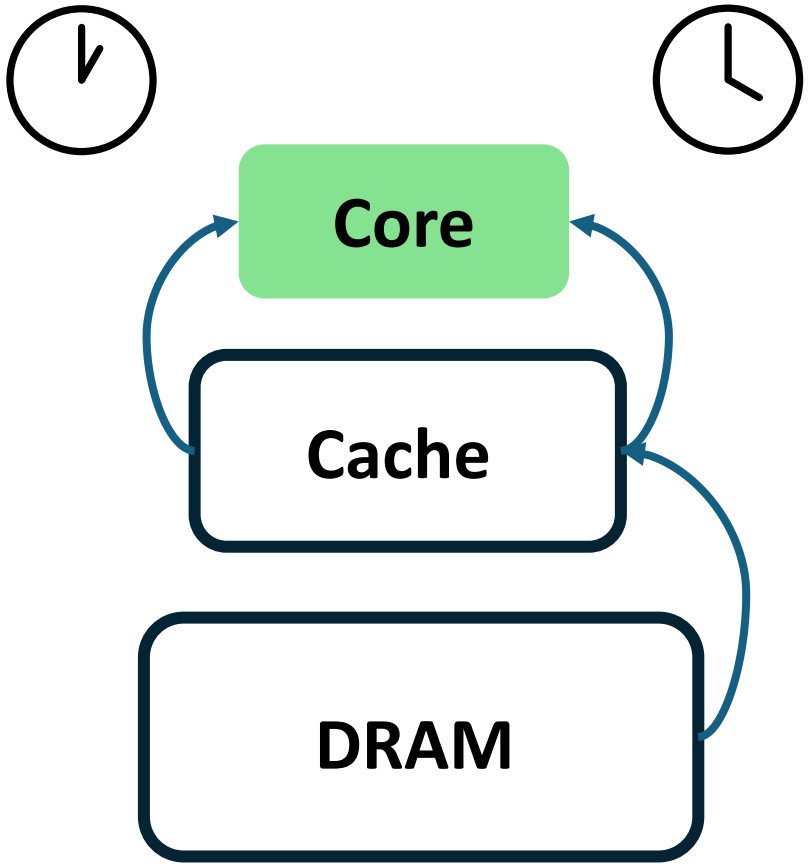

```
// secret = 1 or 0
if (secret)
{
  trash = *addr_A
}
```

Flush
Cache

Load
addr_A



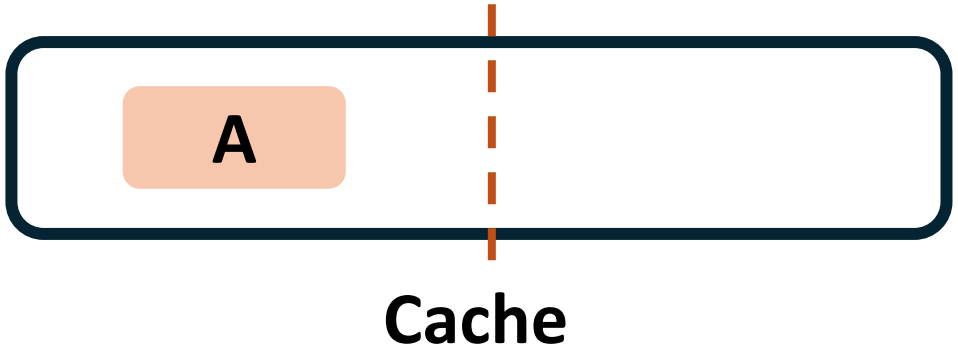
Timing Attacks




```
// secret = 1 or 0
if (secret)
{
  trash = *addr_A
}
```

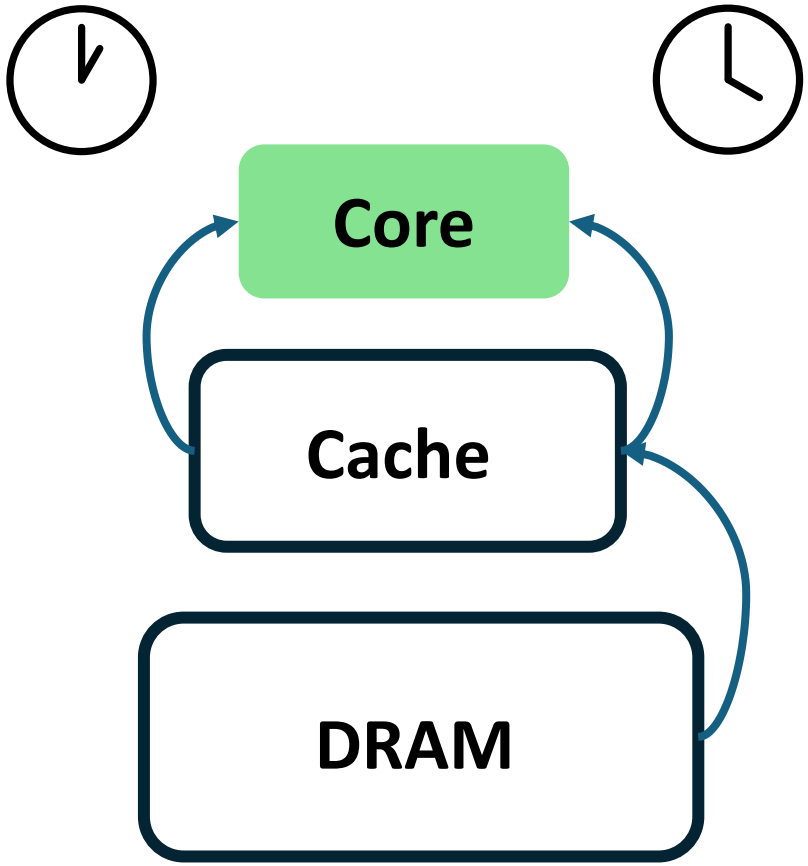
Flush
Cache

Load
addr_A



secret = 1

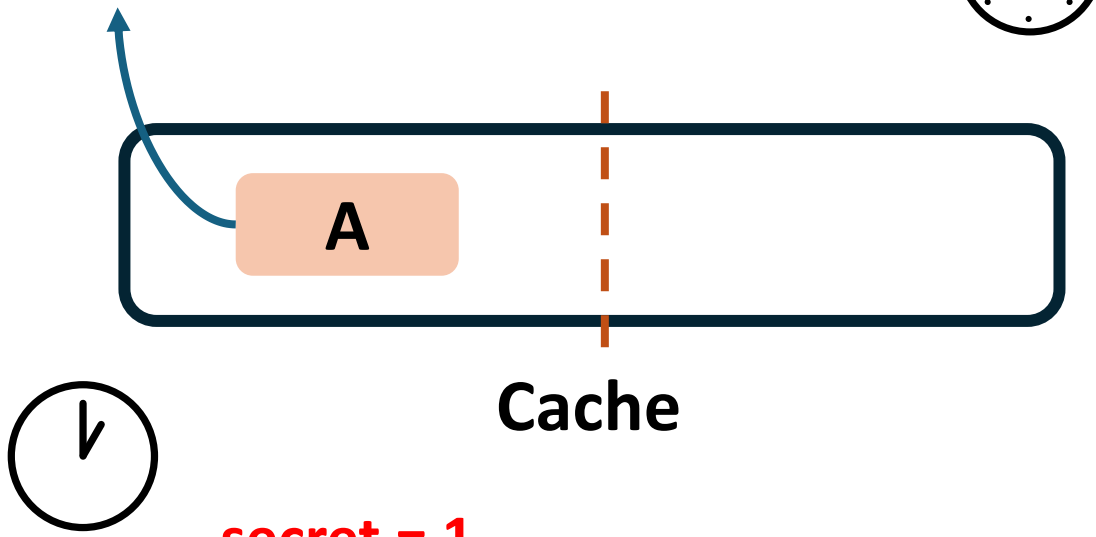
Timing Attacks




```
// secret = 1 or 0
if (secret)
{
  trash = *addr_A
}
```

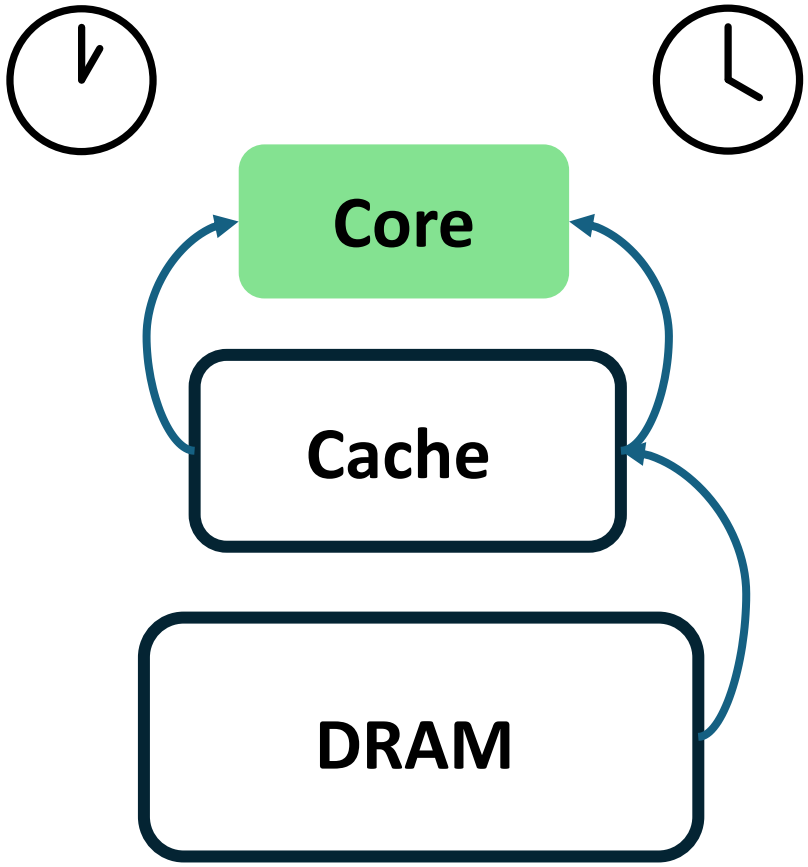
Flush Cache

Load addr_A



secret = 1

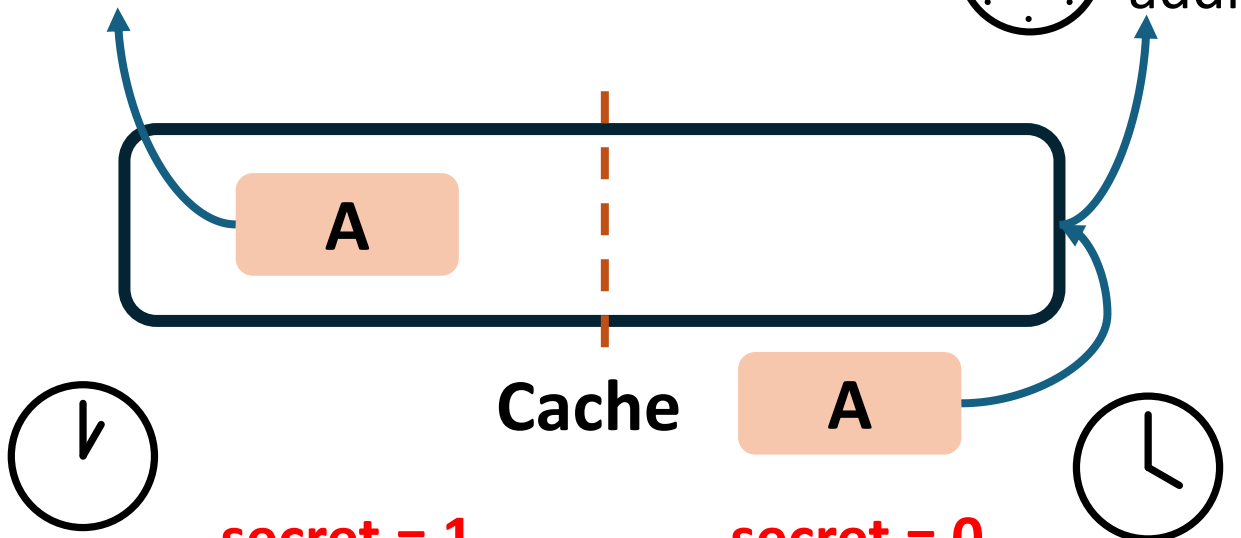
Timing Attacks



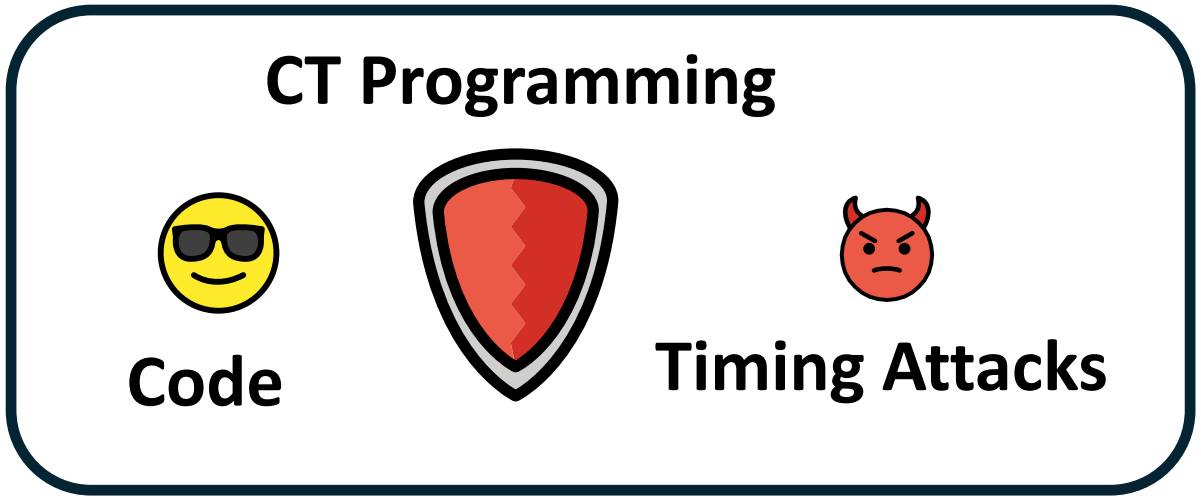

```
// secret = 1 or 0
if (secret)
{
   trash = *addr_A
}
```

Flush Cache

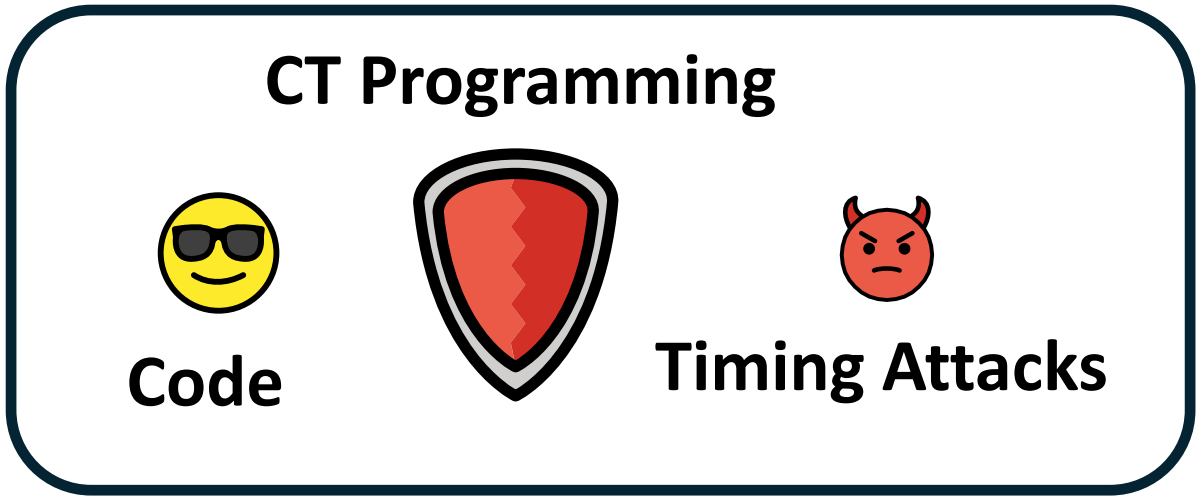
Load addr_A



Constant-Time Programming



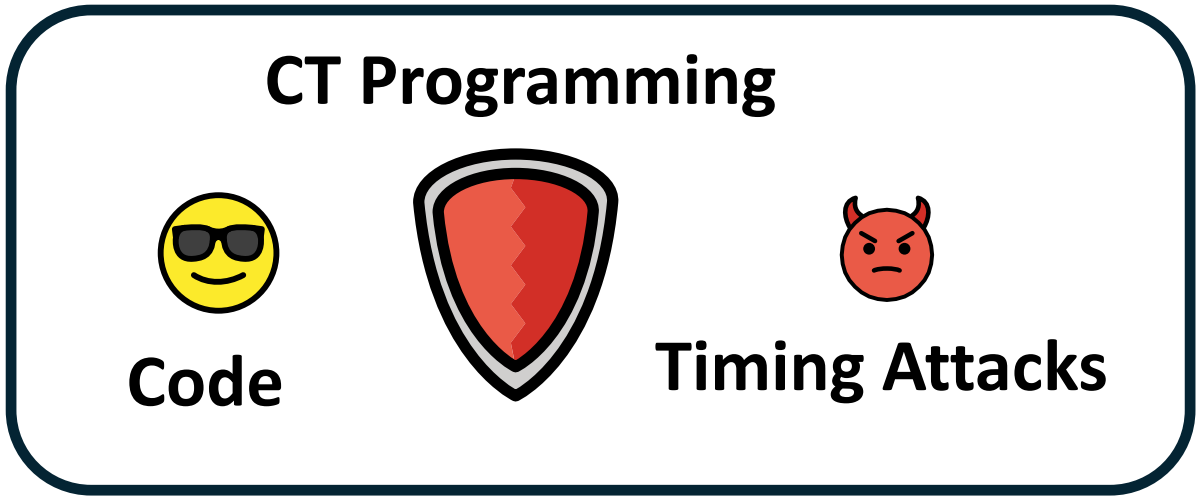
Constant-Time Programming



Independent

`secret` ⟷ `time`

Constant-Time Programming



Independent

`secret` ⟷ `time`

❌ **Control-Flow**

```
if (secret)
{
    ...
}
```

❌ **Time-Variable**

```
secret / 100.0
```

❌ **Memory Address**

```
trash = *secret
```

Constant-Time Programming

CT Programming

Code

Timing Attacks

Independent

`secret` ⟷ `time`

❌ **Control-Flow**

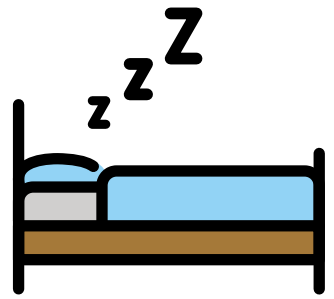
```
if (secret)
{
    ...
}
```

❌ **Time-Variable**

```
secret / 100.0
```

❌ **Memory Address**

```
trash = *secret
```



CT Cryptographic Implementation

Constant-Time Programming

<u>This talk:</u>

Show that these principles are insufficient.

Constant-Time Programming

<u>This talk:</u>

Show that these principles are insufficient.

Data Memory Dependent Prefetcher

Constant-Time Programming

<u>This talk:</u>

Show that these principles are insufficient.

Data Memory Dependent Prefetcher

```
secret = *non-sec-addr
secret = *non-sec-addr
secret = *non-sec-addr
secret = *non-sec-addr
```

program *without* secret-dependent load

Constant-Time Programming

<u>This talk:</u>

Show that these principles are insufficient.

Data Memory Dependent Prefetcher

```
secret = *non-sec-addr
secret = *non-sec-addr
secret = *non-sec-addr
secret = *non-sec-addr
```

*secret

*secret

program *without* secret-dependent load

Constant-Time Programming

This talk:

Show that these principles are insufficient.

Data Memory Dependent Prefetcher

```
secret = *non-sec-addr
secret = *non-sec-addr
secret = *non-sec-addr
secret = *non-sec-addr
```

*secret

*secret

Cache Timing Attacks

program *without* secret-dependent load

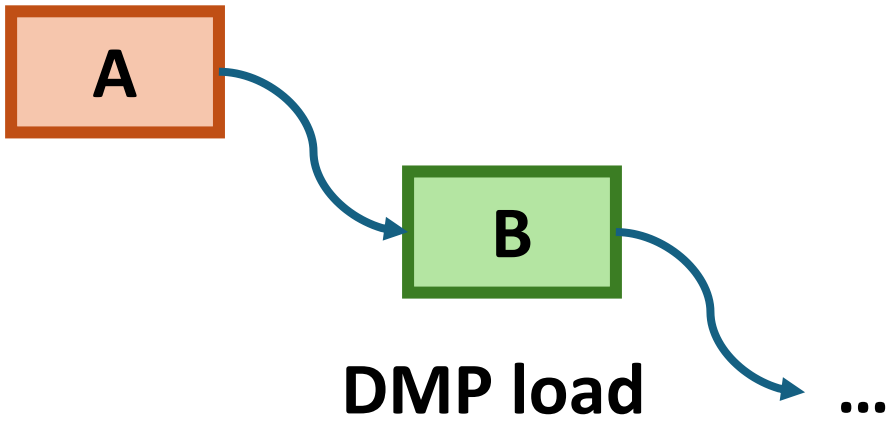
A Constant-Time Code Example

Apple DMP could **treat loaded data as memory address** and perform access.

A Constant-Time Code Example

Apple DMP could **treat loaded data as memory address** and perform access.

program load

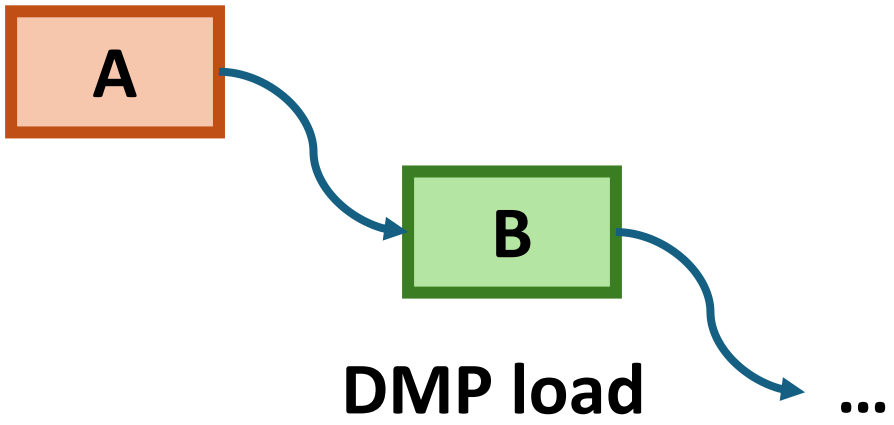


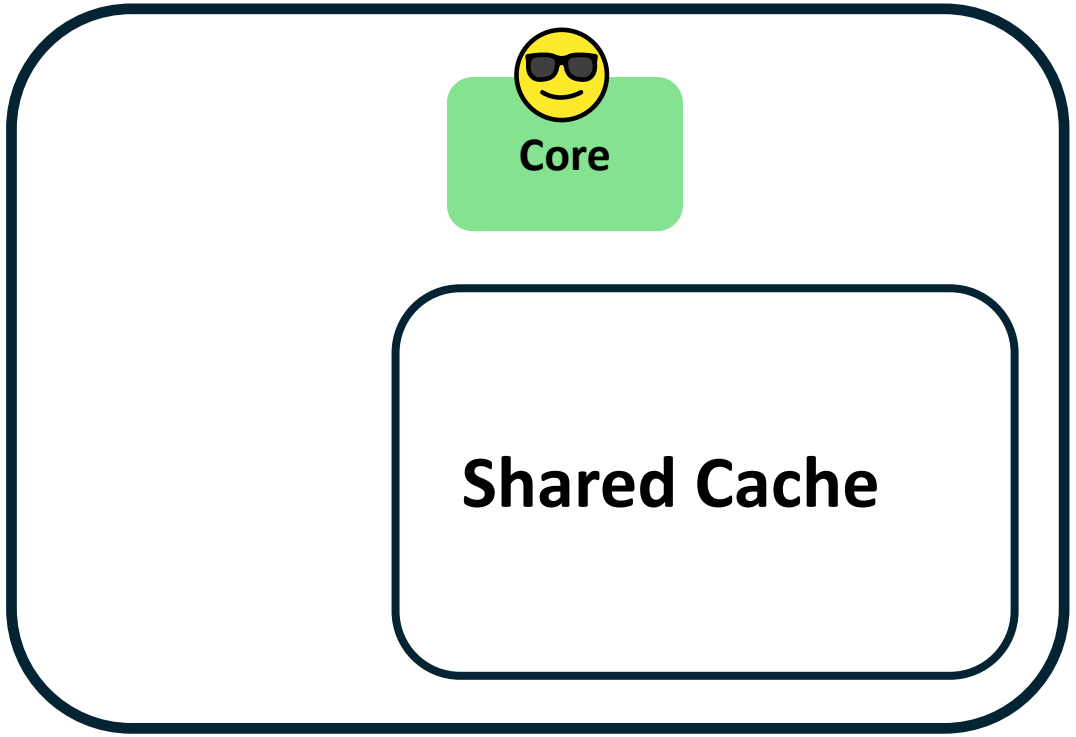
A Constant-Time Code Example

Apple DMP could **treat loaded data as memory address** and perform access.

```
// secret = ptr1 or ptr2
secret = *non-sec-addr
```

program load



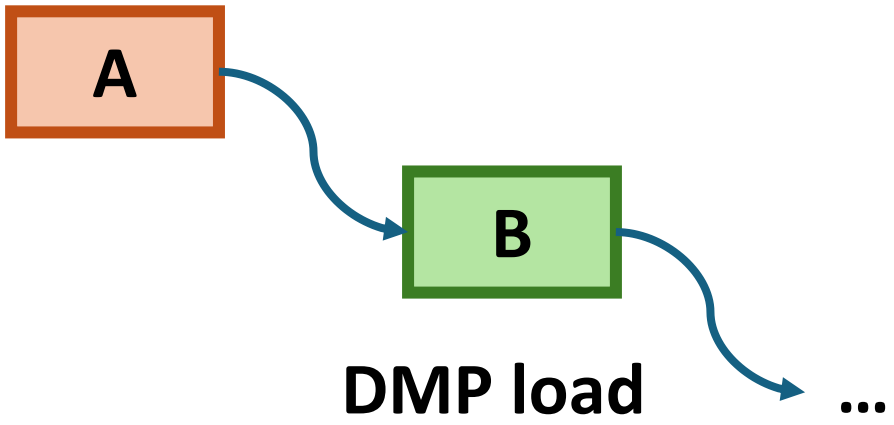


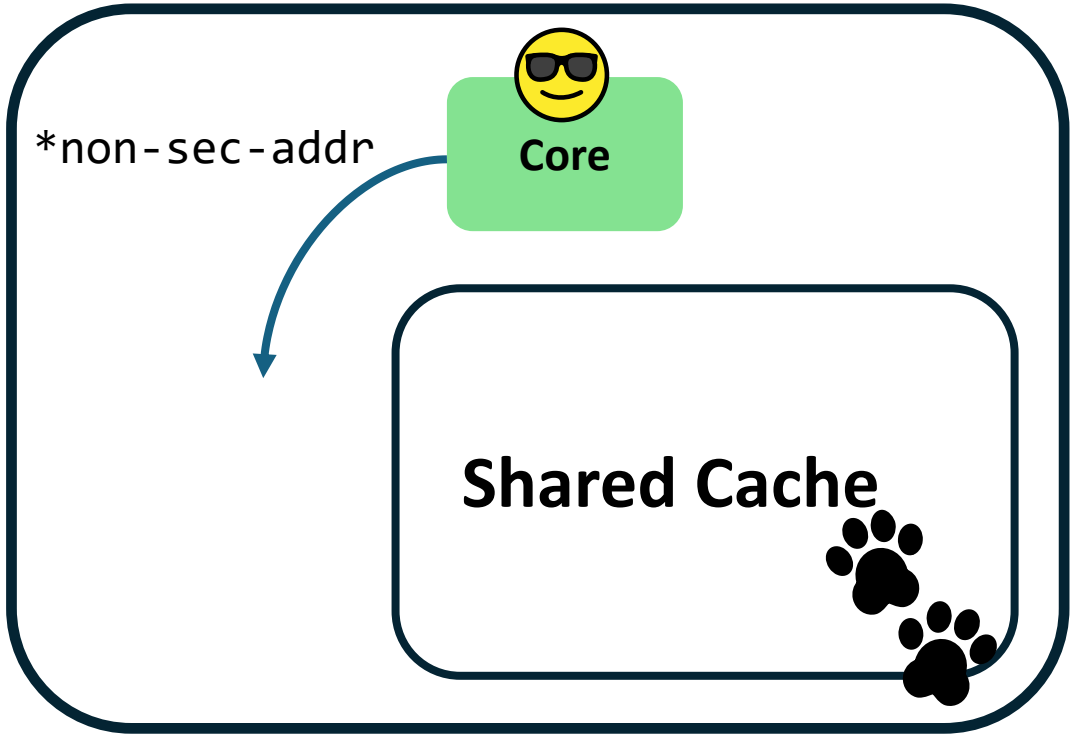
A Constant-Time Code Example

Apple DMP could **treat loaded data as memory address** and perform access.

```
// secret = ptr1 or ptr2
secret = *non-sec-addr
```

program load



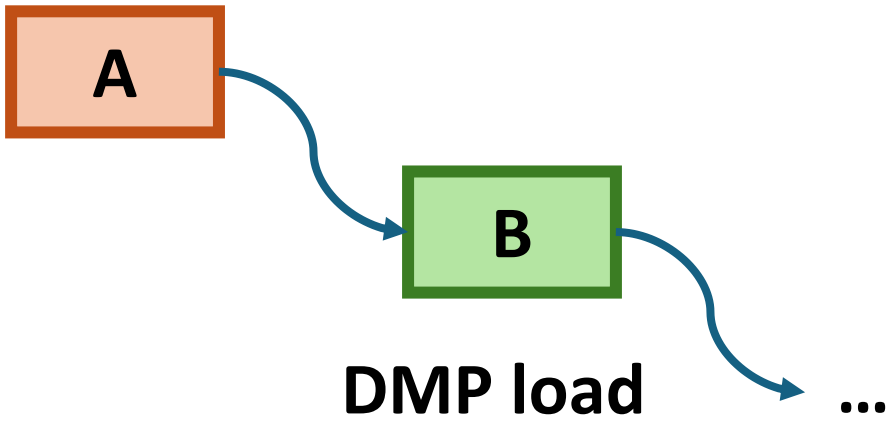


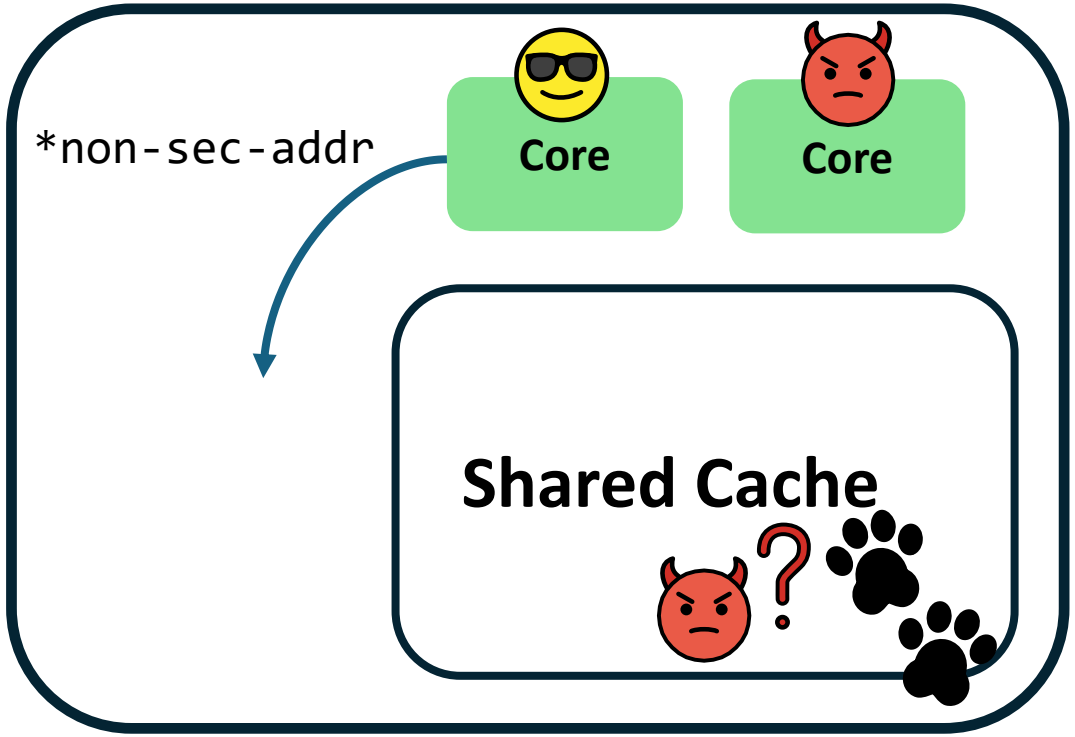
A Constant-Time Code Example

Apple DMP could **treat loaded data as memory address** and perform access.

```
// secret = ptr1 or ptr2
secret = *non-sec-addr
```

program load



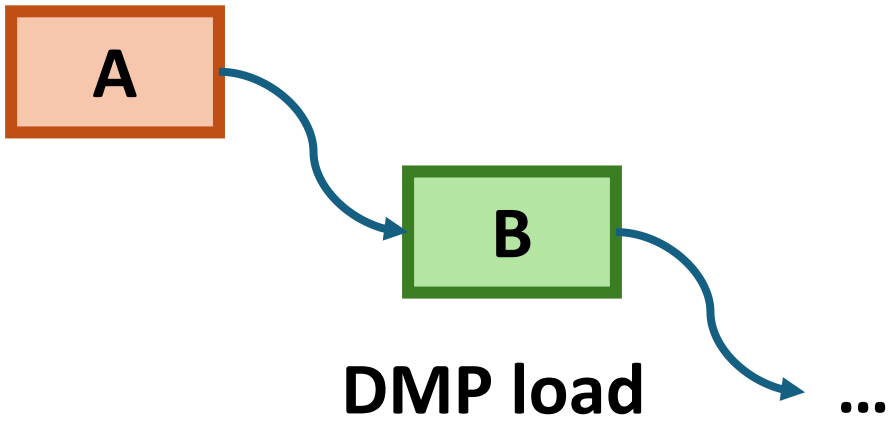


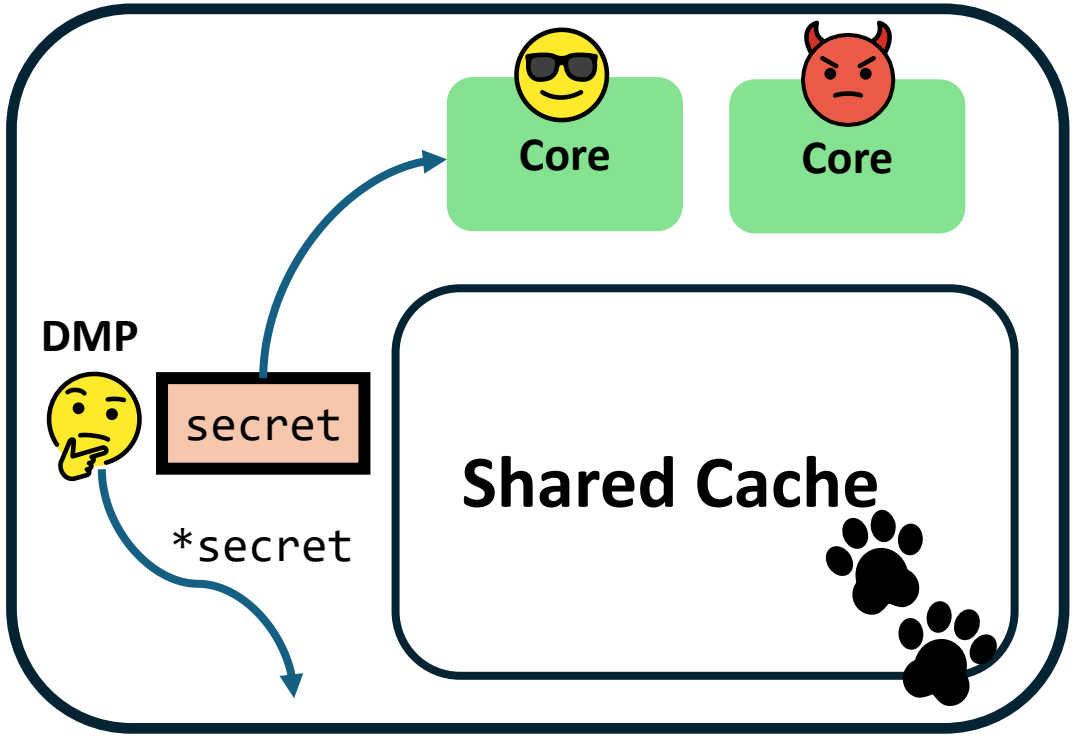
A Constant-Time Code Example

Apple DMP could **treat loaded data as memory address** and perform access.

```
// secret = ptr1 or ptr2
secret = *non-sec-addr
```

program load



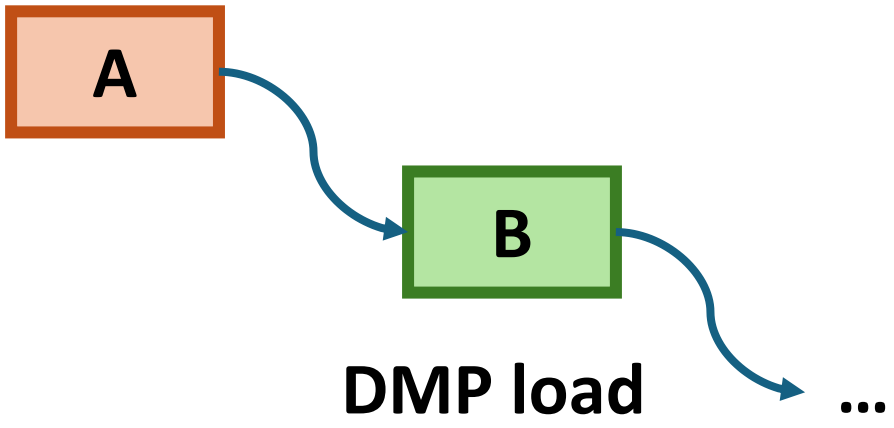


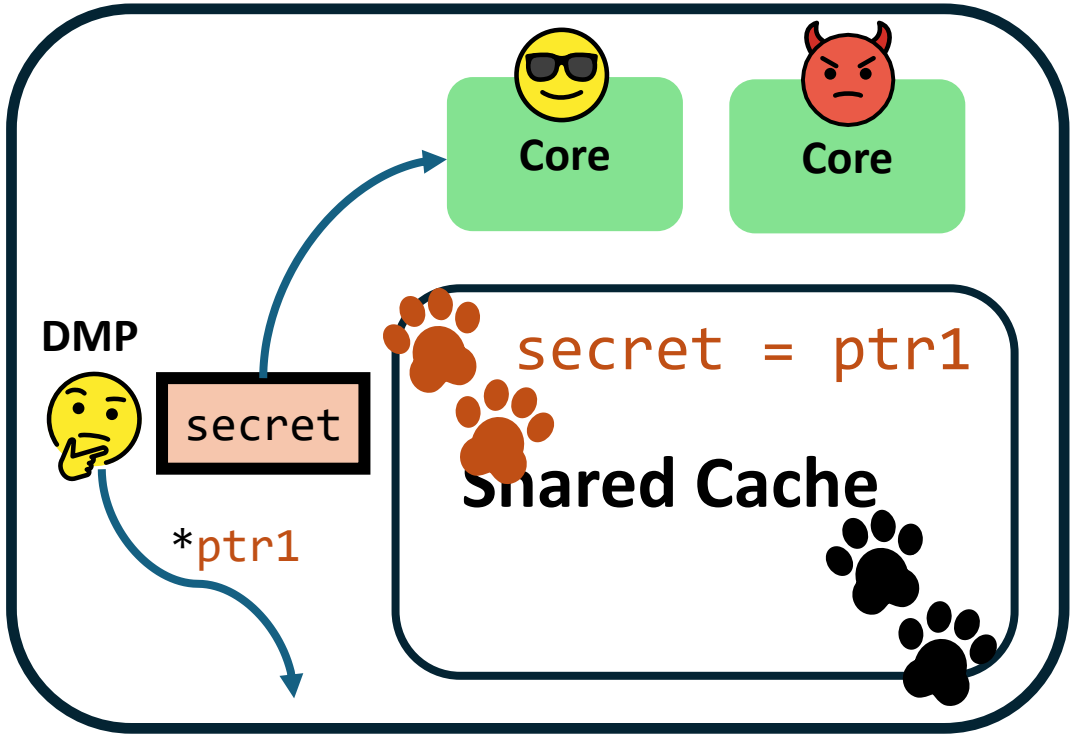
A Constant-Time Code Example

Apple DMP could **treat loaded data as memory address** and perform access.

```
// secret = ptr1 or ptr2
secret = *non-sec-addr
```

program load



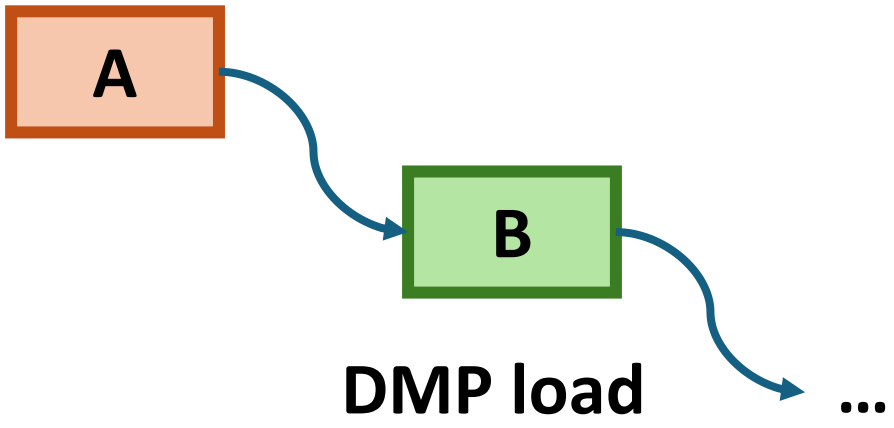


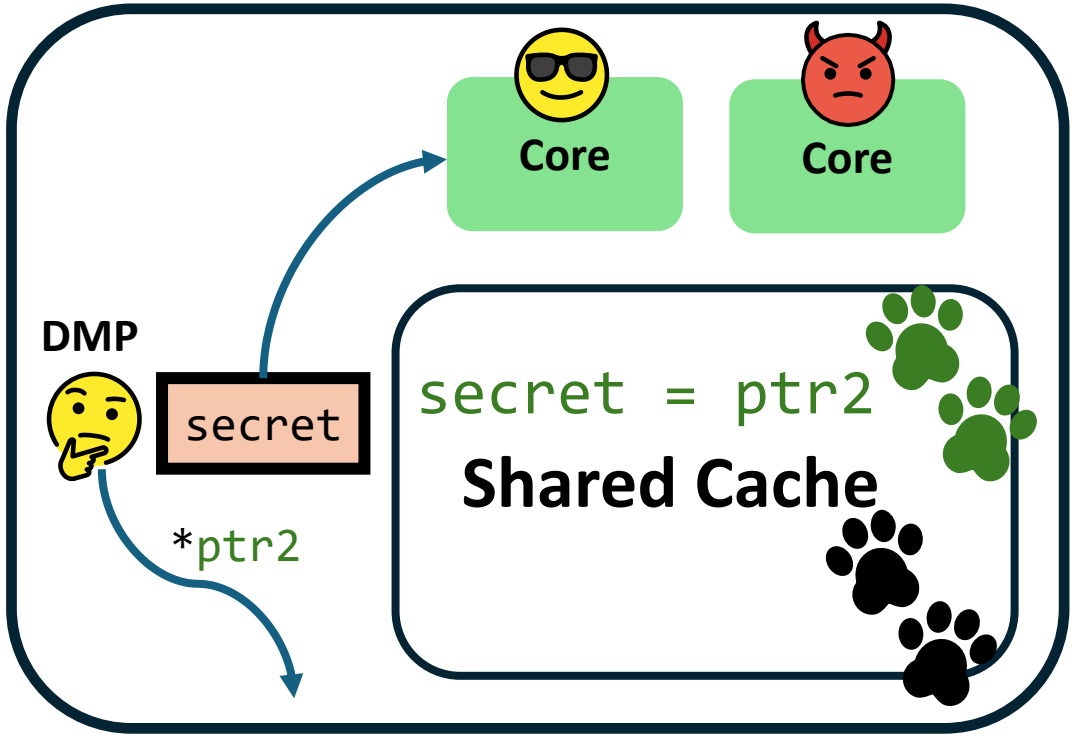
A Constant-Time Code Example

Apple DMP could **treat loaded data as memory address** and perform access.

```
// secret = ptr1 or ptr2
secret = *non-sec-addr
```

program load



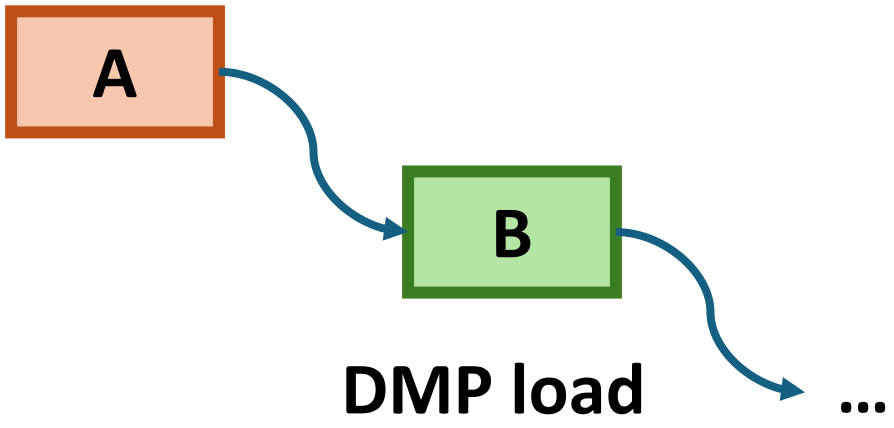


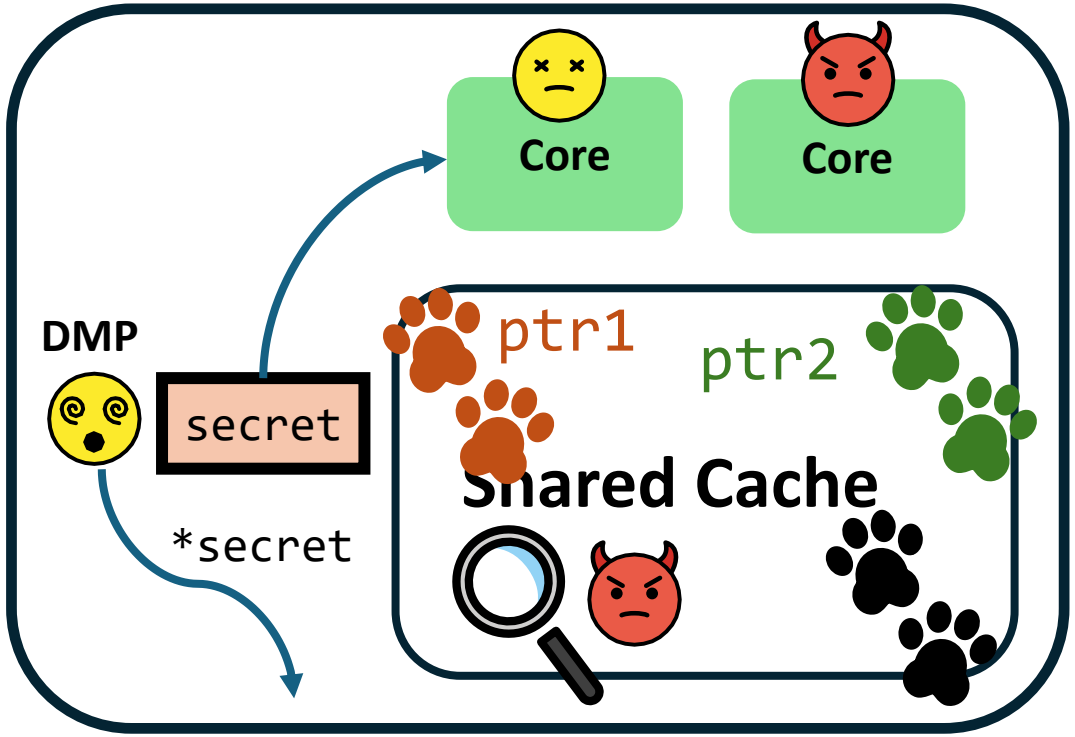
A Constant-Time Code Example

Apple DMP could **treat loaded data as memory address** and perform access.

```
// secret = ptr1 or ptr2
secret = *non-sec-addr
```

program load





Contribution

Augury[1]

1 Vicarte, et al., "Augury: Using data memory-dependent prefetchers to leak data at rest", SP'22.

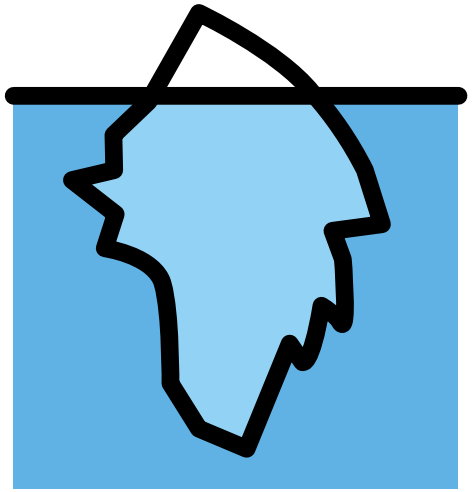
Contribution

- Comprehensive reverse engineering of Apple DMPs.

1 Vicarte, et al., "Augury: Using data memory-dependent prefetchers to leak data at rest", SP'22.

Contribution

- Comprehensive reverse engineering of Apple DMPs.
- Develop DMP-aided chosen-input attack framework.



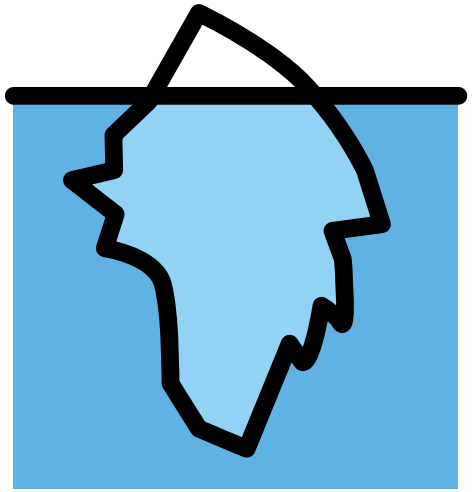
Augury[1]

This Work

1 Vicarte, et al., "Augury: Using data memory-dependent prefetchers to leak data at rest", SP'22.

Contribution

- Comprehensive reverse engineering of Apple DMPs.
- Develop DMP-aided chosen-input attack framework.
- Undermine four cryptographic implementations in the wild or submitted to NIST PQC standardization.



Augury[1]

This Work

1 Vicarte, et al., "Augury: Using data memory-dependent prefetchers to leak data at rest", SP'22.

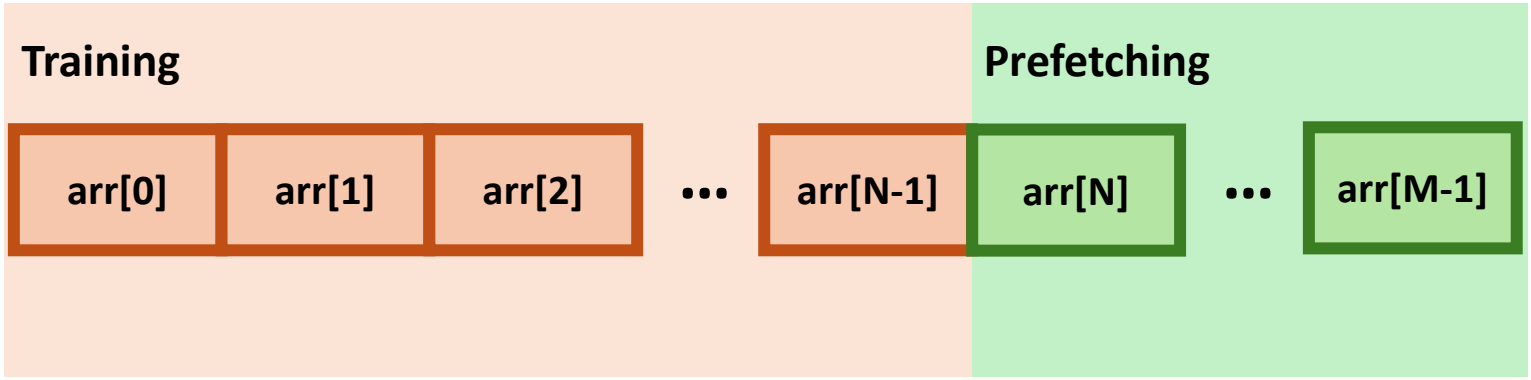
Key Observations of Apple DMP

How do classical prefetchers work?

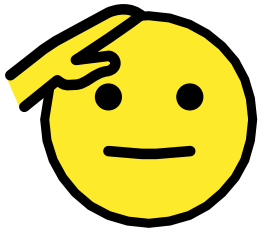
Key Observations of Apple DMP

How do classical prefetchers work?

```
// stride pattern
for (i = 0; i < M; i++)
    trash += arr[i];
```



The program reads arr[0], arr[1], …
The stride is 1! Prefetch arr[N], arr[N+1],…



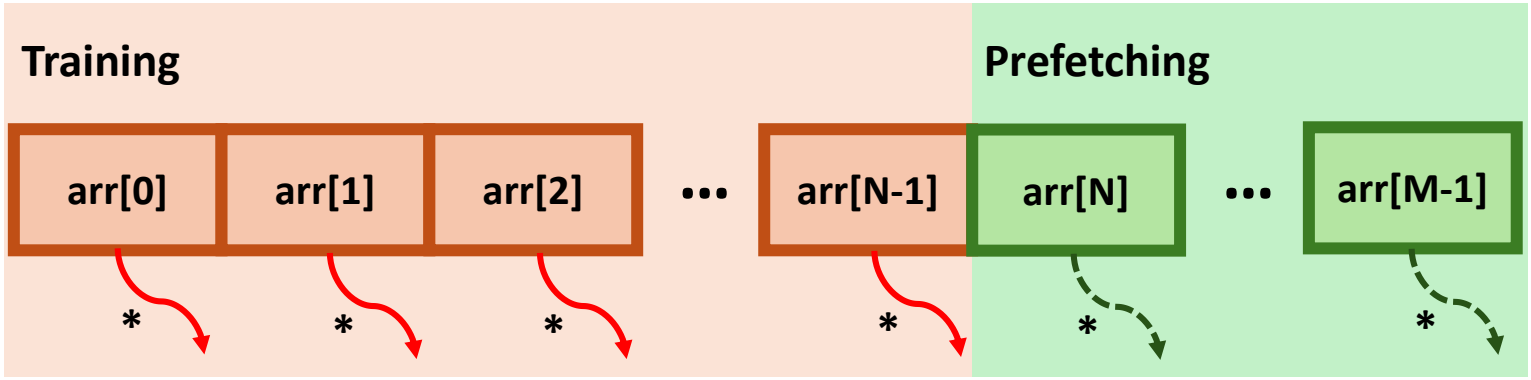
stride prefetcher

Key Observations of Apple DMP

What is the finding of prior work, Augury?

DMP recognizes and prefetches Array-of-pointers access pattern!

```
// Array-of-pointer pattern
for (i = 0; i < M; i++)
    trash += *arr[i];
```



Dereferenced by code

Dereferenced by DMP

Key Observations of Apple DMP

Does memory access pattern even matter?

```
// Array-of-pointer pattern
for (i = 0; i < M; i++)
    trash += *arr[i];
```

Really? Is it necessary?

GoFetch

Key Observations of Apple DMP

Does memory access pattern even matter?

Really? Is it necessary?

GoFetch

```
// Array-of-pointer pattern
for (i = 0; i < M; i++)
    trash += *arr[i];
```

```
// Single load
trash += arr[0];
```

Key Observations of Apple DMP

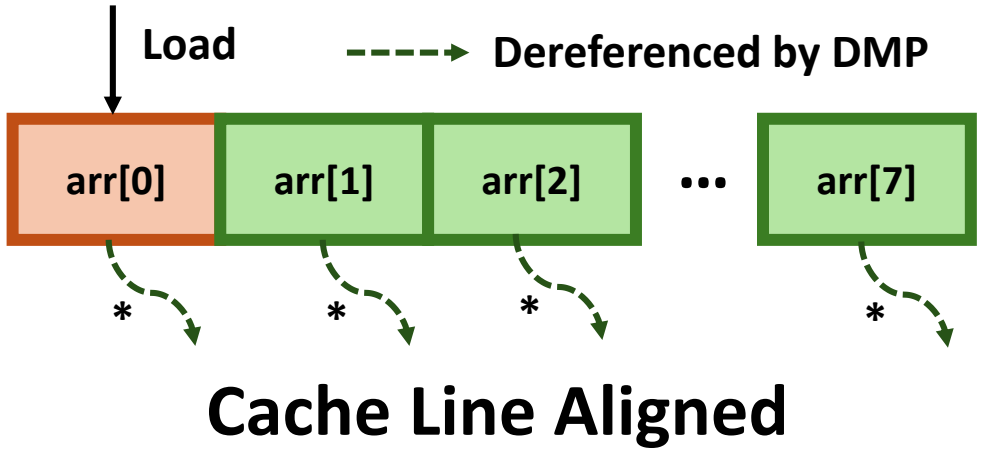
Does memory access pattern even matter?

Really? Is it necessary?

GoFetch

```
// Array-of-pointer pattern
for (i = 0; i < M; i++)
    trash += *arr[i];
```

```
// Single load
trash += arr[0];
```

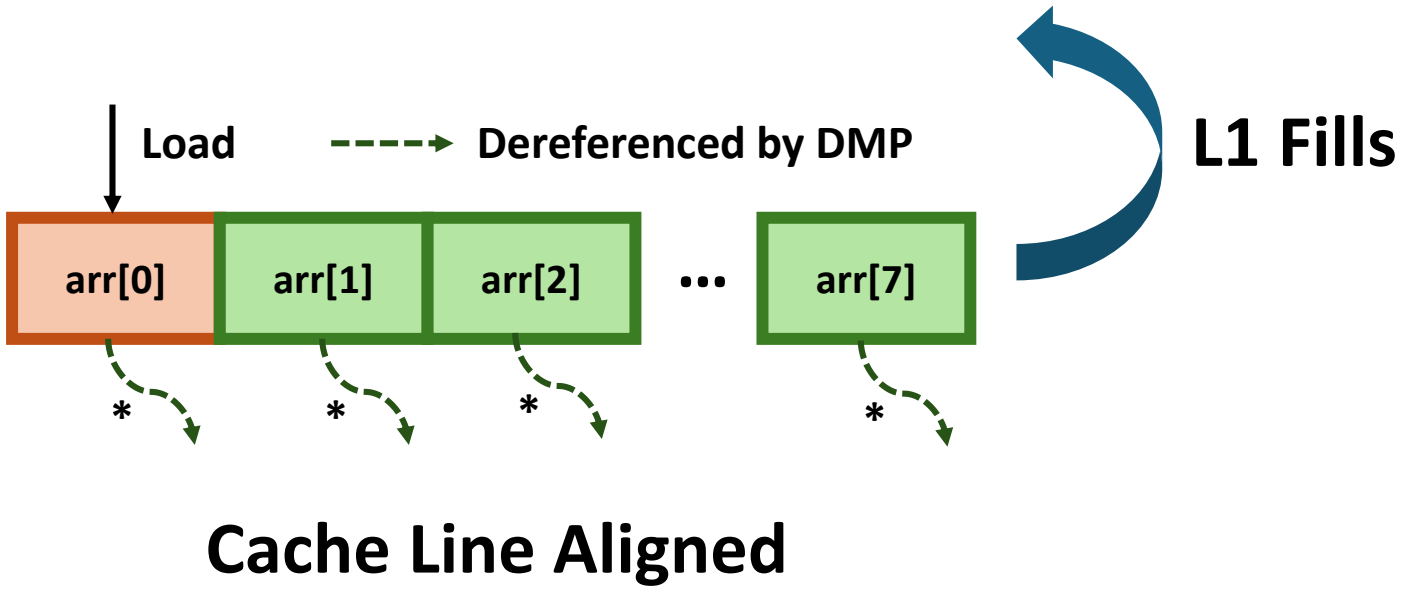


Cache Line Aligned

Key Observations of Apple DMP

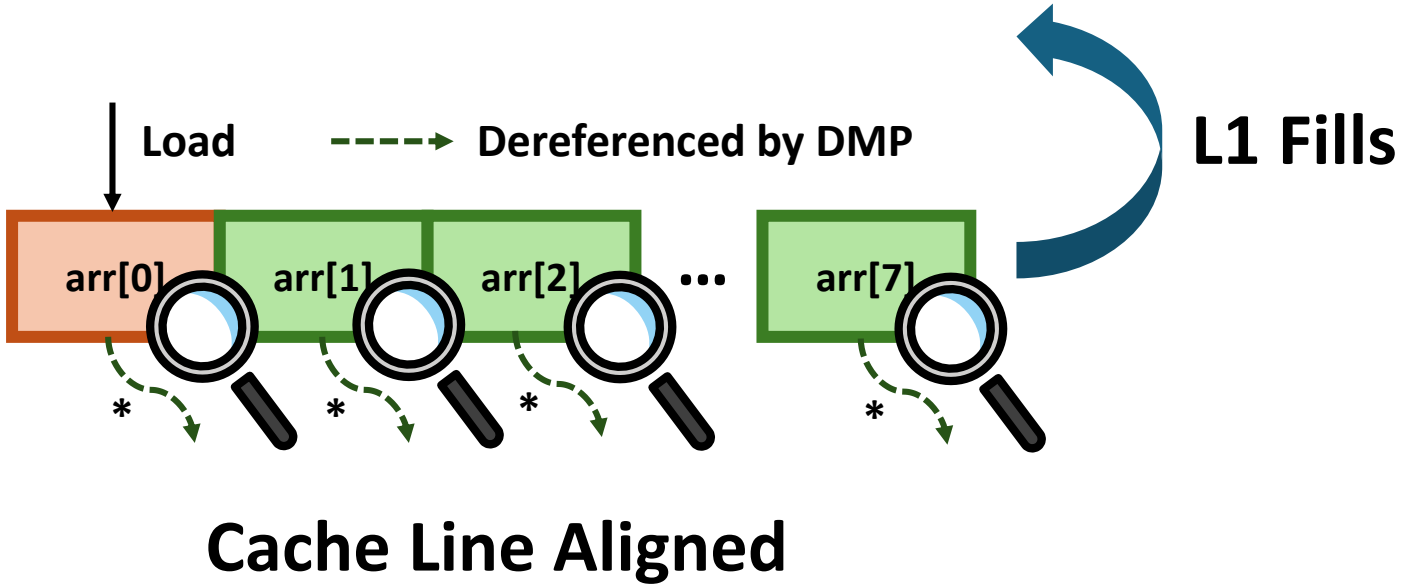
Where does the DMP scan for pointers?

```
// Single load
trash += arr[0];
```



Key Observations of Apple DMP

How does DMP determine pointers to dereference in each line?



Key Observations of Apple DMP

History filter: ***how DMP avoids redundant dereference?***

4GByte region: ***heuristic of predicting pointer value.***

Do-not-scan hint: ***how DMP avoids redundant scan?***

Top byte ignore: ***how DMP synergizes with TBI?***

… **…**

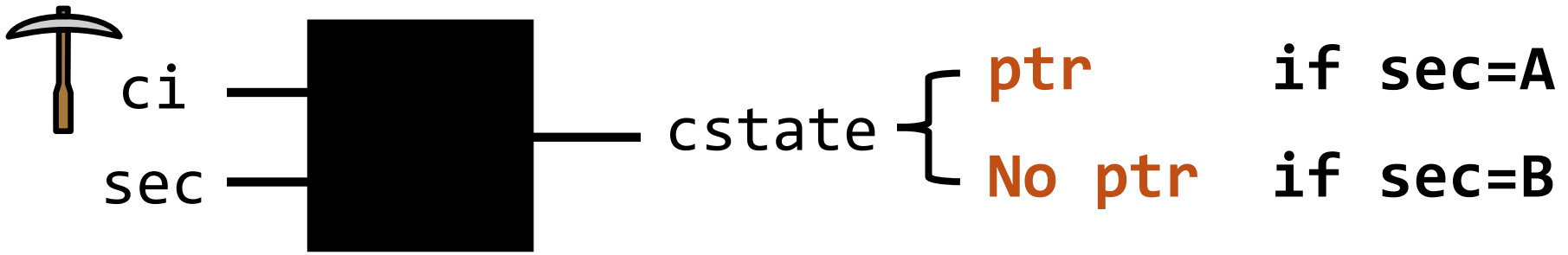
Check out the paper!

How to use DMP to break CT Crypto?

How to use DMP to break CT Crypto?

`cstate`: crypto state

`sec`: secret `ci`: chosen input



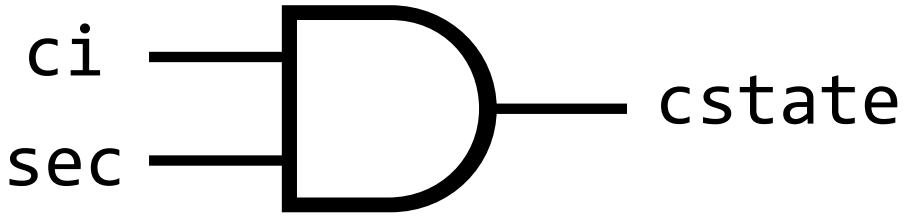
How to use DMP to break CT Crypto?

`cstate`: crypto state

`sec`: secret `ci`: chosen input

```
cstate = ci AND sec
```

```
sec { 0xffffffffffffffff
      0x0000000000000000
```



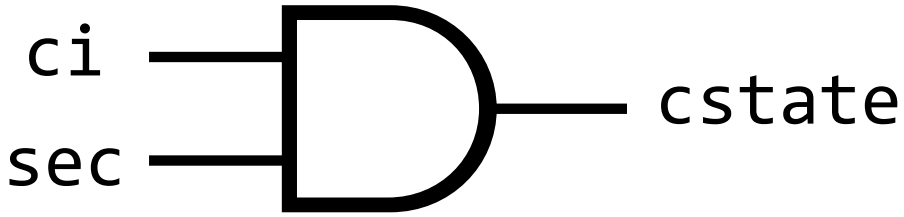
How to use DMP to break CT Crypto?

`cstate`: crypto state

`sec`: secret `ci`: chosen input

```
cstate = ci AND sec
```

```
sec { 0xffffffffffffffff
      0x0000000000000000
```



Choose `ci` as valid pointer!

```
cstate = ptr AND sec
if sec = 0xffffffffffffffff
   => cstate = ptr

if sec = 0x0000000000000000
   => cstate = 0
```

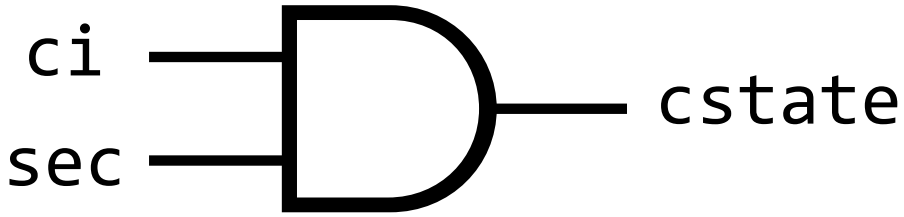
How to use DMP to break CT Crypto?

`cstate`: crypto state

`sec`: secret `ci`: chosen input

```
cstate = ci AND sec
```

`sec` {
- `0xffffffffffffffff`
- `0x0000000000000000`



Choose `ci` as valid pointer!

```
cstate = ptr AND sec
if sec = 0xffffffffffffffff
    => cstate = ptr
```

DMP  *ptr

```
if sec = 0x0000000000000000
    => cstate = 0
```

DMP 

Proof-of-Concept Attacks

Cryptanalysis for DMP exploit

End-to-end key extraction PoCs

Proof-of-Concept Attacks

Cryptanalysis for DMP exploit

End-to-end key extraction PoCs

OpenSSL DHKE

Proof-of-Concept Attacks

Cryptanalysis for DMP exploit

End-to-end key extraction PoCs

OpenSSL DHKE

Go RSA

Proof-of-Concept Attacks

Cryptanalysis for DMP exploit

End-to-end key extraction PoCs

ML-DSA

ML-KEM

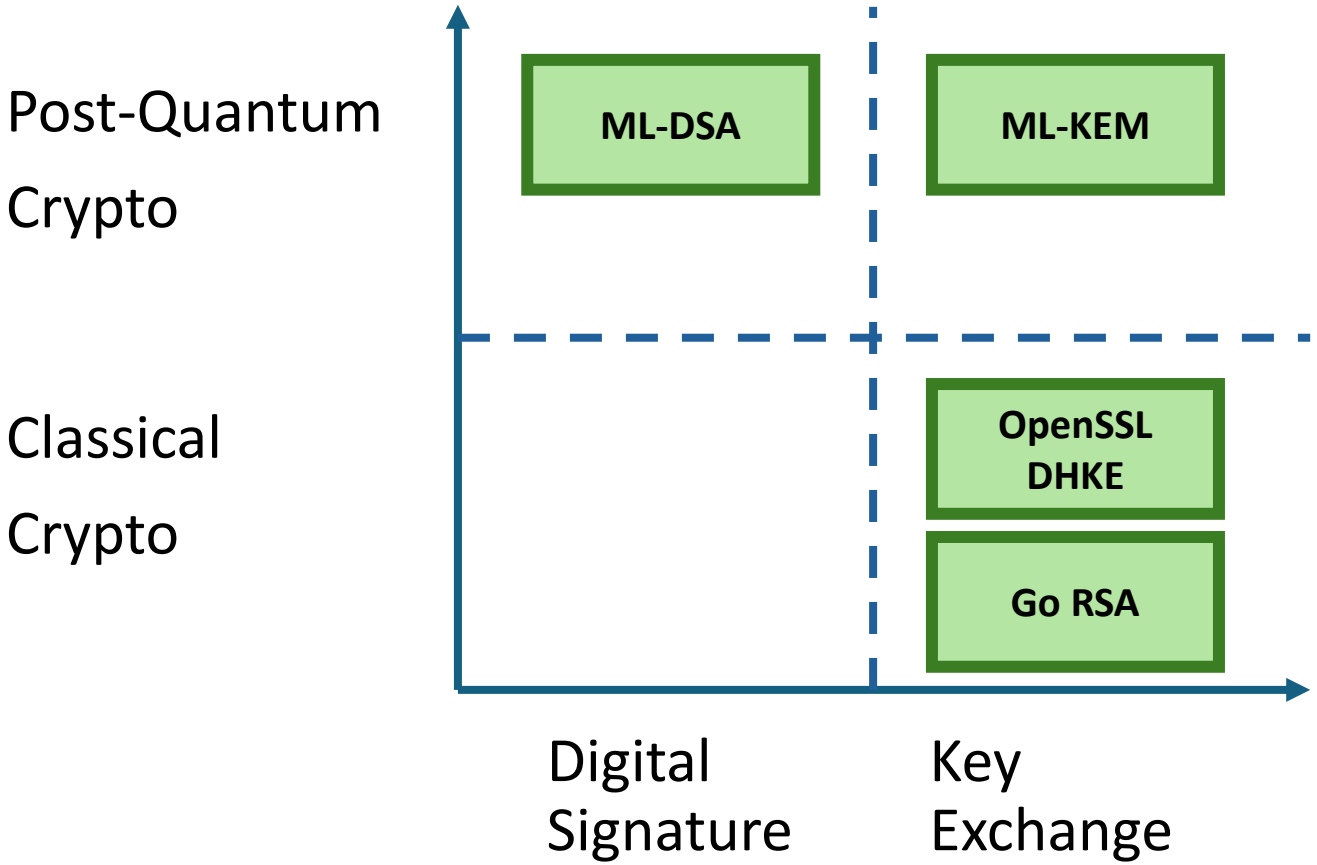
OpenSSL DHKE

Go RSA

Proof-of-Concept Attacks

Cryptanalysis for DMP exploit

End-to-end key extraction PoCs



Go RSA

$$N = p \times q$$

Go RSA

RSA Decryption

$$m \equiv c^d \ mod \ N$$

$$N = p \times q$$

Go RSA

RSA Decryption

$$m \equiv c^d \ mod \ N$$

$$N = p \times q$$

CRT Optimization

$$c^{D_p} \ mod \ p$$

$$c^{D_q} \ mod \ q$$

Go RSA

RSA Decryption

$$m \equiv c^d \ mod \ N$$

$$N = p \times q$$

CRT Optimization

$$c^{D_p} \ mod \ p$$

$$c^{D_q} \ mod \ q$$

First Step

$$c \ mod \ p$$

Go RSA

RSA Decryption

$$m \equiv c^d \ mod \ N$$

$$N = p \times q$$

CRT Optimization

$$c^{D_p} \ mod \ p$$

$$c^{D_q} \ mod \ q$$

First Step

$$c \ mod \ p$$

Observations of crypto state $c \ mod \ p$:

- c is chosen input and p is secret.
- If $c < p$, then $c \ mod \ p = c$, else $c \bmod p = c - lp$.

Go RSA

Target i-th bit of p and set input as $c = prefix \parallel 10000 \ldots \parallel ptr$!

c $\quad$ $\boldsymbol{prefix} \parallel \mathbf{1}0000 \ldots \parallel \boldsymbol{ptr}$

p $\quad$ $\boldsymbol{prefix} \parallel \boldsymbol{x}xxxxxxxx \ldots x$

i-th bit

Observation of $c \bmod p$:

- c is chosen input and p is victim's secret.
- if $c < p$, $c \bmod p = c$
- if $c \geq p$, $c \bmod p = c - lp$

Go RSA

Target i-th bit of p and set input as $c = prefix \parallel 10000 \ldots \parallel ptr$!

c $\quad$ $prefix \parallel 10000 \ldots \parallel ptr$

p $\quad$ $prefix \parallel xxxxxxxxx \ldots x$

i-th bit

- if $p = prefix \parallel 1xxxxxxxx \ldots x$ ➡ $p > c$
  - then $c \bmod p = c = \cdots \parallel ptr$

- if $p = prefix \parallel 0xxxxxxxx \ldots x.$ ➡ $p < c$
  - then $c \bmod p = c - lp = unknown$

Observation of $c \bmod p$:

- c is chosen input and p is victim's secret.
- if $c < p$, $c \bmod p = c$
- if $c \geq p$, $c \bmod p = c - lp$

Go RSA

Target i-th bit of p and set input as $c = prefix \parallel 10000 \ldots \parallel ptr$!

c $\quad prefix \parallel 10000 \ldots \parallel ptr$

p $\quad prefix \parallel xxxxxxxxx \ldots x$

i-th bit

DMP recovers upper **560** bits
+ **Coppersmith**[2]
= **full** extraction!

Observation of $c \bmod p$:

- c is chosen input and p is victim's secret.
- if $c < p$, $c \bmod p = c$
- if $c \geq p$, $c \bmod p = c - lp$

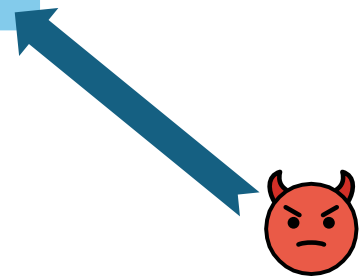
2 Coppersmith, "Finding a small root of a bivariate integer equation; factoring with high bits known", EUROCRYPT'96.

CRYSTALS-Kyber (ML-KEM)

- CCA-KEM built on a CPA-PKE scheme.

CRYSTALS-Kyber (ML-KEM)

- CCA-KEM built on a CPA-PKE scheme.



Key Mismatch Attack[3]

3 Qin et al., ”A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs”, ASIACRYPT’21.

CRYSTALS-Kyber (ML-KEM)

Key Mismatch Attack[3]

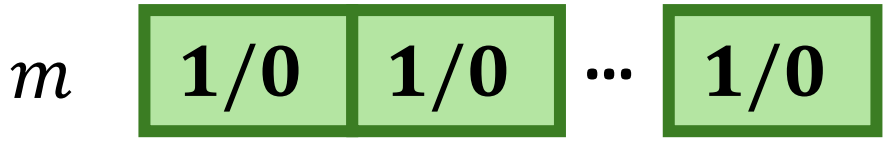
$$c = Enc(m, pk, e)$$

3 Qin et al., ”A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs”, ASIACRYPT’21.

CRYSTALS-Kyber (ML-KEM)

Key Mismatch Attack[3]

$$c = Enc(m, pk, e)$$



3 Qin et al., ”A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs”, ASIACRYPT’21.

CRYSTALS-Kyber (ML-KEM)

Key Mismatch Attack[3]

$$c = Enc(m, pk, e)$$

m | **1/0** | **1/0** | ⋯ | **1/0**

e | $\mathbf{0 \sim q}$ | $\mathbf{0 \sim q}$ | ⋯ | $\mathbf{0 \sim q}$

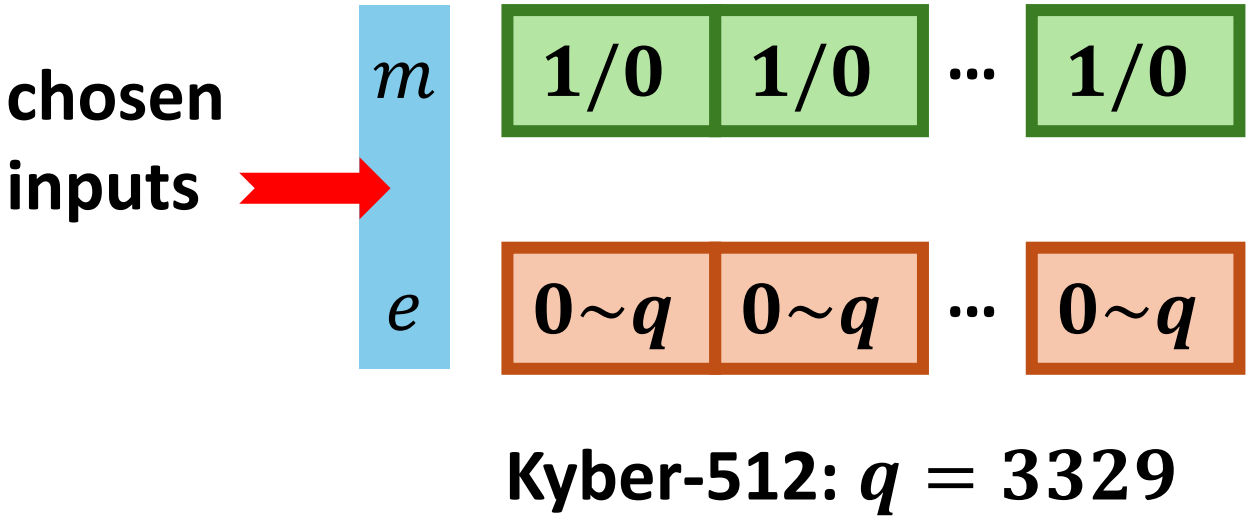
Kyber-512: $q = 3329$

3 Qin et al., ”A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs”, ASIACRYPT’21.

CRYSTALS-Kyber (ML-KEM)

Key Mismatch Attack[3]

$$c = Enc(m, pk, e)$$

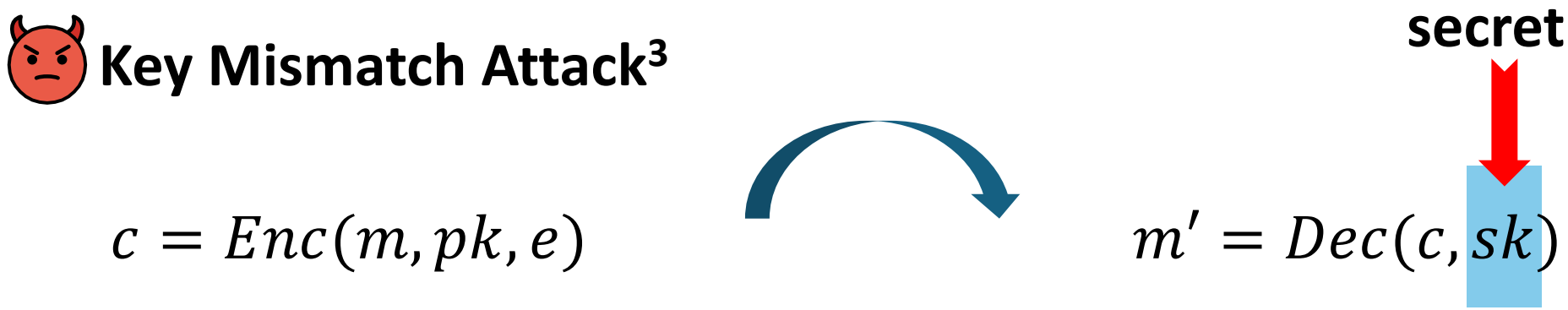


Kyber-512: $q = 3329$

3 Qin et al., "A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs", ASIACRYPT'21.

CRYSTALS-Kyber (ML-KEM)

Key Mismatch Attack[3]



Kyber-512: $q = 3329$

3 Qin et al., "A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs", ASIACRYPT'21.

CRYSTALS-Kyber (ML-KEM)

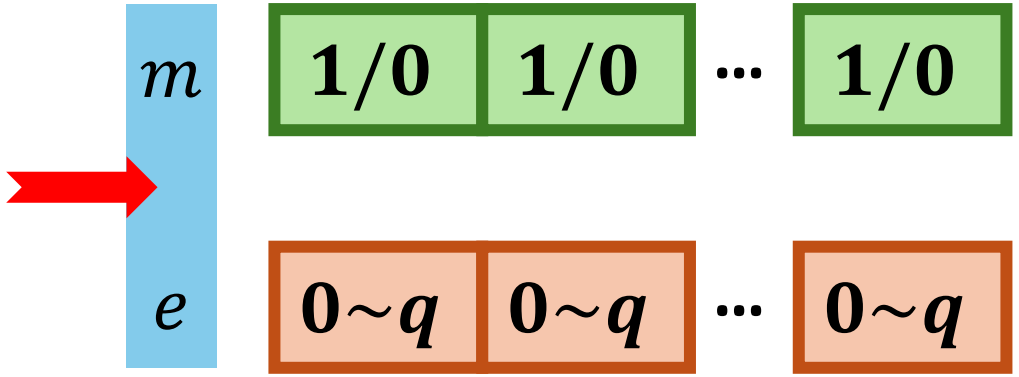
Key Mismatch Attack[3]

$c = Enc(m, pk, e)$

$m' = Dec(c, sk)$

secret

chosen inputs



Kyber-512: $q = 3329$

$m' = m$

$(sk^*, e^*) \rightarrow m' \neq m$

3 Qin et al., "A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs", ASIACRYPT'21.

CRYSTALS-Kyber (ML-KEM)

Key Mismatch Attack[3]

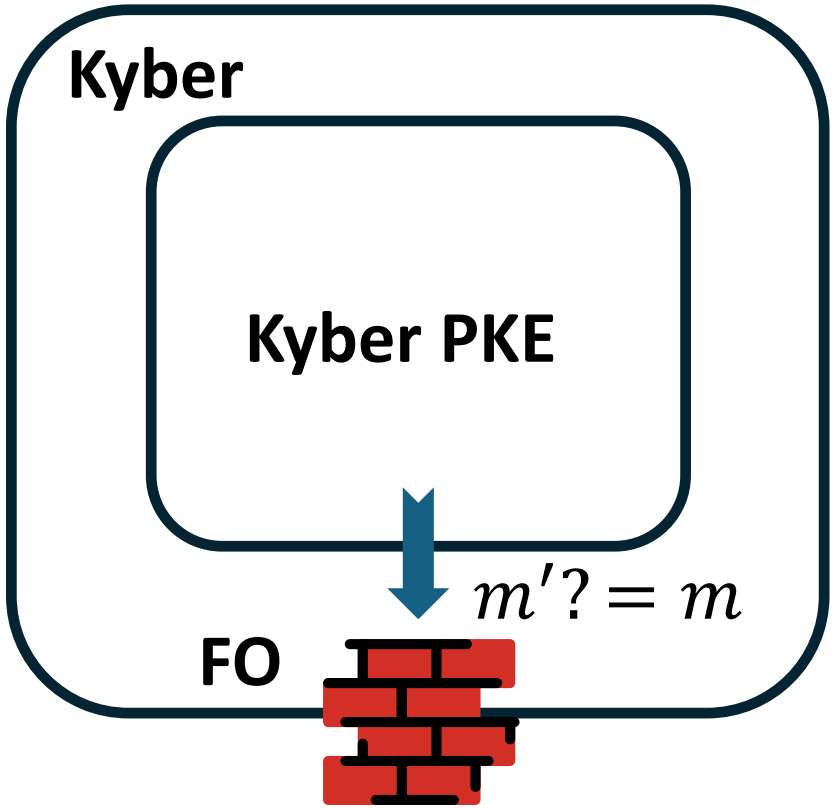
$$sk[i] < \frac{q}{4} - e[i] \quad \Rightarrow \quad m'[i] = m[i]$$

$$sk[i] \geq \frac{q}{4} - e[i] \quad \Rightarrow \quad m'[i] \neq m[i]$$

3 Qin et al., ”A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs”, ASIACRYPT’21.

CRYSTALS-Kyber (ML-KEM)

- CCA-KEM built on a CPA-PKE scheme.
- FO transformation wraps CPA to CCA.

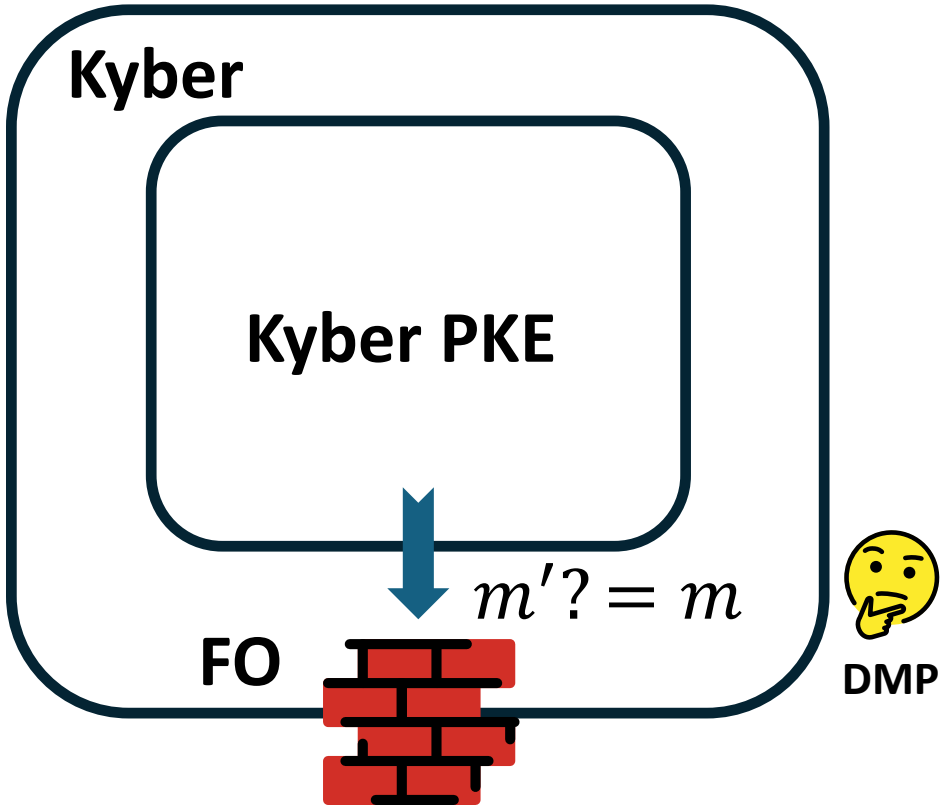


Observations of decrypted message m':

- sk is secret, m and e are chosen inputs.
- if $(e + sk)[i] < \frac{q}{4}$, $m'[i] = m[i]$.
- if $(e + sk)[i] \geq \frac{q}{4}$, $m'[i] \neq m[i]$.

CRYSTALS-Kyber (ML-KEM)

- CCA-KEM built on a CPA-PKE scheme.
- FO transformation wraps CPA to CCA.



Observations of decrypted message m':

- sk is secret, m and e are chosen inputs.
- if $(e + sk)[i] < \frac{q}{4}$, $m'[i] = m[i]$.
- if $(e + sk)[i] \geq \frac{q}{4}$, $m'[i] \neq m[i]$.

CRYSTALS-Kyber (ML-KEM)

Set m as $ptr \parallel 0 \ldots 0$ and tweak $e[0]$!

- if $(e + sk)[\mathbf{0}] < \frac{q}{4}$
  - then $m' = \boldsymbol{ptr}||0 \ldots 0$ $\boldsymbol{ptr}$
- if $(e + sk)[\mathbf{0}] \geq \frac{q}{4}$
  - then $m' = \boldsymbol{ptr'}||0 \ldots 0$ **No** $\boldsymbol{ptr}$

Observations of decrypted message m':

- sk is secret, m and e are chosen inputs.
- if $(e + sk)[i] < \frac{q}{4}$, $m'[i] = m[i]$.
- if $(e + sk)[i] \geq \frac{q}{4}$, $m'[i] \neq m[i]$.

CRYSTALS-Kyber (ML-KEM)

Set m as $ptr \parallel 0 \dots 0$ and tweak $e[0]$!

- if $(e + sk)[\mathbf{0}] < \frac{q}{4}$
  - then $m' = \boldsymbol{ptr}||0 \dots 0$ $\quad \boldsymbol{ptr}$
- if $(e + sk)[\mathbf{0}] \geq \frac{q}{4}$
  - then $m' = \boldsymbol{ptr'}||0 \dots 0$ **No** $\boldsymbol{ptr}$

Kyber-512:

DMP recovers **392** indexes

+ **Lattice Reduction**[3]

= **full** extraction!

Observations of decrypted message m':

- sk is secret, m and e are chosen inputs.
- if $(e + sk)[i] < \frac{q}{4}$, $m'[i] = m[i]$.
- if $(e + sk)[i] \geq \frac{q}{4}$, $m'[i] \neq m[i]$.

3 May et al., "Too Many Hints – When LLL Breaks LWE", ASIACRYPT'23.

Impact

Apple: Disable DMP with DIT=1

- Only works on M3.

```
bool set_DIT_enabled(void) {
    bool was_DIT_enabled = get_DIT_enabled();
    __asm__ __volatile__("msr dit, #1");
    return was_DIT_enabled;
}
```

Enable DIT for constant-time cryptographic operations

Impact

Apple: Disable DMP with DIT=1

- Only works on M3.

Go: Propose an opt-in DIT mode in Go binary.

proposal: runtime: implement a DIT/DOIT mode #66450

Open rolandshoemaker opened this issue on Mar 21 · 20 comments

Impact

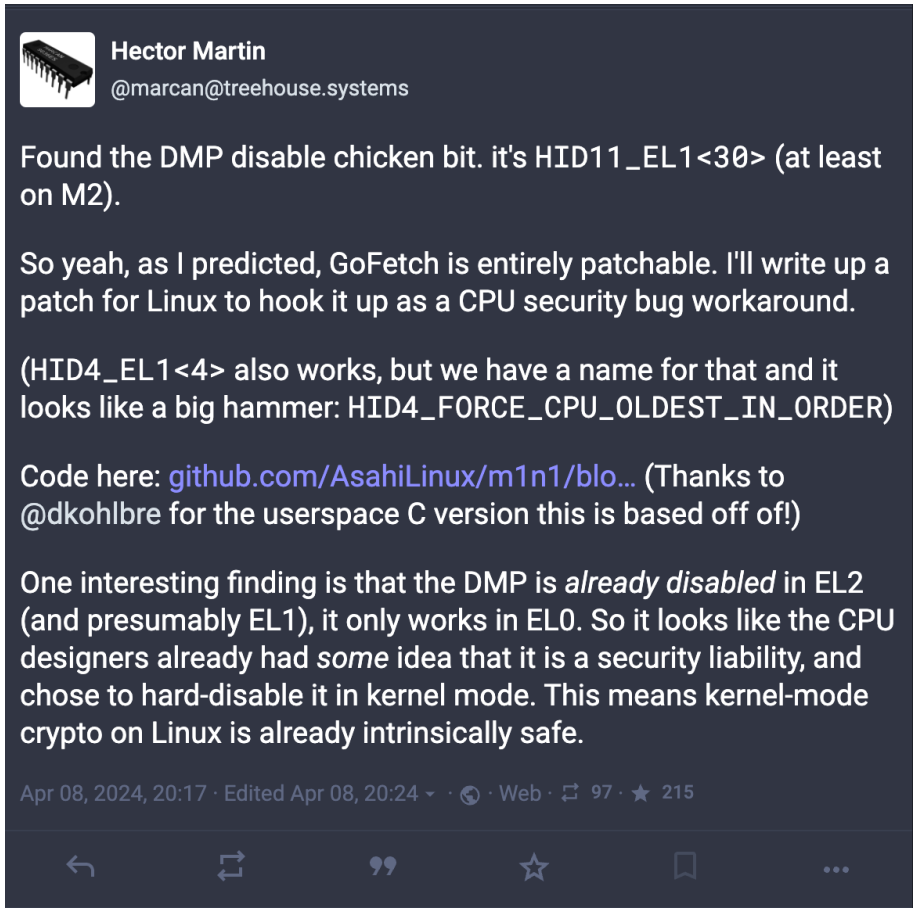
Apple: Disable DMP with DIT=1

- Only works on M3.

Go: Propose an opt-in DIT mode in Go binary.

Asahi Linux: Found chicken bit to disable DMP on M1/M2.

Asahi Linux



Impact

Apple: Disable DMP with DIT=1

- Only works on M3.

Go: Propose an opt-in DIT mode in Go binary.

Asahi Linux: Found chicken bit to disable DMP on M1/M2.

Asahi Linux

Pwine Awards: Best Cryptographic Attack winner.

Conclusion

- Data memory-dependent prefetchers (DMPs) performs secret-dependent memory access to leak data.
- Exploiting DMPs to perform key extraction attacks to constant-time cryptography is feasible.

GoFetch

Check our Website:
gofetch.fail

boruc2@illinois.edu