A Primer on Cache Attacks

Chitchanok Chuengsatiansup

The University of Adelaide, Australia

Summer School on Real-World Crypto and Privacy
Šibenik, Croatia

Outline

- Mastik
- Flush + Reload
- Prime + Probe

Mastik

- **M**icro-**A**rchitectural **S**ide-channel **T**ool**K**it
- https://github.com/0xADE1A1DE/Mastik
- Aims:
  - Collate information on side-channel attacks
  - Overcome the barrier to entry into the area
  - Shift focus to cryptanalysis

Mastik: status

- Reasonably solid implementation of four attacks:
  - Prime+Probe (L1-D, L1-I, L3)
  - Flush+Reload
  - Flush+Flush
  - CacheBleed
- Only Intel x86-64 on Linux
- Limited documentation and testing
- No user feedback

Computer Revolution

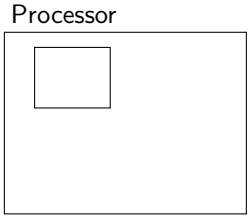
		Processor speed	Memory latency
1977		1 MHz	500 ns
2020		20×3700 MHz	50 ns

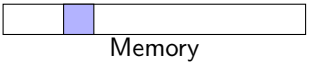
Bridging Speed Gap

- Utilize locality
  - spatial: divide memory into lines
  - temporal: store recently used lines in cache

Bridging Speed Gap

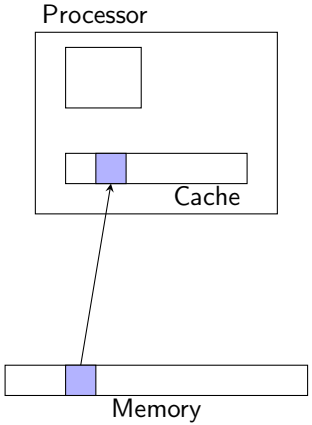
- Utilize locality
  - spatial: divide memory into lines
  - temporal: store recently used lines in cache





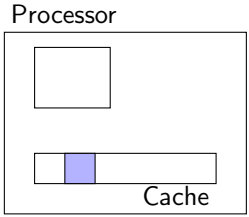
Bridging Speed Gap

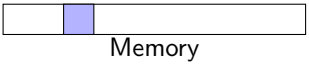
- Utilize locality
  - spatial: divide memory into lines
  - temporal: store recently used lines in cache



Bridging Speed Gap

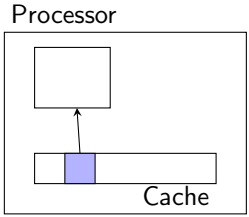
- Utilize locality
  - spatial: divide memory into lines
  - temporal: store recently used lines in cache
- Check if data is in cache
  - cache hit: serve data from cache
  - cache miss: get data from memory and put in cache

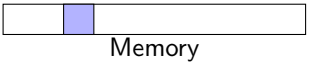




Bridging Speed Gap

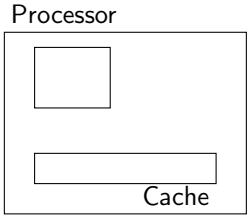
- Utilize locality
  - spatial: divide memory into lines
  - temporal: store recently used lines in cache
- Check if data is in cache
  - cache hit: serve data from cache
  - cache miss: get data from memory and put in cache

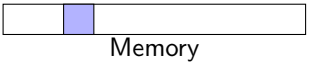




Bridging Speed Gap

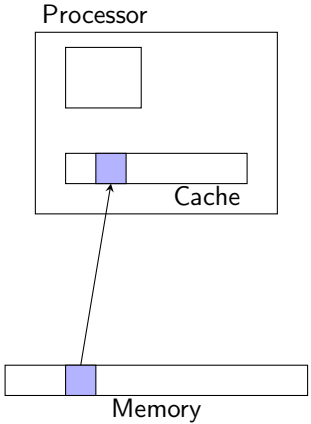
- Utilize locality
  - spatial: divide memory into lines
  - temporal: store recently used lines in cache
- Check if data is in cache
  - cache hit: serve data from cache
  - cache miss: get data from memory and put in cache





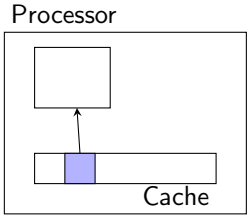
Bridging Speed Gap

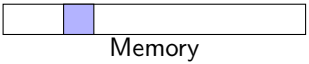
- Utilize locality
  - spatial: divide memory into lines
  - temporal: store recently used lines in cache
- Check if data is in cache
  - cache hit: serve data from cache
  - cache miss: get data from memory and put in cache



Bridging Speed Gap

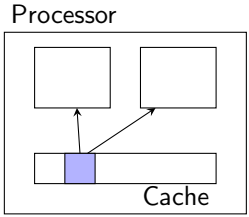
- Utilize locality
  - spatial: divide memory into lines
  - temporal: store recently used lines in cache
- Check if data is in cache
  - cache hit: serve data from cache
  - cache miss: get data from memory and put in cache

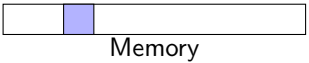




Bridging Speed Gap

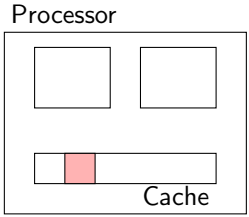
- Utilize locality
  - spatial: divide memory into lines
  - temporal: store recently used lines in cache
- Check if data is in cache
  - cache hit: serve data from cache
  - cache miss: get data from memory and put in cache
- Share cache
  - improve performance of multi-core processors

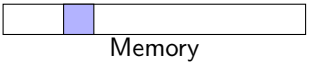




Maintaining Consistency

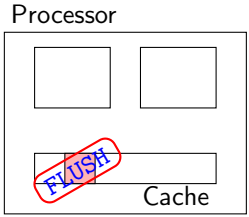
- Memory and cache can be inconsistent
  - rare but possible

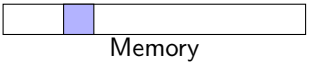




Maintaining Consistency

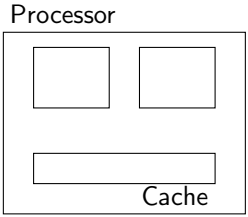
- Memory and cache can be inconsistent
  - rare but possible
- Solution: flush cache contents
  - ensure next load to serve from memory

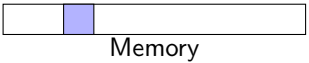




Maintaining Consistency

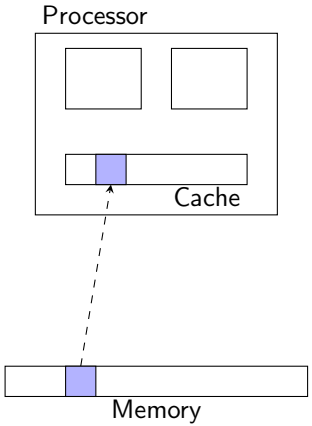
- Memory and cache can be inconsistent
  - rare but possible
- Solution: flush cache contents
  - ensure next load to serve from memory





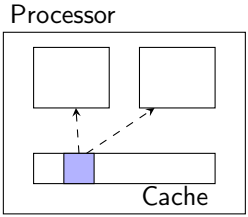
Maintaining Consistency

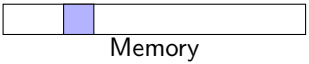
- Memory and cache can be inconsistent
  - rare but possible
- Solution: flush cache contents
  - ensure next load to serve from memory



Maintaining Consistency

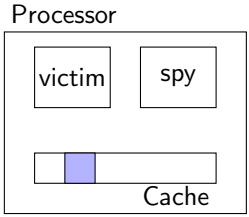
- Memory and cache can be inconsistent
  - rare but possible
- Solution: flush cache contents
  - ensure next load to serve from memory

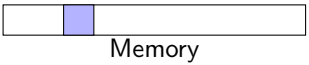




Flush + Reload [GBK11, YF14]

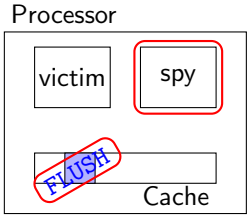
- **Flush** memory line
- Wait (victim executes)
- Measure time to **Reload** line
  - slow → no victim access
  - fast → victim accessed
- (Repeat)

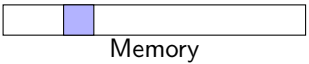




Flush + Reload [GBK11, YF14]

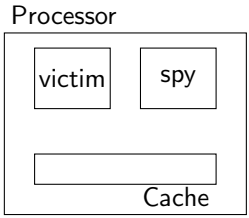
- **Flush** memory line
- Wait (victim executes)
- Measure time to **Reload** line
  - slow → no victim access
  - fast → victim accessed
- (Repeat)

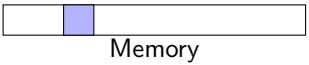




Flush + Reload [GBK11, YF14]

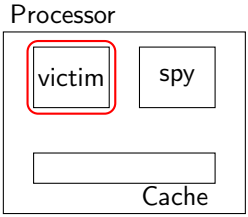
- **Flush** memory line
- Wait (victim executes)
- Measure time to **Reload** line
  - slow → no victim access
  - fast → victim accessed
- (Repeat)

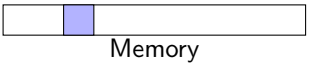




Flush + Reload [GBK11, YF14]

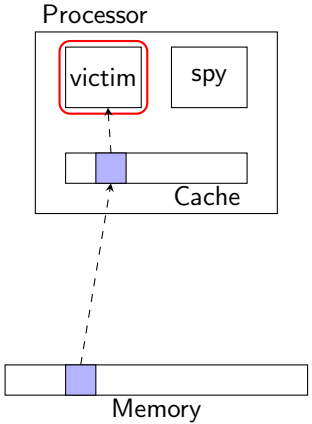
- **Flush** memory line
- Wait (victim executes)
- Measure time to **Reload** line
  - slow → no victim access
  - fast → victim accessed
- (Repeat)





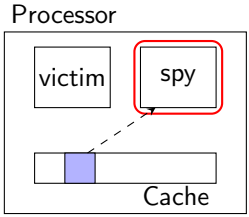
Flush + Reload [GBK11, YF14]

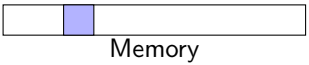
- **Flush** memory line
- Wait (victim executes)
- Measure time to **Reload** line
  - slow → no victim access
  - fast → victim accessed
- (Repeat)



Flush + Reload [GBK11, YF14]

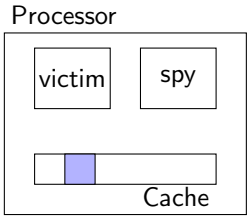
- **Flush** memory line
- Wait (victim executes)
- Measure time to **Reload** line
  - slow → no victim access
  - fast → victim accessed
- (Repeat)

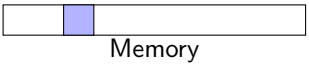




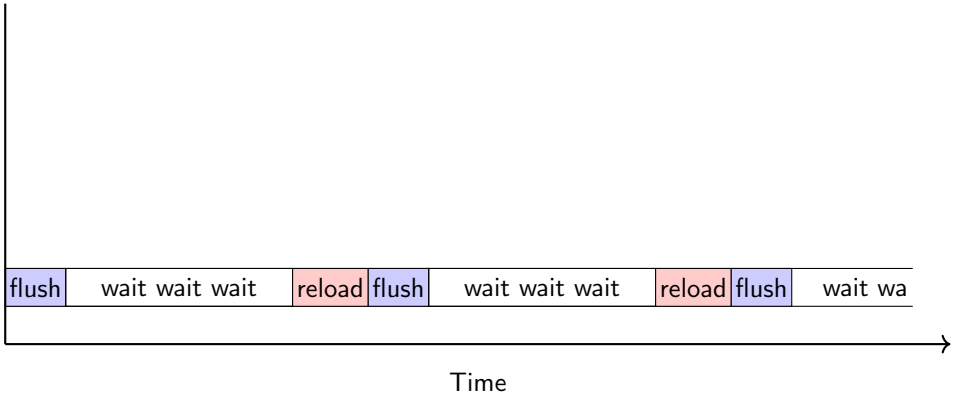
Flush + Reload [GBK11, YF14]

- **Flush** memory line
- Wait (victim executes)
- Measure time to **Reload** line
  - slow → no victim access
  - fast → victim accessed
- (Repeat)

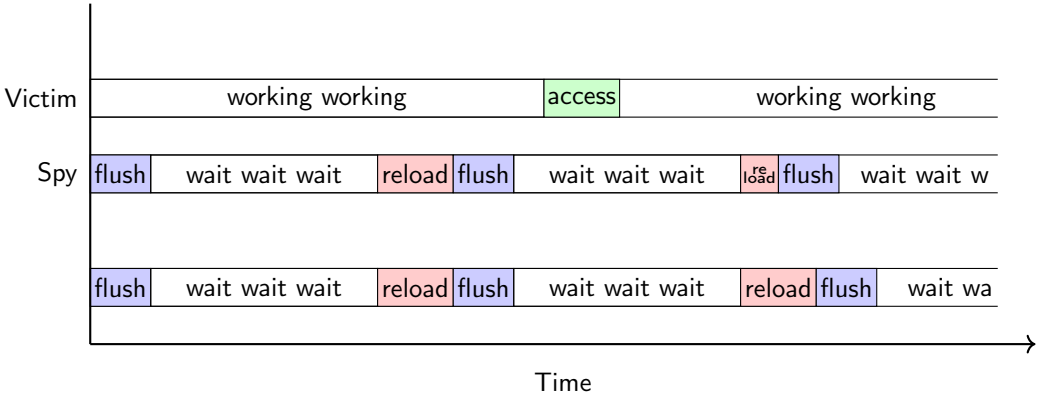




Flush + Reload: in action



Flush + Reload: in action



Flush + Reload: code

```
mfence                  ; ensure in-order execution
rdtscp                  ; get start time
mov %eax, %esi          ; store start time
mov (%ebx), %eax        ; access memory
rdtscp                  ; get finish time
sub %esi, %eax          ; subtract start and finish time
clflush 0 (%ebx)        ; flush cache line
```

Flush + Reload: demo1

- FR-access

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

$x \leftarrow 1$
for $i \leftarrow |d|-1$ **downto** 0 **do**
    $x \leftarrow x^2 \bmod n$
    if $(d_i = 1)$ **then**
        $x \leftarrow x \cdot b \bmod n$
return x

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ & \quad = 161051 \bmod 100 \\ & \quad = 51 \end{aligned}$$

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Operation	x	i	d_i
	1	2	101

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ &= 161051 \bmod 100 \\ &= 51 \end{aligned}$$

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Operation	x	i	d_i
	1	2	101
Square	1	2	101

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ &= 161051 \bmod 100 \\ &= 51 \end{aligned}$$

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Operation	x	i	d_i
	1	2	101
Square	1	2	101
Reduce	1	2	101

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ &= 161051 \bmod 100 \\ &= 51 \end{aligned}$$

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Operation	x	i	d_i
	1	2	101
Square	1	2	101
Reduce	1	2	101
Multiply	11	2	101

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ &= 161051 \bmod 100 \\ &= 51 \end{aligned}$$

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Operation	x	i	d_i
	1	2	101
Square	1	2	101
Reduce	1	2	101
Multiply	11	2	101
Reduce	11	2	101

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ &= 161051 \bmod 100 \\ &= 51 \end{aligned}$$

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Operation	x	i	d_i
	1	2	101
Square	1	2	101
Reduce	1	2	101
Multiply	11	2	101
Reduce	11	2	101
Square	121	1	101

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ &= 161051 \bmod 100 \\ &= 51 \end{aligned}$$

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ & \quad = 161051 \bmod 100 \\ & \quad = 51 \end{aligned}$$

Operation	x	i	d_i
	1	2	101
Square	1	2	101
Reduce	1	2	101
Multiply	11	2	101
Reduce	11	2	101
Square	121	1	101
Reduce	21	1	101

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ & \quad = 161051 \bmod 100 \\ & \quad = 51 \end{aligned}$$

Operation	x	i	d_i
	1	2	101
Square	1	2	101
Reduce	1	2	101
Multiply	11	2	101
Reduce	11	2	101
Square	121	1	101
Reduce	21	1	101
Square	441	0	101

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ &= 161051 \bmod 100 \\ &= 51 \end{aligned}$$

Operation	x	i	d_i
	1	2	101
Square	1	2	101
Reduce	1	2	101
Multiply	11	2	101
Reduce	11	2	101
Square	121	1	101
Reduce	21	1	101
Square	441	0	101
Reduce	41	0	101

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ & \quad = 161051 \bmod 100 \\ & \quad = 51 \end{aligned}$$

Operation	x	i	d_i
	1	2	101
Square	1	2	101
Reduce	1	2	101
Multiply	11	2	101
Reduce	11	2	101
Square	121	1	101
Reduce	21	1	101
Square	441	0	101
Reduce	41	0	101
Multiply	451	0	101

GnuPG 1.4.13 Modular Exponentiation

Compute $x = b^d \bmod n$

```
x ← 1
for i ← |d|−1 downto 0 do
    x ← x^2 mod n
    if (d_i = 1) then
        x ← x·b mod n
return x
```

Example: $b{=}11,\ d{=}5{=}101_2,\ n{=}100$

$$\begin{aligned} & 11^5 \bmod 100 \\ & \quad = 161051 \bmod 100 \\ & \quad = 51 \end{aligned}$$

Operation	x	i	d_i
	1	2	101
Square	1	2	101
Reduce	1	2	101
Multiply	11	2	101
Reduce	11	2	101
Square	121	1	101
Reduce	21	1	101
Square	441	0	101
Reduce	41	0	101
Multiply	451	0	101
Reduce	51	0	101

Flush + Reload: demo

- FR-gnupg-1.4.13

Flush + Reload: pros & cons

+ Simple
+ Very fews false positive
+ Resolution of memory line

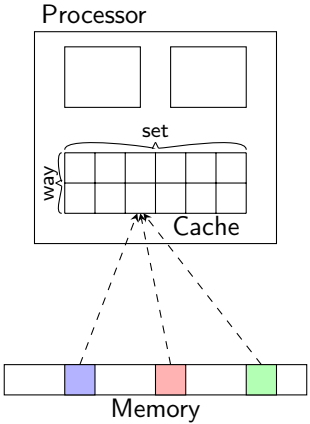
Flush + Reload: pros & cons

+ Simple
+ Very fews false positive
+ Resolution of memory line

- Only work with shared memory
- Require flush instruction

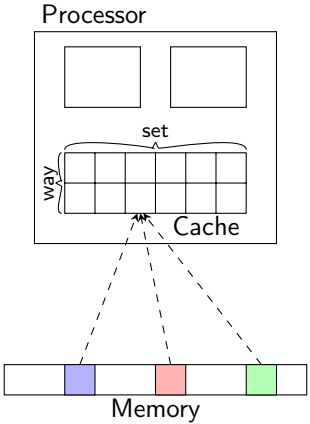
Set Associative Caches

- Memory lines map to cache sets



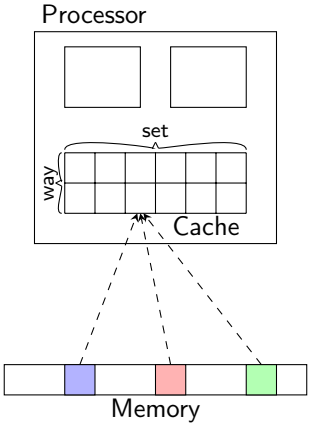
Set Associative Caches

- Memory lines map to cache sets
- Each line maps to one particular set and can be stored in any of its ways



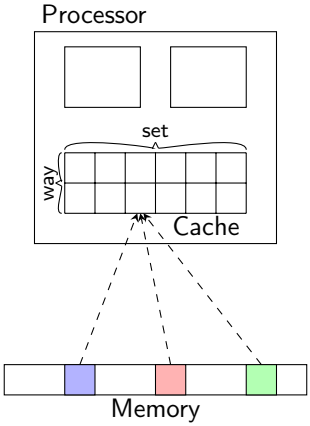
Set Associative Caches

- Memory lines map to cache sets
- Each line maps to one particular set and can be stored in any of its ways
- Multiple lines map to the same set



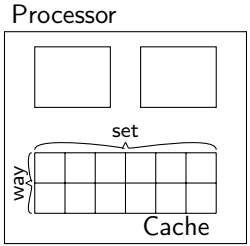
Set Associative Caches

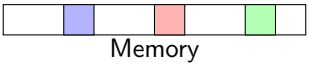
- Memory lines map to cache sets
- Each line maps to one particular set and can be stored in any of its ways
- Multiple lines map to the same set
- When cache miss and that set is full, one of the lines in that set is evicted



Set Associative Caches

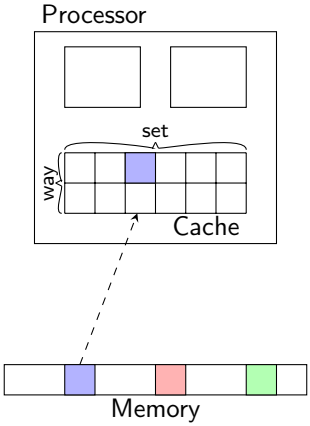
- Memory lines map to cache sets
- Each line maps to one particular set and can be stored in any of its ways
- Multiple lines map to the same set
- When cache miss and that set is full, one of the lines in that set is evicted





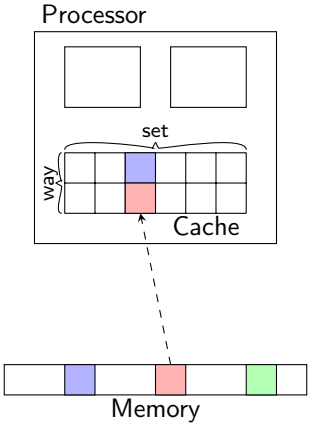
Set Associative Caches

- Memory lines map to cache sets
- Each line maps to one particular set and can be stored in any of its ways
- Multiple lines map to the same set
- When cache miss and that set is full, one of the lines in that set is evicted



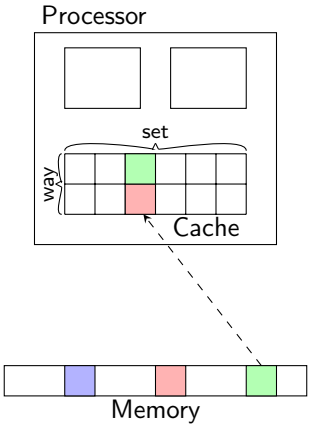
Set Associative Caches

- Memory lines map to cache sets
- Each line maps to one particular set and can be stored in any of its ways
- Multiple lines map to the same set
- When cache miss and that set is full, one of the lines in that set is evicted



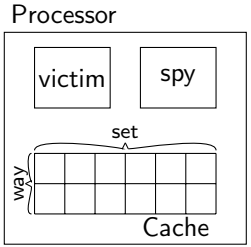
Set Associative Caches

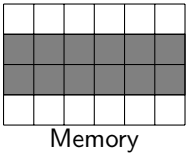
- Memory lines map to cache sets
- Each line maps to one particular set and can be stored in any of its ways
- Multiple lines map to the same set
- When cache miss and that set is full, one of the lines in that set is evicted



Prime + Probe [OST05, Per05]

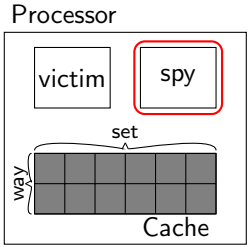
- (Allocate cache-sized memory buffer)
- **Prime**: access all lines in buffer
  → fill cache with attacker's data
- Wait (victim executes)
  → may evict some cache lines
- **Probe**: measure time to access buffer
  - slow → cache line evicted
  - fast → no victim access

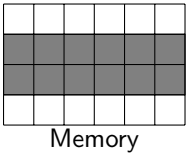




Prime + Probe [OST05, Per05]

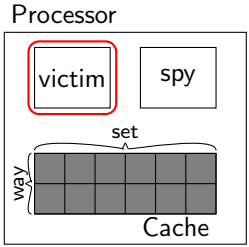
- (Allocate cache-sized memory buffer)
- **Prime**: access all lines in buffer
  → fill cache with attacker's data
- Wait (victim executes)
  → may evict some cache lines
- **Probe**: measure time to access buffer
  - slow → cache line evicted
  - fast → no victim access

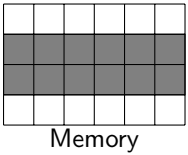




Prime + Probe [OST05, Per05]

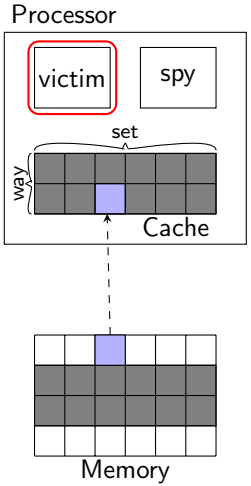
- (Allocate cache-sized memory buffer)
- **Prime**: access all lines in buffer
  → fill cache with attacker's data
- Wait (victim executes)
  → may evict some cache lines
- **Probe**: measure time to access buffer
  - slow → cache line evicted
  - fast → no victim access





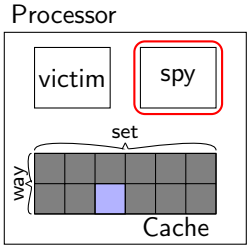
Prime + Probe [OST05, Per05]

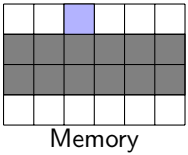
- (Allocate cache-sized memory buffer)
- **Prime**: access all lines in buffer
  → fill cache with attacker's data
- Wait (victim executes)
  → may evict some cache lines
- **Probe**: measure time to access buffer
  - slow → cache line evicted
  - fast → no victim access



Prime + Probe [OST05, Per05]

- (Allocate cache-sized memory buffer)
- **Prime**: access all lines in buffer
  → fill cache with attacker's data
- Wait (victim executes)
  → may evict some cache lines
- **Probe**: measure time to access buffer
  - slow → cache line evicted
  - fast → no victim access





Prime + Probe: sample victim data rattle

```
volatile char buffer[4096];

int main(int ac, char **av) {
  for (;;) {
    for (int i=0; i<64000; i++)
      buffer[800] += i;

    for (int i=0; i<64000; i++)
      buffer[1800] += i;
  }
}
```

Prime + Probe: demo

- Cache finger print of the rattle program

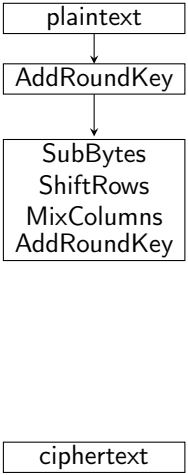
Prime + Probe: attack first-round AES

plaintext

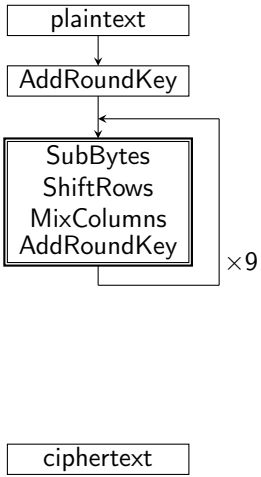
ciphertext

Prime + Probe: attack first-round AES

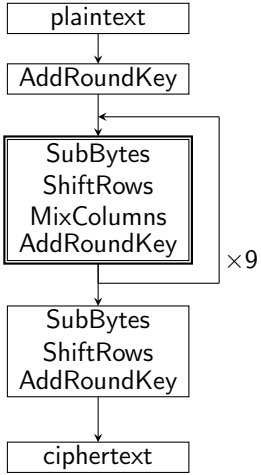
Prime + Probe: attack first-round AES



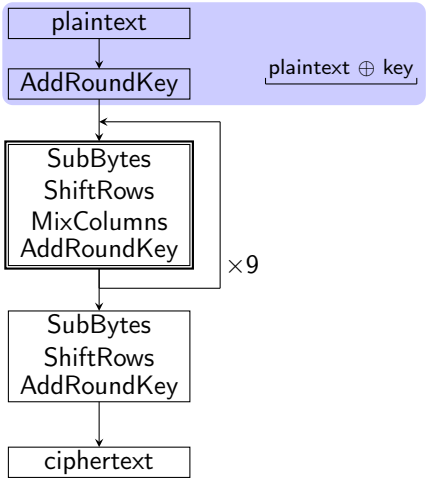
Prime + Probe: attack first-round AES



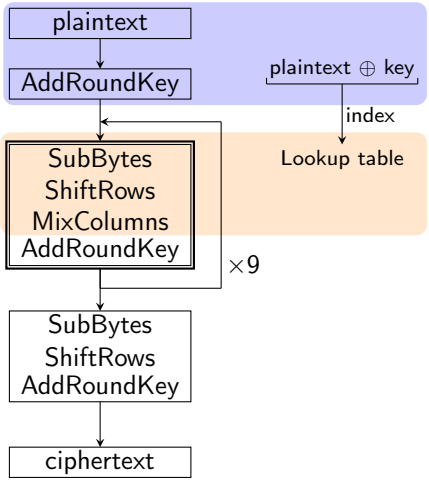
Prime + Probe: attack first-round AES



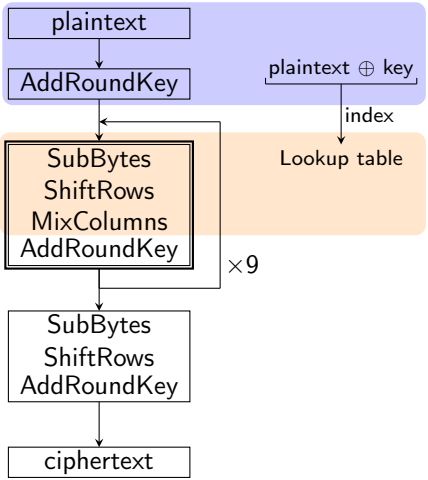
Prime + Probe: attack first-round AES



Prime + Probe: attack first-round AES

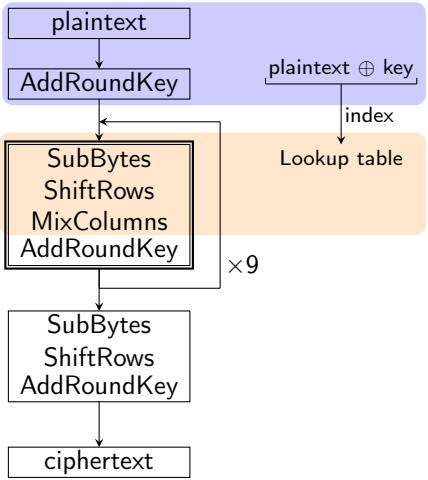


Prime + Probe: attack first-round AES



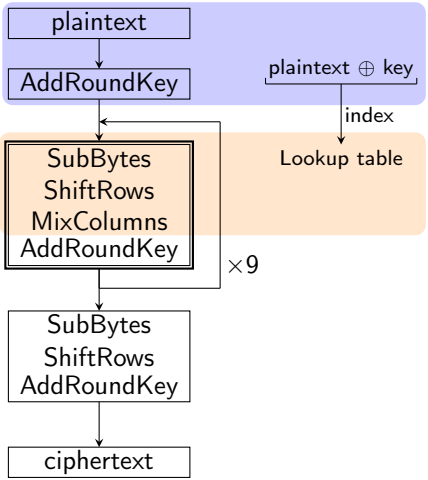
- (Assume a cache with 64 sets)

Prime + Probe: attack first-round AES



- (Assume a cache with 64 sets)
- A table has 256 entries, each of size 4 bytes

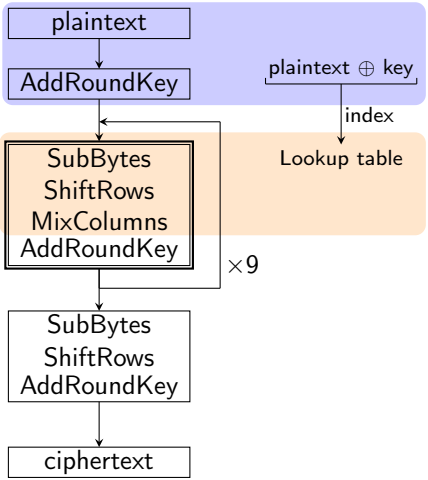
Prime + Probe: attack first-round AES



- (Assume a cache with 64 sets)
- A table has 256 entries, each of size 4 bytes

→ occupy $\frac{256 \times 4}{64} = 16$ cache lines

Prime + Probe: attack first-round AES

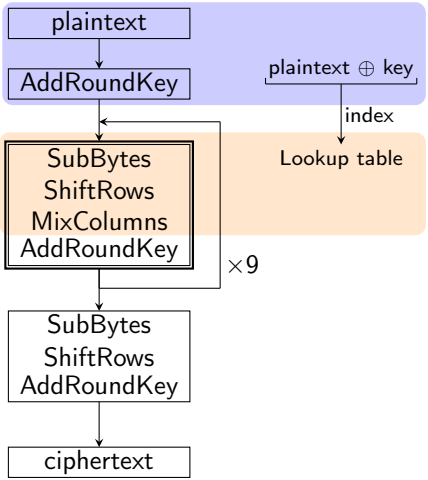


- (Assume a cache with 64 sets)
- A table has 256 entries, each of size 4 bytes

→ occupy $\frac{256 \times 4}{64} = 16$ cache lines

- Steps: Prime, AES, Probe

Prime + Probe: attack first-round AES



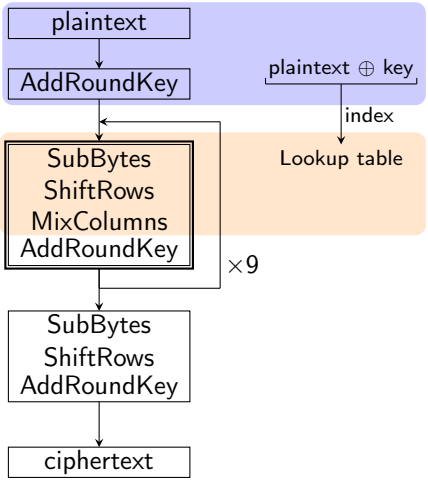
- (Assume a cache with 64 sets)
- A table has 256 entries, each of size 4 bytes

→ occupy $\frac{256 \times 4}{64} = 16$ cache lines

- Steps: Prime, AES, Probe
- Probability of no access to a cache set:

$$(\frac{15}{16})^{40} \approx 7.6\%$$

Prime + Probe: attack first-round AES



- (Assume a cache with 64 sets)
- A table has 256 entries, each of size 4 bytes

→ occupy $\frac{256 \times 4}{64} = 16$ cache lines

- Steps: Prime, AES, Probe
- Probability of no access to a cache set:

$$(\frac{15}{16})^{40} \approx 7.6\%$$

→ eliminate impossible keys to one cache set

Prime + Probe: attack first-round AES (cont.)

- Actual experiments have noise

Prime + Probe: attack first-round AES (cont.)

- Actual experiments have noise

→ sometimes cannot distinguish cache hit/miss
(always show that all sets are accessed)

Prime + Probe: attack first-round AES (cont.)

- Actual experiments have noise

→ sometimes cannot distinguish cache hit/miss
(always show that all sets are accessed)

- Average access time per cache set

Prime + Probe: attack first-round AES (cont.)

- Actual experiments have noise

→ sometimes cannot distinguish cache hit/miss
(always show that all sets are accessed)

- Average access time per cache set
  - p_μ : average of all n samples

Prime + Probe: attack first-round AES (cont.)

- Actual experiments have noise

→ sometimes cannot distinguish cache hit/miss
(always show that all sets are accessed)

- Average access time per cache set
  - p_μ : average of all n samples
  - p_{μ_0} : average of samples whose 4 MSBs of the plaintext byte are 0000

Prime + Probe: attack first-round AES (cont.)

- Actual experiments have noise

→ sometimes cannot distinguish cache hit/miss
(always show that all sets are accessed)

- Average access time per cache set
  - p_μ : average of all n samples
  - p_{μ_0} : average of samples whose 4 MSBs of the plaintext byte are 0000
  - p_{μ_1} : average of samples whose 4 MSBs of the plaintext byte are 0001

Prime + Probe: attack first-round AES (cont.)

- Actual experiments have noise

→ sometimes cannot distinguish cache hit/miss
(always show that all sets are accessed)

- Average access time per cache set
  - p_μ : average of all n samples
  - p_{μ_0} : average of samples whose 4 MSBs of the plaintext byte are 0000
  - p_{μ_1} : average of samples whose 4 MSBs of the plaintext byte are 0001
  - ...
  - $p_{\mu_{15}}$: average of samples whose 4 MSBs of the plaintext byte are 1111

Prime + Probe: attack first-round AES (cont.)

$$p_{t_0} = [\quad t_0^0 \quad , \quad t_0^1 \quad , \ldots , \quad t_0^{63} \quad]$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\quad t_0^0 \quad , \quad t_0^1 \quad , \ldots , \quad t_0^{63} \quad] \\
p_{t_1} &= [\quad t_1^0 \quad , \quad t_1^1 \quad , \ldots , \quad t_1^{63} \quad]
\end{aligned}
$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0 \ , \ t_0^1 \ , \ldots , \ t_0^{63} \ ] \\
p_{t_1} &= [\ t_1^0 \ , \ t_1^1 \ , \ldots , \ t_1^{63} \ ] \\
&\ldots \\
p_{t_n} &= [\ t_n^0 \ , \ t_n^1 \ , \ldots , \ t_n^{63} \ ]
\end{aligned}
$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{array}{lcccccc}
p_{t_0} & = [& t_0^0 & , & t_0^1 & , \ldots , & t_0^{63} \quad] \\
p_{t_1} & = [& t_1^0 & , & t_1^1 & , \ldots , & t_1^{63} \quad] \\
\ldots & & & & & & \\
p_{t_n} & = [& t_n^0 & , & t_n^1 & , \ldots , & t_n^{63} \quad]
\end{array}
$$

$$p_{\mu_0} = [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ , \ldots ,\ \mathsf{avg}_0(t^{63})\ ]$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{array}{lcccccc}
p_{t_0} & = [& t_0^0 & , & t_0^1 & , \ldots , & t_0^{63} \quad] \\
p_{t_1} & = [& t_1^0 & , & t_1^1 & , \ldots , & t_1^{63} \quad] \\
\ldots & & & & & & \\
p_{t_n} & = [& t_n^0 & , & t_n^1 & , \ldots , & t_n^{63} \quad]
\end{array}
$$

$$
\begin{array}{ll}
p_{\mu_0} & = [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ , \ldots ,\ \mathsf{avg}_0(t^{63})\ ] \\
p_{\mu_1} & = [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ , \ldots ,\ \mathsf{avg}_1(t^{63})\ ]
\end{array}
$$

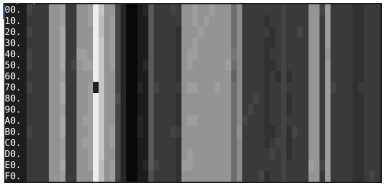
Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0\ ,\ t_0^1\ , \ldots ,\ t_0^{63}\ ] \\
p_{t_1} &= [\ t_1^0\ ,\ t_1^1\ , \ldots ,\ t_1^{63}\ ] \\
&\cdots \\
p_{t_n} &= [\ t_n^0\ ,\ t_n^1\ , \ldots ,\ t_n^{63}\ ]
\end{aligned}
$$

$$
\begin{aligned}
p_{\mu_0} &= [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ , \ldots ,\ \mathsf{avg}_0(t^{63})\ ] \\
p_{\mu_1} &= [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ , \ldots ,\ \mathsf{avg}_1(t^{63})\ ] \\
&\cdots \\
p_{\mu_{15}} &= [\ \mathsf{avg}_{15}(t^0),\ \mathsf{avg}_{15}(t^1), \ldots ,\ \mathsf{avg}_{15}(t^{63})\ ]
\end{aligned}
$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0\ ,\ t_0^1\ ,\ldots,\ t_0^{63}\ ] \\
p_{t_1} &= [\ t_1^0\ ,\ t_1^1\ ,\ldots,\ t_1^{63}\ ] \\
&\cdots \\
p_{t_n} &= [\ t_n^0\ ,\ t_n^1\ ,\ldots,\ t_n^{63}\ ]
\end{aligned}
$$



$$
\begin{aligned}
p_{\mu_0} &= [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ ,\ldots,\ \mathsf{avg}_0(t^{63})\ ] \\
p_{\mu_1} &= [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ ,\ldots,\ \mathsf{avg}_1(t^{63})\ ] \\
&\cdots \\
p_{\mu_{15}} &= [\ \mathsf{avg}_{15}(t^0),\ \mathsf{avg}_{15}(t^1),\ldots,\ \mathsf{avg}_{15}(t^{63})\ ]
\end{aligned}
$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0\ ,\ t_0^1\ , \ldots ,\ t_0^{63}\ ] \\
p_{t_1} &= [\ t_1^0\ ,\ t_1^1\ , \ldots ,\ t_1^{63}\ ] \\
&\cdots \\
p_{t_n} &= [\ t_n^0\ ,\ t_n^1\ , \ldots ,\ t_n^{63}\ ]
\end{aligned}
$$

$$
\begin{aligned}
p_{\mu} &= [\ \mathsf{avg}_n(t^0)\ ,\ \mathsf{avg}_n(t^1)\ , \ldots ,\ \mathsf{avg}_n(t^{63})\ ] \\
p_{\mu_0} &= [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ , \ldots ,\ \mathsf{avg}_0(t^{63})\ ] \\
p_{\mu_1} &= [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ , \ldots ,\ \mathsf{avg}_1(t^{63})\ ] \\
&\cdots \\
p_{\mu_{15}} &= [\ \mathsf{avg}_{15}(t^0),\ \mathsf{avg}_{15}(t^1), \ldots ,\ \mathsf{avg}_{15}(t^{63})\ ]
\end{aligned}
$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0\ ,\ t_0^1\ , \ldots ,\ t_0^{63}\ ] \\
p_{t_1} &= [\ t_1^0\ ,\ t_1^1\ , \ldots ,\ t_1^{63}\ ] \\
&\cdots \\
p_{t_n} &= [\ t_n^0\ ,\ t_n^1\ , \ldots ,\ t_n^{63}\ ]
\end{aligned}
$$

$$
\begin{aligned}
p_{\mu} &= [\ \mathsf{avg}_n(t^0)\ ,\ \mathsf{avg}_n(t^1)\ , \ldots ,\ \mathsf{avg}_n(t^{63})\ ] \\
p_{\mu_0} &= [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ , \ldots ,\ \mathsf{avg}_0(t^{63})\ ] \\
p_{\mu_1} &= [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ , \ldots ,\ \mathsf{avg}_1(t^{63})\ ] \\
&\cdots \\
p_{\mu_{15}} &= [\ \mathsf{avg}_{15}(t^0),\ \mathsf{avg}_{15}(t^1), \ldots ,\ \mathsf{avg}_{15}(t^{63})\ ]
\end{aligned}
$$

$$\tilde{p}_{\mu_0} = p_{\mu_0} - p_{\mu}$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0\ ,\ t_0^1\ , \ldots ,\ t_0^{63}\ ] \\
p_{t_1} &= [\ t_1^0\ ,\ t_1^1\ , \ldots ,\ t_1^{63}\ ] \\
&\cdots \\
p_{t_n} &= [\ t_n^0\ ,\ t_n^1\ , \ldots ,\ t_n^{63}\ ]
\end{aligned}
$$

$$
\begin{aligned}
p_{\mu} &= [\ \mathsf{avg}_n(t^0)\ ,\ \mathsf{avg}_n(t^1)\ , \ldots ,\ \mathsf{avg}_n(t^{63})\ ] \\
p_{\mu_0} &= [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ , \ldots ,\ \mathsf{avg}_0(t^{63})\ ] \\
p_{\mu_1} &= [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ , \ldots ,\ \mathsf{avg}_1(t^{63})\ ] \\
&\cdots \\
p_{\mu_{15}} &= [\ \mathsf{avg}_{15}(t^0),\ \mathsf{avg}_{15}(t^1), \ldots ,\ \mathsf{avg}_{15}(t^{63})\ ]
\end{aligned}
$$

$$
\begin{aligned}
\tilde{p}_{\mu_0} &= p_{\mu_0} - p_{\mu} \\
\tilde{p}_{\mu_1} &= p_{\mu_1} - p_{\mu}
\end{aligned}
$$

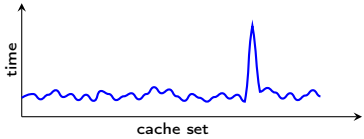
Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0\ ,\ t_0^1\ ,\ \ldots,\ t_0^{63}\ ] \\
p_{t_1} &= [\ t_1^0\ ,\ t_1^1\ ,\ \ldots,\ t_1^{63}\ ] \\
&\ldots \\
p_{t_n} &= [\ t_n^0\ ,\ t_n^1\ ,\ \ldots,\ t_n^{63}\ ]
\end{aligned}
$$

$$
\begin{aligned}
p_{\mu} &= [\ \mathsf{avg}_n(t^0)\ ,\ \mathsf{avg}_n(t^1)\ ,\ldots,\ \mathsf{avg}_n(t^{63})\ ] \\
p_{\mu_0} &= [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ ,\ldots,\ \mathsf{avg}_0(t^{63})\ ] \qquad \tilde{p}_{\mu_0} = p_{\mu_0} - p_{\mu} \\
p_{\mu_1} &= [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ ,\ldots,\ \mathsf{avg}_1(t^{63})\ ] \qquad \tilde{p}_{\mu_1} = p_{\mu_1} - p_{\mu} \\
&\ldots \\
p_{\mu_{15}} &= [\ \mathsf{avg}_{15}(t^0),\ \mathsf{avg}_{15}(t^1),\ldots,\ \mathsf{avg}_{15}(t^{63})\ ] \qquad \tilde{p}_{\mu_{15}} = p_{\mu_{15}} - p_{\mu}
\end{aligned}
$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0\ ,\ t_0^1\ ,\ \ldots,\ t_0^{63}\ ] \\
p_{t_1} &= [\ t_1^0\ ,\ t_1^1\ ,\ \ldots,\ t_1^{63}\ ] \\
&\ldots \\
p_{t_n} &= [\ t_n^0\ ,\ t_n^1\ ,\ \ldots,\ t_n^{63}\ ]
\end{aligned}
$$

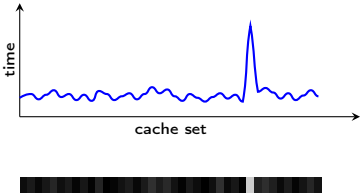


$$
\begin{aligned}
p_{\mu} &= [\ \mathsf{avg}_n(t^0)\ ,\ \mathsf{avg}_n(t^1)\ ,\ldots,\ \mathsf{avg}_n(t^{63})\ ] \\
p_{\mu_0} &= [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ ,\ldots,\ \mathsf{avg}_0(t^{63})\ ] \\
p_{\mu_1} &= [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ ,\ldots,\ \mathsf{avg}_1(t^{63})\ ] \\
&\ldots \\
p_{\mu_{15}} &= [\ \mathsf{avg}_{15}(t^0),\ \mathsf{avg}_{15}(t^1),\ldots,\ \mathsf{avg}_{15}(t^{63})\ ]
\end{aligned}
$$

$$
\begin{aligned}
\tilde{p}_{\mu_0} &= p_{\mu_0} - p_{\mu} \\
\tilde{p}_{\mu_1} &= p_{\mu_1} - p_{\mu} \\
\tilde{p}_{\mu_{15}} &= p_{\mu_{15}} - p_{\mu}
\end{aligned}
$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0\ ,\ t_0^1\ ,\ \ldots,\ t_0^{63}\ ] \\
p_{t_1} &= [\ t_1^0\ ,\ t_1^1\ ,\ \ldots,\ t_1^{63}\ ] \\
&\ldots \\
p_{t_n} &= [\ t_n^0\ ,\ t_n^1\ ,\ \ldots,\ t_n^{63}\ ]
\end{aligned}
$$

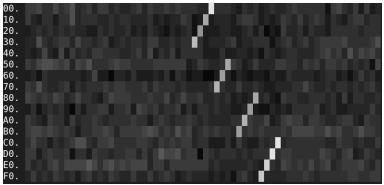


$$
\begin{aligned}
p_{\mu} &= [\ \mathsf{avg}_n(t^0)\ ,\ \mathsf{avg}_n(t^1)\ ,\ \ldots,\ \mathsf{avg}_n(t^{63})\ ] \\
p_{\mu_0} &= [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ ,\ \ldots,\ \mathsf{avg}_0(t^{63})\ ] \\
p_{\mu_1} &= [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ ,\ \ldots,\ \mathsf{avg}_1(t^{63})\ ] \\
&\ldots \\
p_{\mu_{15}} &= [\ \mathsf{avg}_{15}(t^0),\ \mathsf{avg}_{15}(t^1),\ \ldots,\ \mathsf{avg}_{15}(t^{63})\ ]
\end{aligned}
$$

$$
\begin{aligned}
\tilde{p}_{\mu_0} &= p_{\mu_0} - p_{\mu} \\
\tilde{p}_{\mu_1} &= p_{\mu_1} - p_{\mu} \\
\tilde{p}_{\mu_{15}} &= p_{\mu_{15}} - p_{\mu}
\end{aligned}
$$

Prime + Probe: attack first-round AES (cont.)

$$
\begin{aligned}
p_{t_0} &= [\ t_0^0\ ,\ t_0^1\ ,\ \ldots,\ t_0^{63}\ ] \\
p_{t_1} &= [\ t_1^0\ ,\ t_1^1\ ,\ \ldots,\ t_1^{63}\ ] \\
&\cdots \\
p_{t_n} &= [\ t_n^0\ ,\ t_n^1\ ,\ \ldots,\ t_n^{63}\ ]
\end{aligned}
$$



$$
\begin{aligned}
p_{\mu} &= [\ \mathsf{avg}_n(t^0)\ ,\ \mathsf{avg}_n(t^1)\ ,\ldots,\ \mathsf{avg}_n(t^{63})\ ] \\
p_{\mu_0} &= [\ \mathsf{avg}_0(t^0)\ ,\ \mathsf{avg}_0(t^1)\ ,\ldots,\ \mathsf{avg}_0(t^{63})\ ] \\
p_{\mu_1} &= [\ \mathsf{avg}_1(t^0)\ ,\ \mathsf{avg}_1(t^1)\ ,\ldots,\ \mathsf{avg}_1(t^{63})\ ] \\
&\cdots \\
p_{\mu_{15}} &= [\ \mathsf{avg}_{15}(t^0),\ \mathsf{avg}_{15}(t^1),\ldots,\ \mathsf{avg}_{15}(t^{63})\ ]
\end{aligned}
$$

$$
\begin{aligned}
\tilde{p}_{\mu_0} &= p_{\mu_0} - p_{\mu} \\
\tilde{p}_{\mu_1} &= p_{\mu_1} - p_{\mu} \\
\tilde{p}_{\mu_{15}} &= p_{\mu_{15}} - p_{\mu}
\end{aligned}
$$

Prime + Probe: demo

- First-round AES

Prime + Probe: attack first-round AES (cont.)

- Recall Prime + Probe has a resolution of cache set

$\rightarrow$ detect which cache set is (not) accessed

Prime + Probe: attack first-round AES (cont.)

- Recall Prime + Probe has a resolution of cache set

$\rightarrow$ detect which cache set is (not) accessed

- A table spans over $16 = 2^4$ sets

$\rightarrow$ reveal top 4 bits at best

Prime + Probe: attack first-round AES (cont.)

- Recall Prime + Probe has a resolution of cache set

$\rightarrow$ detect which cache set is (not) accessed

- A table spans over $16 = 2^4$ sets

$\rightarrow$ reveal top 4 bits at best

- To recover the remaining bits
  - combine with second-round cryptanalysis
  - attack final round instead
  - etc.

Prime + Probe: attack first-round AES (cont.)

- Recall Prime + Probe has a resolution of cache set

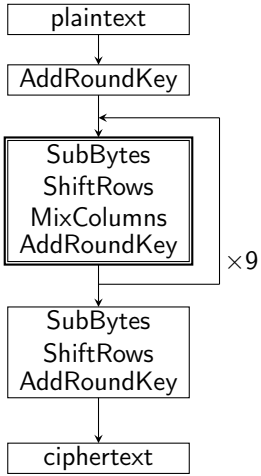
$\rightarrow$ detect which cache set is (not) accessed

- A table spans over $16 = 2^4$ sets

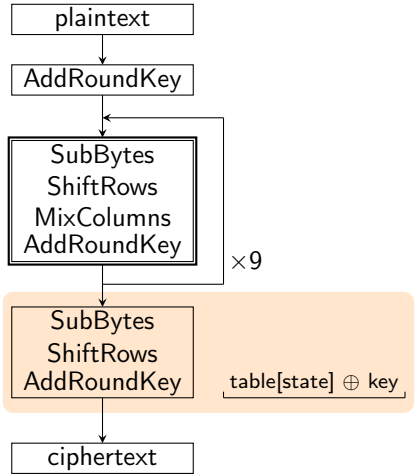
$\rightarrow$ reveal top 4 bits at best

- To recover the remaining bits
  - combine with second-round cryptanalysis
  - attack final round instead
  - etc.

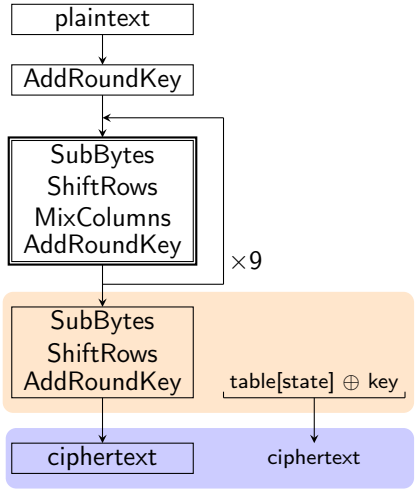
Prime + Probe: attack final-round AES



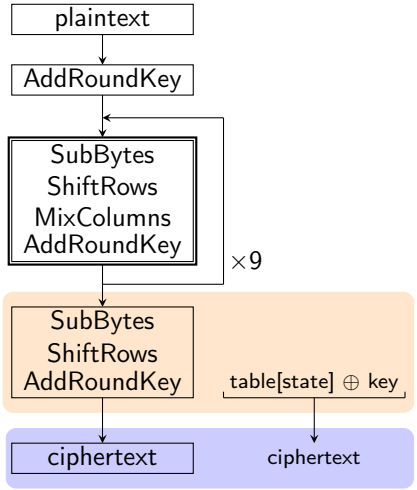
Prime + Probe: attack final-round AES



Prime + Probe: attack final-round AES

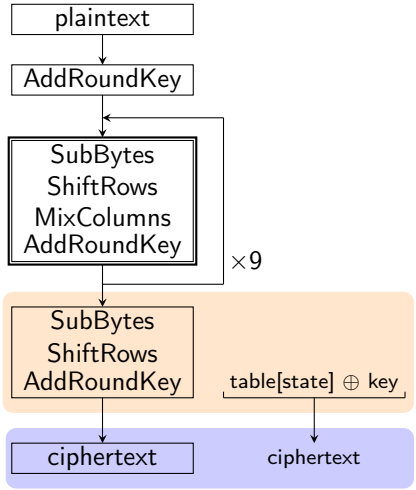


Prime + Probe: attack final-round AES



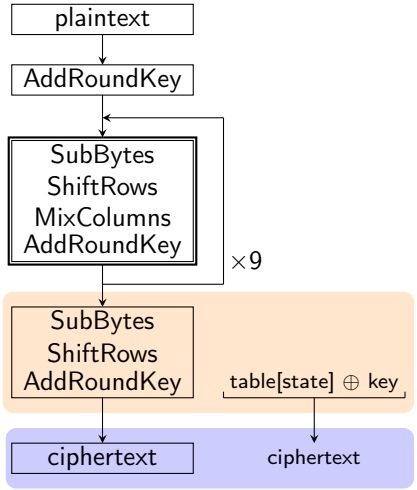
- Know ciphertext + guess key → table index

Prime + Probe: attack final-round AES



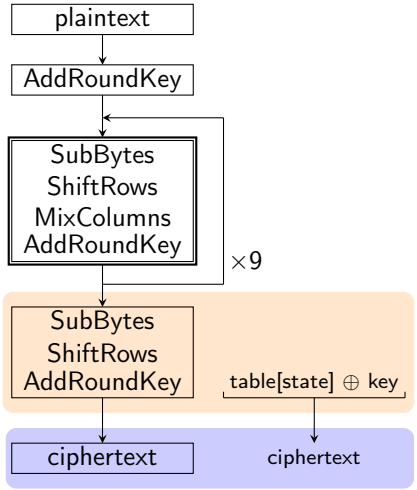
- Know ciphertext + guess key → table index
- Exploit SBox confusion property

Prime + Probe: attack final-round AES



- Know ciphertext + guess key → table index
- Exploit SBox confusion property
- Compute correlation between known and guessed key
  - high correlation indicates correct key guess

Prime + Probe: attack final-round AES



- Know ciphertext + guess key → table index
- Exploit SBox confusion property
- Compute correlation between known and guessed key
  - high correlation indicates correct key guess
- Recover full key

Prime + Probe: attack final-round AES

$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$ 

Prime + Probe: attack final-round AES

$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$

$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$

Prime + Probe: attack final-round AES

$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$

$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$ 

$\ldots$

$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

Prime + Probe: attack final-round AES

$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

$$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$$

Prime + Probe: attack final-round AES

$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

$$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$$

Prime + Probe: attack final-round AES

$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

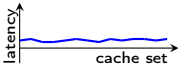
$$k_{t_{00}} = [\ 0,\ 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1,\ 15, \ldots,\ 0]$$
$$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$$

Prime + Probe: attack final-round AES

$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

$$k_{t_{00}} = [\ 0,\ 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1,\ 15, \ldots,\ 0]$$
$$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$$
$$\ldots$$
$$k_{t_{\mathrm{FF}}} = [\ 0, -1, \ldots,\ 15, -1, \ldots,\ 0]$$

Prime + Probe: attack final-round AES

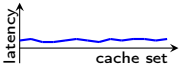


$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

$$\tilde{p}_{t_{0,00}} = p_{t_0} \odot k_{t_0}$$

$$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$$
$$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$$
$$\ldots$$
$$k_{t_{\mathrm{FF}}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$$

Prime + Probe: attack final-round AES



$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$

$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$

$\ldots$

$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$

$\tilde{p}_{t_{0,00}} = p_{t_0} \odot k_{t_0}$

$\tilde{p}_{t_{1,00}} = p_{t_1} \odot k_{t_0}$

$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$

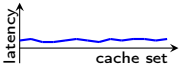
$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$

$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$

$\ldots$

$k_{t_{FF}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$

Prime + Probe: attack final-round AES

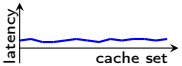


$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

$$\tilde{p}_{t_{0,00}} = p_{t_0} \odot k_{t_0}$$
$$\tilde{p}_{t_{1,00}} = p_{t_1} \odot k_{t_0}$$
$$\ldots$$
$$\tilde{p}_{t_{n,00}} = p_{t_n} \odot k_{t_0}$$

$$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$$
$$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$$
$$\ldots$$
$$k_{t_{\mathrm{FF}}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$$

Prime + Probe: attack final-round AES



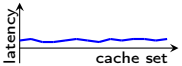
$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

$$\tilde{p}_{t_{0,00}} = p_{t_0} \odot k_{t_0}$$
$$\tilde{p}_{t_{1,00}} = p_{t_1} \odot k_{t_0}$$
$$\ldots$$
$$\tilde{p}_{t_{n,00}} = p_{t_n} \odot k_{t_0}$$

$$\tilde{p}_{\mu_0} = \mathsf{sum}(\tilde{p}_{t_0,\cdot})$$
$$\rho_0 = \sum \tilde{p}_{\mu_0}$$

$$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$$
$$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$$
$$\ldots$$
$$k_{t_{\mathrm{FF}}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$$

Prime + Probe: attack final-round AES



$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$

$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$

$\ldots$

$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$

$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$

$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$

$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$

$\ldots$

$k_{t_{FF}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$

$\tilde{p}_{t_{0,00}} = p_{t_0} \odot k_{t_0}$

$\tilde{p}_{t_{1,00}} = p_{t_1} \odot k_{t_0}$

$\ldots$

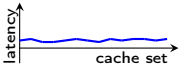
$\tilde{p}_{t_{n,00}} = p_{t_n} \odot k_{t_0}$

$\tilde{p}_{t_{0,01}} = p_{t_0} \odot k_{t_1}$

$\tilde{p}_{\mu_0} = \mathsf{sum}(\tilde{p}_{t_0,\cdot})$

$\rho_0 = \sum \tilde{p}_{\mu_0}$

Prime + Probe: attack final-round AES



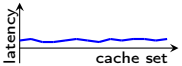
$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

$$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$$
$$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$$
$$\ldots$$
$$k_{t_{FF}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$$

$$\tilde{p}_{t_0,00} = p_{t_0} \odot k_{t_0}$$
$$\tilde{p}_{t_1,00} = p_{t_1} \odot k_{t_0}$$
$$\ldots$$
$$\tilde{p}_{t_n,00} = p_{t_n} \odot k_{t_0}$$
$$\tilde{p}_{t_0,01} = p_{t_0} \odot k_{t_1}$$
$$\tilde{p}_{t_1,01} = p_{t_1} \odot k_{t_1}$$

$$\tilde{p}_{\mu_0} = \mathsf{sum}(\tilde{p}_{t_0,\cdot})$$
$$\rho_0 = \sum \tilde{p}_{\mu_0}$$

Prime + Probe: attack final-round AES



$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

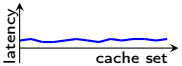
$$\tilde{p}_{t_{0,00}} = p_{t_0} \odot k_{t_0}$$
$$\tilde{p}_{t_{1,00}} = p_{t_1} \odot k_{t_0}$$
$$\ldots$$
$$\tilde{p}_{t_{n,00}} = p_{t_n} \odot k_{t_0}$$
$$\tilde{p}_{t_{0,01}} = p_{t_0} \odot k_{t_1}$$
$$\tilde{p}_{t_{1,01}} = p_{t_1} \odot k_{t_1}$$
$$\ldots$$

$$\tilde{p}_{\mu_0} = \mathsf{sum}(\tilde{p}_{t_0,\cdot})$$
$$\rho_0 = \sum \tilde{p}_{\mu_0}$$

$$\tilde{p}_{\mu_1} = \mathsf{sum}(\tilde{p}_{t_1,\cdot})$$
$$\rho_1 = \sum \tilde{p}_{\mu_1}$$

$$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$$
$$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$$
$$\ldots$$
$$k_{t_{\mathrm{FF}}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$$

Prime + Probe: attack final-round AES



$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$

$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$

$\ldots$

$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$

$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$

$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$

$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$

$\ldots$

$k_{t_{\mathrm{FF}}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$

$\tilde{p}_{t_{0,00}} = p_{t_0} \odot k_{t_0}$

$\tilde{p}_{t_{1,00}} = p_{t_1} \odot k_{t_0}$

$\ldots$

$\tilde{p}_{t_{n,00}} = p_{t_n} \odot k_{t_0}$

$\tilde{p}_{t_{0,01}} = p_{t_0} \odot k_{t_1}$

$\tilde{p}_{t_{1,01}} = p_{t_1} \odot k_{t_1}$

$\ldots$

$\ldots$

$\ldots$

$\tilde{p}_{t_{n,\mathrm{FF}}} = p_{t_n} \odot k_{t_{\mathrm{FF}}}$

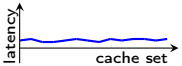
$\tilde{p}_{\mu_0} = \mathsf{sum}(\tilde{p}_{t_0,\cdot})$

$\rho_0 = \sum \tilde{p}_{\mu_0}$

$\tilde{p}_{\mu_1} = \mathsf{sum}(\tilde{p}_{t_1,\cdot})$

$\rho_1 = \sum \tilde{p}_{\mu_1}$

Prime + Probe: attack final-round AES

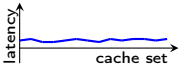


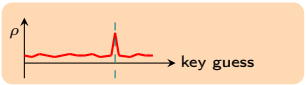
$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

$$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$$
$$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$$
$$\ldots$$
$$k_{t_{\mathrm{FF}}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$$

$$\tilde{p}_{t_{0,00}} = p_{t_0} \odot k_{t_0} \qquad \tilde{p}_{\mu_0} = \mathsf{sum}(\tilde{p}_{t_0,\cdot}) \qquad \rho_0 = \sum \tilde{p}_{\mu_0}$$
$$\tilde{p}_{t_{1,00}} = p_{t_1} \odot k_{t_0}$$
$$\ldots$$
$$\tilde{p}_{t_{n,00}} = p_{t_n} \odot k_{t_0}$$
$$\tilde{p}_{t_{0,01}} = p_{t_0} \odot k_{t_1} \qquad \tilde{p}_{\mu_1} = \mathsf{sum}(\tilde{p}_{t_1,\cdot}) \qquad \rho_1 = \sum \tilde{p}_{\mu_1}$$
$$\tilde{p}_{t_{1,01}} = p_{t_1} \odot k_{t_1}$$
$$\ldots \qquad \ldots$$
$$\ldots \qquad \tilde{p}_{\mu_{\mathrm{FF}}} = \mathsf{sum}(\tilde{p}_{t_{\mathrm{FF},\cdot}})$$
$$\ldots \qquad \rho_{15} = \sum \tilde{p}_{\mu_{\mathrm{FF}}}$$
$$\tilde{p}_{t_{n,\mathrm{FF}}} = p_{t_n} \odot k_{t_{\mathrm{FF}}}$$

Prime + Probe: attack final-round AES





$$p_{t_0} = [t_0^0,\ t_0^1, \ldots, t_0^{16}, t_0^{17}, \ldots, t_0^{63}]$$
$$p_{t_1} = [t_1^0,\ t_1^1, \ldots, t_1^{16}, t_1^{17}, \ldots, t_1^{63}]$$
$$\ldots$$
$$p_{t_n} = [t_n^0,\ t_n^1, \ldots, t_n^{16}, t_n^{17}, \ldots, t_n^{63}]$$

$$k_{t_{00}} = [\ 0, 15, \ldots, -1, -1, \ldots,\ 0]$$
$$k_{t_{01}} = [\ 0, -1, \ldots, -1, 15, \ldots,\ 0]$$
$$k_{t_{02}} = [\ 0, -1, \ldots, -1, -1, \ldots,\ 0]$$
$$\ldots$$
$$k_{t_{\mathrm{FF}}} = [\ 0, -1, \ldots, 15, -1, \ldots,\ 0]$$

$$\tilde{p}_{t_{0,00}} = p_{t_0} \odot k_{t_0}$$
$$\tilde{p}_{t_{1,00}} = p_{t_1} \odot k_{t_0}$$
$$\ldots$$
$$\tilde{p}_{t_{n,00}} = p_{t_n} \odot k_{t_0}$$
$$\tilde{p}_{t_{0,01}} = p_{t_0} \odot k_{t_1}$$
$$\tilde{p}_{t_{1,01}} = p_{t_1} \odot k_{t_1}$$
$$\ldots$$
$$\ldots$$
$$\ldots$$
$$\tilde{p}_{t_{n,\mathrm{FF}}} = p_{t_n} \odot k_{t_{\mathrm{FF}}}$$

$$\tilde{p}_{\mu_0} = \mathsf{sum}(\tilde{p}_{t_0,\cdot})$$
$$\rho_0 = \sum \tilde{p}_{\mu_0}$$

$$\tilde{p}_{\mu_1} = \mathsf{sum}(\tilde{p}_{t_1,\cdot})$$
$$\rho_1 = \sum \tilde{p}_{\mu_1}$$
$$\ldots$$
$$\tilde{p}_{\mu_{\mathrm{FF}}} = \mathsf{sum}(\tilde{p}_{t_{\mathrm{FF},\cdot}})$$
$$\rho_{15} = \sum \tilde{p}_{\mu_{\mathrm{FF}}}$$

Prime + Probe: pros & cons

+ Work on non-shared memory

Prime + Probe: pros & cons

+ Work on non-shared memory

- Resolution of cache set
- Suffer from false positive/negative

Comparison

Flush+Reload	**Prime+Probe**
more accurate	more generic
require shared memory	fewer prerequisites
identify cache line	identify cache set

Other Variants

- Evict + Time [OST05]
- Evict + Reload [GSM15]
- Flush + Flush [GMWM16]
- Prime + Abort [DKPT17]
- Prime + Scope [PTV21]
- Prefetch + Prefetch [GZZY22]
- etc.

Summary

- Mastik: toolkit to collect cache timing side-channel information
- General steps in cache attacks:

1. Prepare cache
2. (victim executes)
3. Get timing difference

Summary

- Mastik: toolkit to collect cache timing side-channel information
- General steps in cache attacks:

	Flush + Reload
1. Prepare cache	clear cache line
2. (victim executes)	
3. Get timing difference	reload cache line

Summary

- Mastik: toolkit to collect cache timing side-channel information
- General steps in cache attacks:

	Flush + Reload	Prime + Probe
1. Prepare cache	clear cache line	fill cache with own data
2. (victim executes)		
3. Get timing difference	reload cache line	re-access own data