Cache Games

Bringing Access-Based Cache Attacks on AES to Practice

David Gullasch [1,2] Endre Bangerter [1]
Stephan Krenn [1,3]

[1]Bern University of Applied Sciences

[2]Dreamlab Technologies AG

[3]University of Fribourg

COSADE 2011-02-25

Attacking AES via CPU Cache

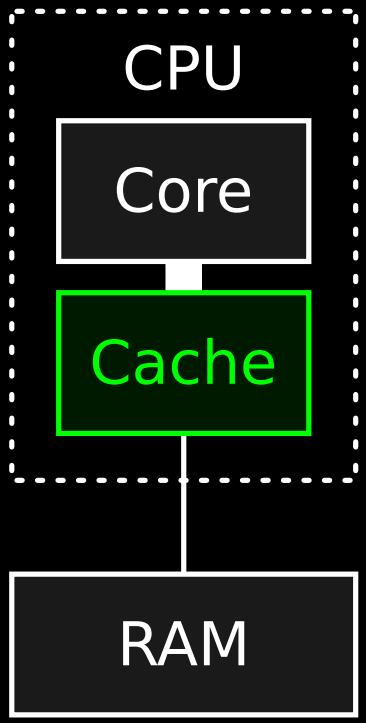
- fully working access-driven attack
- recovery of key, plain- and ciphertext
  - needs to observe ≈ 100 encryptions
  - victim experiences only minor delay (< 3 s)
  - no access to ciphertext needed
- solutions to practical difficulties
  - DoS on Linux' task scheduler (CFS)
  - noise removal
  - fast (≈ 3 minutes) key search

Overview

- Basics
  - Cache Sidechannel
  - AES
- Attack
  - Measuring Cache Timings
  - ⋮
  - Key Recovery
- Live Demonstration
- Outlook

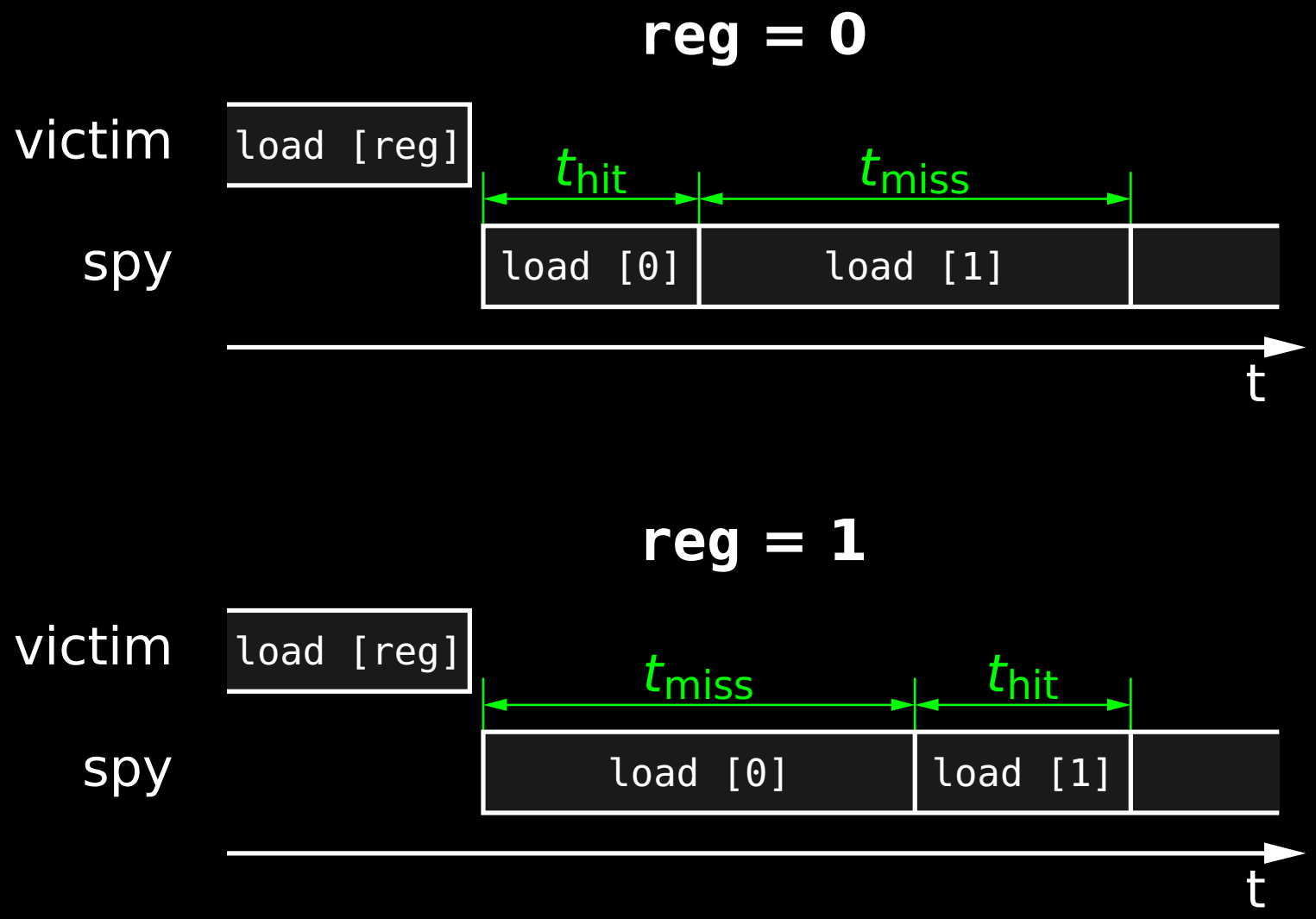
Details: Cache Games, IEEE S&P 2011 (to appear)
http://eprint.iacr.org/2010/594

Cache Side Channel



- cache keeps state
- state leaks via memory access timings:
  - $t_{\text{hit}} \approx 10\,\text{ns}$
  - $t_{\text{miss}} \approx 100\,\text{ns}$

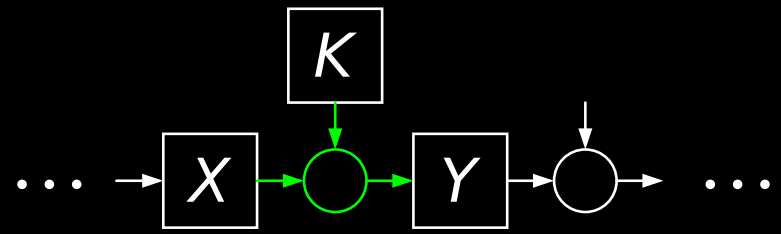
Cache Side Channel



AES: Notation

One Round

$$Y = M \bullet s(\tilde{X}) \oplus K$$



X, Y, M, K	4×4 byte matrices
$\tilde{X}$	ShiftRows(X)
$s(X)$	SubBytes(X)
$M \bullet X$	MixColumns(X)

AES: Implementation

$$\underline{\mathbf{y}_i} = \underline{\mathbf{m}_0} \bullet \mathrm{s}(\tilde{x}_{0,i}) \oplus \ldots \oplus \underline{\mathbf{m}_3} \bullet \mathrm{s}(\tilde{x}_{3,i}) \oplus \underline{\mathbf{k}_i}$$

$\underline{\mathbf{x}_i}$ column vector in X

AES: Implementation

$$\underline{\mathbf{y}_i} = \underline{\mathbf{m}_0} \bullet \mathrm{s}(\tilde{x}_{0,i}) \oplus \ldots \oplus \underline{\mathbf{m}_3} \bullet \mathrm{s}(\tilde{x}_{3,i}) \oplus \underline{\mathbf{k}_i}$$

$$\underline{\mathbf{m}_j} \bullet s(x) = T[x][j \ldots j+3]$$

$$\underline{\mathbf{y}_i} = T[\tilde{x}_{0,i}][0 \ldots 3] \oplus \ldots \oplus T[\tilde{x}_{3,i}][3 \ldots 6] \oplus \underline{\mathbf{k}_i}$$

$\underline{\mathbf{x}_i}$ column vector in X

$T[a \ldots b]$ $(T[a], T[a+1], \ldots, T[b])$

Attack

- Measuring Cache Timings
- Scheduler DoS
- Noise Removal
- Leaking Bits
- Key Recovery

Measuring Cache Timings

How

Where

When

Measuring Cache Timings

How

- `rdtsc`, memory access, `rdtsc`

Where

When

Measuring Cache Timings

How

- `rdtsc`, memory access, `rdtsc`

Where

- AES lookup table T in shared memory

When

Measuring Cache Timings

How

- `rdtsc`, memory access, `rdtsc`

Where

- AES lookup table T in shared memory
- for each cacheline (64 bytes)

When

Measuring Cache Timings

How

- `rdtsc`, memory access, `rdtsc`

Where

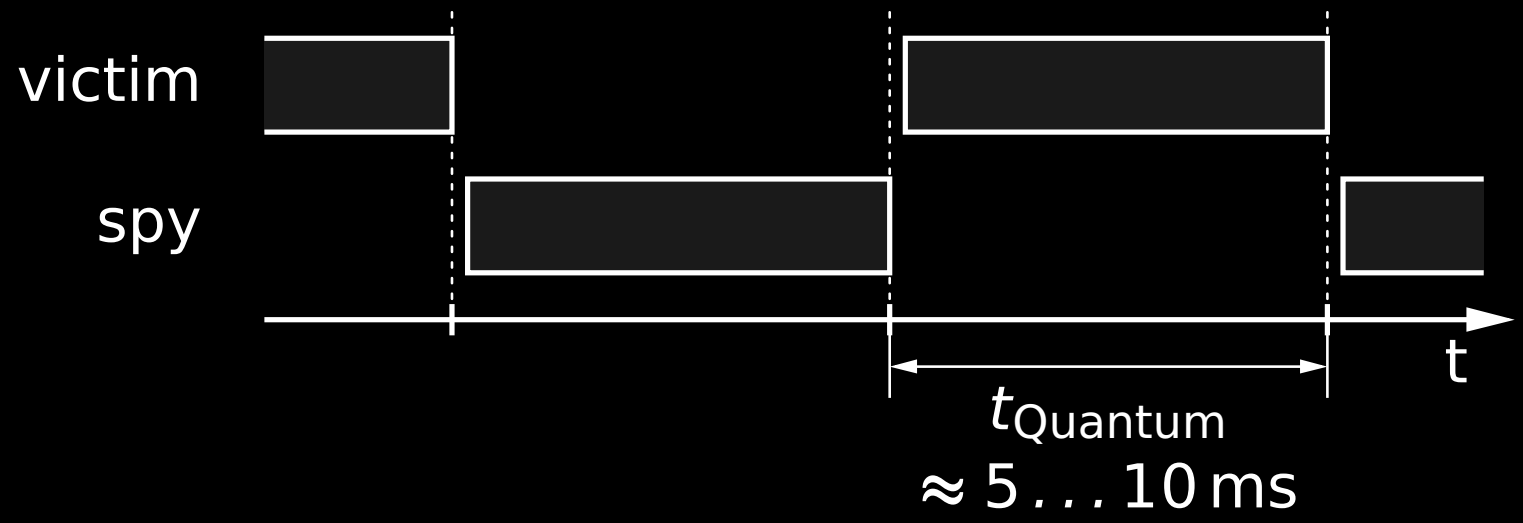
- AES lookup table T in shared memory
- for each cacheline (64 bytes)

When

- after every memory access in victim

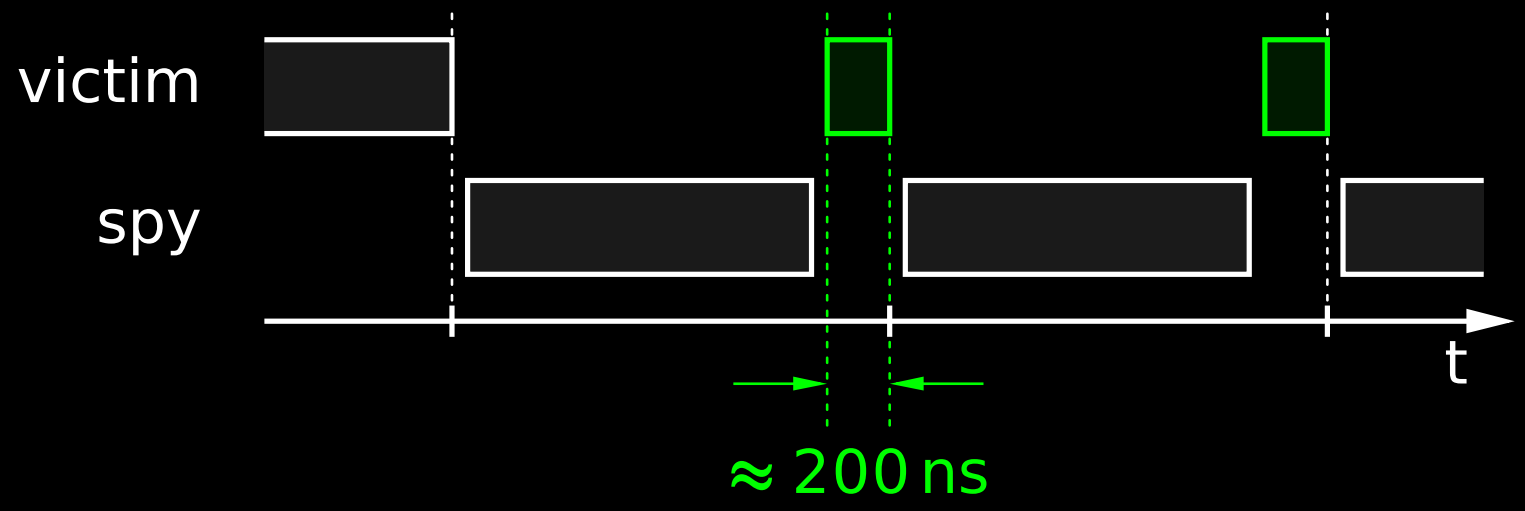
Scheduler DoS

textbook task scheduling



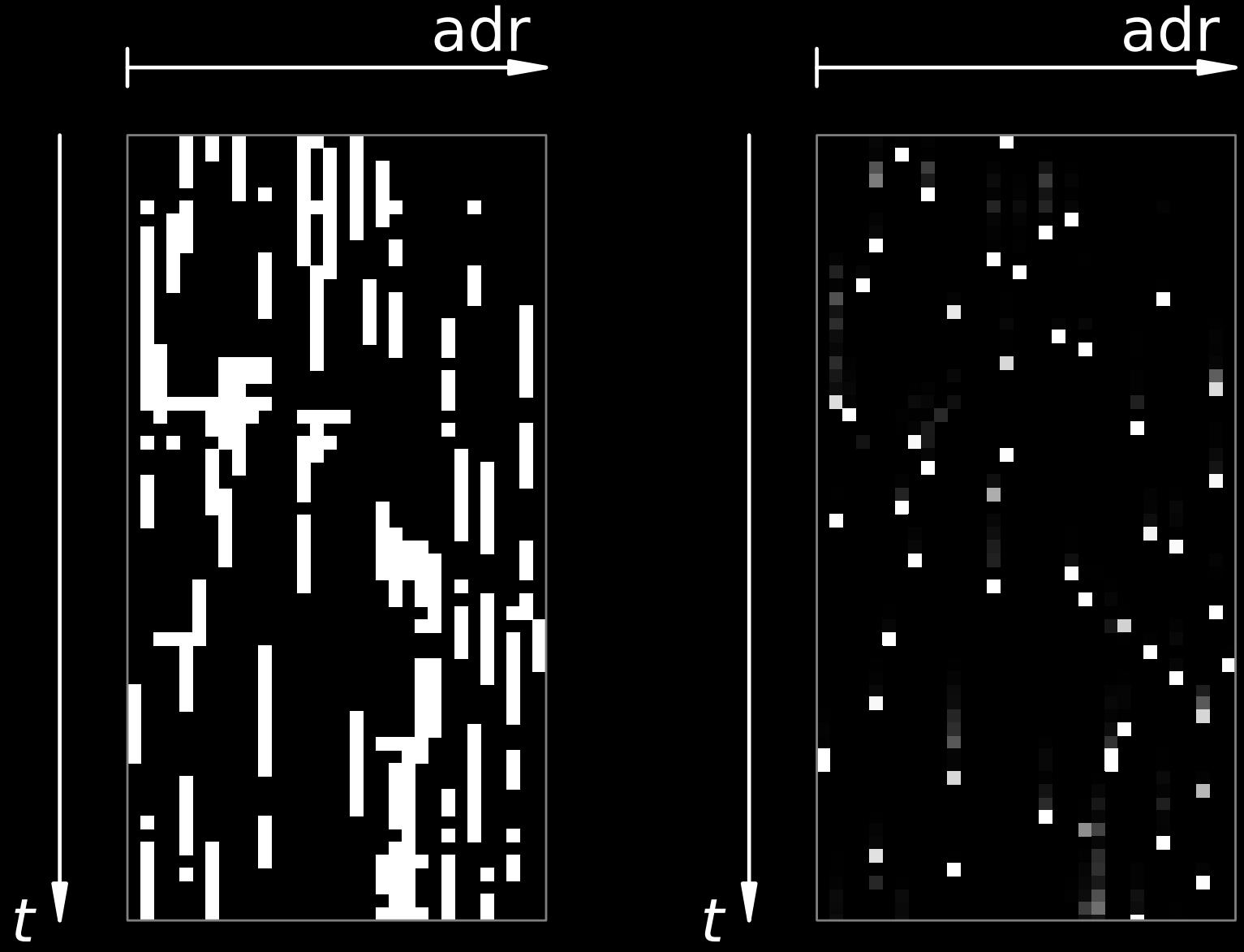
Scheduler DoS

yield CPU before interrupt



⇒ one memory access retires in victim

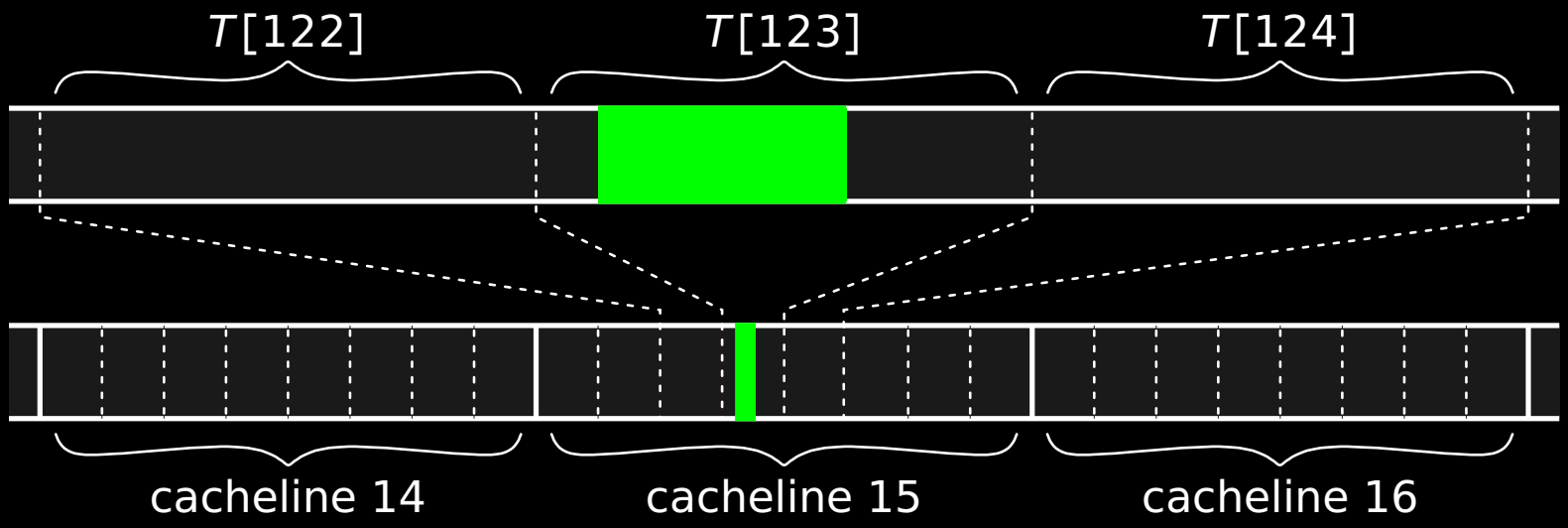
Noise Removal



Leaking Bits

$$x = 123 = 01111011_b$$

$$\underline{\mathbf{y}} = \ldots \oplus T[x][1\ldots4] \oplus \ldots \oplus \underline{\mathbf{k}}$$



$$x^* = 15 = 01111_b$$

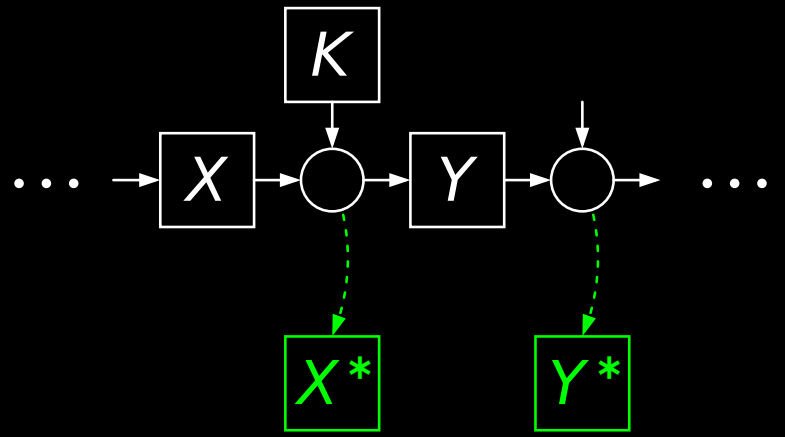
---

x^* 5 most significant bits of x

AES: Consecutive Rounds

$$\underline{\mathbf{y}} = M \bullet \mathrm{s}(\underline{\tilde{\mathbf{x}}}) \oplus \underline{\mathbf{k}}$$

$$\Rightarrow \quad \underline{\mathbf{k}}^* = \left(M \bullet \mathrm{s}(\underline{\tilde{\mathbf{x}}})\right)^* \oplus \underline{\mathbf{y}}^*$$

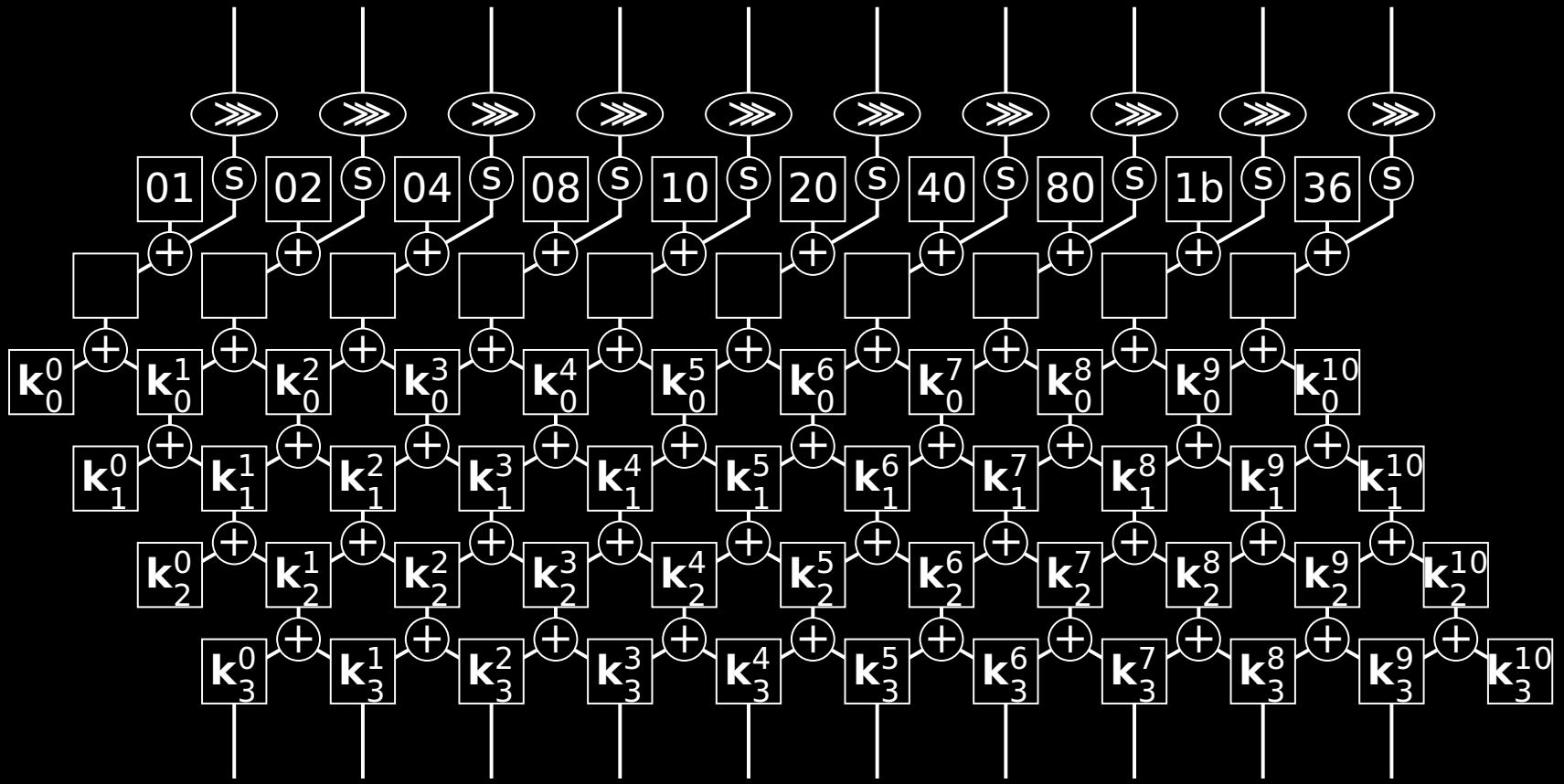


score σ: log-probability of $\underline{\mathbf{k}}^*$ occuring in key

---

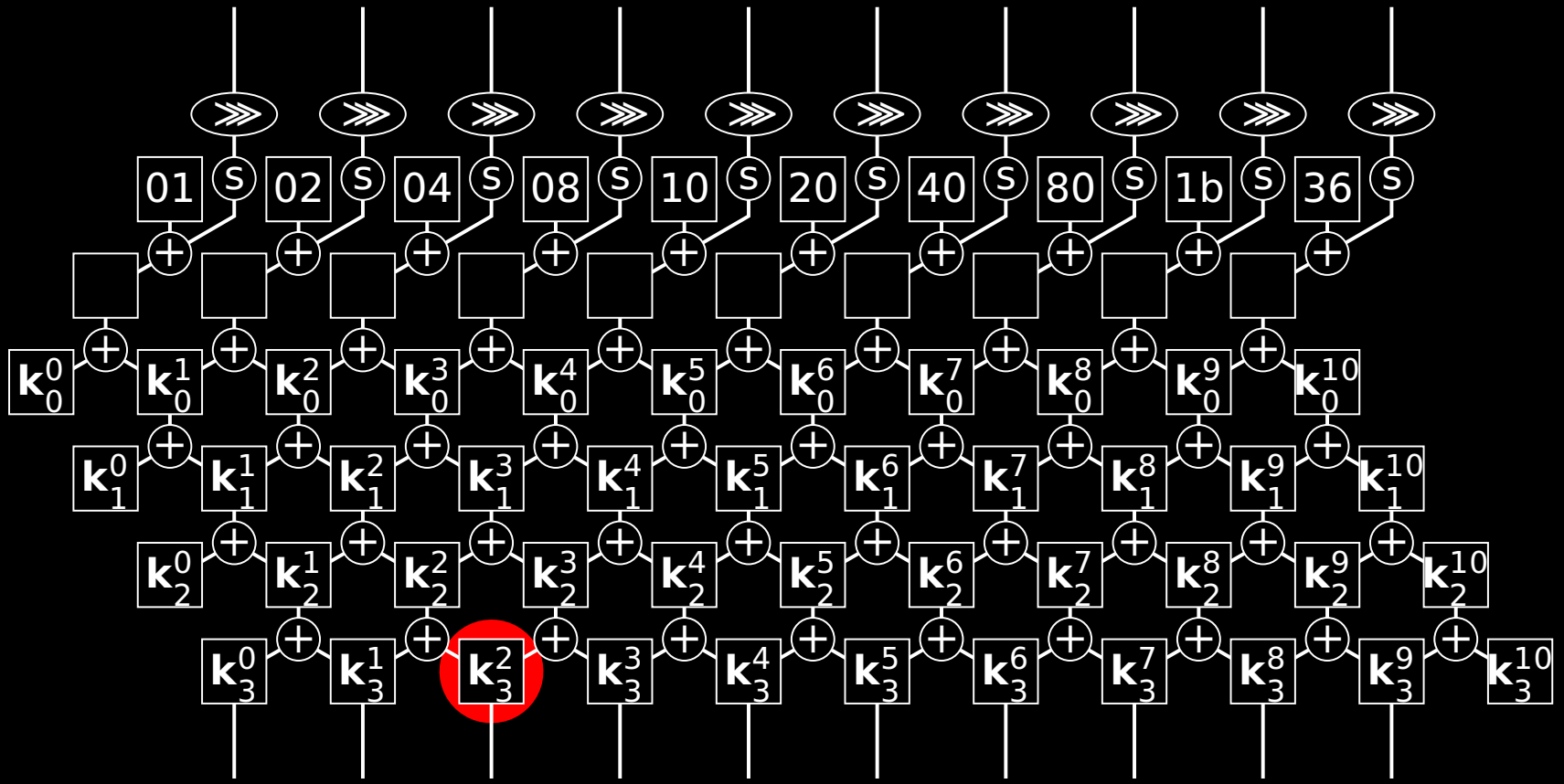
$X^*, \underline{\mathbf{x}}^*$ $X, \underline{\mathbf{x}}$ with all bytes x replaced by x^*

AES-128 Key Recovery



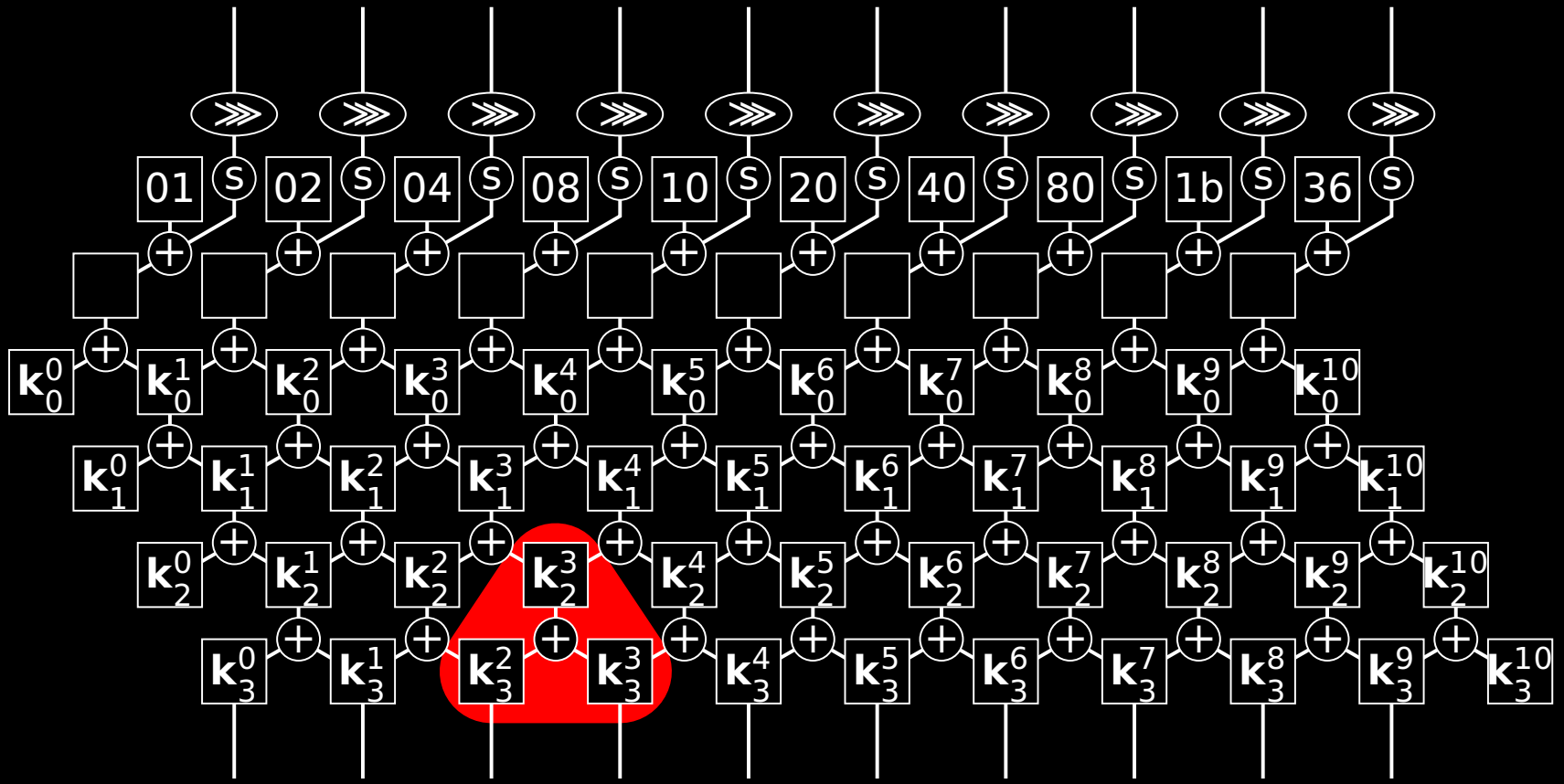
algebraic relation of $\underline{\mathbf{k}}_{column}^{round}$ in complete key

AES-128 Key Recovery



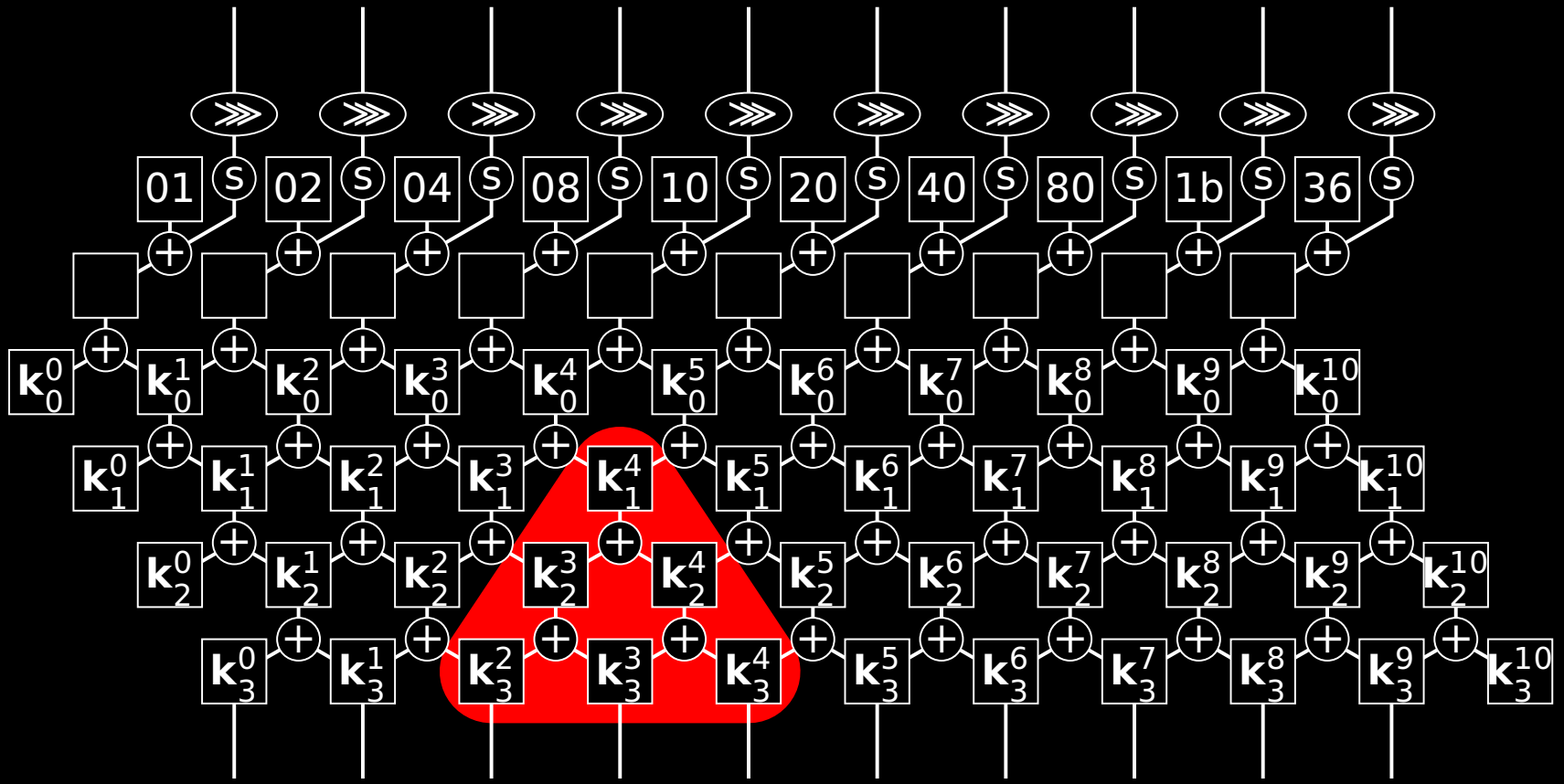
$$\sigma_{\bullet} = \sigma_3^2$$

AES-128 Key Recovery



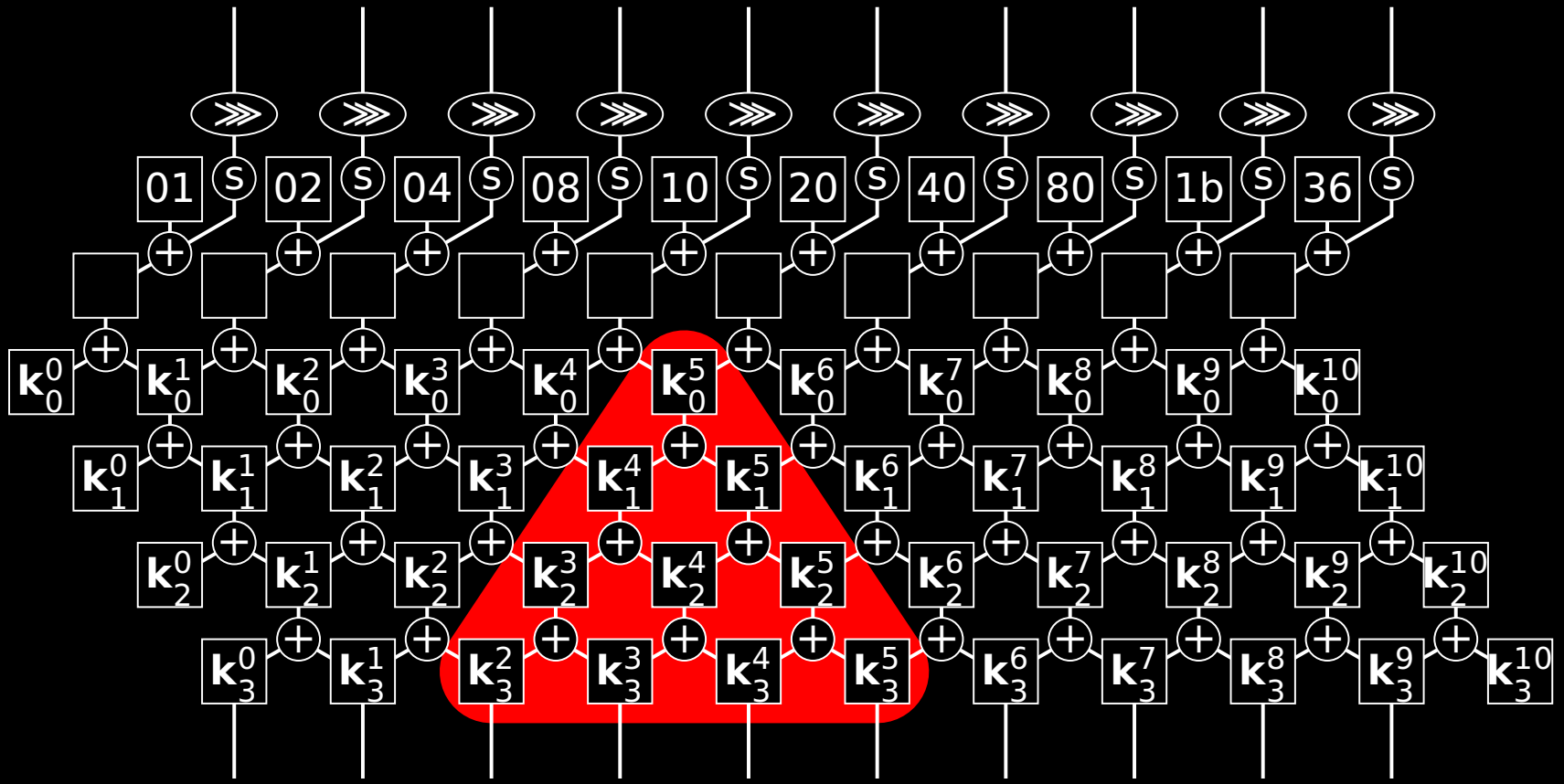
$$\sigma_{\blacktriangle} = \frac{1}{3}\left(\sigma_3^2 + \sigma_2^3 + \sigma_3^3\right)$$

AES-128 Key Recovery



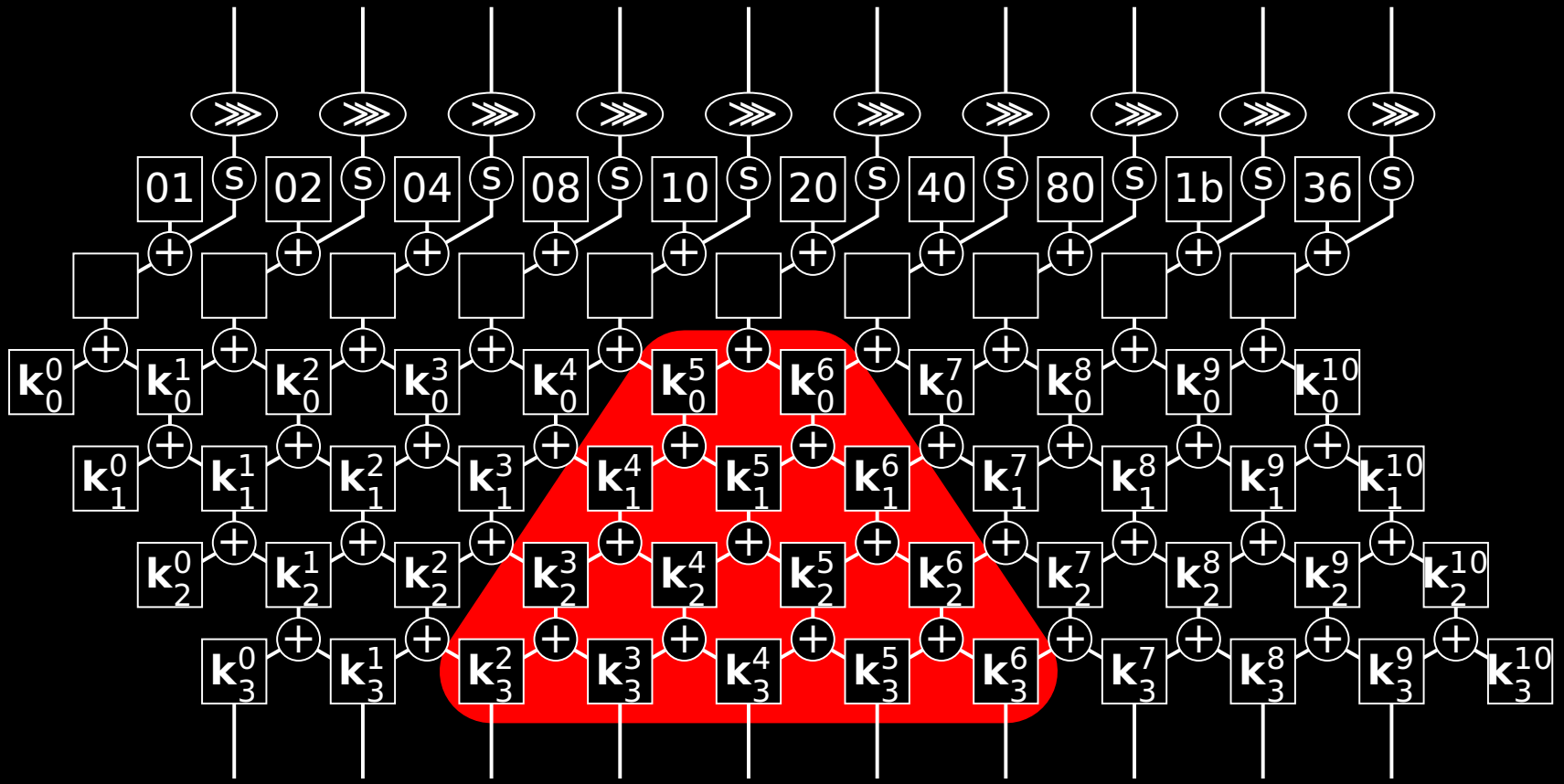
$$\sigma_{\blacktriangle} = \frac{1}{6}\left(\sigma_3^2 + \sigma_2^3 + \sigma_3^3 + \sigma_1^4 + \sigma_2^4 + \sigma_3^4\right)$$

AES-128 Key Recovery



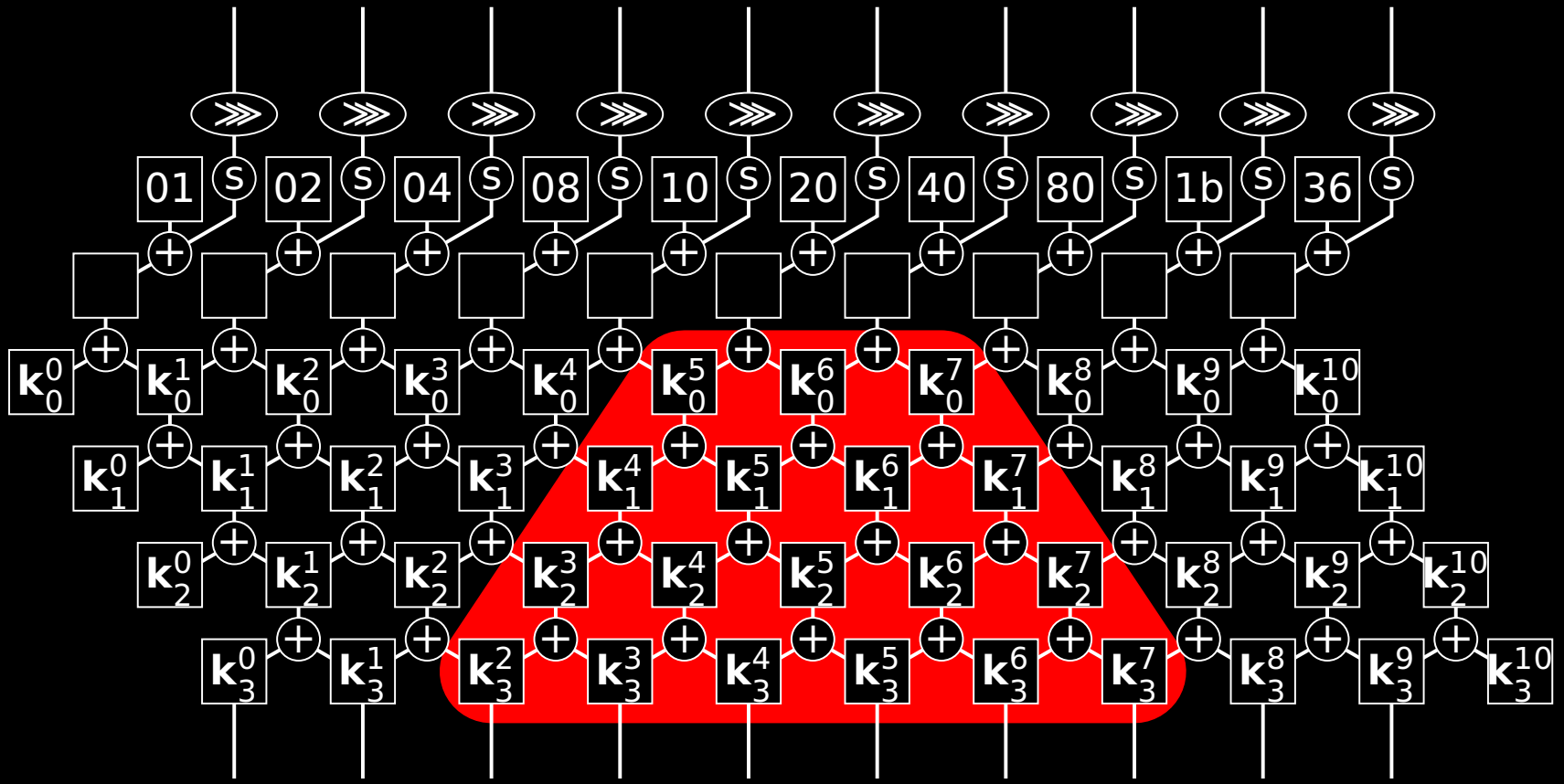
$$\sigma_{\blacktriangle} = \frac{1}{10}\left(\sigma_3^2 + \ldots + \sigma_3^5\right)$$

AES-128 Key Recovery



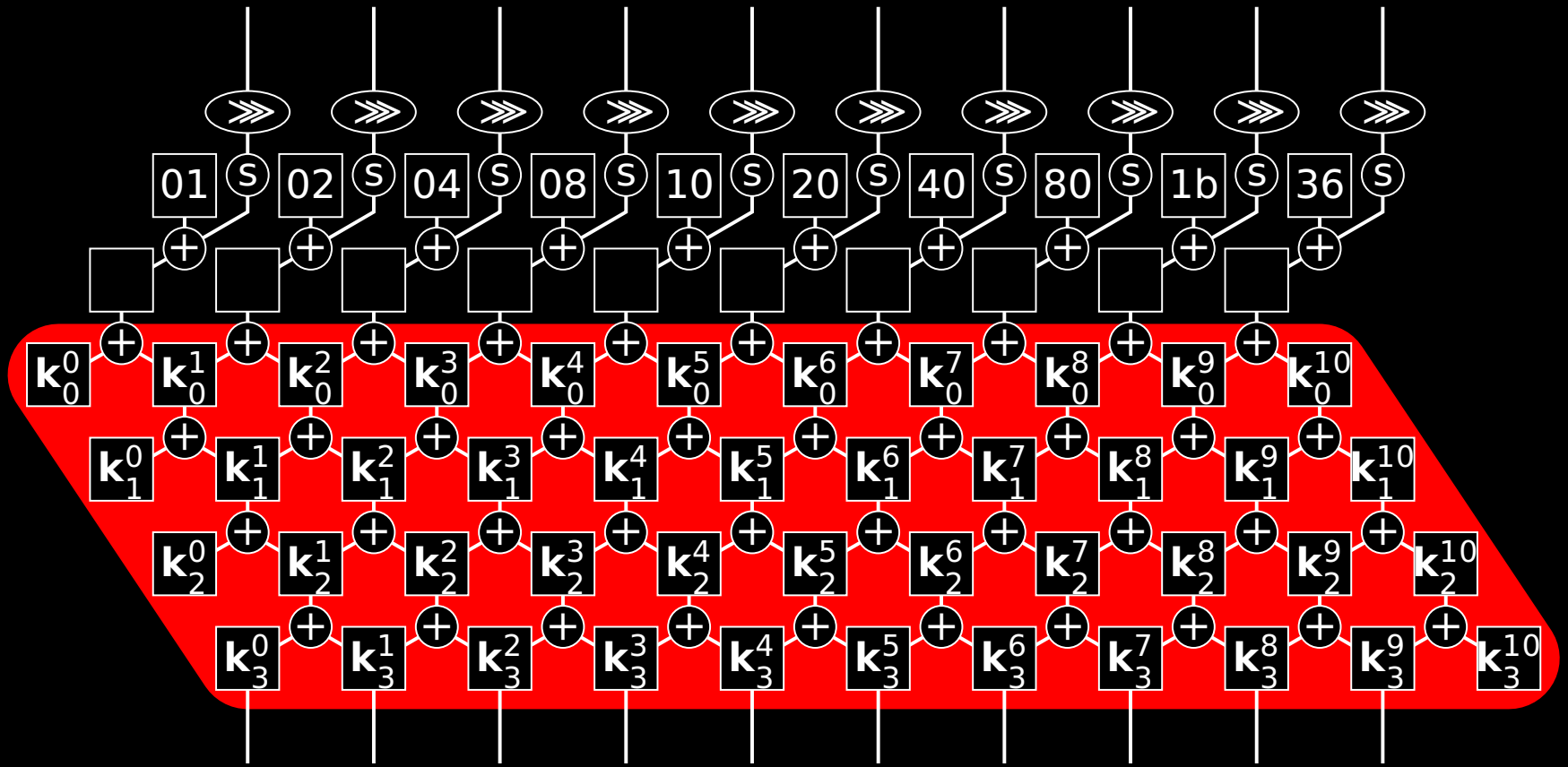
$$\sigma_{\triangle} = \frac{1}{14}\left(\sigma_3^2 + \ldots + \sigma_3^6\right)$$

AES-128 Key Recovery



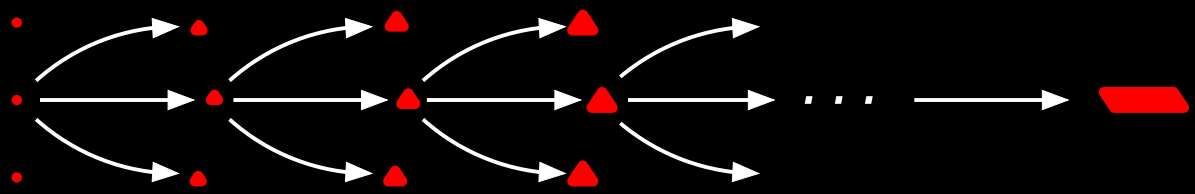
$$\sigma_{\blacktriangle} = \frac{1}{18}\left(\sigma_3^2 + \ldots + \sigma_3^7\right)$$

AES-128 Key Recovery



$$\sigma = \frac{1}{44} \sum_{i=0}^{10} \sum_{j=0}^{3} \sigma_j^i$$

AES-128 Key Recovery



- systematic search over fragments
- mean score: heuristic
- complete key with maximal mean score: key used during encryption

Demo

Setting

- Pentium M (Banias, 1.5GHz, 512kB L2)
- Linux Kernel 2.6.33.4 (Arch Linux default)

Players

- victim process: AES-128, OpenSSL 0.9.8n
- spy process: mmap() libcrypto, RDTSC

Outlook

- AES-192, AES-256
- multicore setting
- OpenSSL 1.0
- attack across VM boundaries