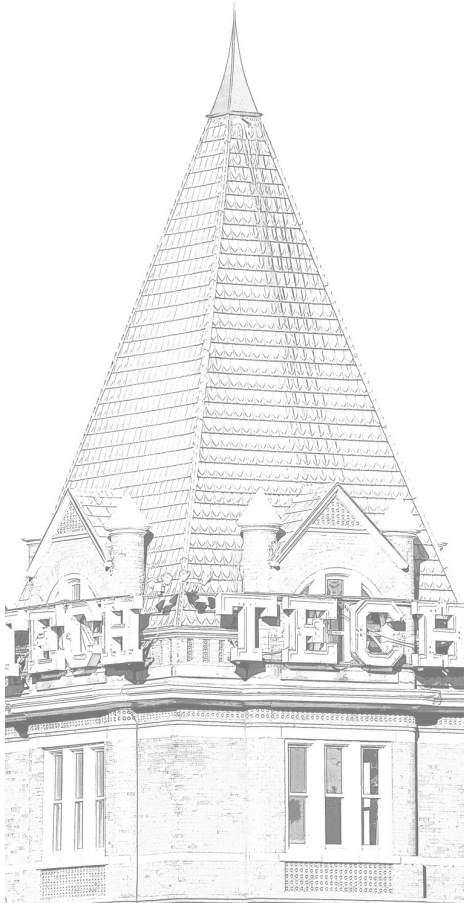
DEUCE: WRITE-EFFICIENT ENCRYPTION FOR PCM



March 16th 2015
ASPLOS-XX Istanbul, Turkey

Vinson Young
Prashant J. Nair
Moinuddin K. Qureshi

Georgia Institute of Technology

EXECUTIVE SUMMARY

- Bit flips expensive in Phase Change Memory (PCM)
  - Affects Lifetime, Power, and Performance
  - PCM system optimized to reduce bit flips (~12%)

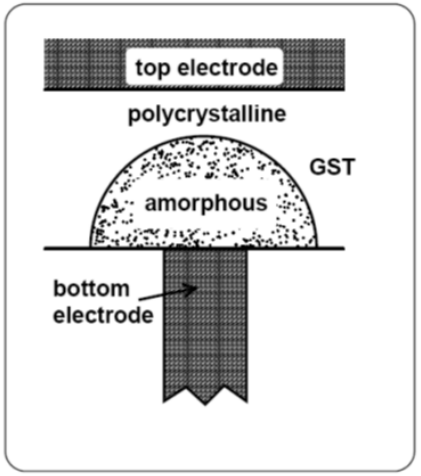
- PCM more vulnerable to attacks → stolen module
- Secure PCM with encryption
- Problem: Encryption increases bit flips 12%→50%
- Goal: Encrypt PCM without causing 4x bit flips

- Insight: re-encrypt only modified words
- DEUCE: write-efficient encryption scheme
- Result: bit flips reduced 50%→23% (27% speedup)

PCM AS MAIN MEMORY

Phase Change Memory (PCM)

- Improved scaling + density
- Non-volatile (no refresh)



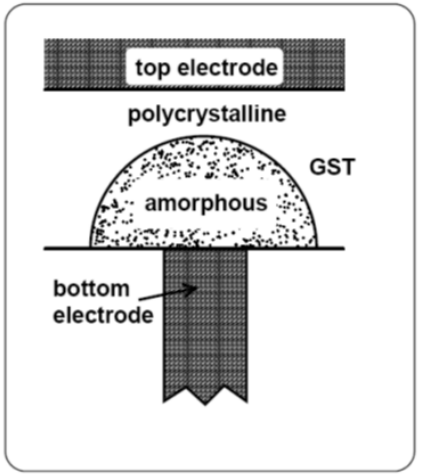
Key Challenge

- Writes: Limited endurance (10-100 million writes)
- Writes: Slow and limited throughput
- Writes: Power-hungry

PCM AS MAIN MEMORY

Phase Change Memory (PCM)

- Improved scaling + density
- Non-volatile (no refresh)



Key Challenge

- Writes: Limited endurance (10-100 million writes)
- Writes: Slow and limited throughput
- Writes: Power-hungry

PCM systems are designed to reduce writes

TYPICAL WRITE OPTIMIZATIONS IN PCM

Data-Comparison-Write

Write only flipped bits

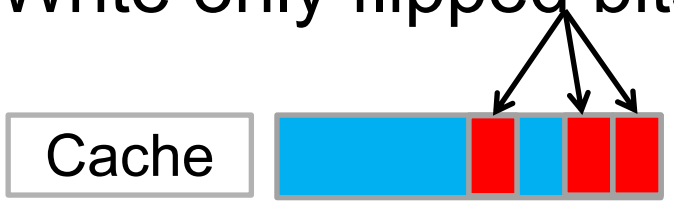
Cache

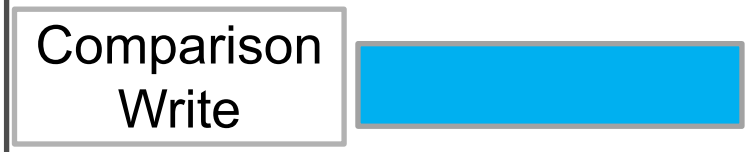
Comparison Write

TYPICAL WRITE OPTIMIZATIONS IN PCM

Data-Comparison-Write

Write only flipped bits

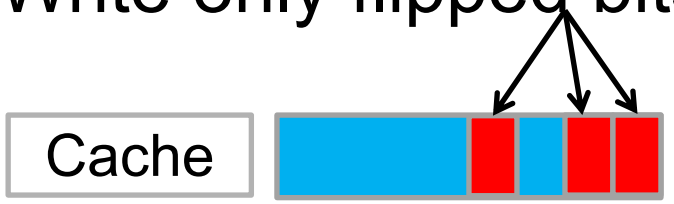


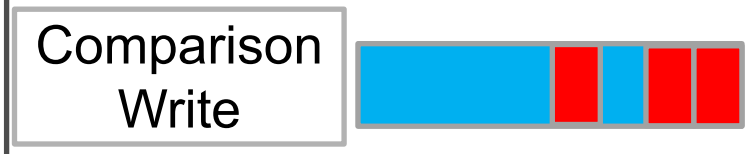


TYPICAL WRITE OPTIMIZATIONS IN PCM

Data-Comparison-Write

Write only flipped bits

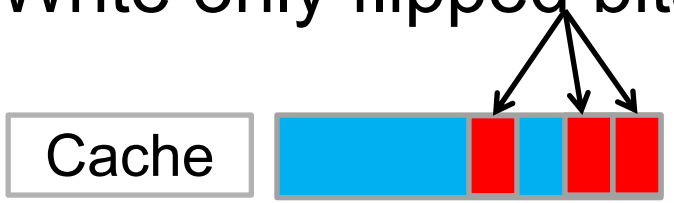




TYPICAL WRITE OPTIMIZATIONS IN PCM

Data-Comparison-Write

Write only flipped bits



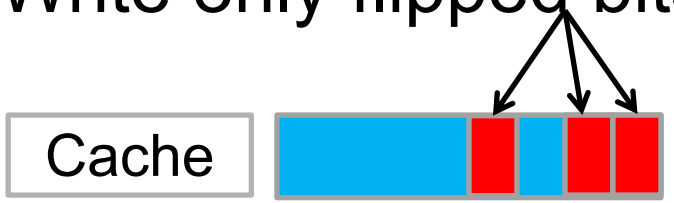
Flip-N-Write

Invert if too many bit flips

TYPICAL WRITE OPTIMIZATIONS IN PCM

Data-Comparison-Write

Write only flipped bits



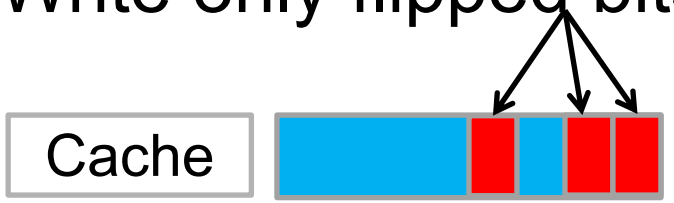
Flip-N-Write

Invert if too many bit flips

TYPICAL WRITE OPTIMIZATIONS IN PCM

Data-Comparison-Write

Write only flipped bits



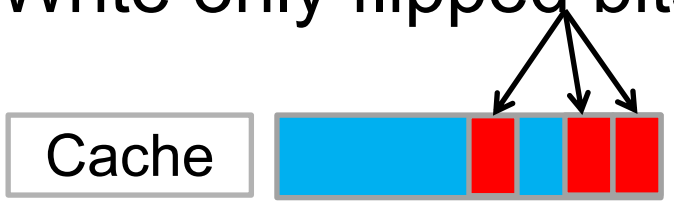
Flip-N-Write

Invert if too many bit flips

TYPICAL WRITE OPTIMIZATIONS IN PCM

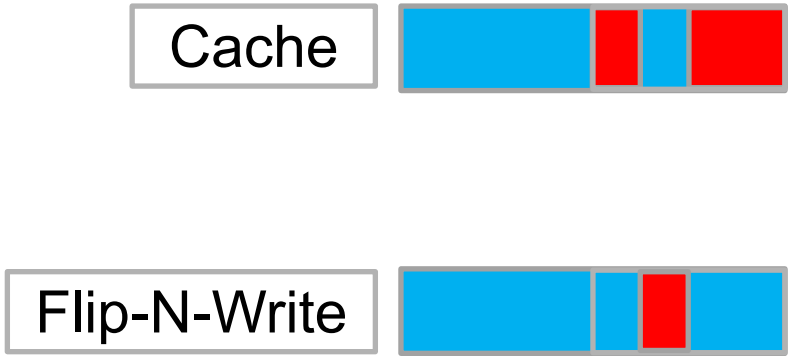
Data-Comparison-Write

Write only flipped bits



Flip-N-Write

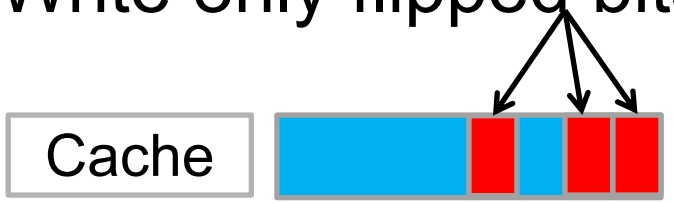
Invert if too many bit flips

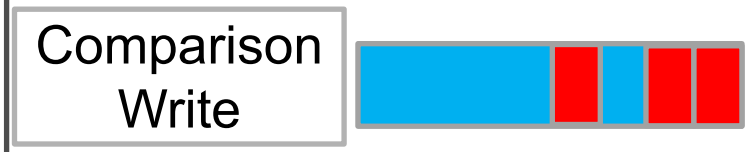


TYPICAL WRITE OPTIMIZATIONS IN PCM

Data-Comparison-Write

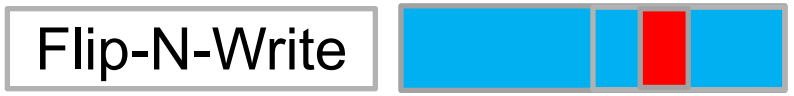
Write only flipped bits





Flip-N-Write

Invert if too many bit flips



Optimizations reduce bit-flips per write to 10-12%

SECURITY VULNERABILITY OF PCM

Non-volatility: Power savings  Security 

More vulnerable to stolen-memory attack

SECURITY VULNERABILITY OF PCM

Non-volatility: Power savings  Security 

More vulnerable to stolen-memory attack

Stolen memory attack

SECURITY VULNERABILITY OF PCM

Non-volatility: Power savings  Security 

More vulnerable to stolen-memory attack

Stolen memory attack

SECURITY VULNERABILITY OF PCM

Non-volatility: Power savings  Security 

More vulnerable to stolen-memory attack

Stolen memory attack

Bus snooping attack also possible

SECURITY VULNERABILITY OF PCM

Non-volatility: Power savings  Security 

More vulnerable to stolen-memory attack

Stolen memory attack

Bus snooping attack also possible

SECURITY VULNERABILITY OF PCM

Non-volatility: Power savings  Security 

More vulnerable to stolen-memory attack

Stolen memory attack

Bus snooping attack also possible

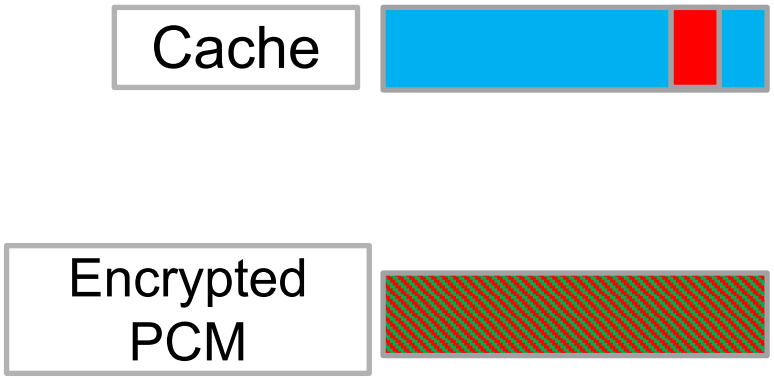
We want to protect PCM from both
"stolen memory attack" and "bus snooping attack"

ENCRYPTION ON PCM

Protect PCM using memory encryption

Encryption causes 50% bit flips on each write

- 1 bit flip in line ➔ 50% bit flips in encrypted line

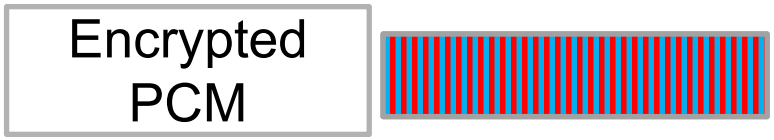


ENCRYPTION ON PCM

Protect PCM using memory encryption

Encryption causes 50% bit flips on each write

- 1 bit flip in line ➔ 50% bit flips in encrypted line



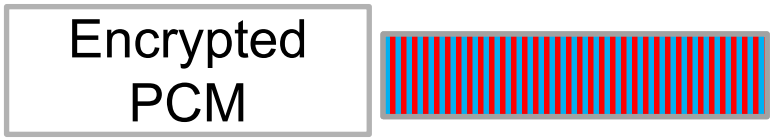
Avalanche Effect

ENCRYPTION ON PCM

Protect PCM using memory encryption

Encryption causes 50% bit flips on each write

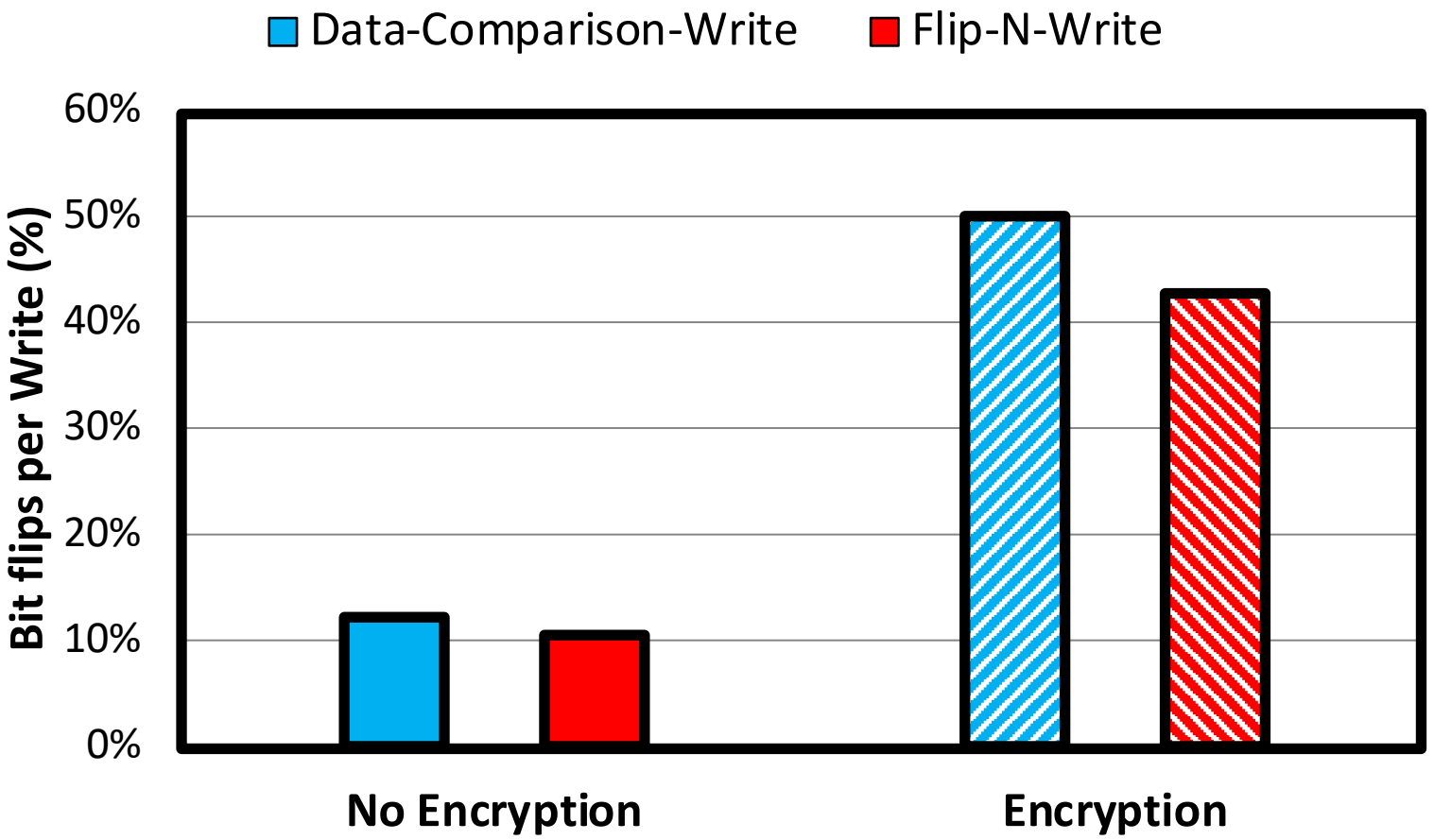
- 1 bit flip in line ➔ 50% bit flips in encrypted line



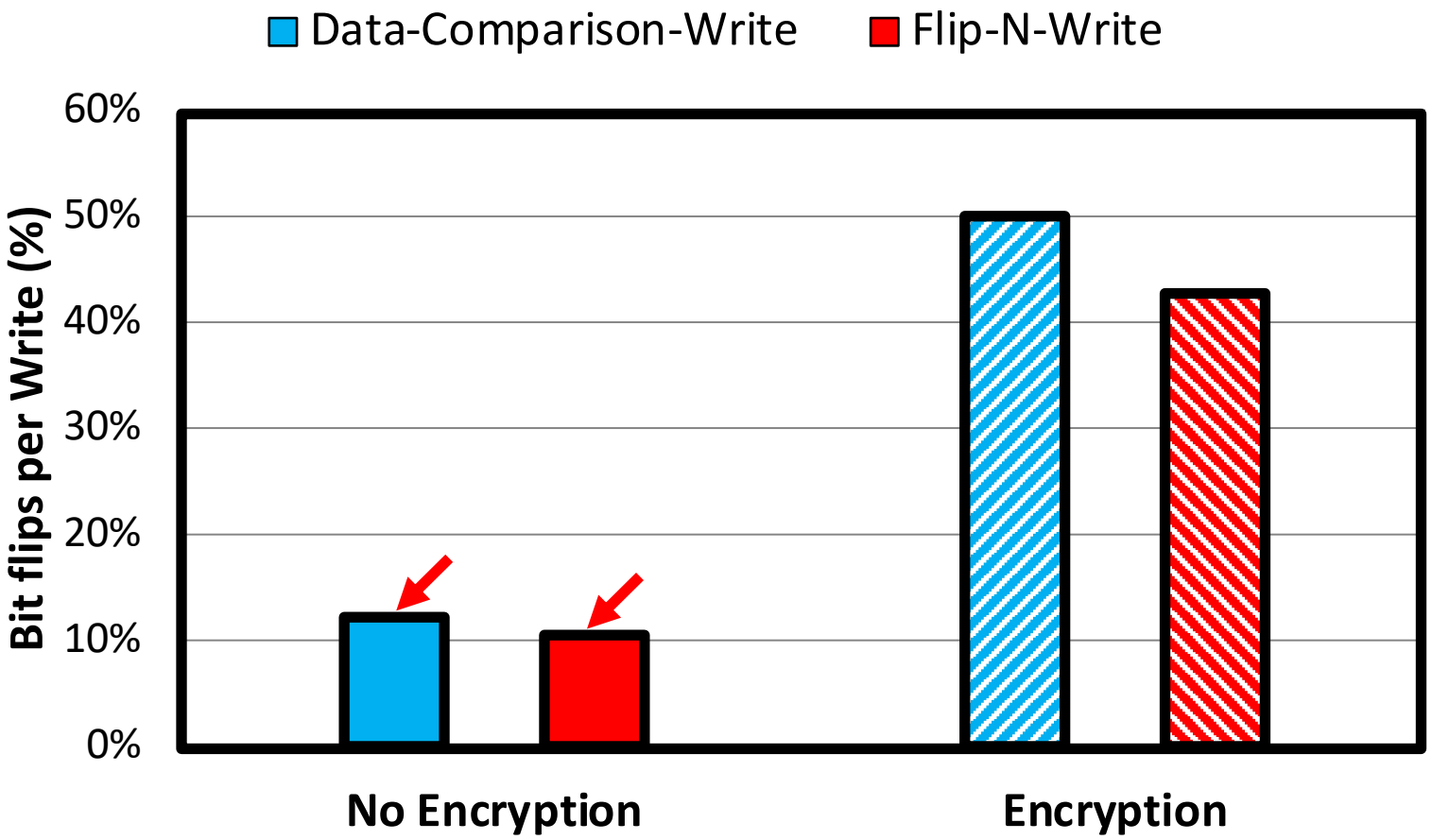
Avalanche Effect

Encryption causes 50% bit flips ➔ Write-intensive

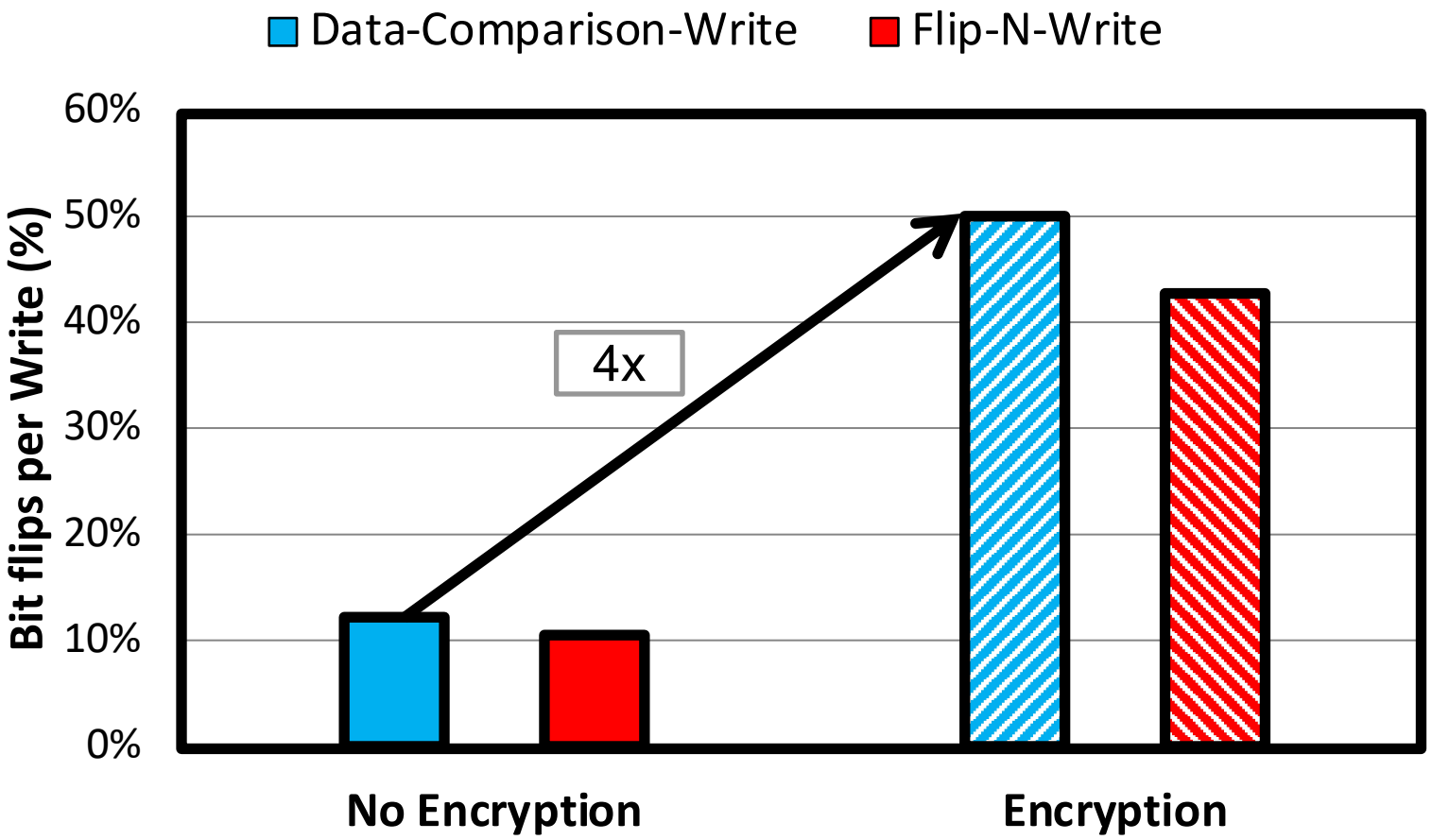
ENCRYPTION ON PCM COSTLY



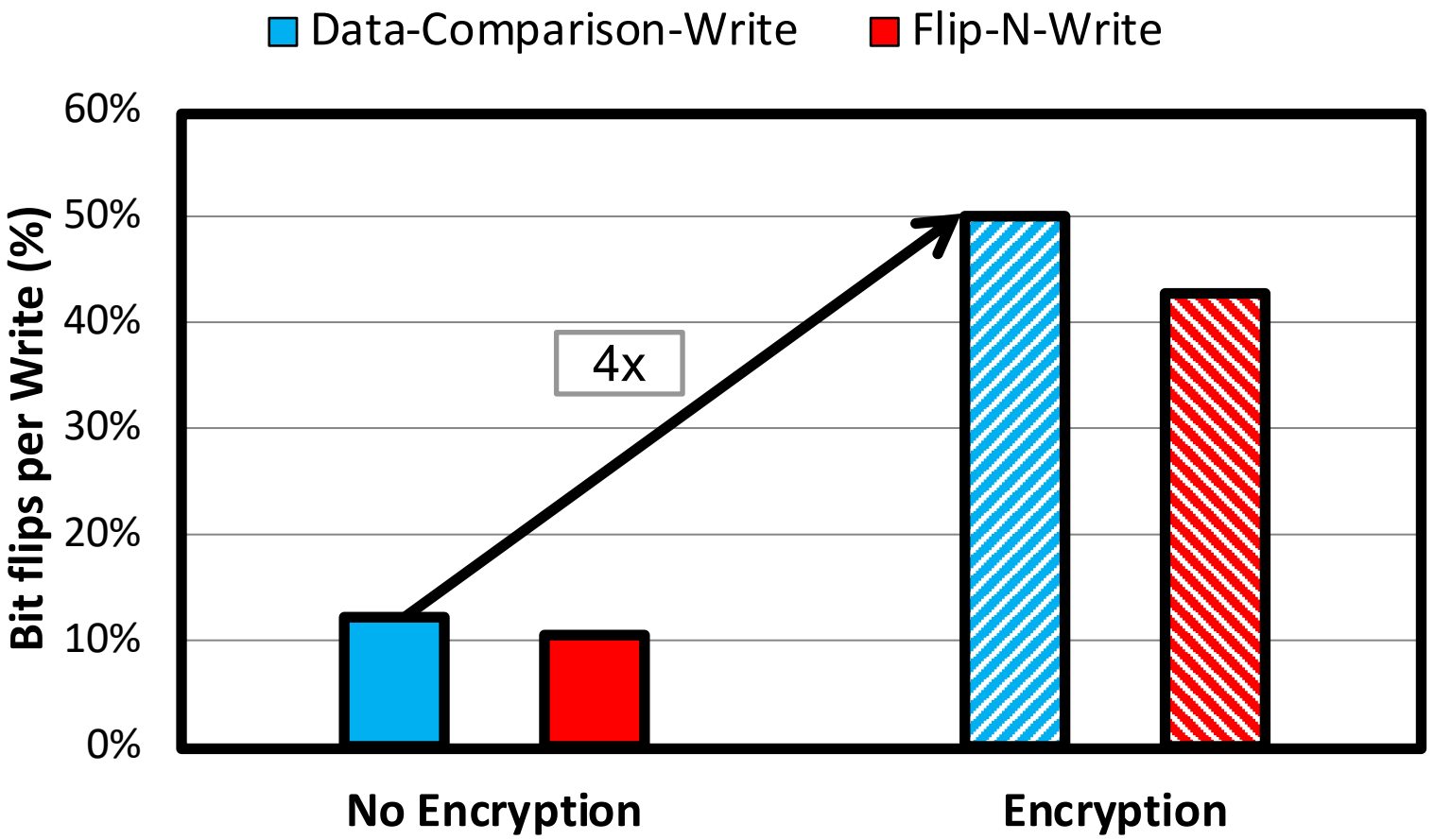
ENCRYPTION ON PCM COSTLY



ENCRYPTION ON PCM COSTLY



ENCRYPTION ON PCM COSTLY



Encryption increases bit flips from 12% to 50% (4x!)

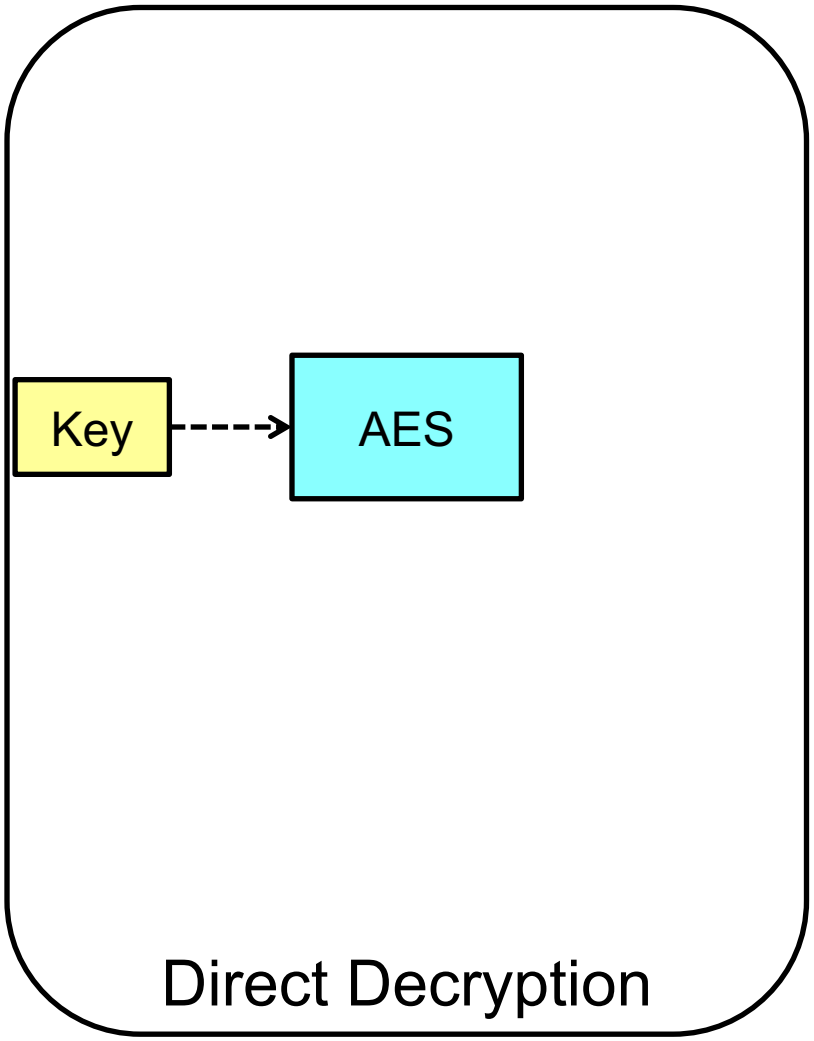
GOAL: WRITE-EFFICIENT ENCRYPTION

Goal: How do we implement memory encryption, without increasing bit flips by 4x?

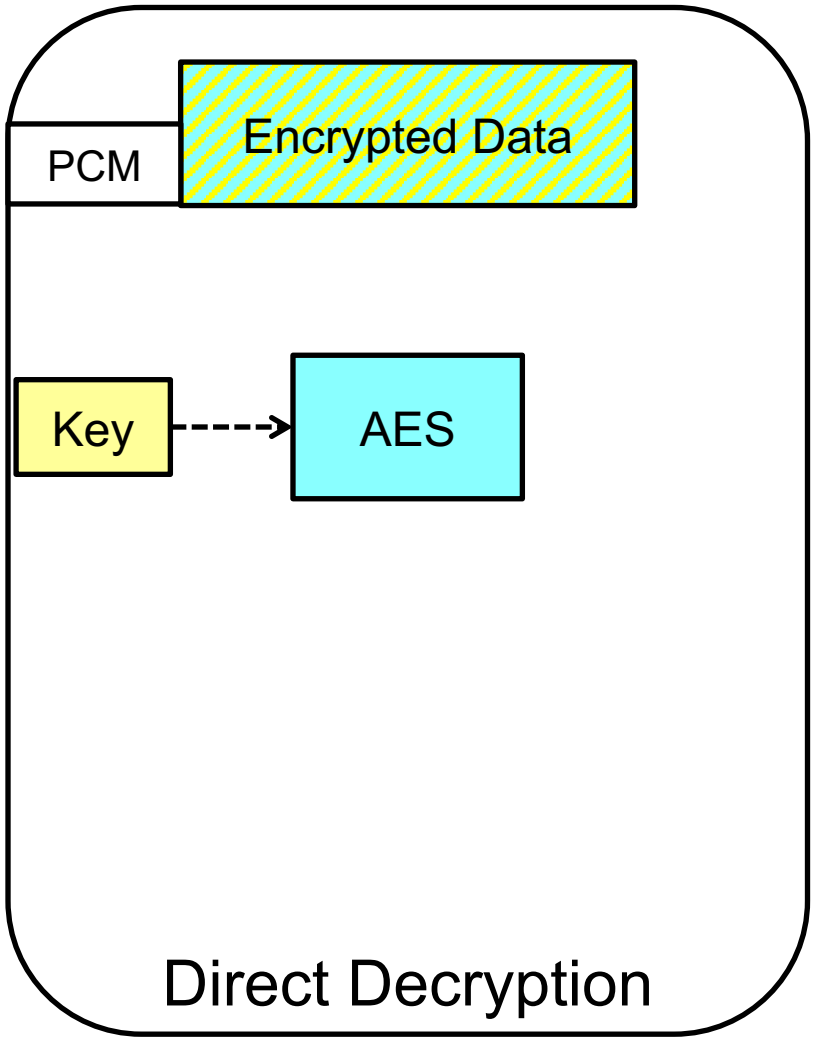
OUTLINE

- Introduction to PCM and encryption
- Background on Counter-mode Encryption 
- DEUCE
- Results
- Summary

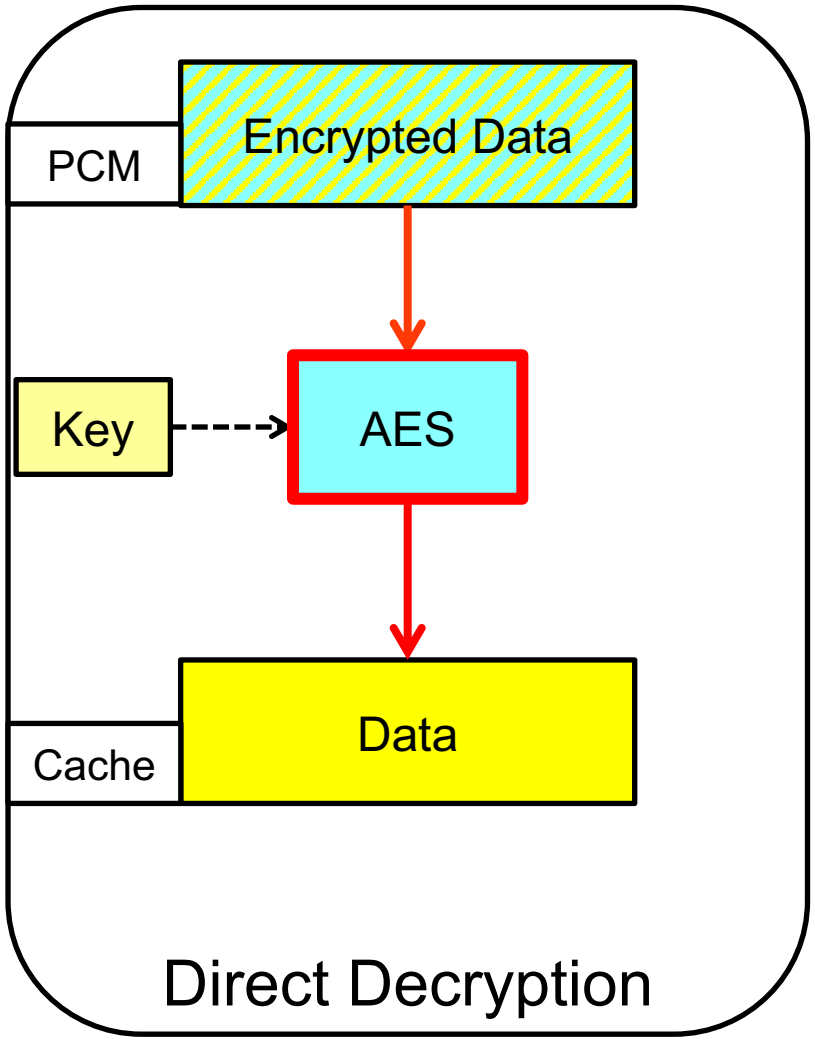
WHY PAD-BASED ENCRYPTION?



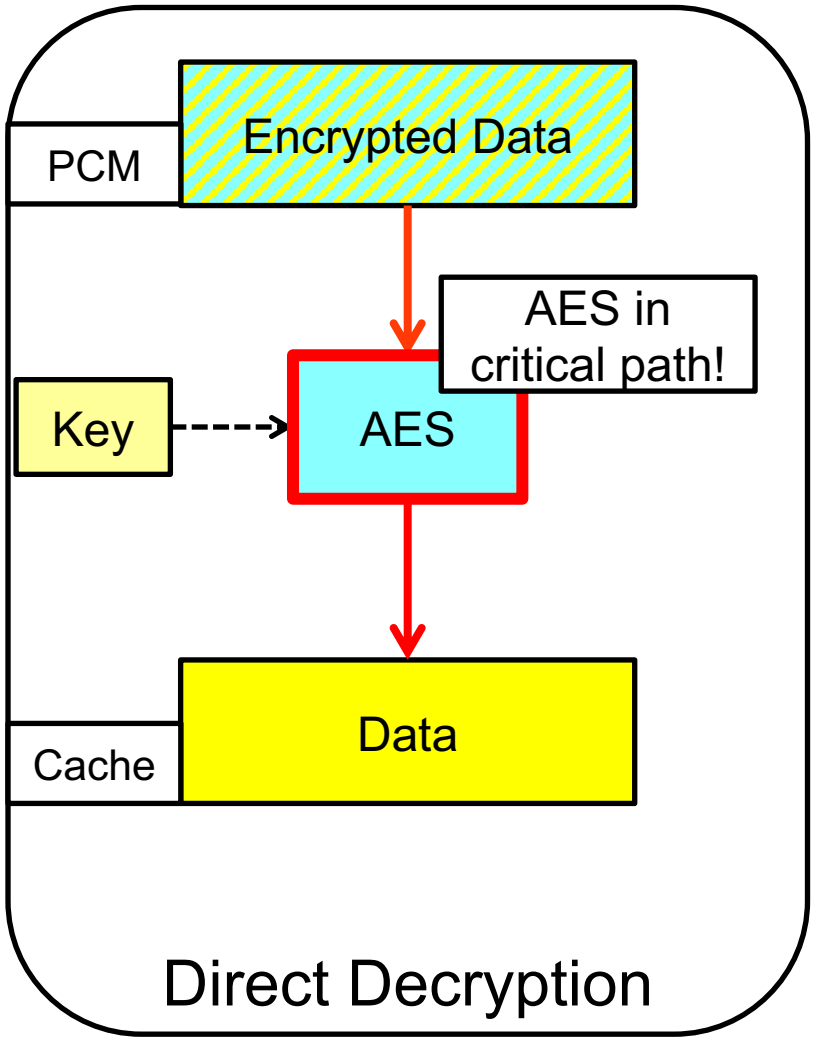
WHY PAD-BASED ENCRYPTION?



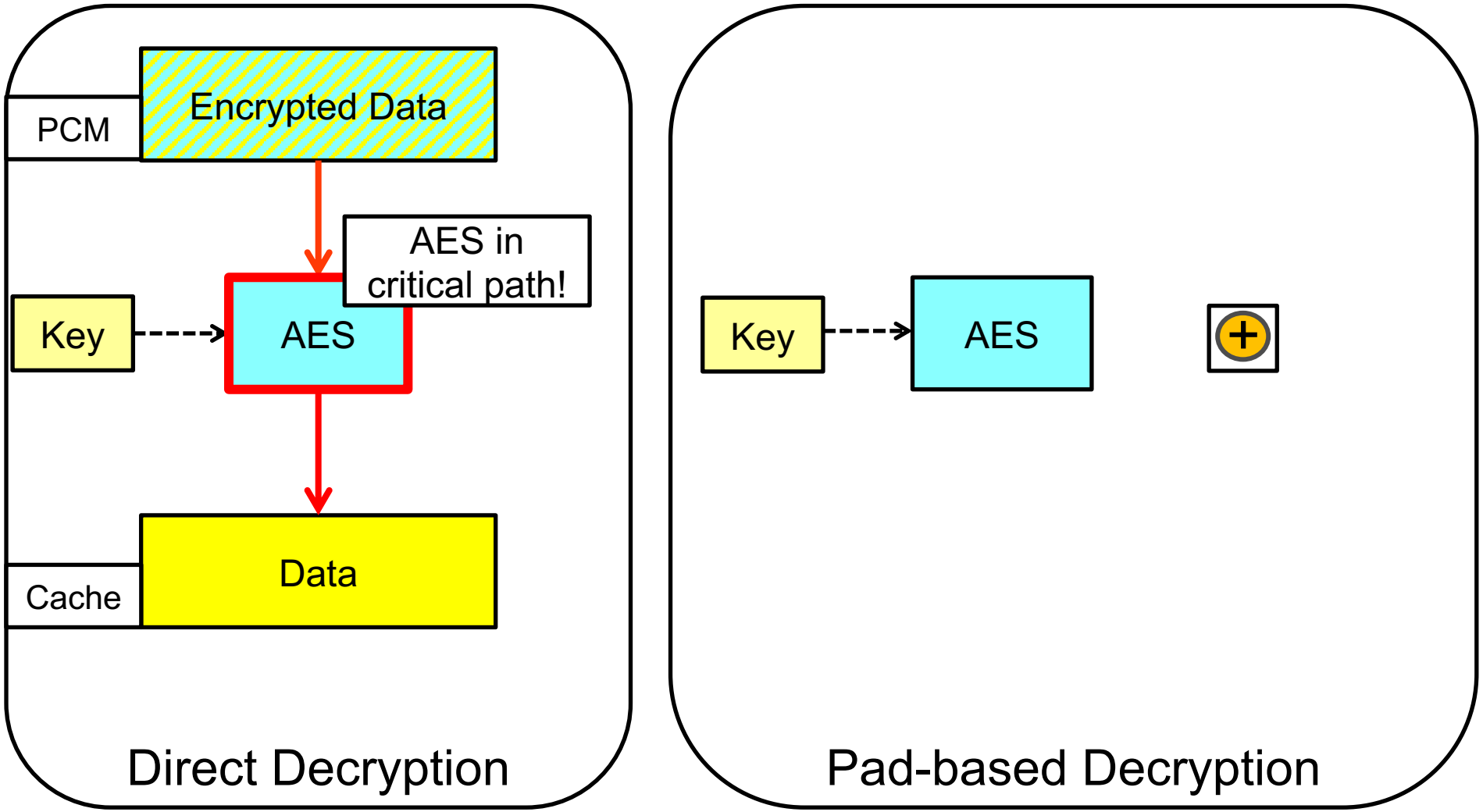
WHY PAD-BASED ENCRYPTION?



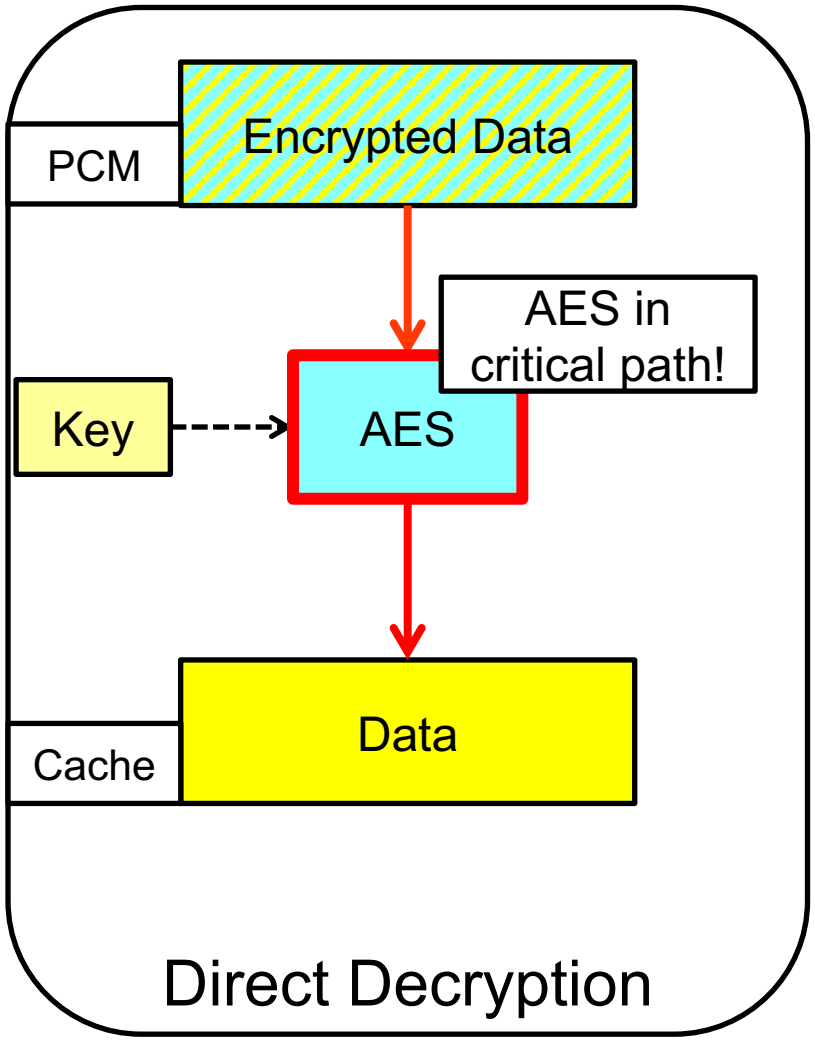
WHY PAD-BASED ENCRYPTION?

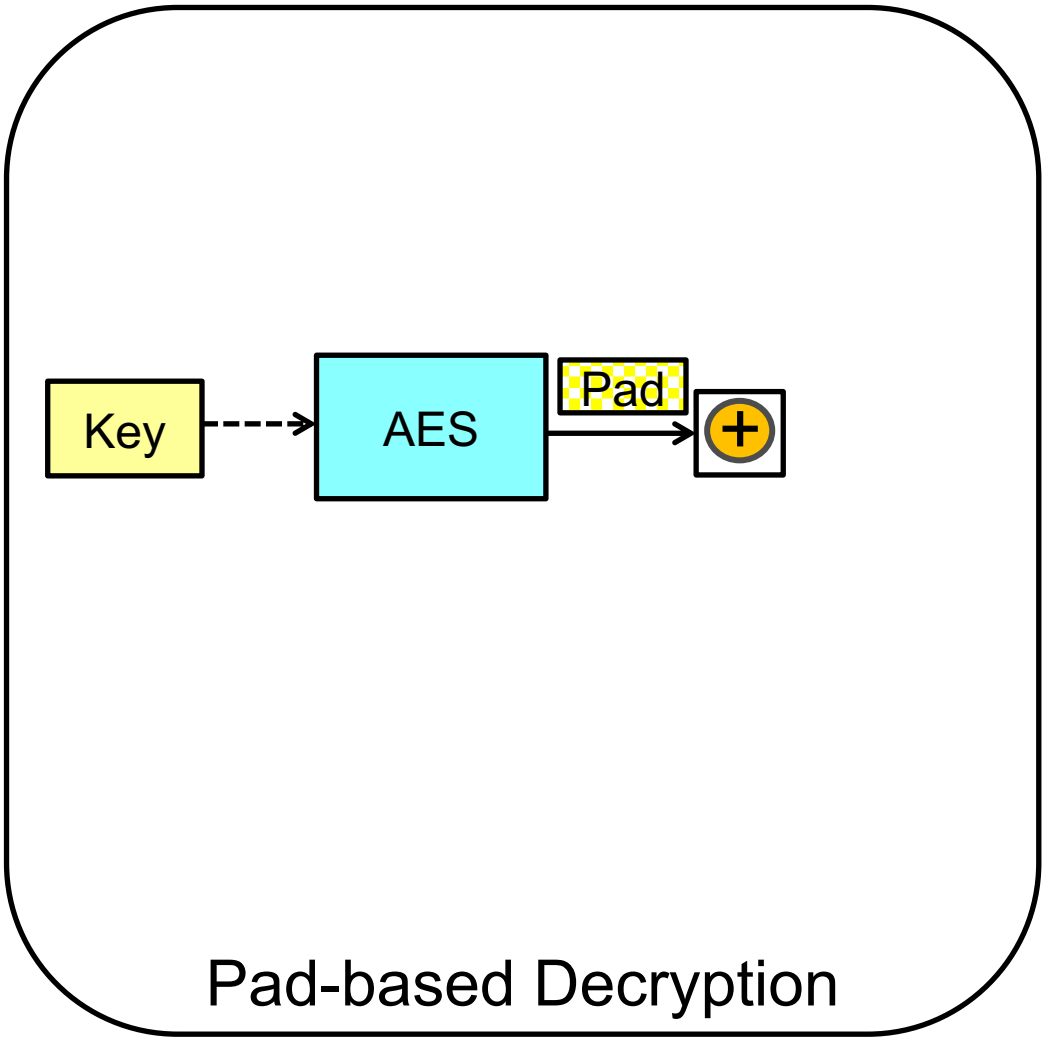


WHY PAD-BASED ENCRYPTION?

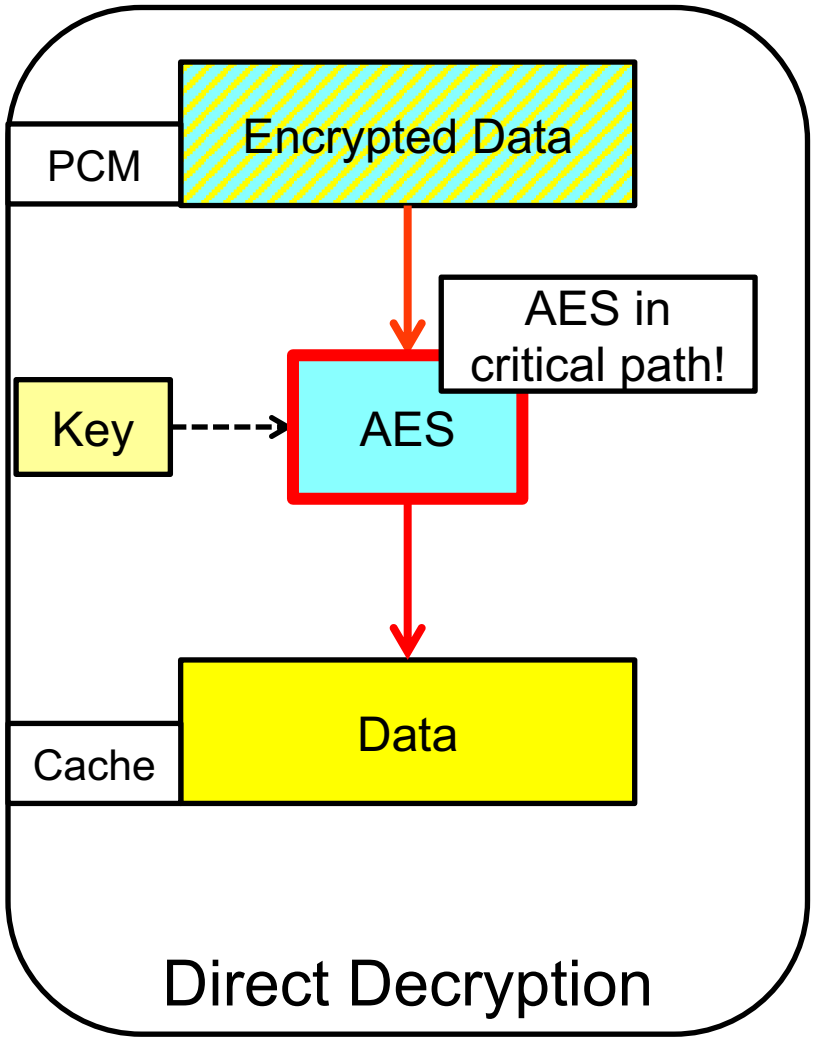


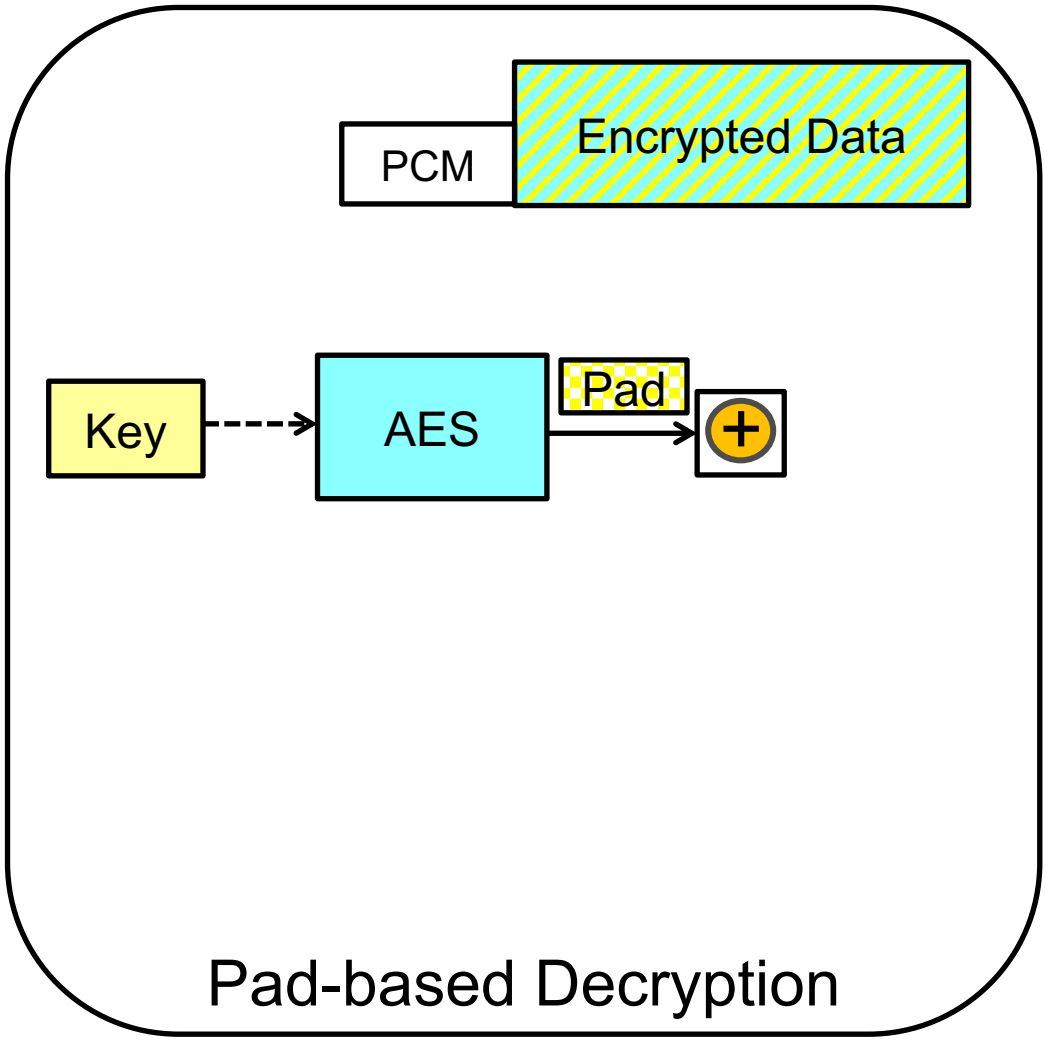
WHY PAD-BASED ENCRYPTION?



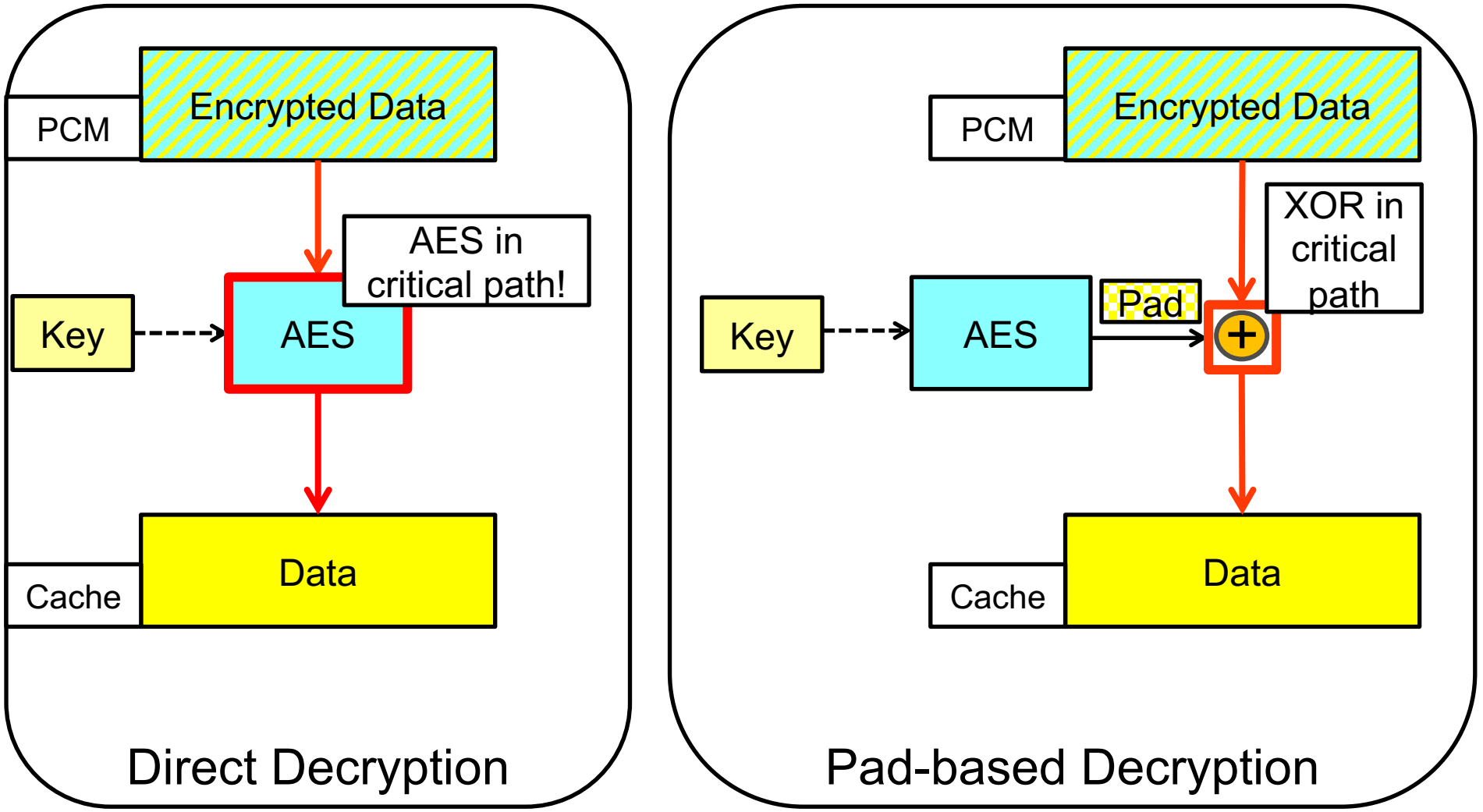


WHY PAD-BASED ENCRYPTION?





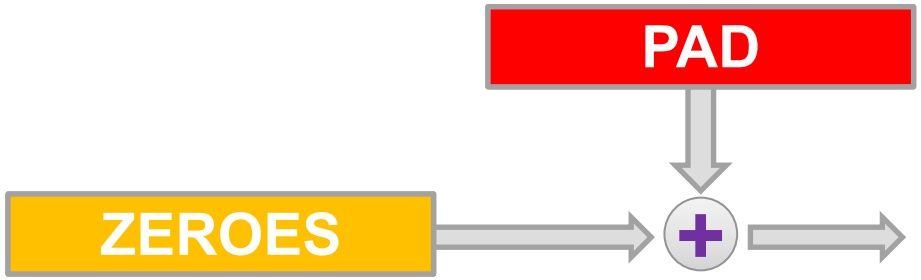
WHY PAD-BASED ENCRYPTION?



Pad-based decryption has low-latency

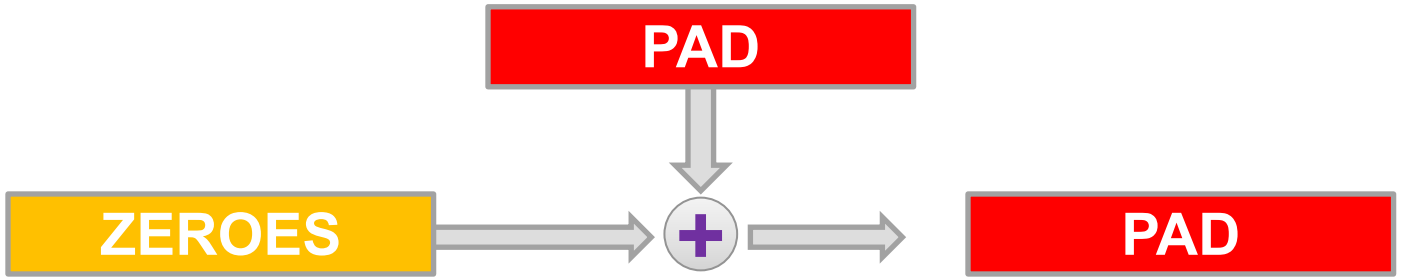
NEED FOR UNIQUE PADS

(0000) $\oplus$ Pad = Pad



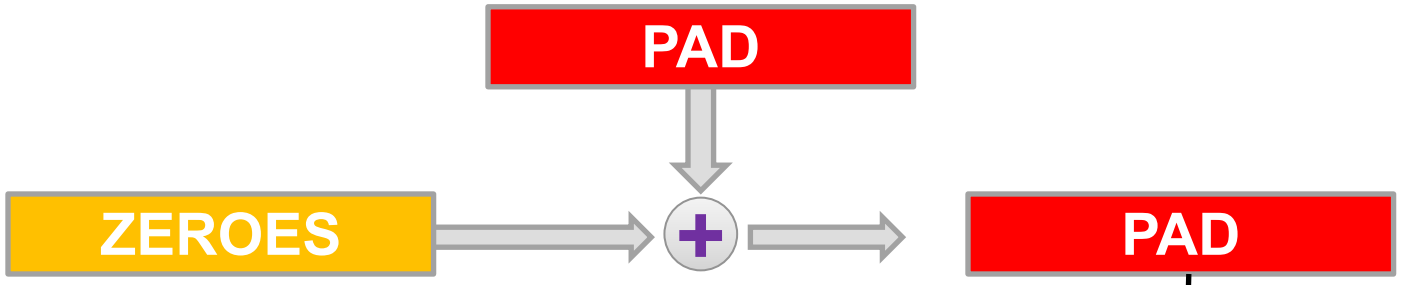
NEED FOR UNIQUE PADS

(0000) ⊕ Pad = Pad

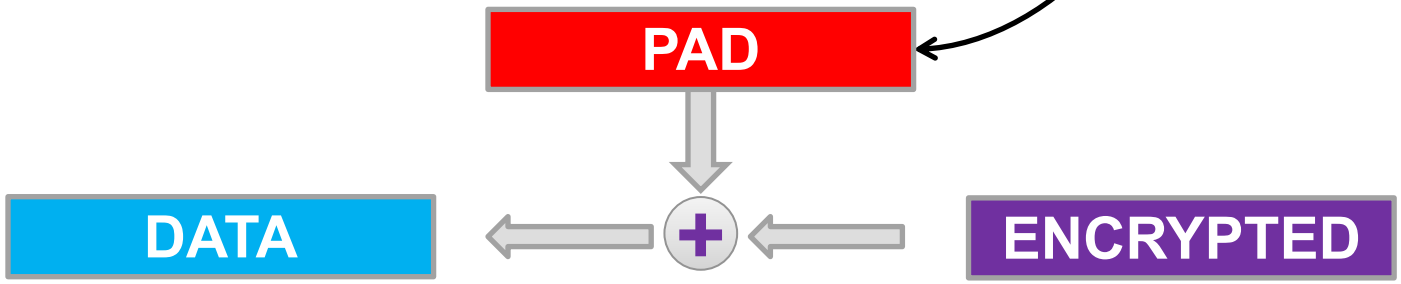


NEED FOR UNIQUE PADS

(0000) ⊕ Pad = Pad



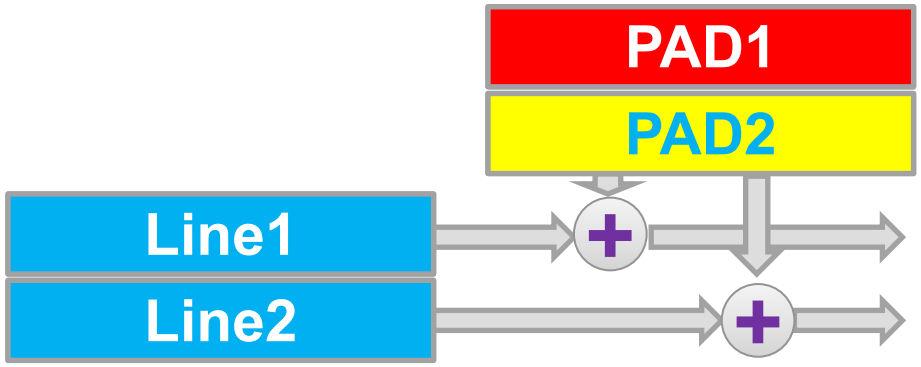
Attack! Insecure if pad learned



Secure pad encryption cannot re-use pads

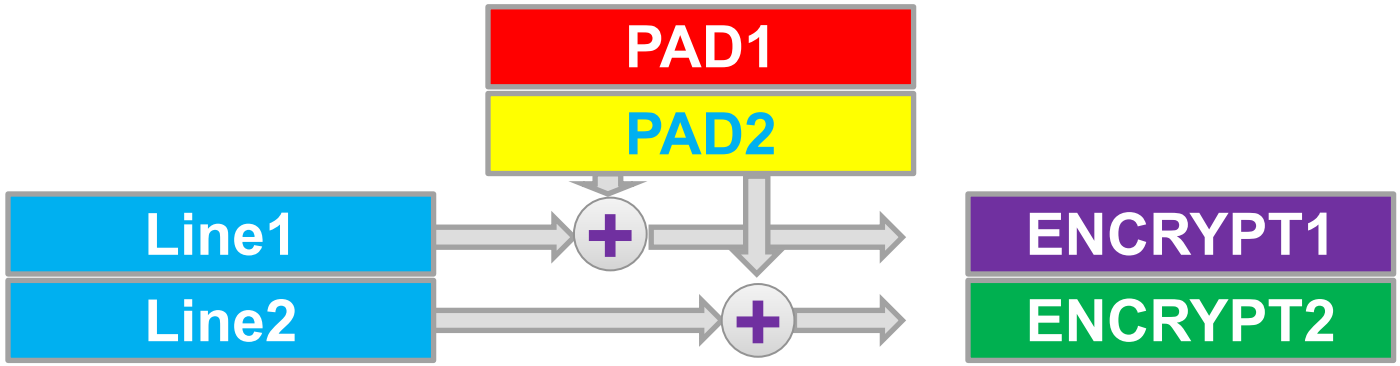
COUNTER-MODE ENCRYPTION

Different pad per line address



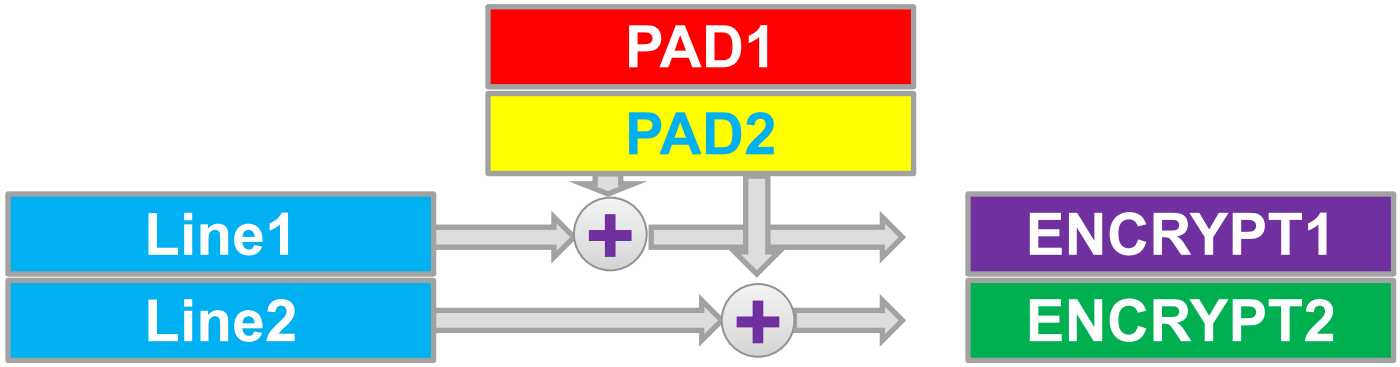
COUNTER-MODE ENCRYPTION

Different pad per line address

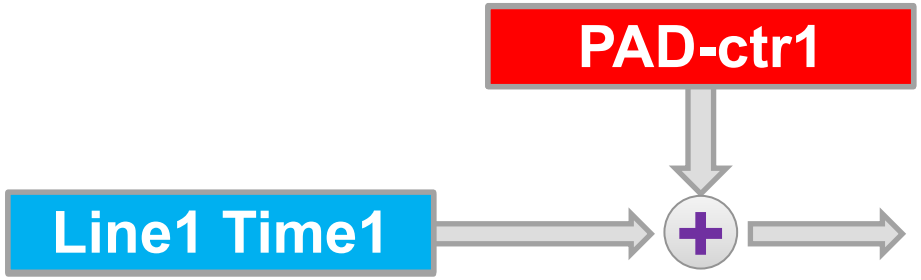


COUNTER-MODE ENCRYPTION

Different pad per line address

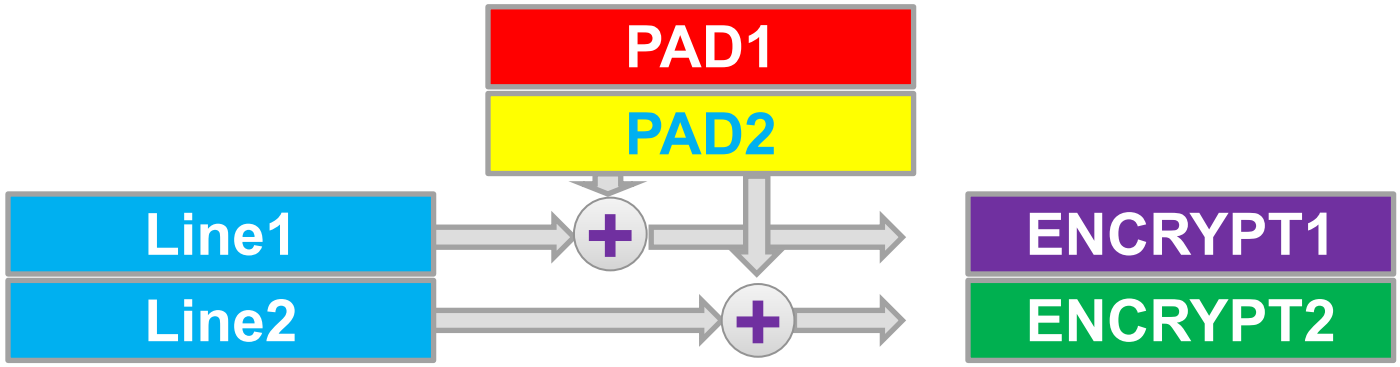


Different pad per write to same line➔per-line counter

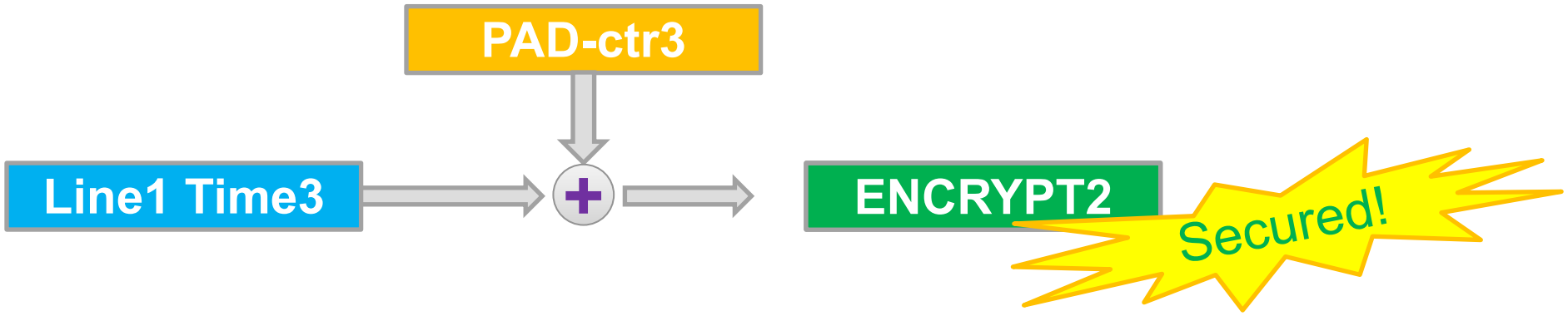


COUNTER-MODE ENCRYPTION

Different pad per line address

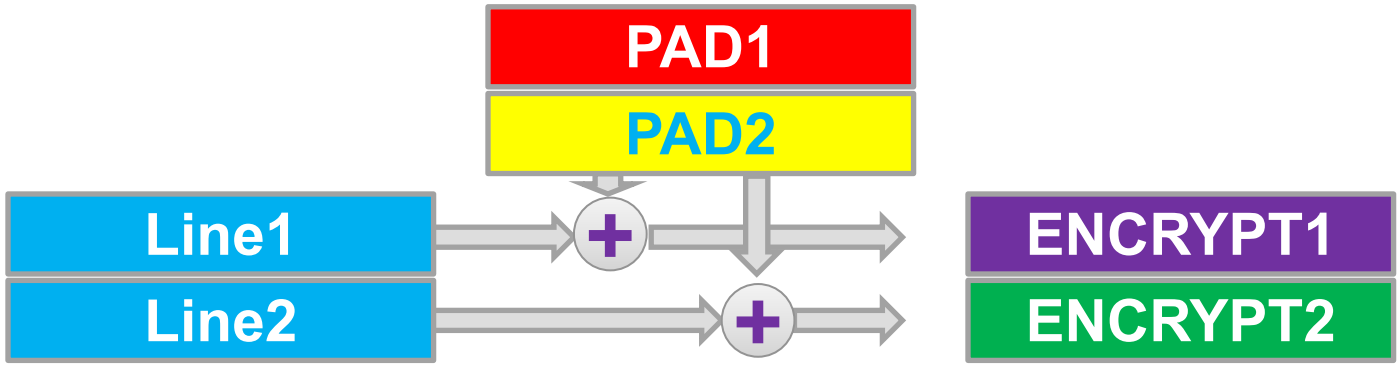


Different pad per write to same line➔per-line counter

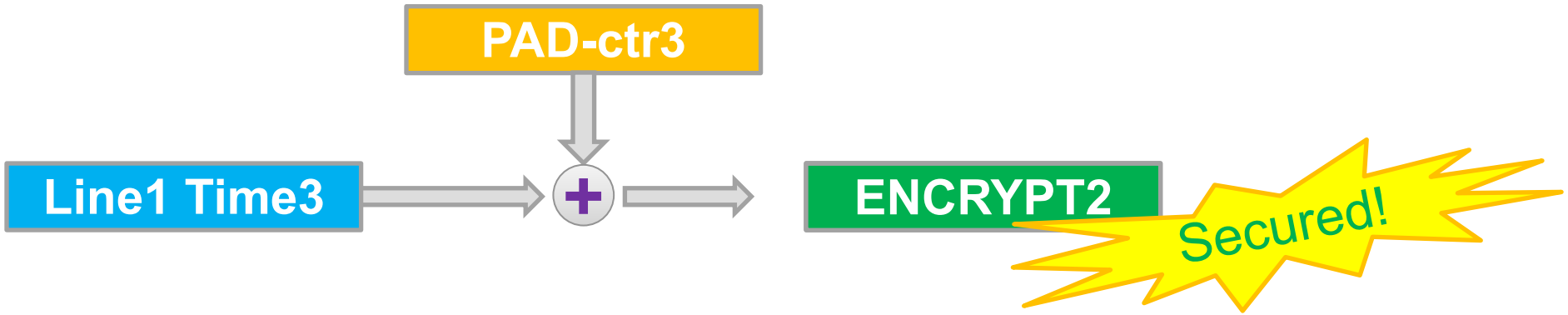


COUNTER-MODE ENCRYPTION

Different pad per line address



Different pad per write to same line➔per-line counter



Pad changes every write ➔ 50% bit flips every write

OUTLINE

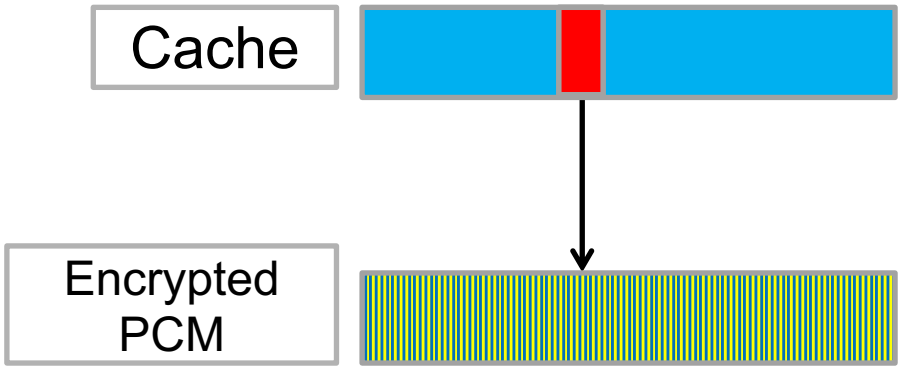
- Introduction to PCM and encryption
- Background on Counter-mode Encryption
- DEUCE 
- Results
- Summary

INSIGHT 1

What if we re-encrypt only modified words?

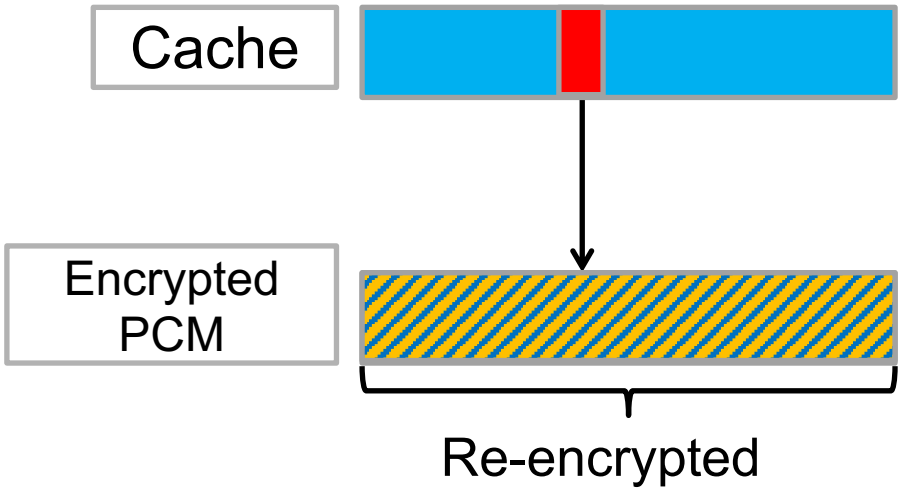
INSIGHT 1

What if we re-encrypt only modified words?



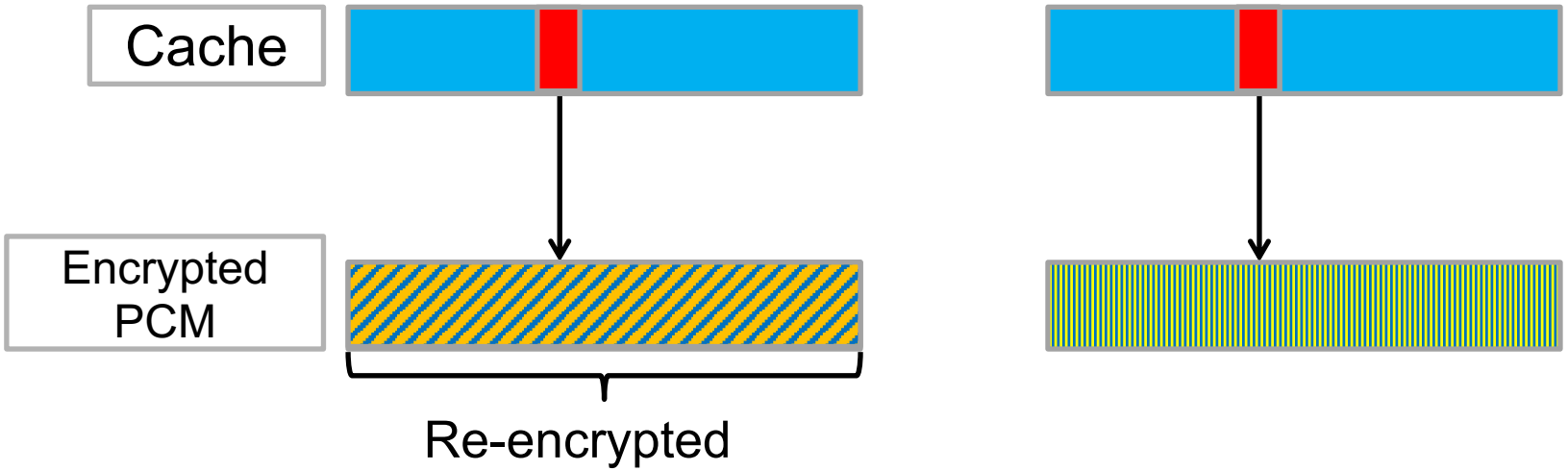
INSIGHT 1

What if we re-encrypt only modified words?



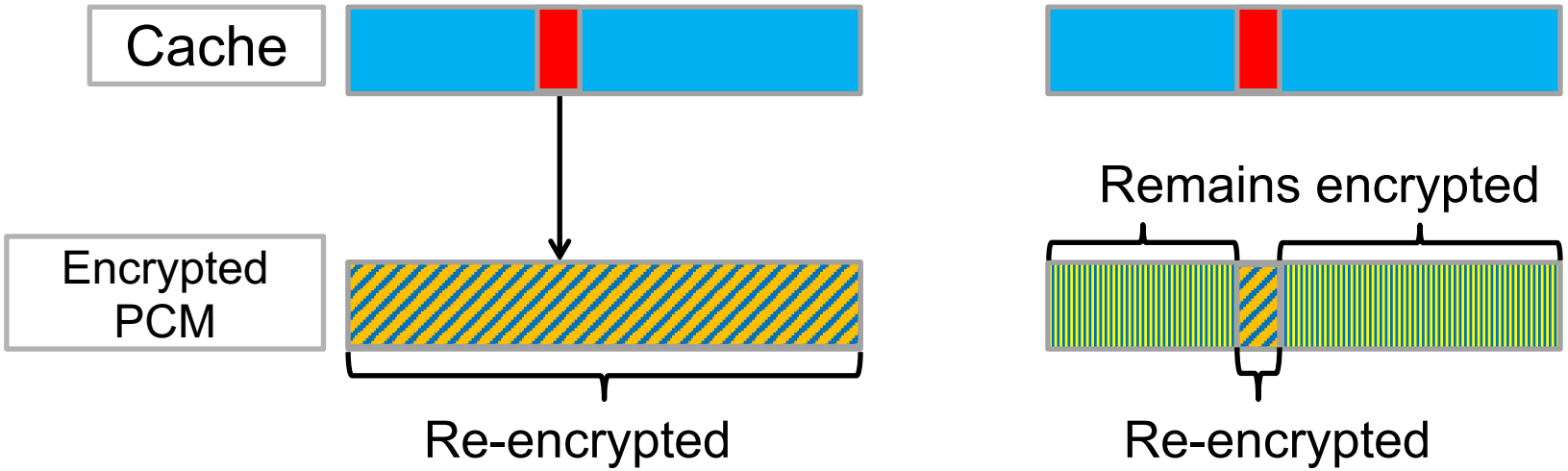
INSIGHT 1

What if we re-encrypt only modified words?



INSIGHT 1

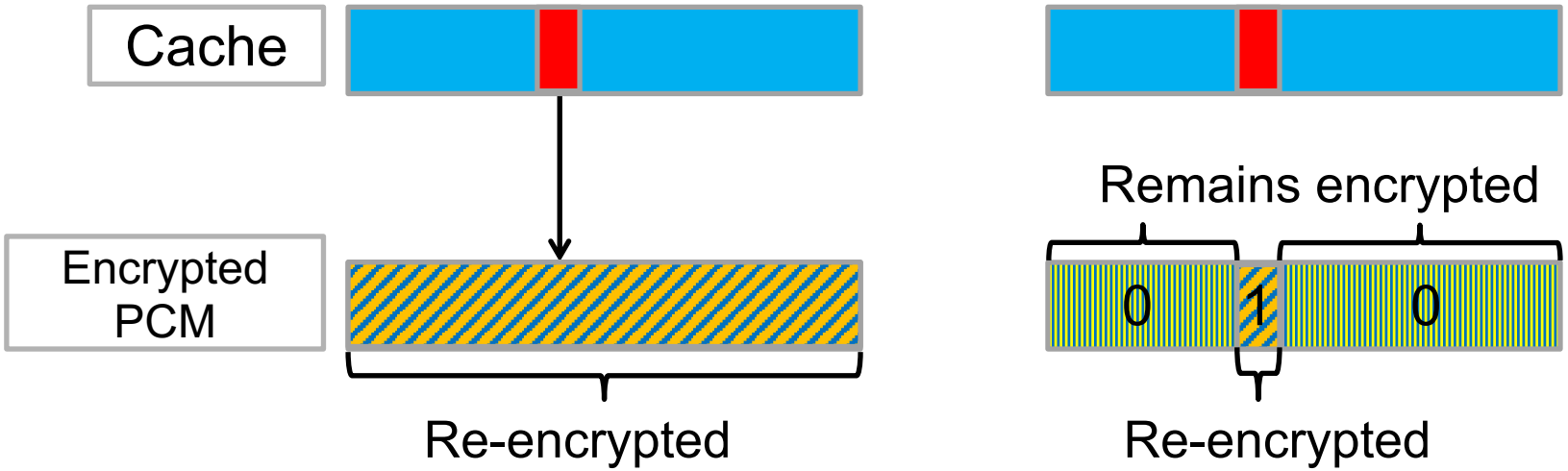
What if we re-encrypt only modified words?



Reduce bit flips by re-encrypting only modified words

INSIGHT 1

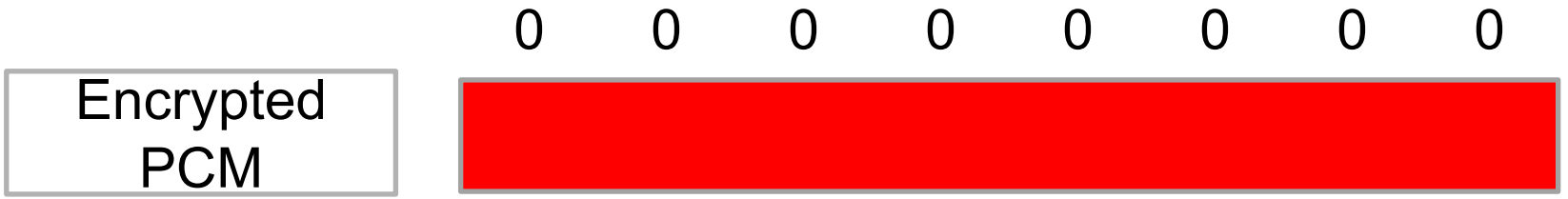
What if we re-encrypt only modified words?



Reduce bit flips by re-encrypting only modified words

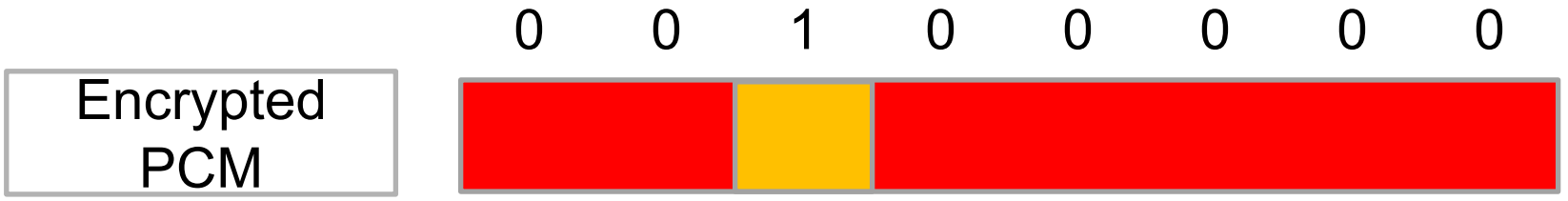
INSIGHT 2: EFFICIENT IMPLEMENTATION

Naïve implementation needs per-word counters/pads



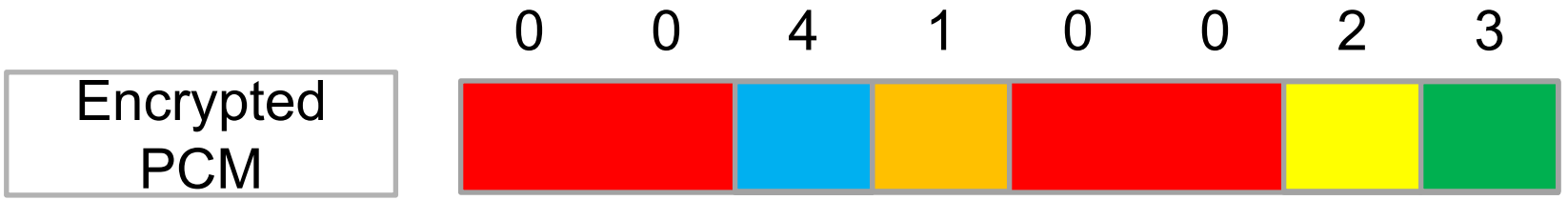
INSIGHT 2: EFFICIENT IMPLEMENTATION

Naïve implementation needs per-word counters/pads



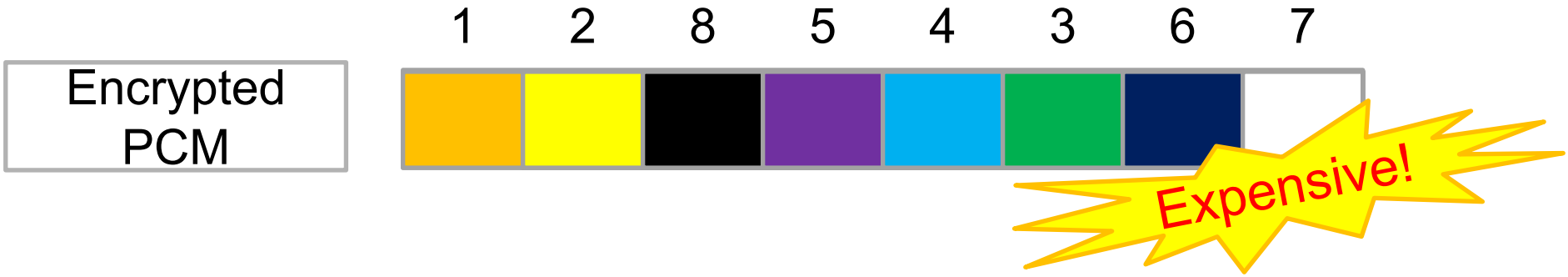
INSIGHT 2: EFFICIENT IMPLEMENTATION

Naïve implementation needs per-word counters/pads



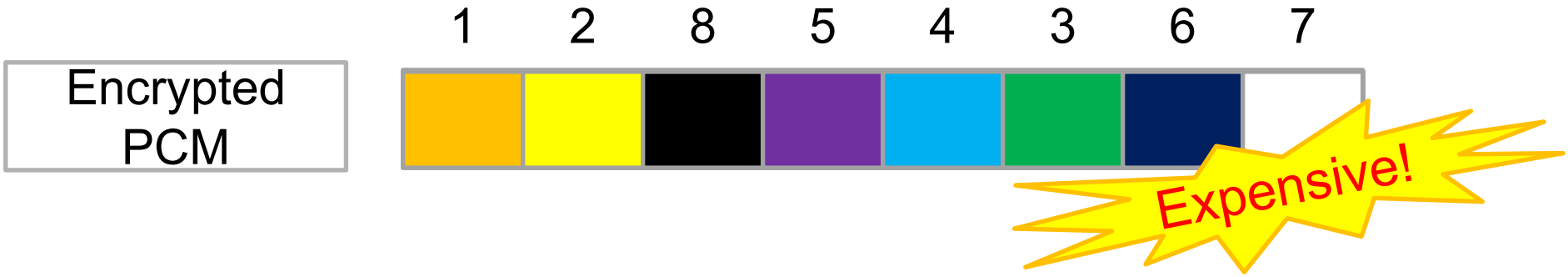
INSIGHT 2: EFFICIENT IMPLEMENTATION

Naïve implementation needs per-word counters/pads



INSIGHT 2: EFFICIENT IMPLEMENTATION

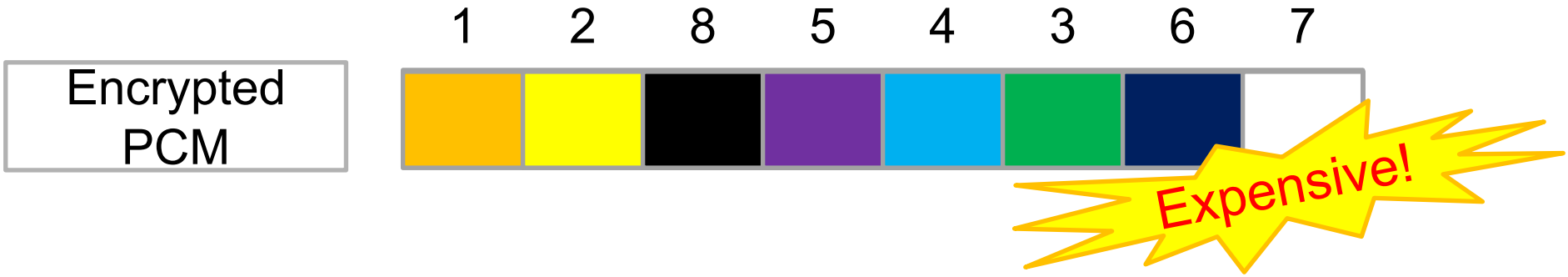
Naïve implementation needs per-word counters/pads



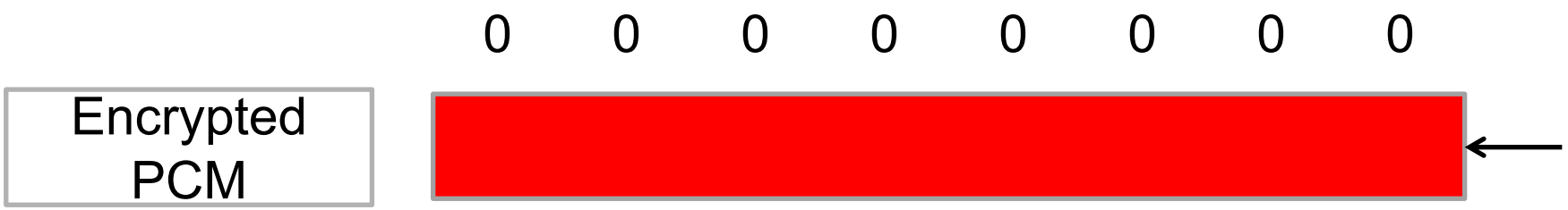
Reduce storage overhead by using only two counters

INSIGHT 2: EFFICIENT IMPLEMENTATION

Naïve implementation needs per-word counters/pads

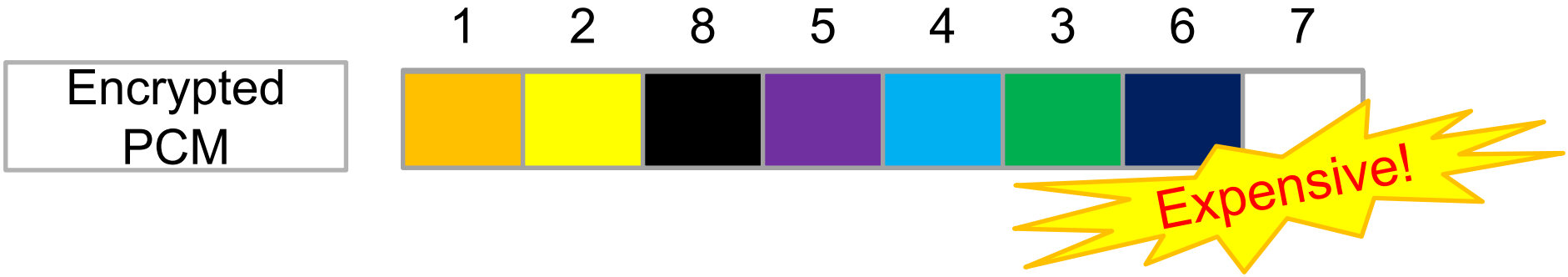


Reduce storage overhead by using only two counters

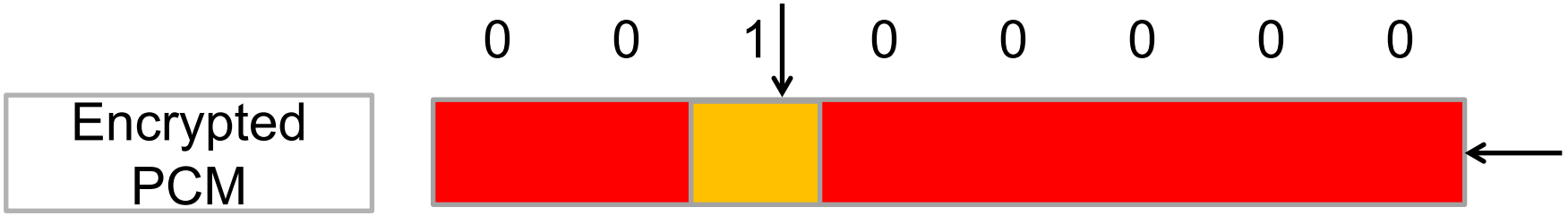


INSIGHT 2: EFFICIENT IMPLEMENTATION

Naïve implementation needs per-word counters/pads

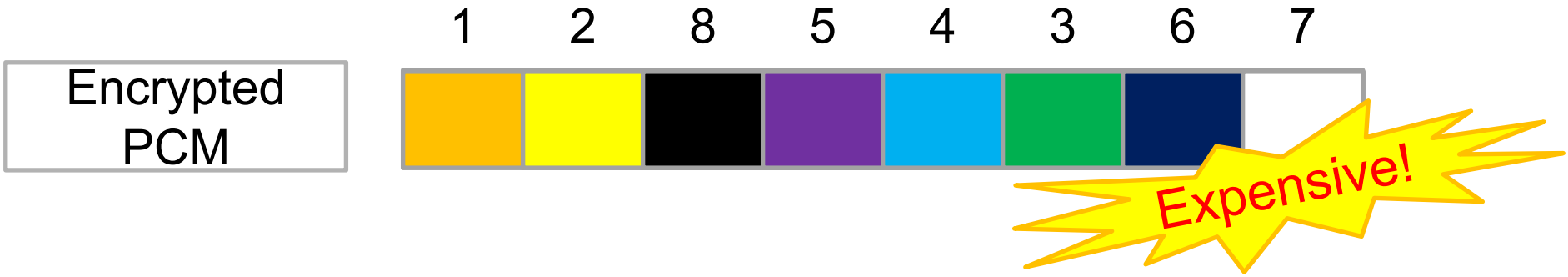


Reduce storage overhead by using only two counters

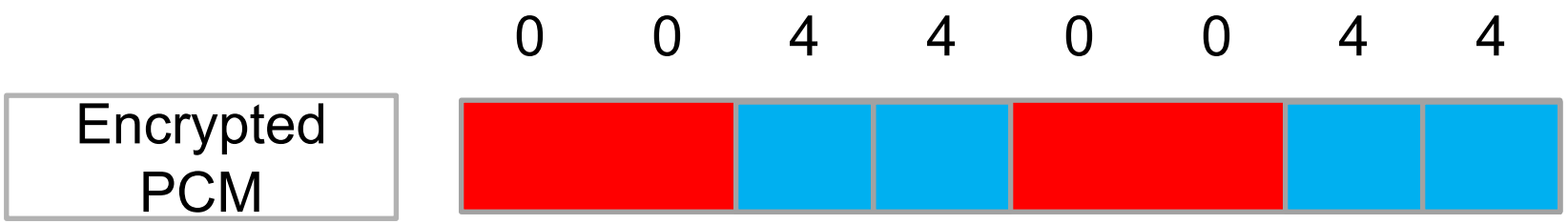


INSIGHT 2: EFFICIENT IMPLEMENTATION

Naïve implementation needs per-word counters/pads

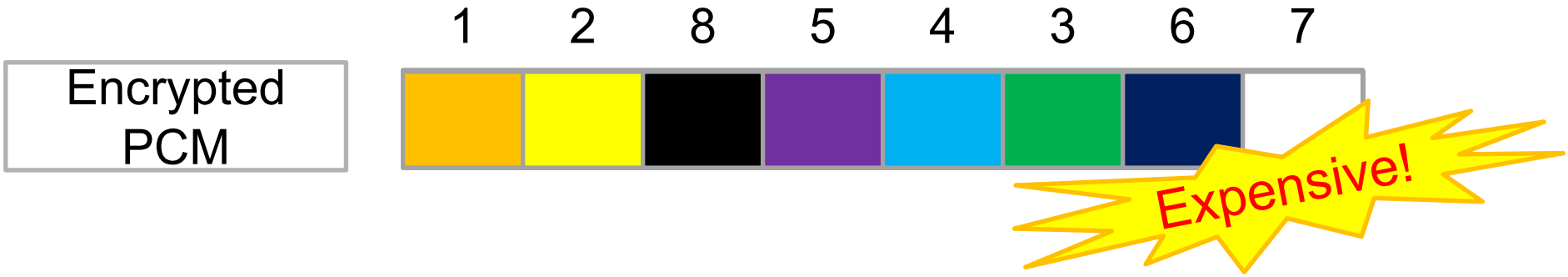


Reduce storage overhead by using only two counters

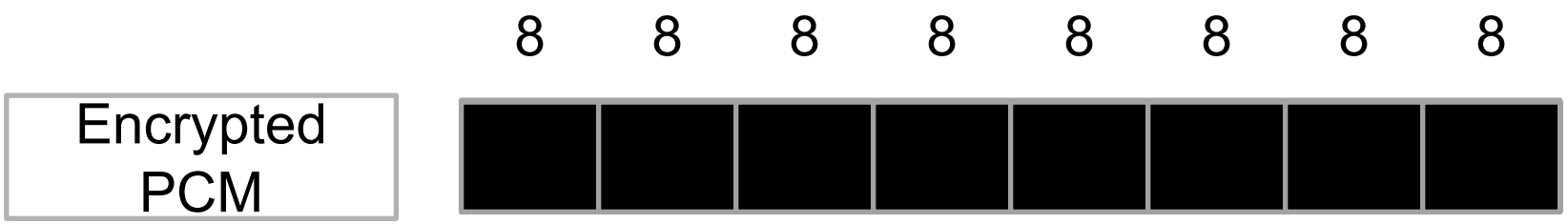


INSIGHT 2: EFFICIENT IMPLEMENTATION

Naïve implementation needs per-word counters/pads



Reduce storage overhead by using only two counters



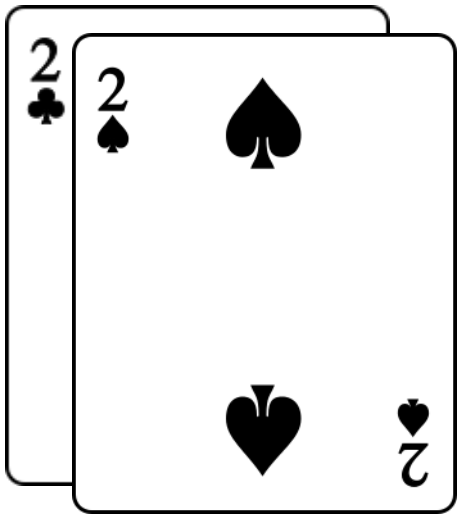
Do efficient partial re-encryption with only two counters

DEUCE: DUAL COUNTER ENCRYPTION

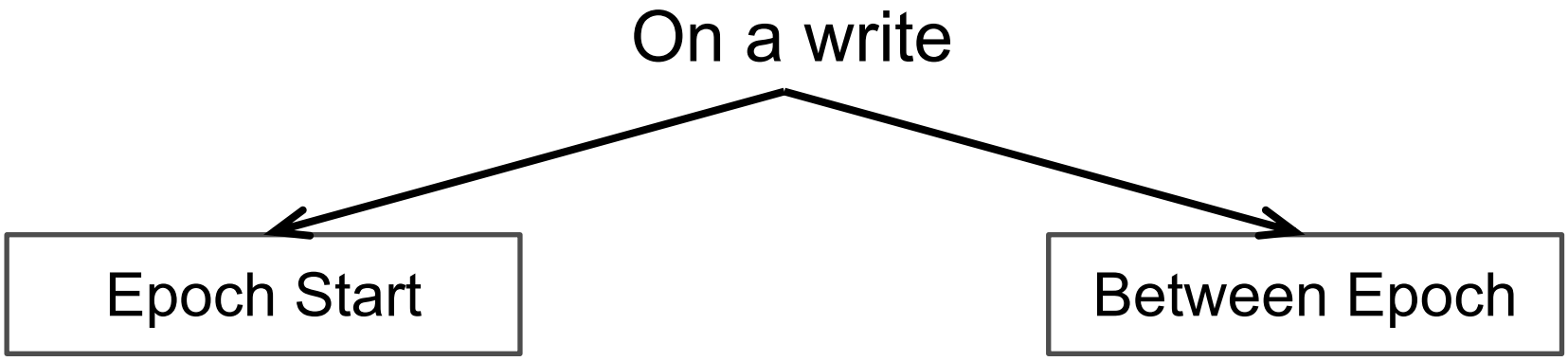
Each line has two counters:

- Leading counter (*LeadCTR*): incremented every write
- Trailing counter (*TrailCTR*): updated every N writes (*Epoch*)

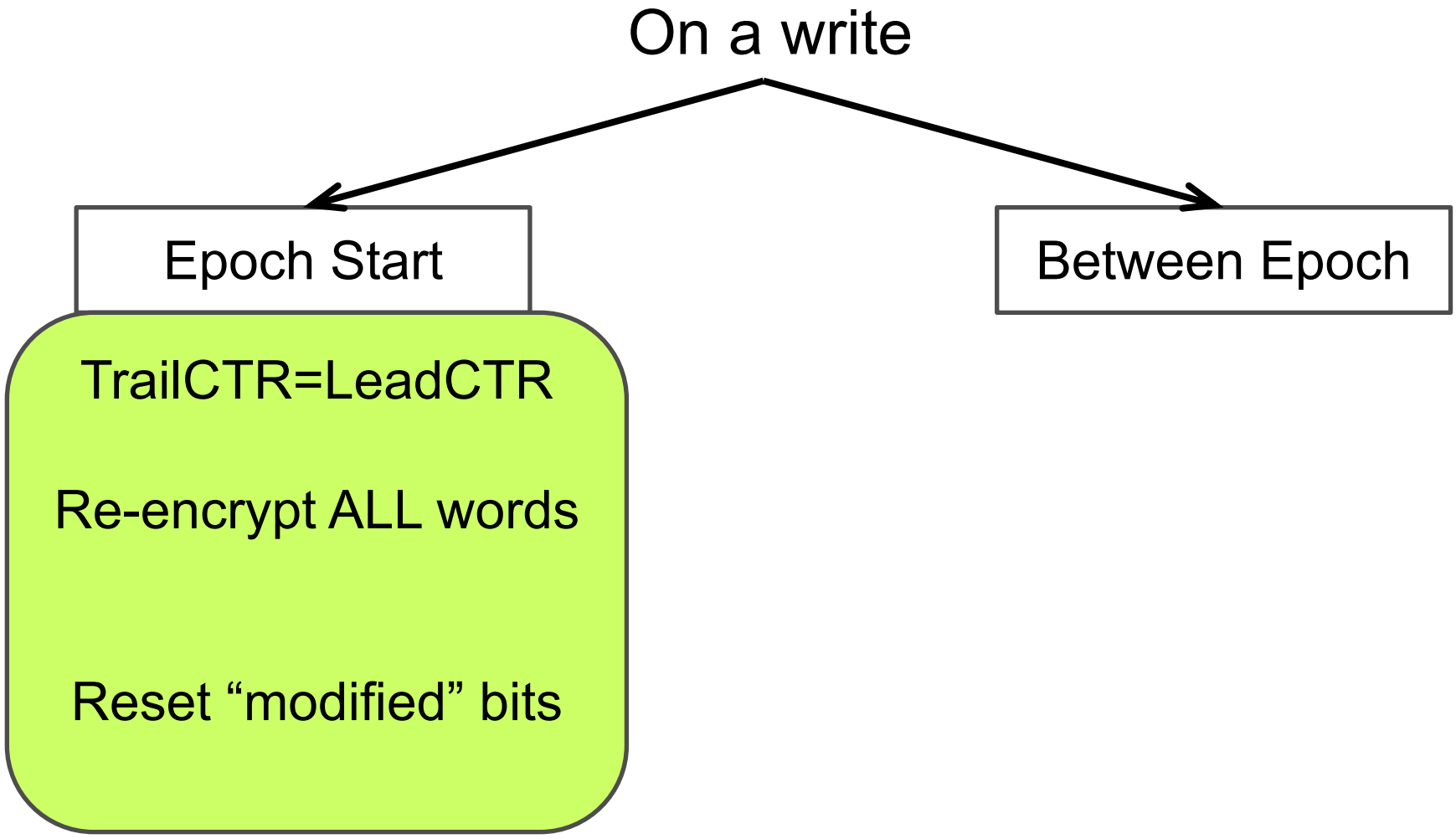
"Modified" bit per word to choose between counters



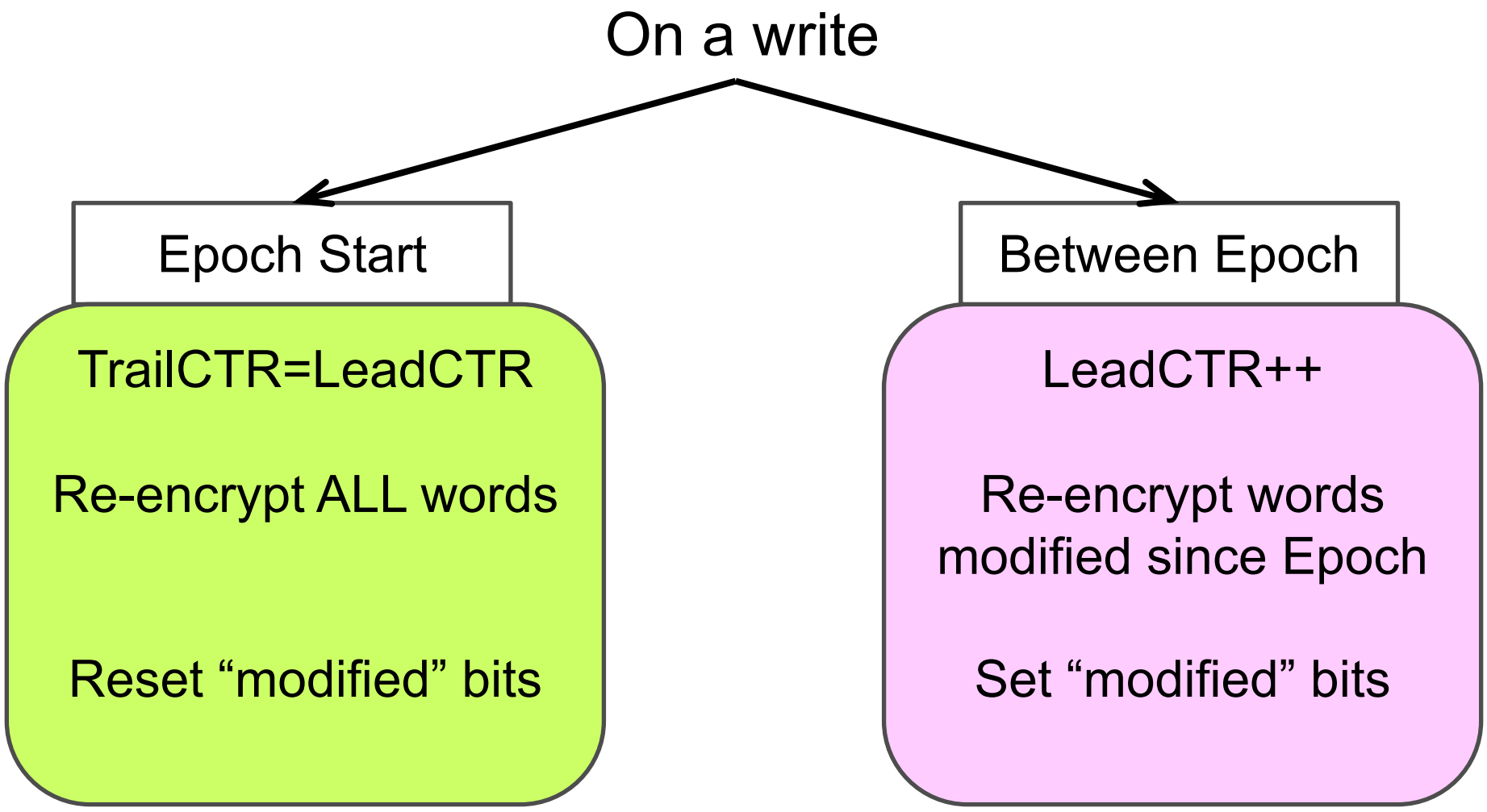
DEUCE: OPERATION



DEUCE: OPERATION

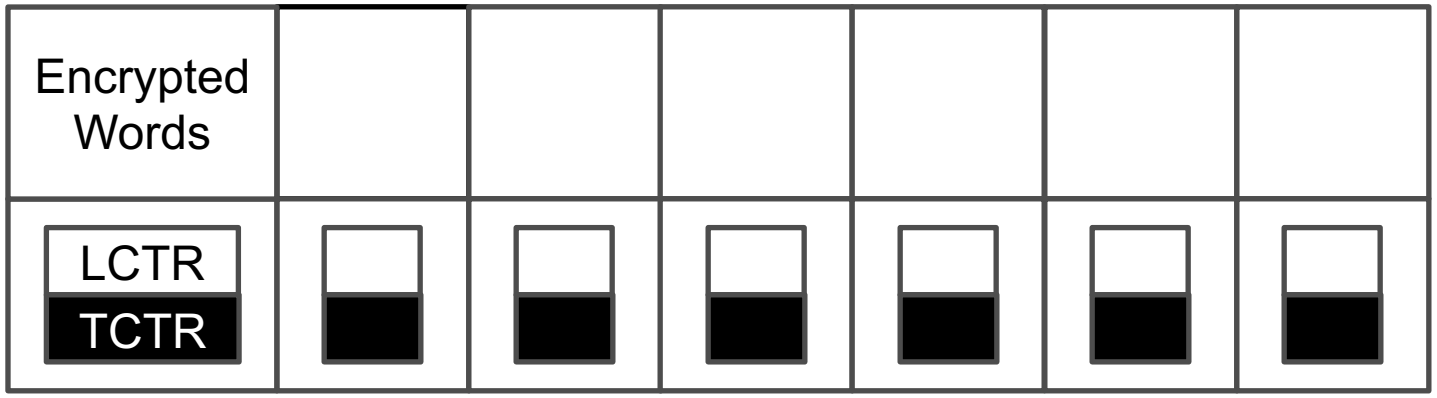


DEUCE: OPERATION

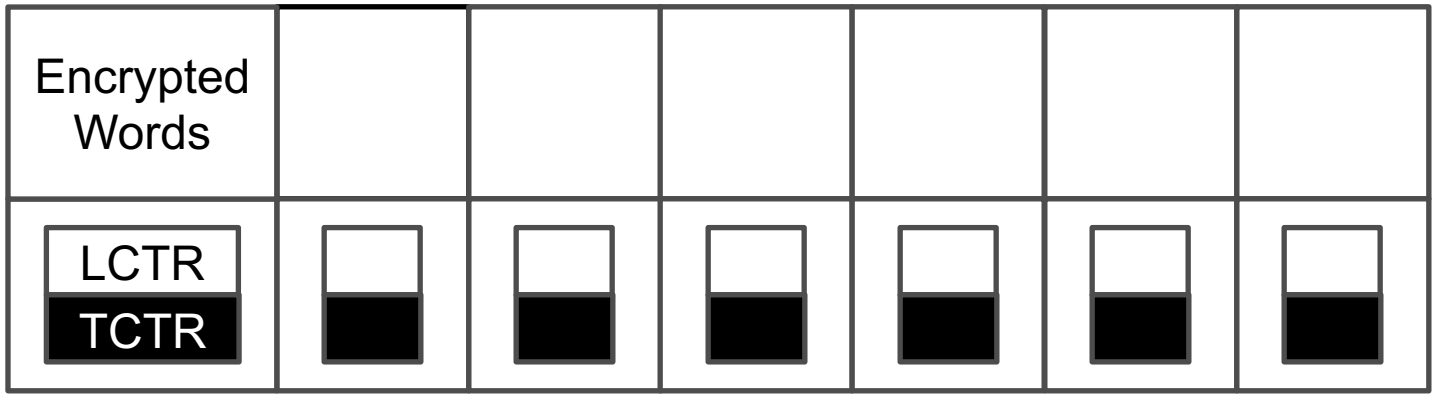


DEUCE re-encrypts only words modified since Epoch

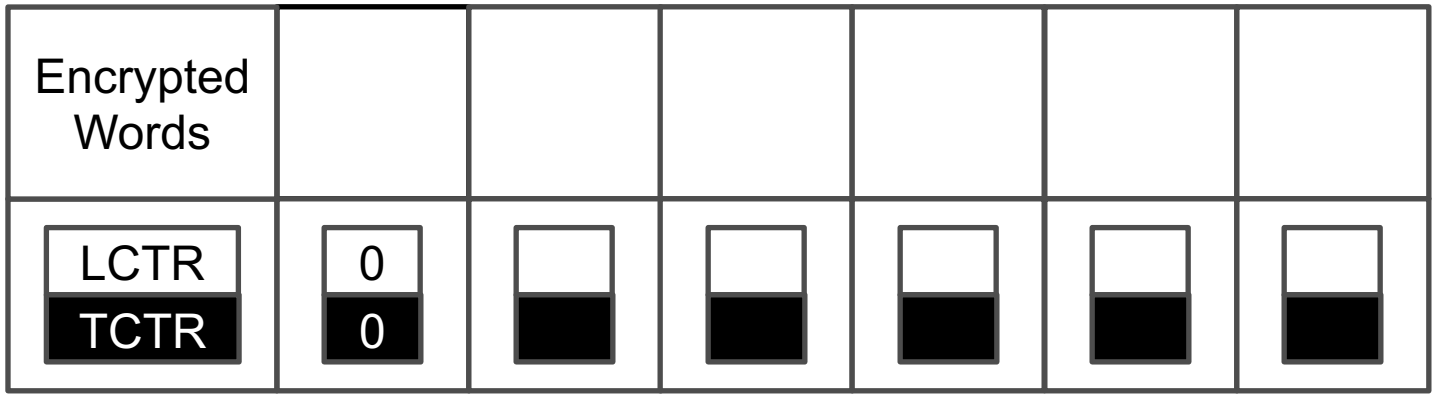
DEUCE: EXAMPLE (EPOCH = 4 WRITES)



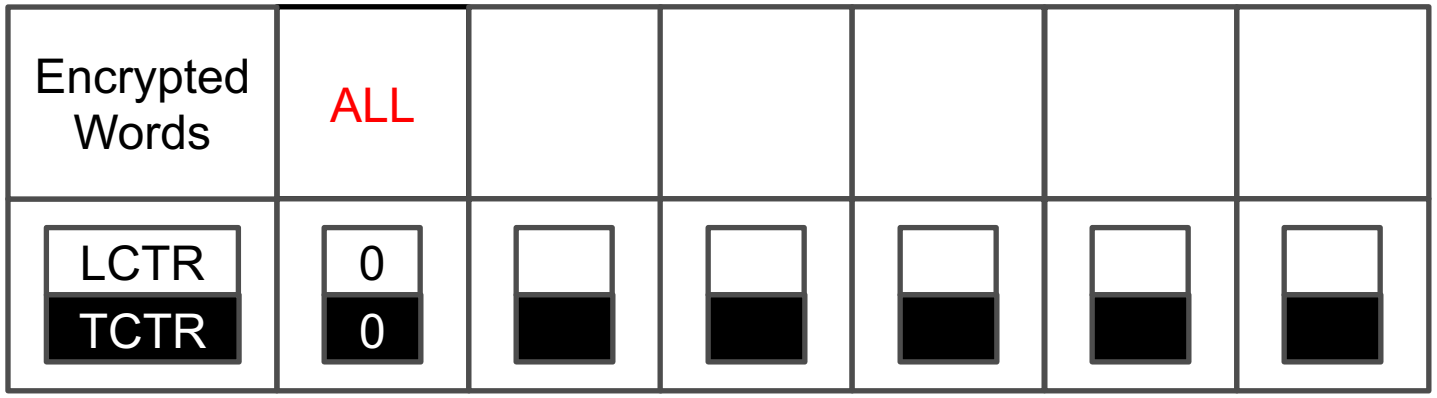
DEUCE: EXAMPLE (EPOCH = 4 WRITES)



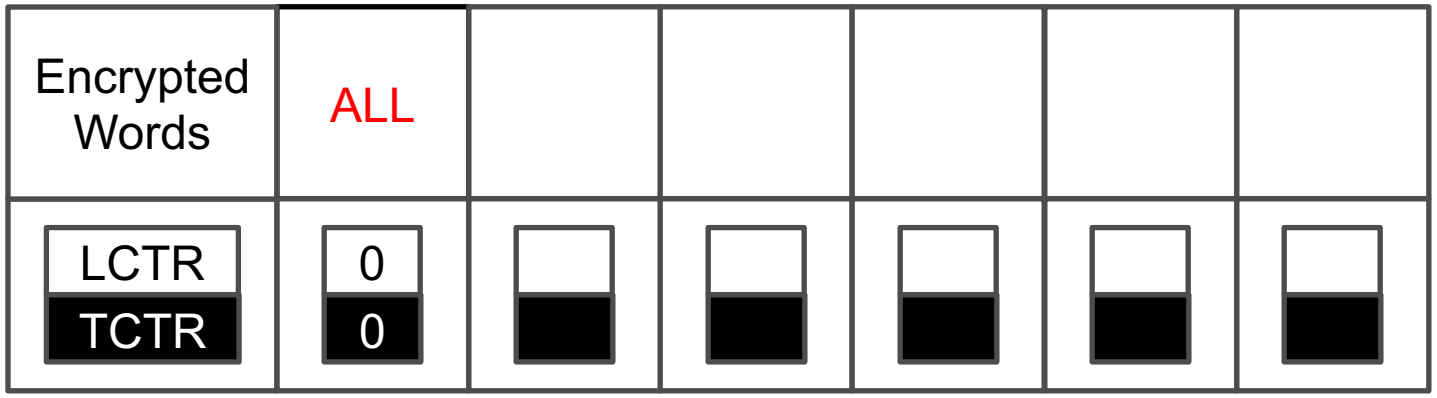
DEUCE: EXAMPLE (EPOCH = 4 WRITES)



DEUCE: EXAMPLE (EPOCH = 4 WRITES)

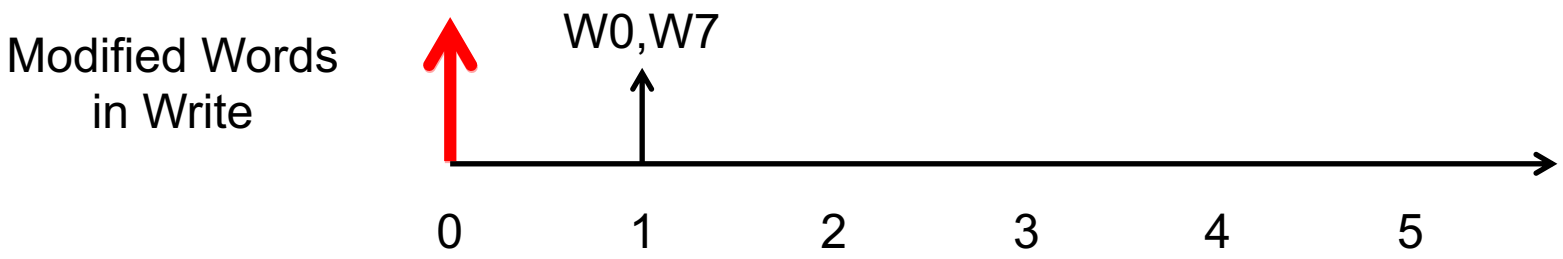


DEUCE: EXAMPLE (EPOCH = 4 WRITES)



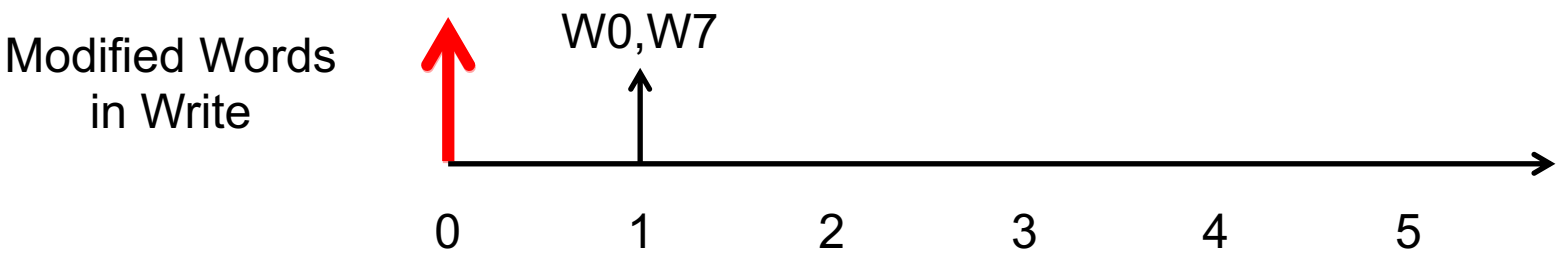
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL					
LCTR / TCTR	0 / 0	1 / 0				



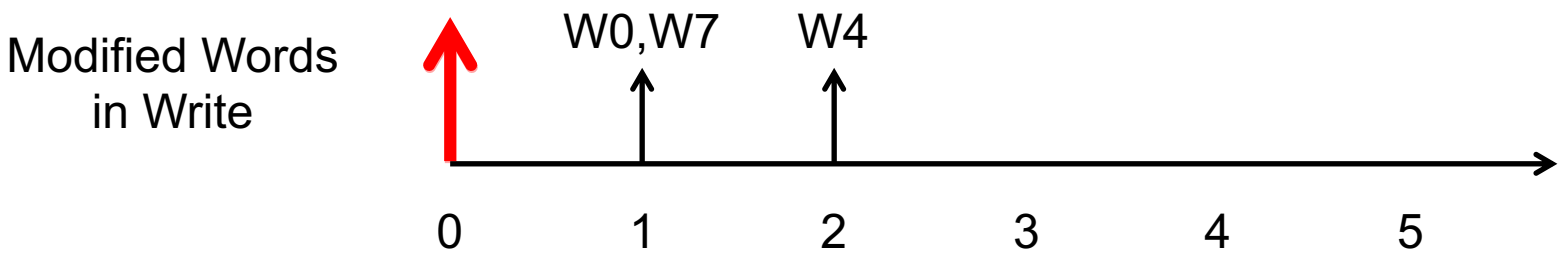
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7				
LCTR / TCTR	0 / 0	1 / 0				



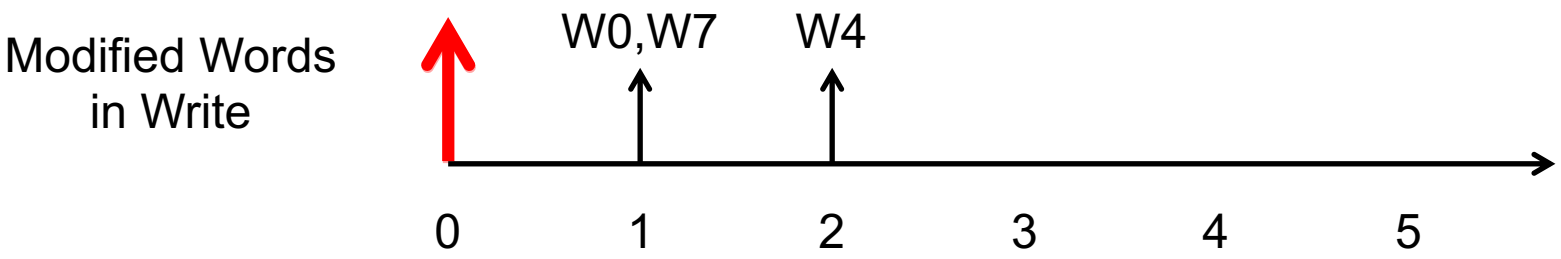
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7				
LCTR / TCTR	0 / 0	1 / 0				



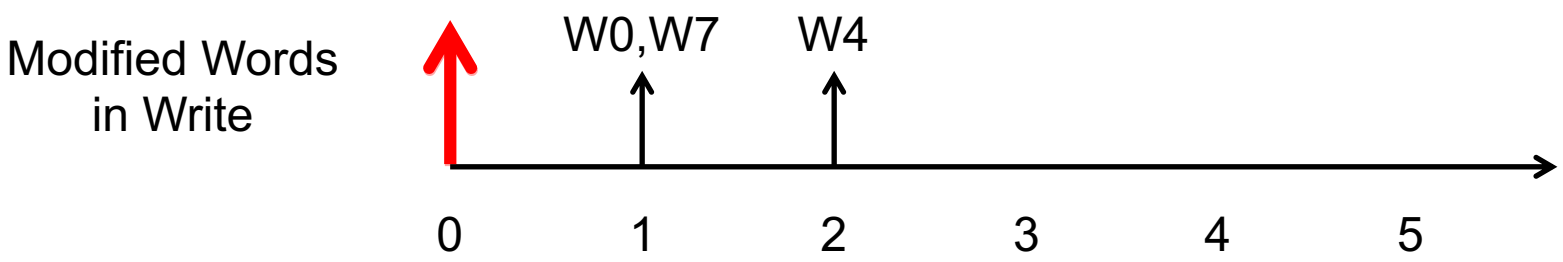
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7				
LCTR / TCTR	0 / 0	1 / 0	2 / 0			



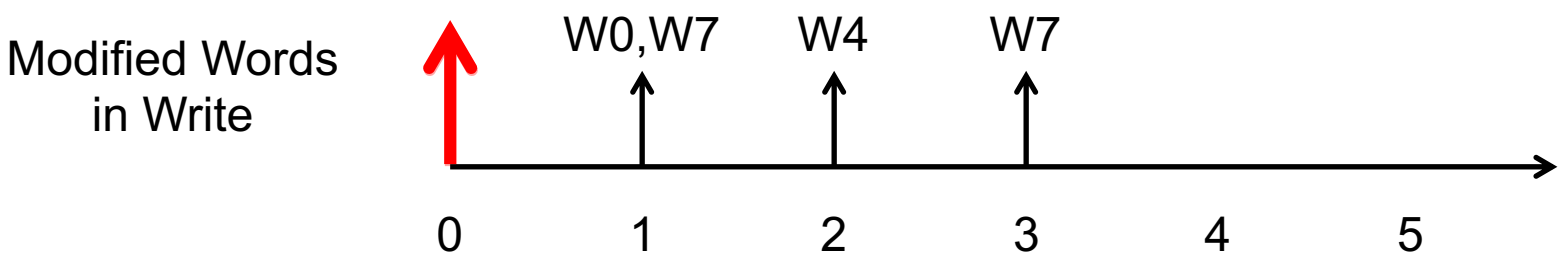
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7			
LCTR / TCTR	0 / 0	1 / 0	2 / 0			



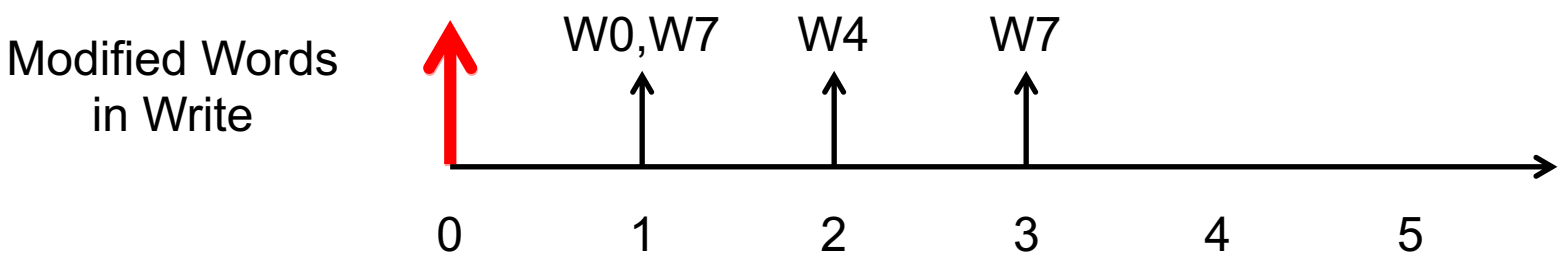
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7			
LCTR / TCTR	0 / 0	1 / 0	2 / 0			



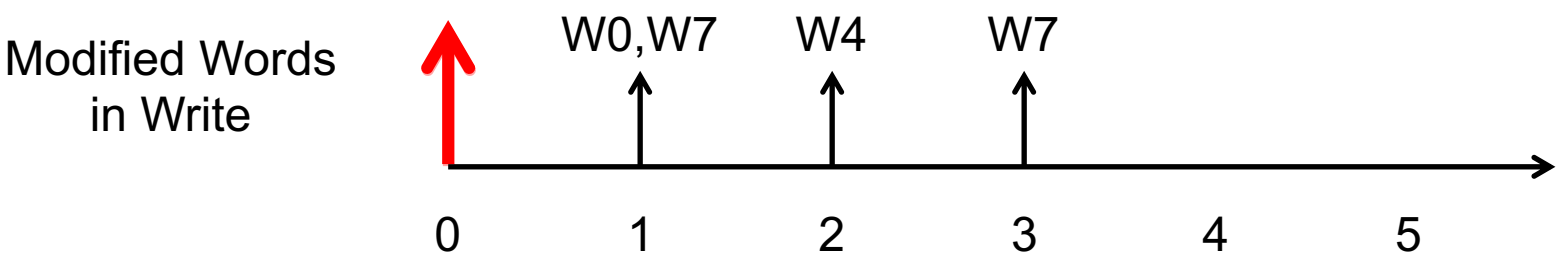
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7			
LCTR / TCTR	0 / 0	1 / 0	2 / 0	3 / 0		



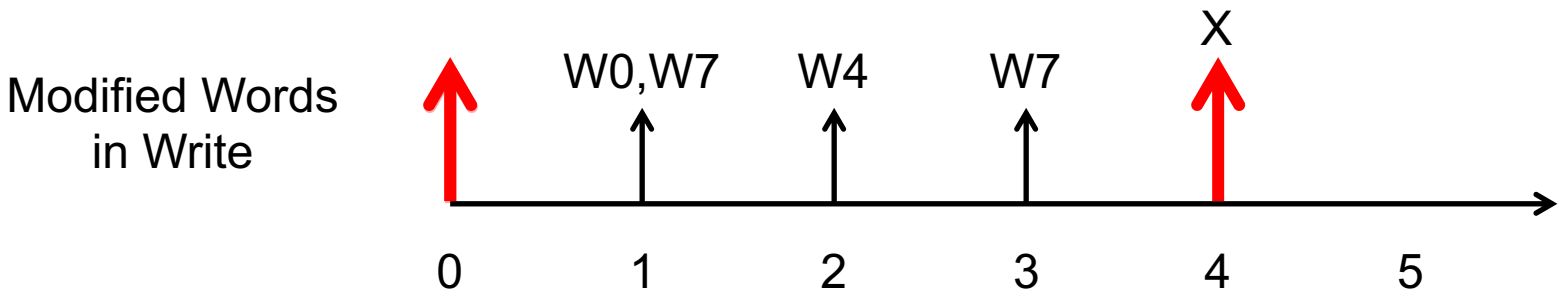
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7	W0 W4 W7		
LCTR / TCTR	0 / 0	1 / 0	2 / 0	3 / 0		



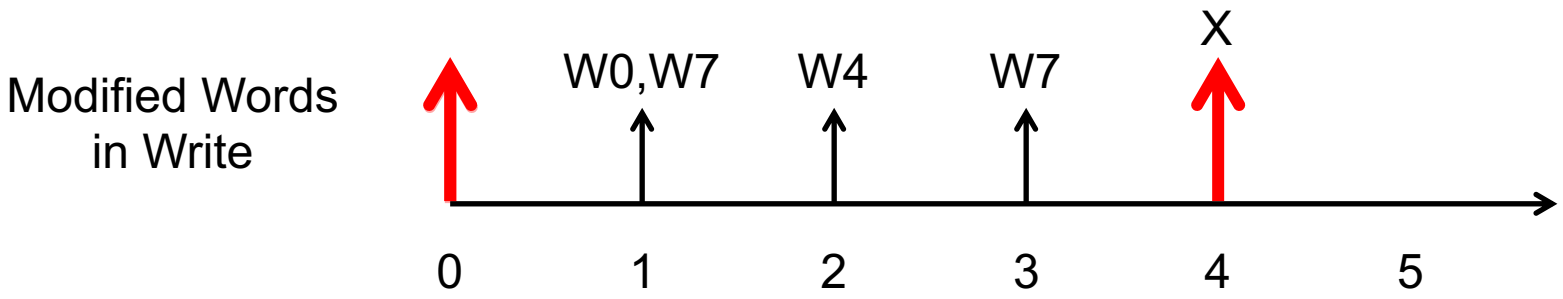
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7	W0 W4 W7		
LCTR / TCTR	0 / 0	1 / 0	2 / 0	3 / 0		



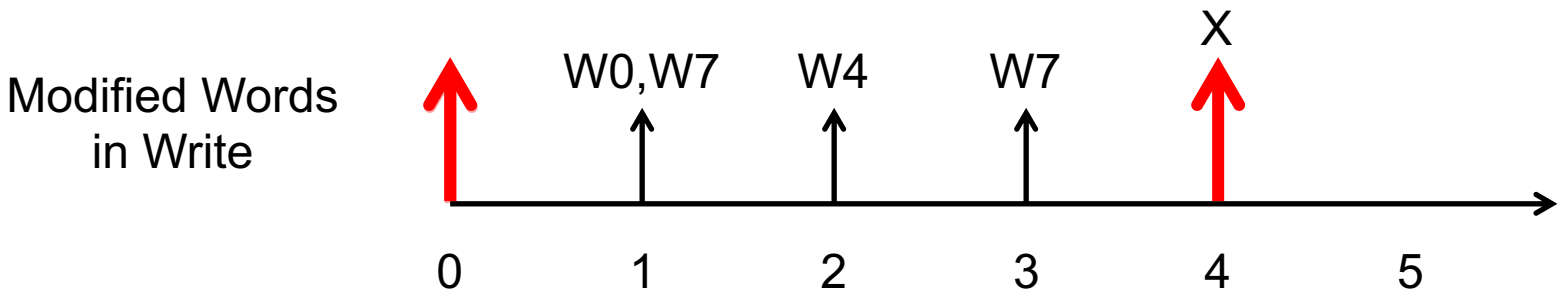
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7	W0 W4 W7		
LCTR	0	1	2	3	4	
TCTR	0	0	0	0	4	



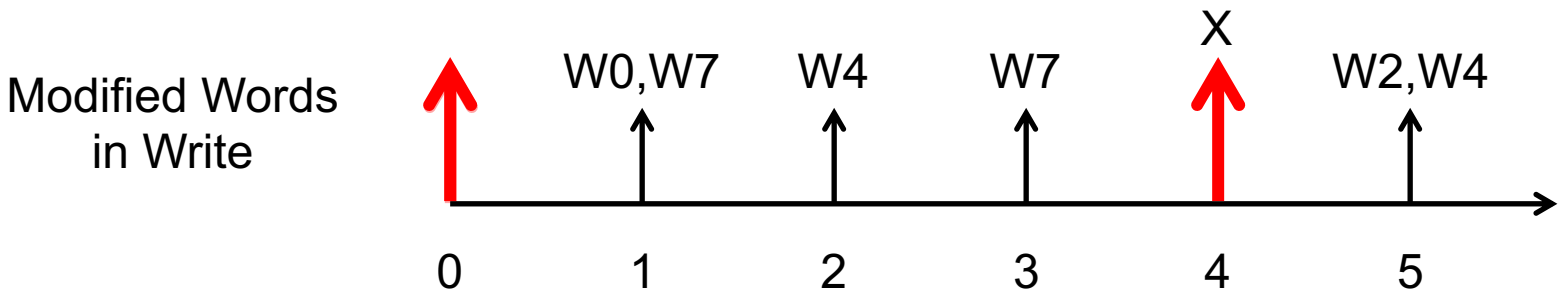
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7	W0 W4 W7	ALL	
LCTR	0	1	2	3	4	
TCTR	0	0	0	0	4	



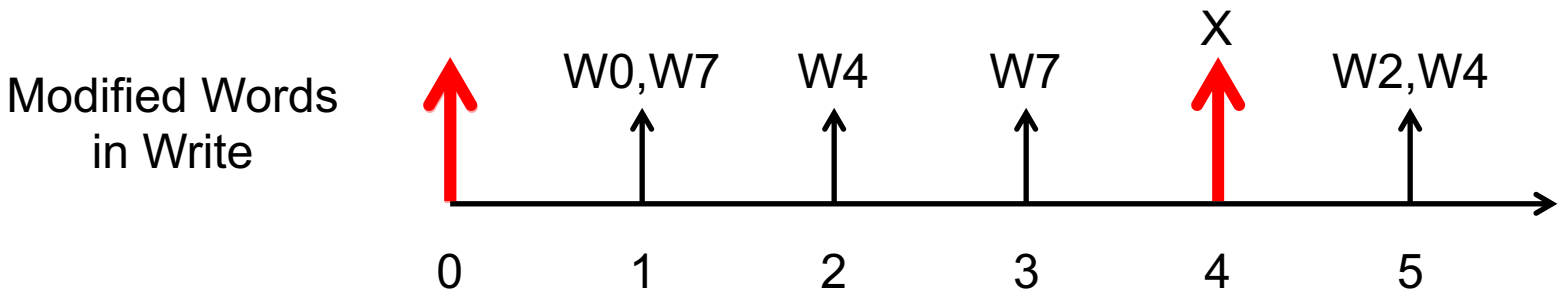
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7	W0 W4 W7	ALL	
LCTR	0	1	2	3	4	
TCTR	0	0	0	0	4	



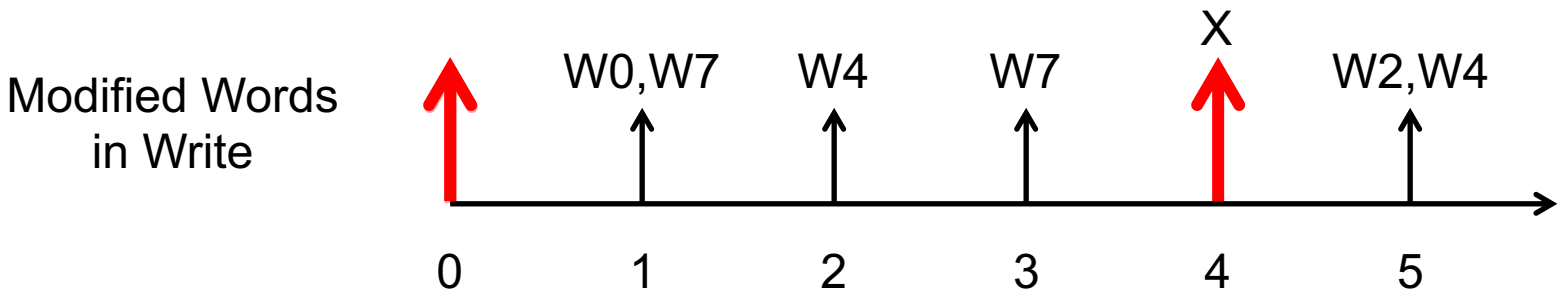
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7	W0 W4 W7	ALL	
LCTR	0	1	2	3	4	5
TCTR	0	0	0	0	4	4



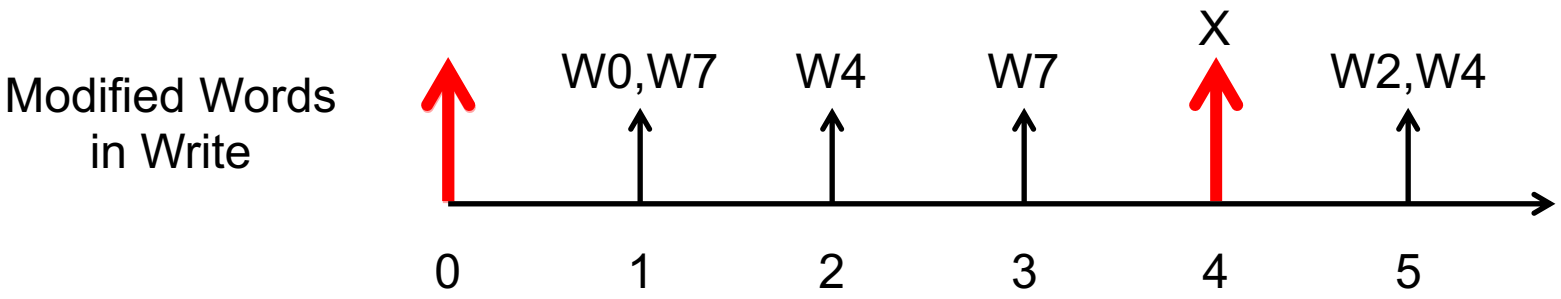
DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7	W0 W4 W7	ALL	W2 W4
LCTR	0	1	2	3	4	5
TCTR	0	0	0	0	4	4



DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7	W0 W4 W7	ALL	W2 W4
LCTR	0	1	2	3	4	5
TCTR	0	0	0	0	4	4

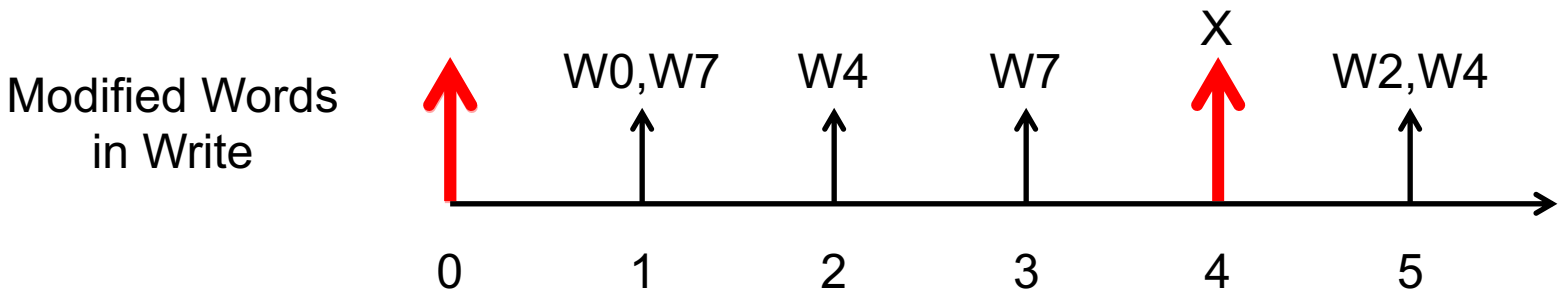


DEUCE does partial re-encryption between Epochs

DEUCE: EXAMPLE (EPOCH = 4 WRITES)

Encrypted Words	ALL	W0 W7	W0 W4 W7	W0 W4 W7	ALL	W2 W4
LCTR	0	1	2	3	4	5
TCTR	0	0	0	0	4	4

Get TrailCTR by masking 2 LSB off LeadCTR
DEUCE needs only one physical counter per line



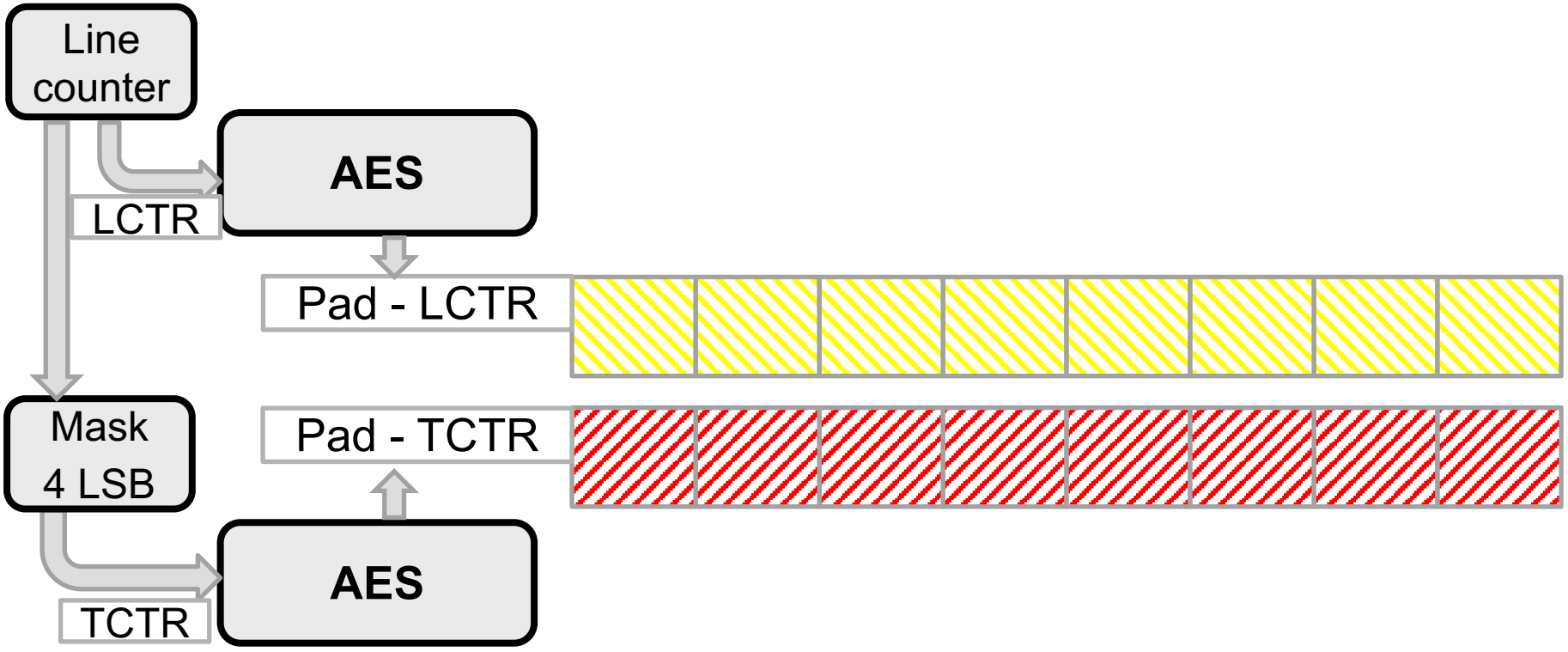
DEUCE does partial re-encryption between Epochs

DEUCE: HOW TO DECRYPT?

DEUCE generates two pads with LCTR and TCTR

DEUCE: HOW TO DECRYPT?

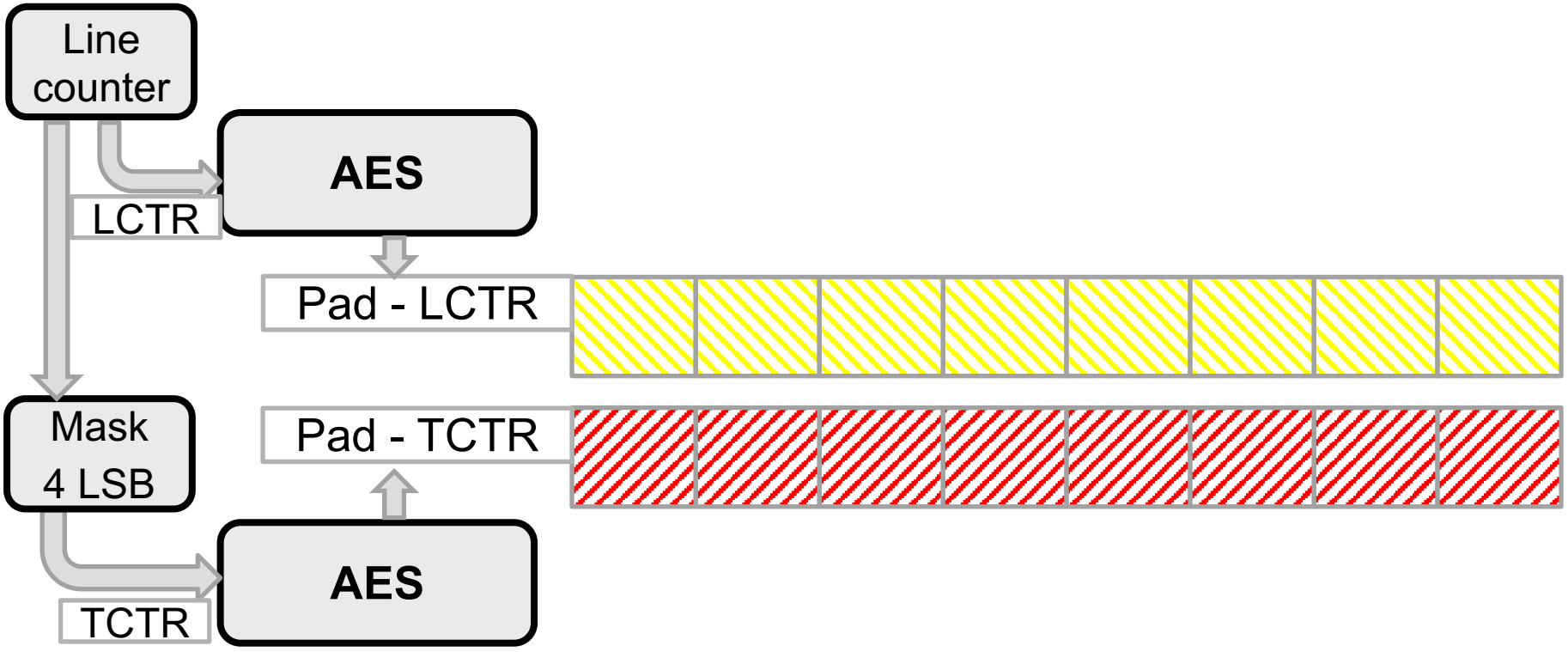
DEUCE generates two pads with LCTR and TCTR



DEUCE: HOW TO DECRYPT?

DEUCE generates two pads with LCTR and TCTR

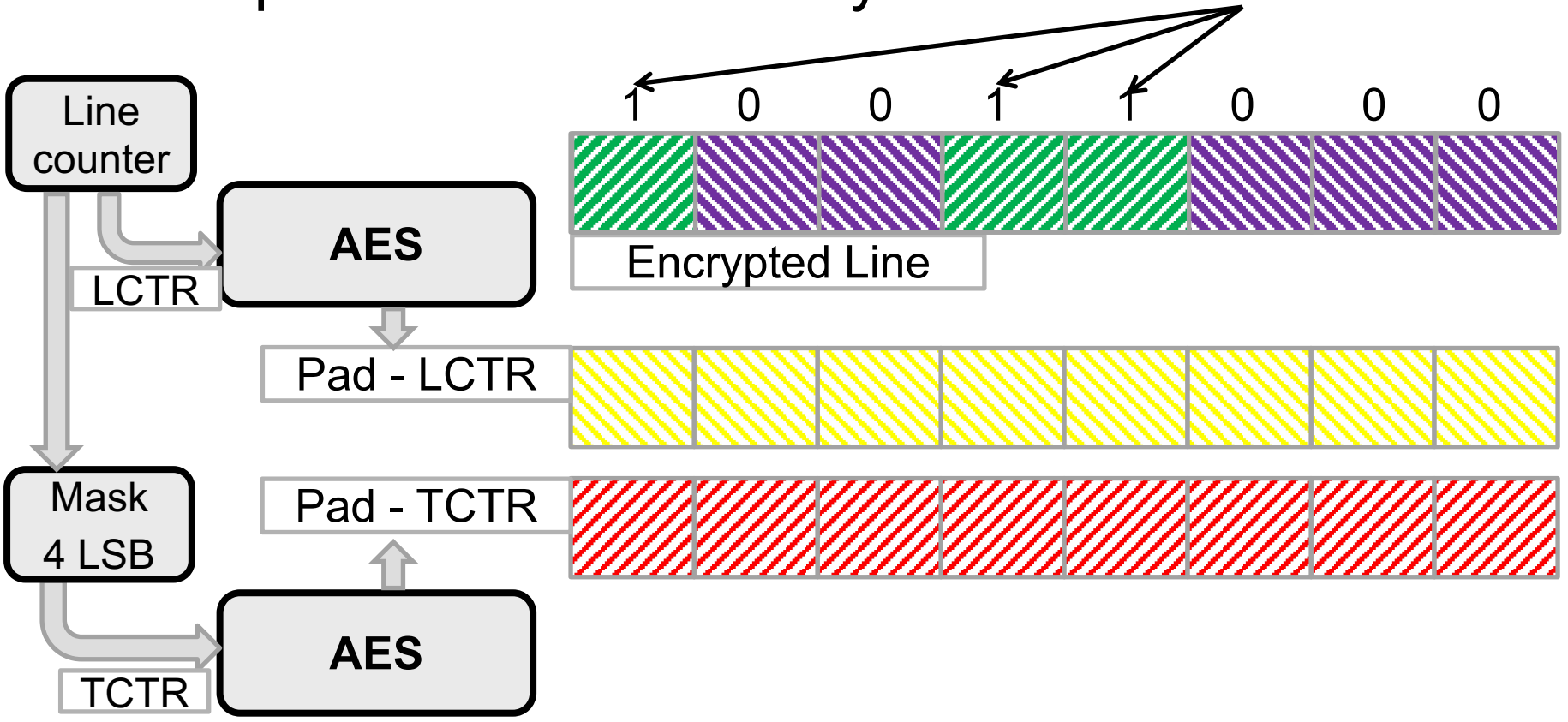
Which pad to use decided by the "Modified" bits



DEUCE: HOW TO DECRYPT?

DEUCE generates two pads with LCTR and TCTR

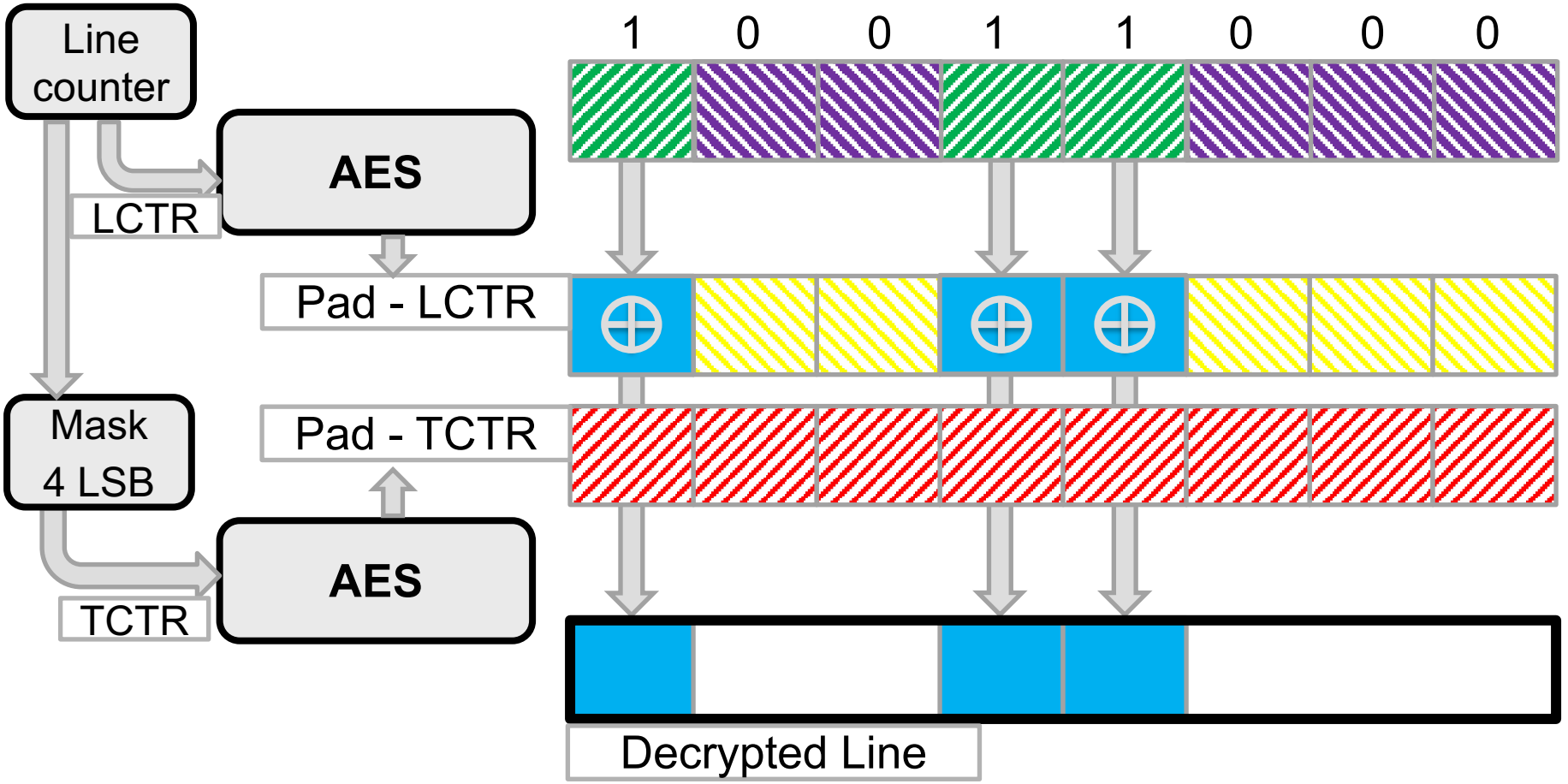
Which pad to use decided by the "Modified" bits



DEUCE: HOW TO DECRYPT?

DEUCE generates two pads with LCTR and TCTR

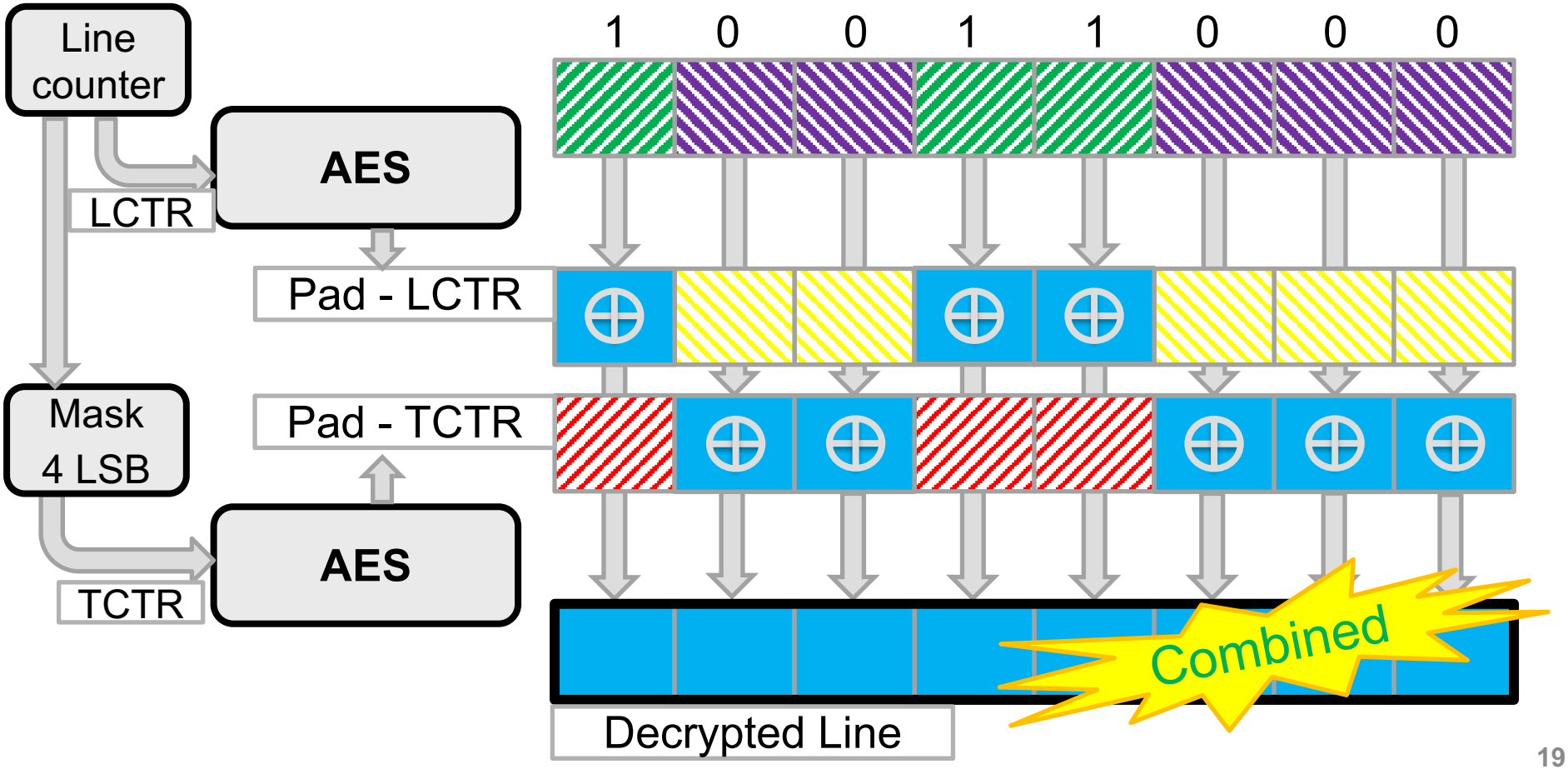
Which pad to use decided by the "Modified" bits



DEUCE: HOW TO DECRYPT?

DEUCE generates two pads with LCTR and TCTR

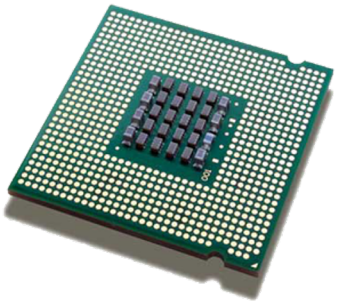
Which pad to use decided by the "Modified" bits



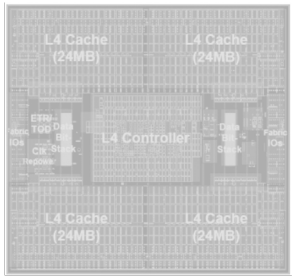
OUTLINE

- Introduction to PCM and encryption
- Background on Counter-mode Encryption
- DEUCE
- Results 
- Summary

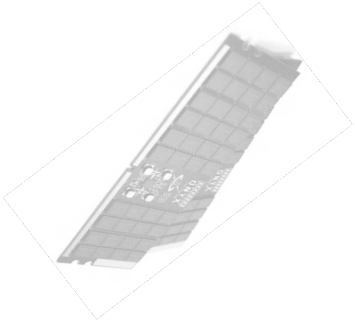
METHODOLOGY



CPU



Shared L4 Cache

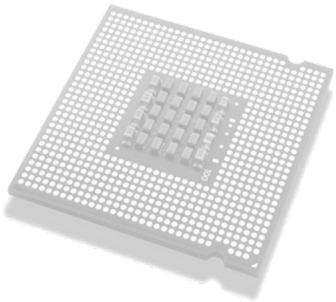


Phase Change Memory

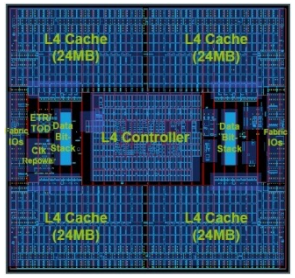
Core Chip

- 8 cores, each 4GHz 4-wide core
- L1/L2/L3➔32KB/256KB/1MB

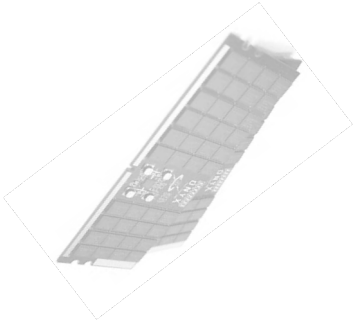
METHODOLOGY



CPU



Shared L4 Cache

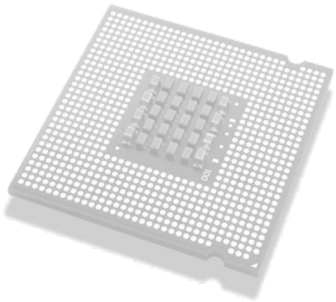


Phase Change Memory

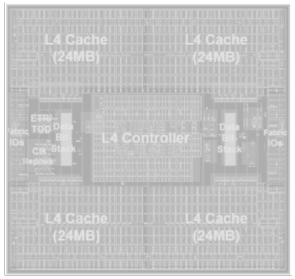
Shared L4 Cache:

- 64 MB capacity
- 50 cycle latency

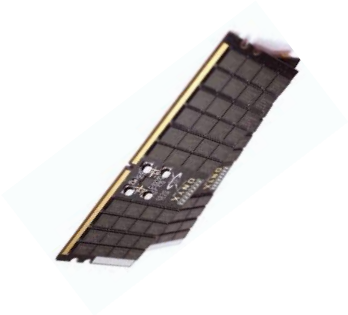
METHODOLOGY



CPU



Shared L4 Cache

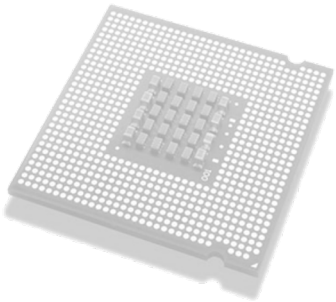


Phase Change Memory

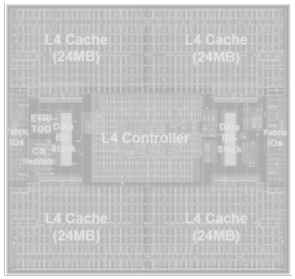
Phase Change Memory [Samsung ISSCC'12]

- 4 ranks, each 8GB ➔ 32GB total
- Read latency 75ns
- Write latency 150ns (per 128-bit write slot)

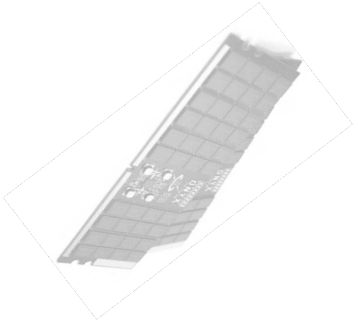
METHODOLOGY



CPU



Shared L4 Cache



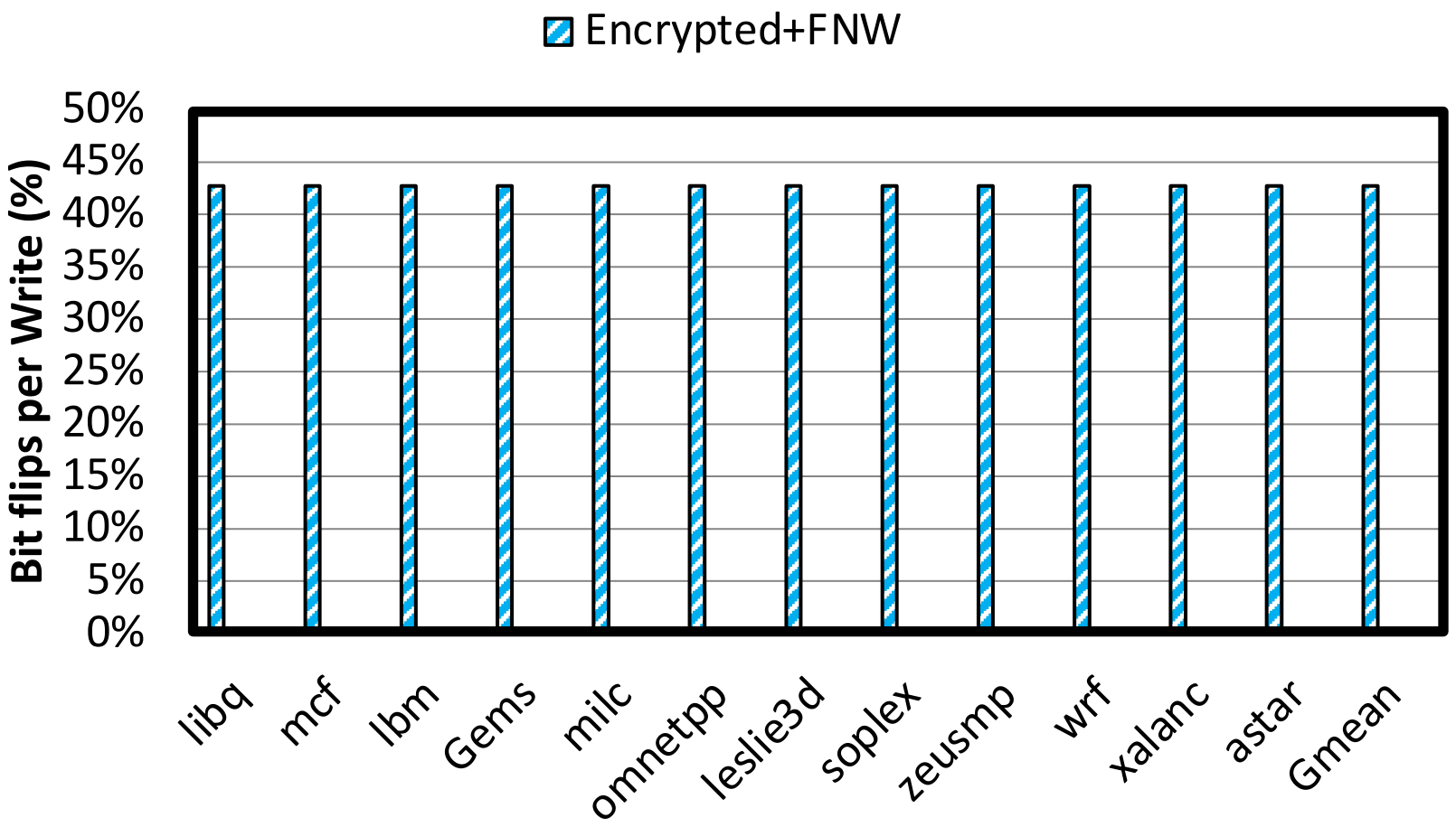
Phase Change Memory

Workloads

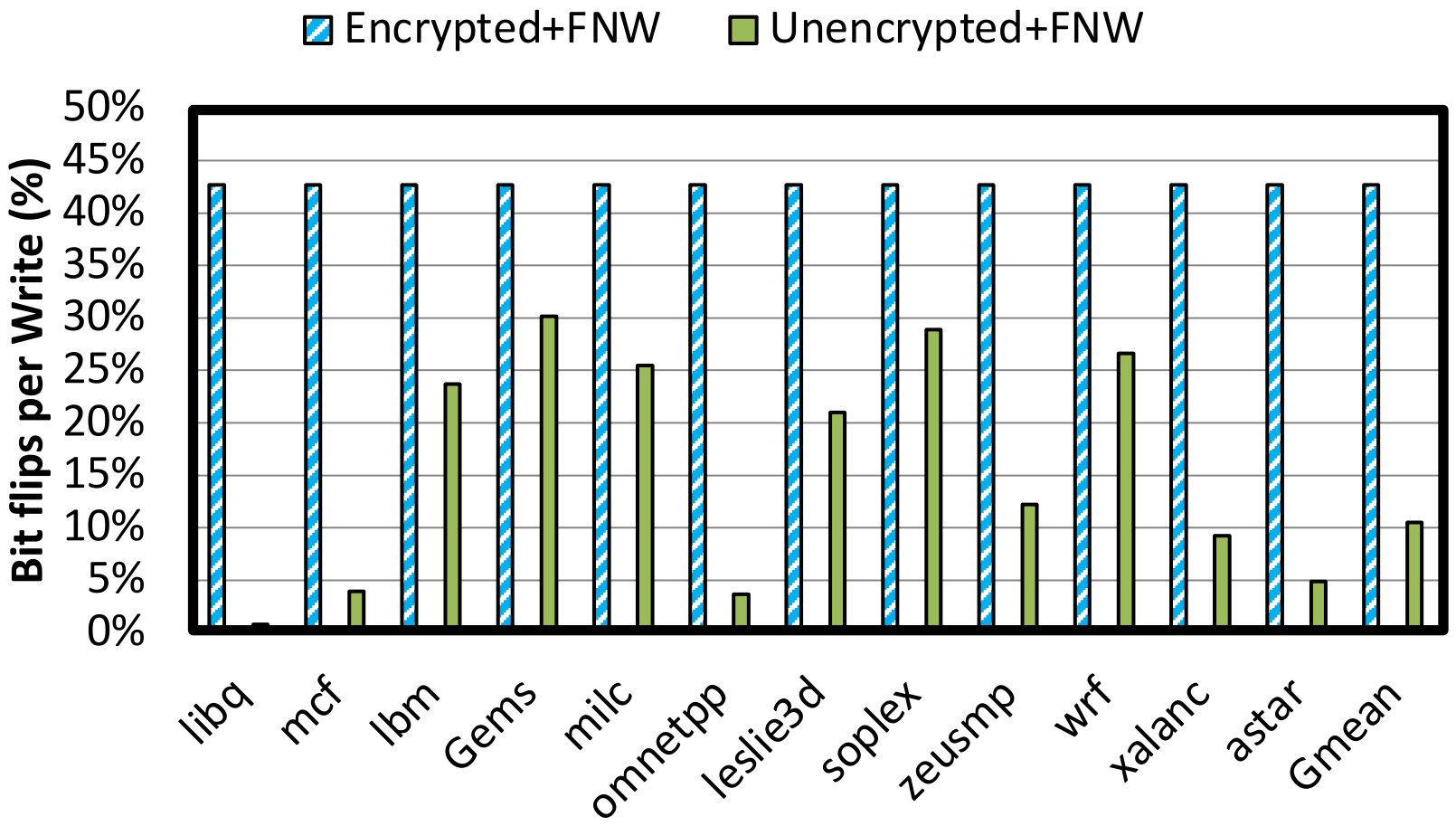
- SPEC2006: High MPKI, rate mode
- 4 billion instruction slice

DEUCE: Epoch=32; Word size=2B

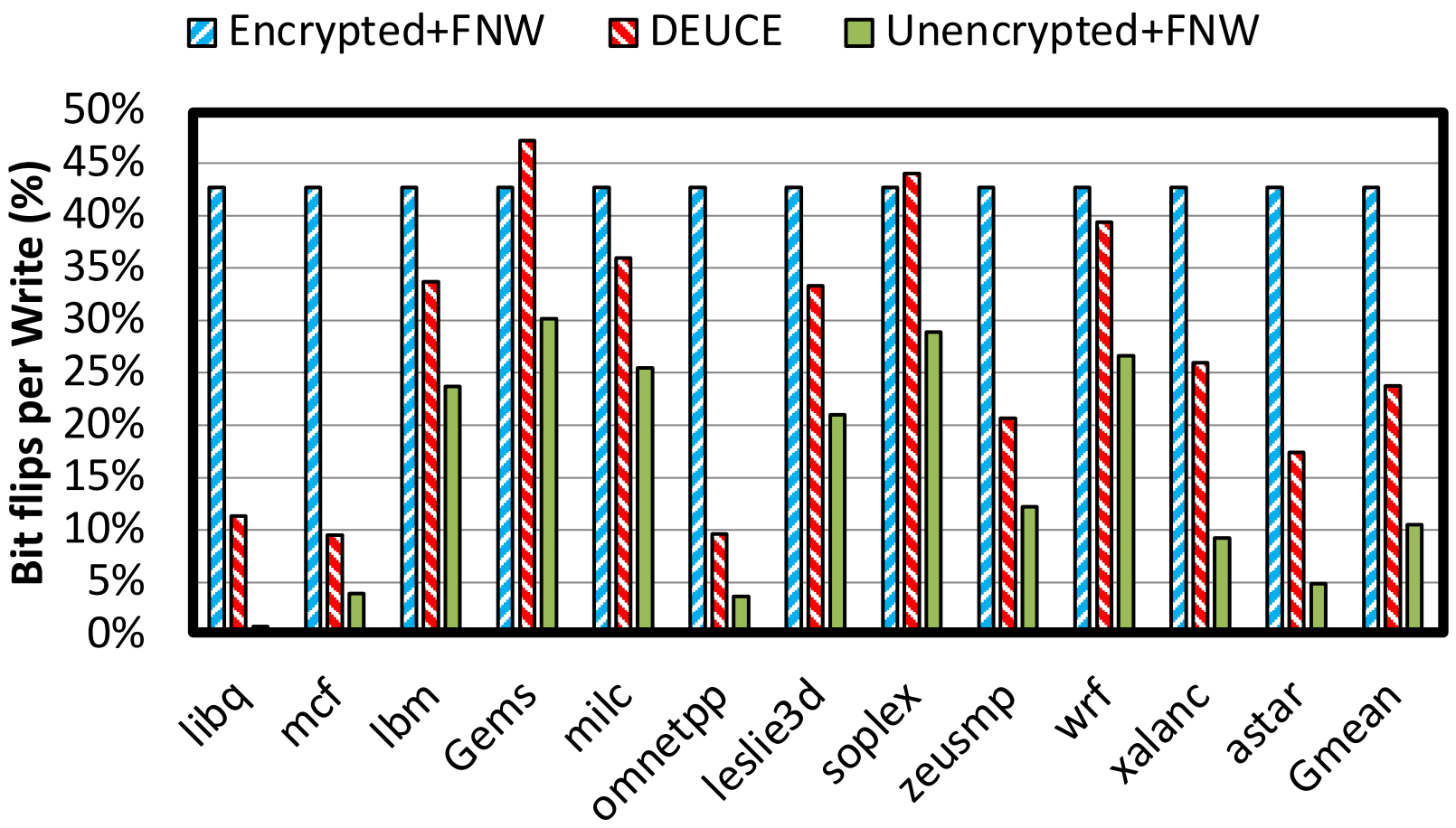
RESULTS: BIT FLIP ANALYSIS



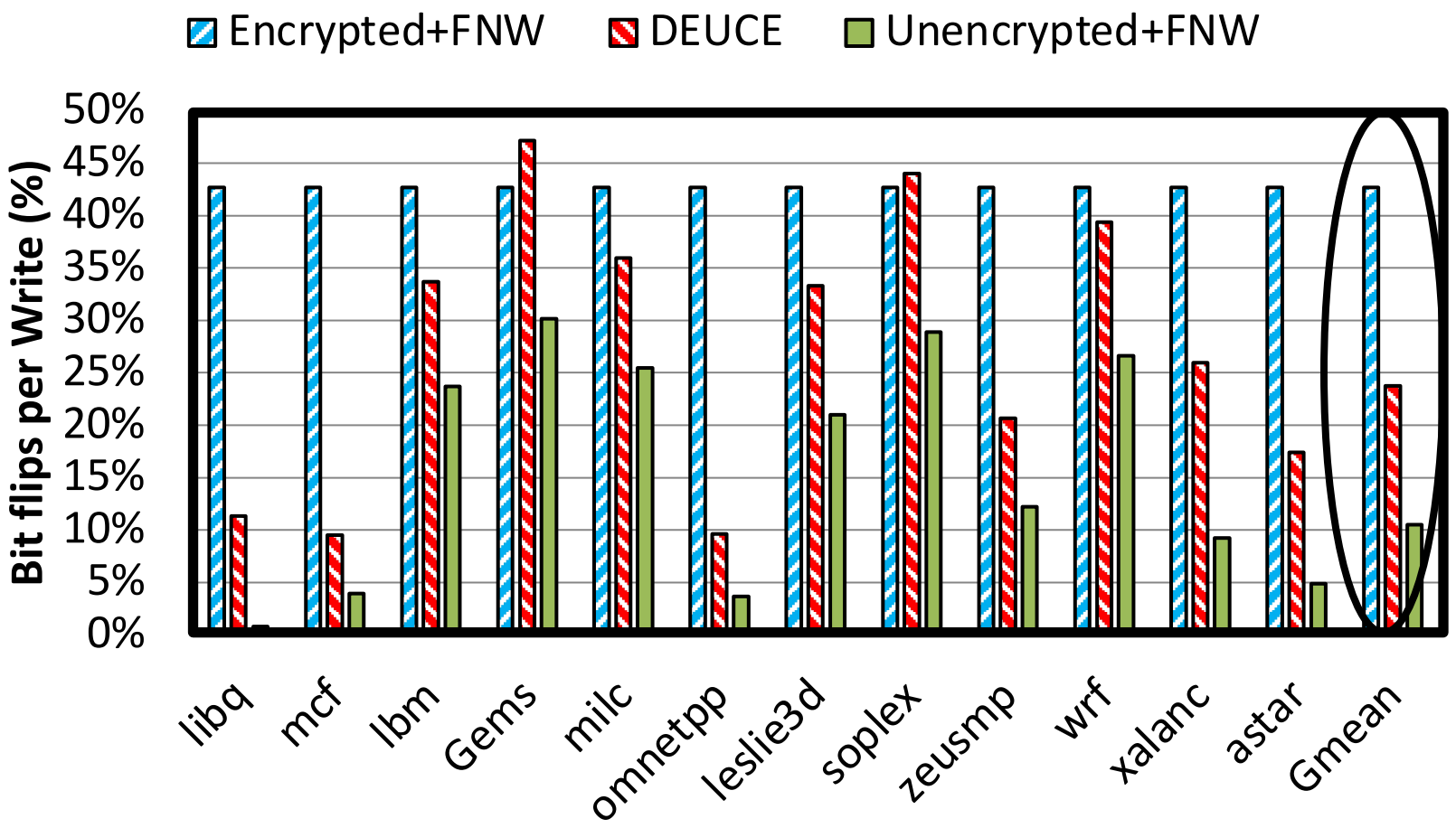
RESULTS: BIT FLIP ANALYSIS



RESULTS: BIT FLIP ANALYSIS

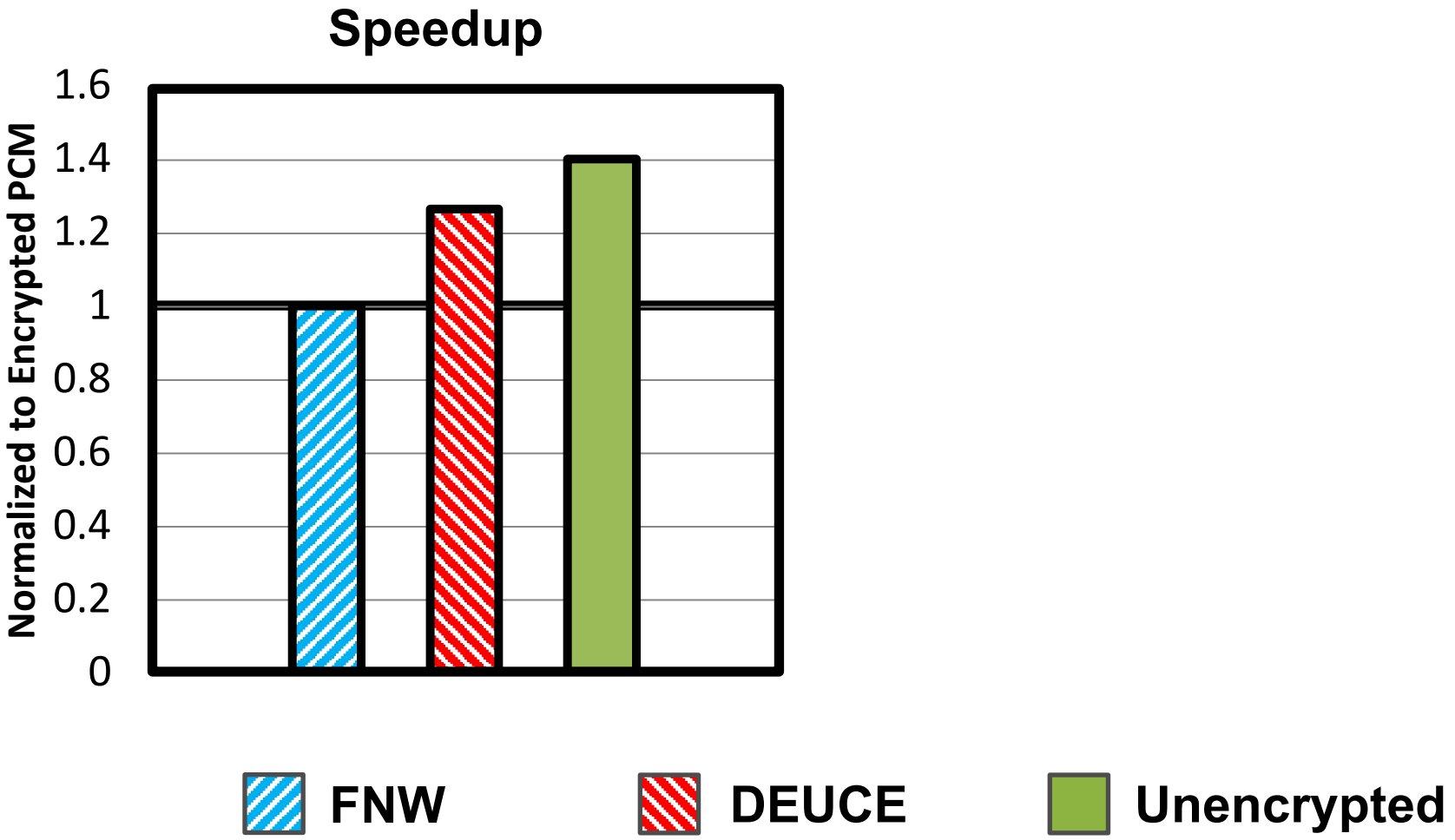


RESULTS: BIT FLIP ANALYSIS

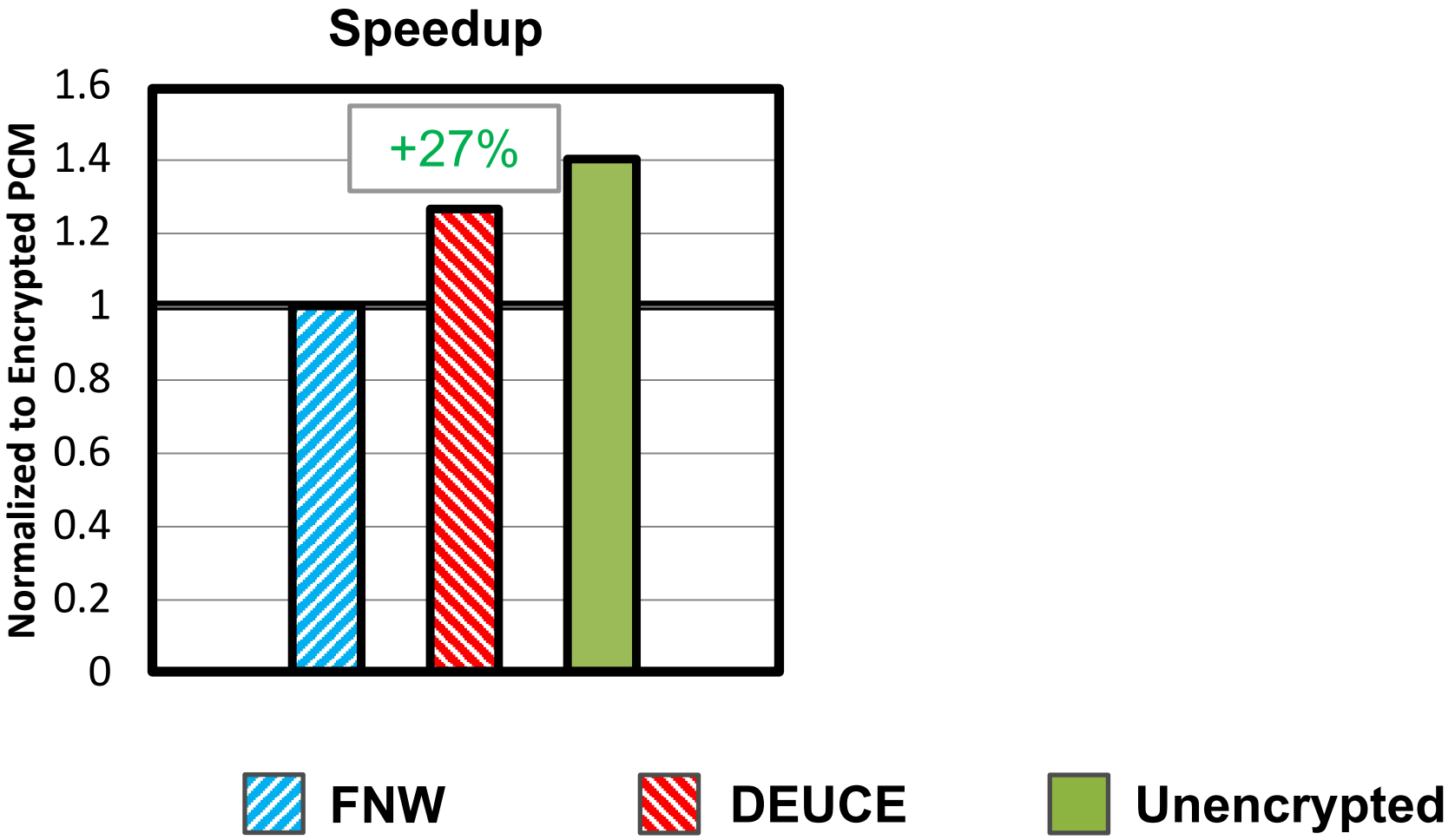


DEUCE eliminates two-thirds of the extra bit flips caused by encryption

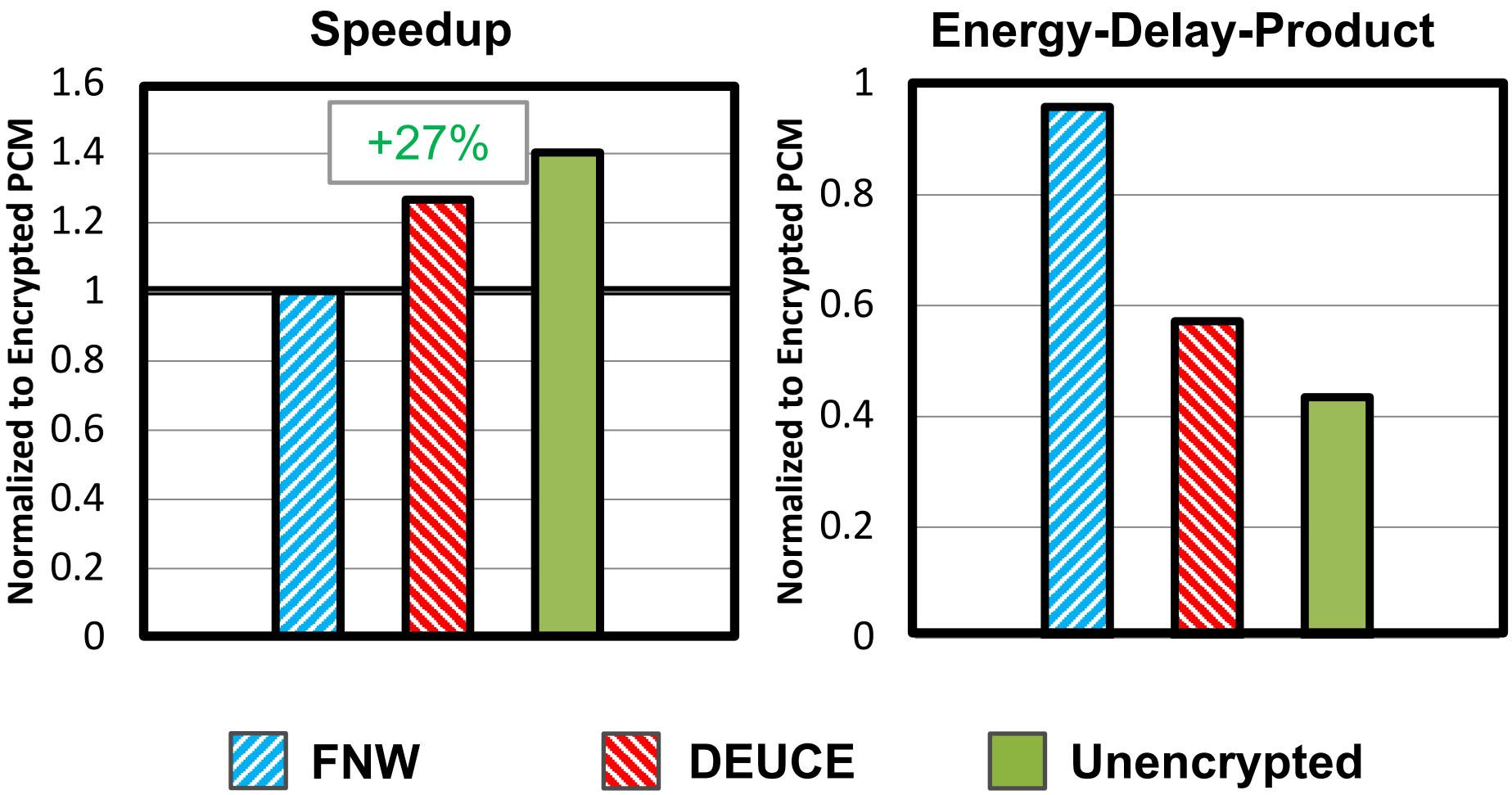
RESULTS: SPEEDUP&POWER ANALYSIS



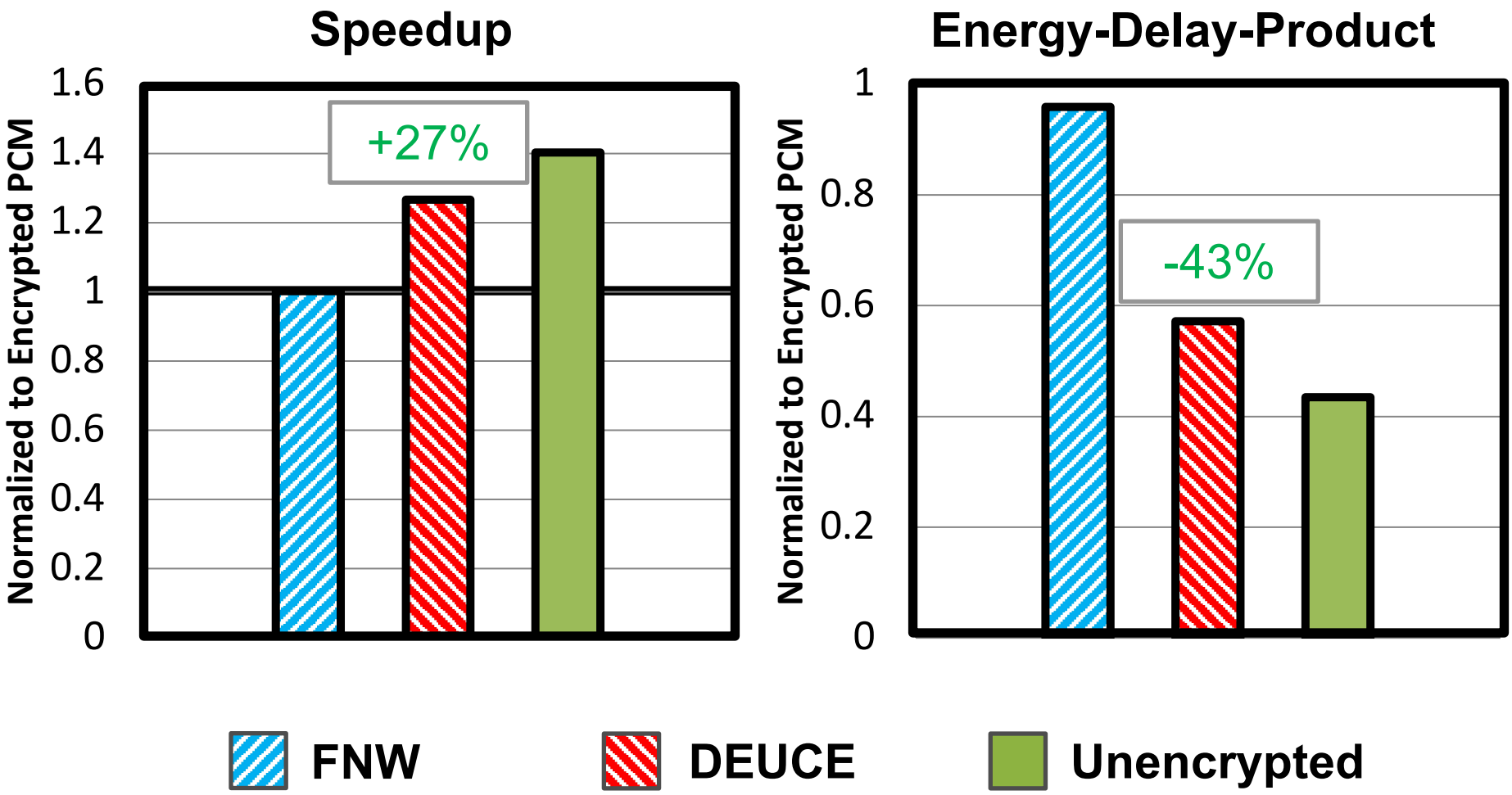
RESULTS: SPEEDUP&POWER ANALYSIS



RESULTS: SPEEDUP&POWER ANALYSIS



RESULTS: SPEEDUP&POWER ANALYSIS



Bit flip reduction improves speedup and EDP

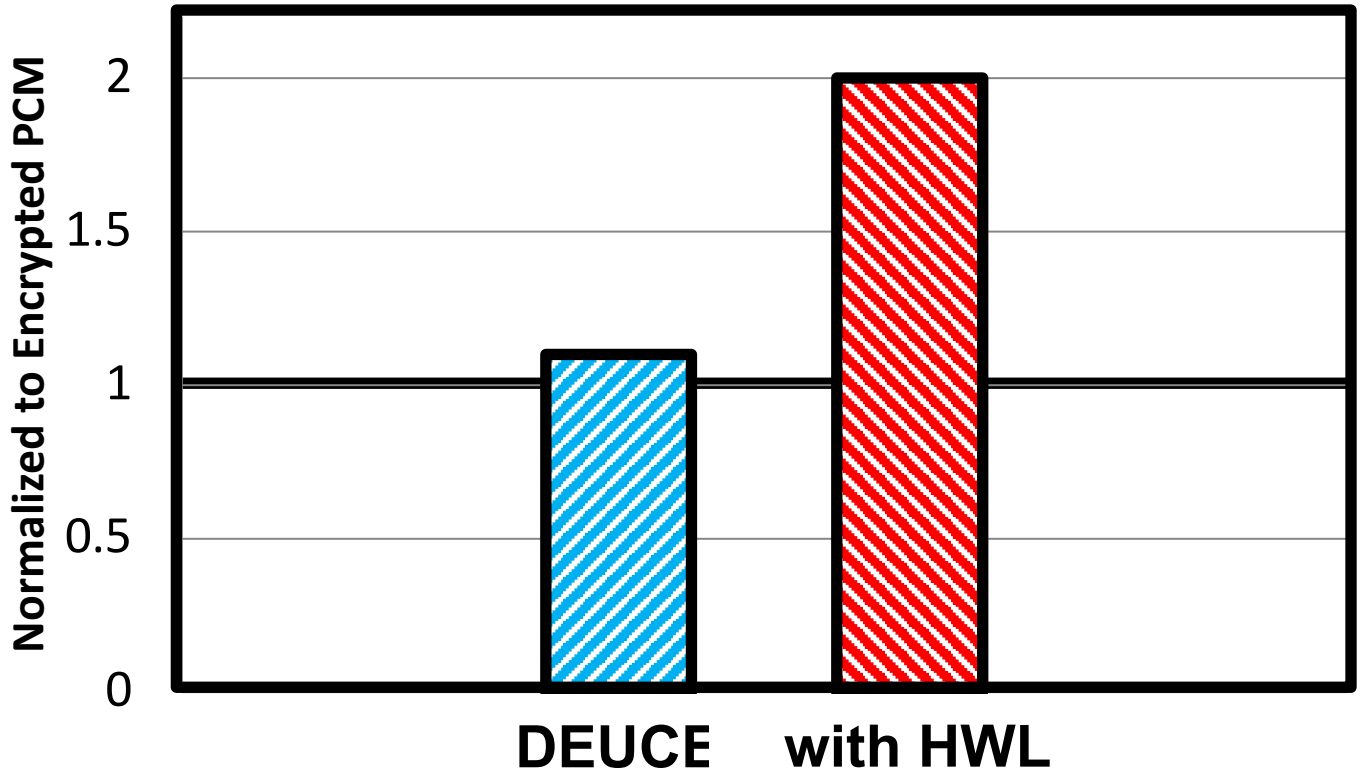
RESULTS: LIFETIME ANALYSIS

Heavily-written bits still heavily-written with DEUCE

RESULTS: LIFETIME ANALYSIS

Heavily-written bits still heavily-written with DEUCE

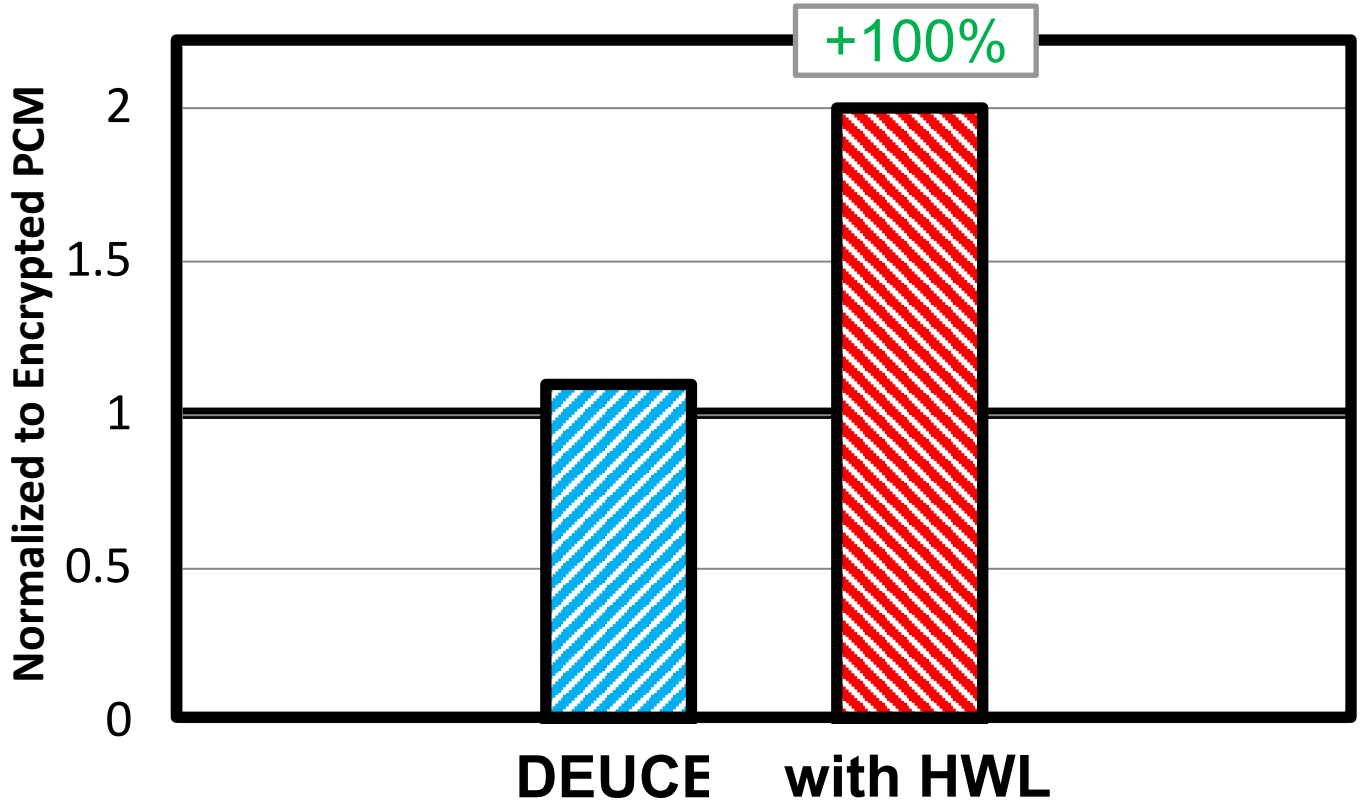
Solution: Zero-cost “Horizontal” Wear Leveling



RESULTS: LIFETIME ANALYSIS

Heavily-written bits still heavily-written with DEUCE

Solution: Zero-cost “Horizontal” Wear Leveling



Lifetime improvement of 2x

OUTLINE

- Introduction to PCM and encryption
- Baseline–Counter-mode Encryption
- DEUCE
- Results
- Summary 

EXECUTIVE SUMMARY

- Bit flips expensive in Phase Change Memory (PCM)
  - Affects Lifetime, Power, and Performance
  - PCM system optimized to reduce bit flips (~12%)

- PCM more vulnerable to attacks → stolen module
- Secure PCM with encryption
- Problem: Encryption increases bit flips 12%→50%
- Goal: Encrypt PCM without causing 4x bit flips

- Insight: re-encrypt only modified words
- DEUCE: write-efficient encryption scheme
- Result: bit flips reduced 50%→23% (27% speedup)

THANK YOU

EXTRA SLIDES

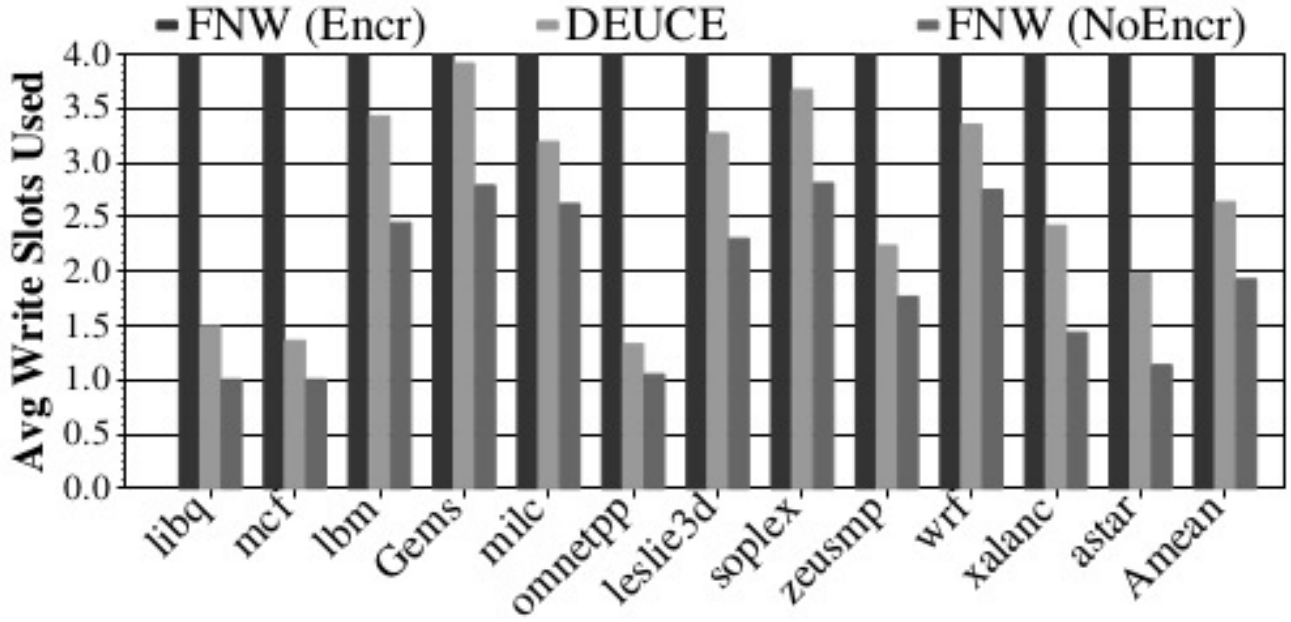
DEUCE FOR OTHER NVM

- In-place writing NVM
  - RRAM
  - STT-MRAM

WRITE SLOTS

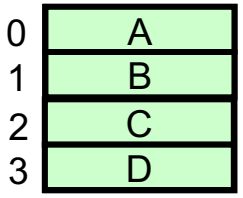
PCM is power-limited

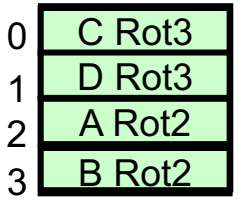
Write-slot of 128 Flip-N-Write-enabled bits (64 bit flips)



Average number of write slots used per write request. On average, DEUCE consumes 2.64 slots whereas unencrypted memory takes 1.92 slots out of the 4 write slots

IMPROVING PCM WEAR LEVELING



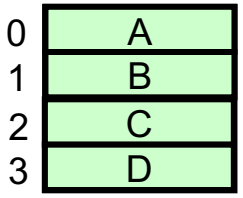


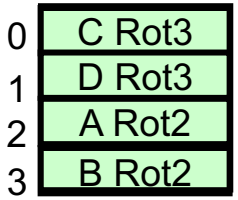
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling



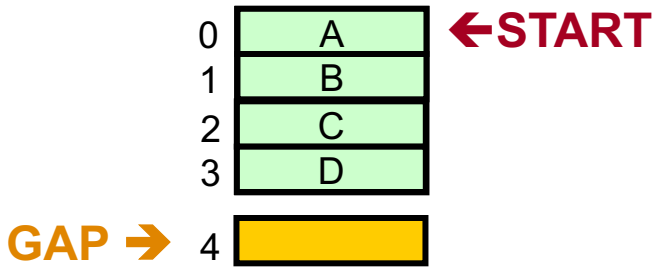


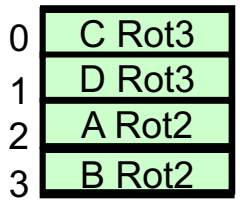
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling



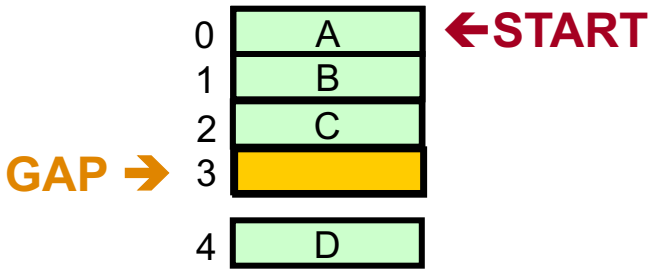


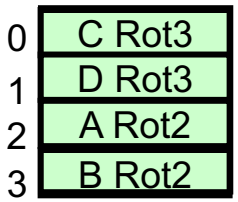
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling



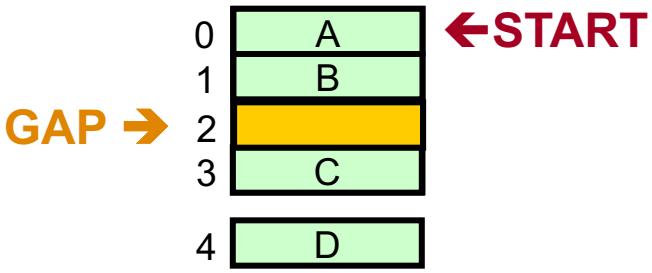


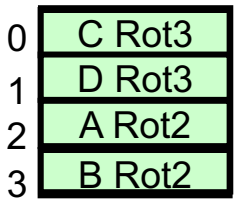
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling



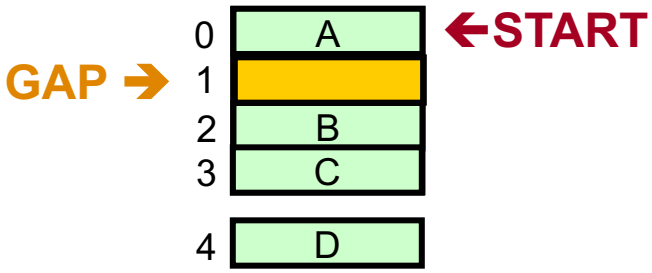


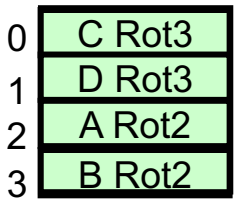
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling



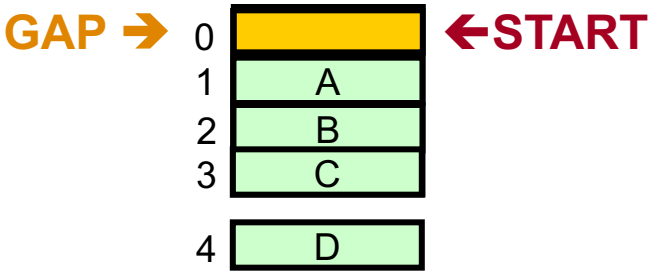


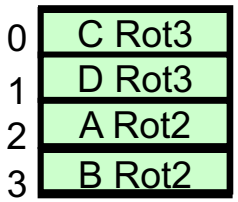
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling



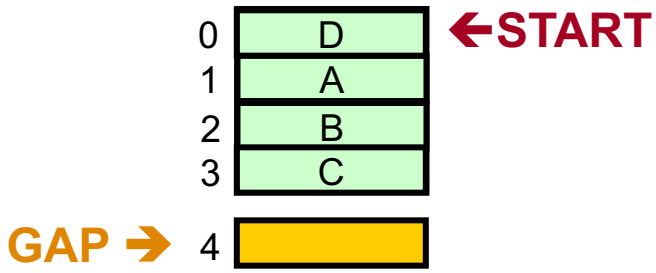


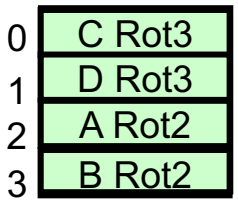
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling





Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

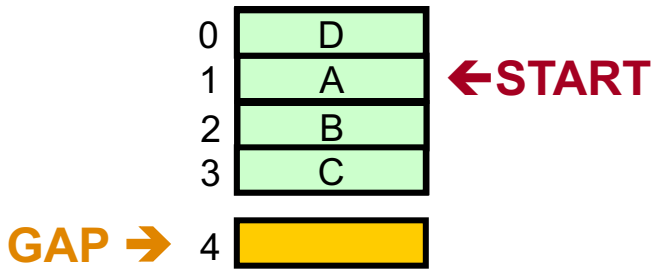
- Vertical Wear Leveling (Start-gap)—Inter-line leveling

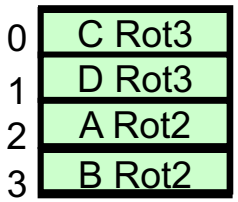
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling
- Horizontal Wear Leveling—Intra-line leveling



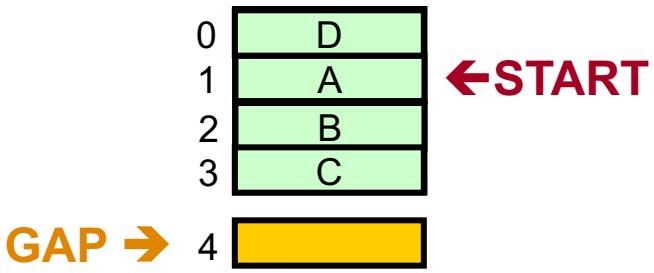


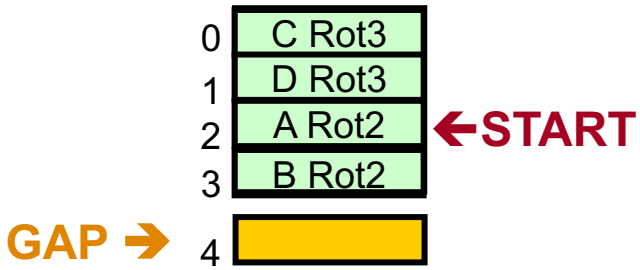
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling
- Horizontal Wear Leveling—Intra-line leveling



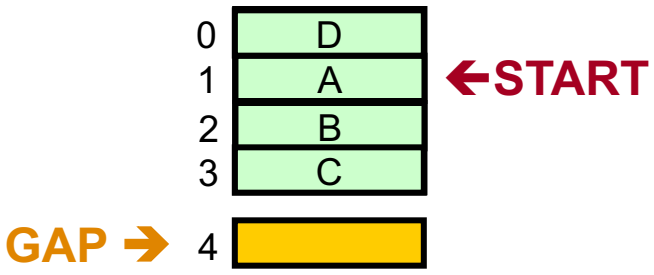


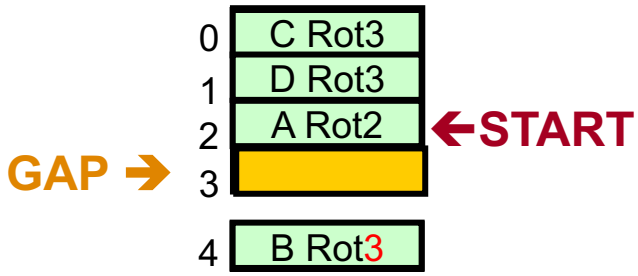
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling
- Horizontal Wear Leveling—Intra-line leveling



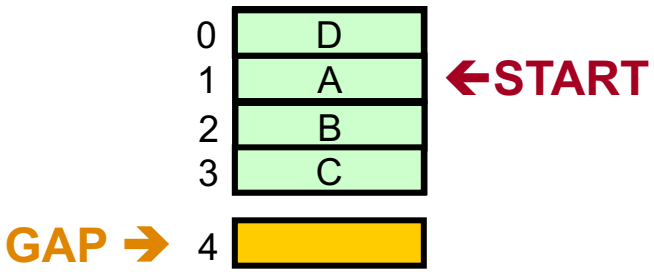


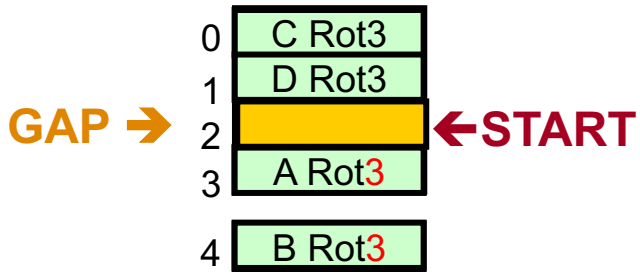
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling
- Horizontal Wear Leveling—Intra-line leveling



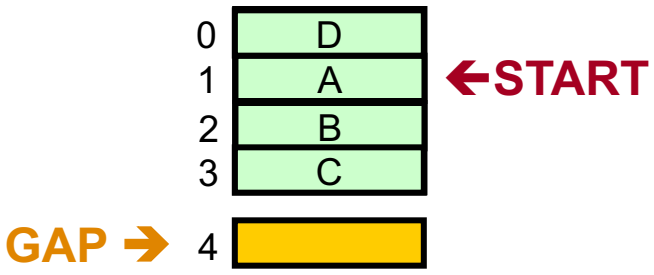


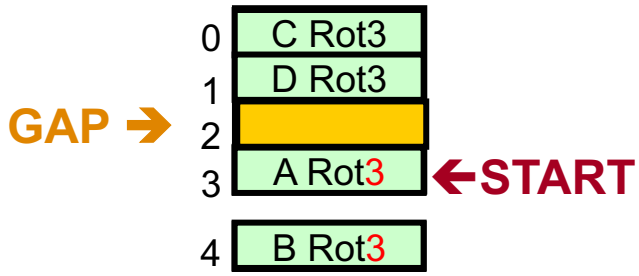
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling
- Horizontal Wear Leveling—Intra-line leveling



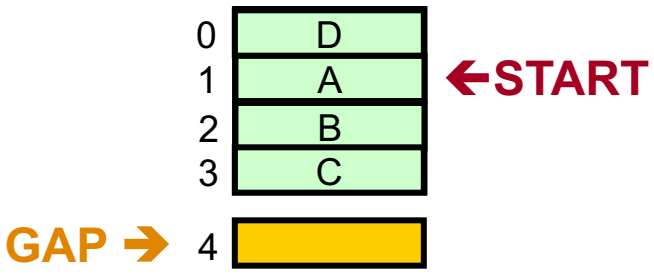


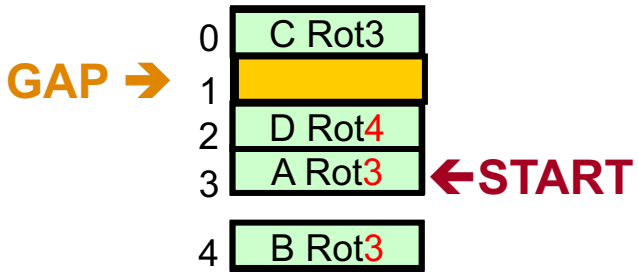
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling
- Horizontal Wear Leveling—Intra-line leveling



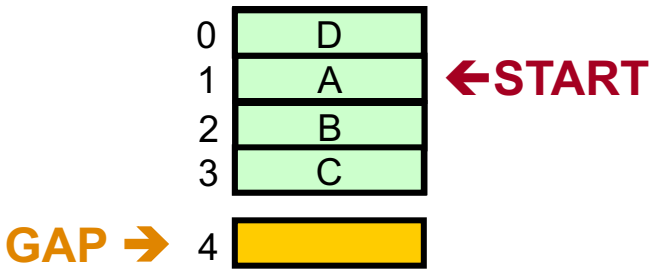


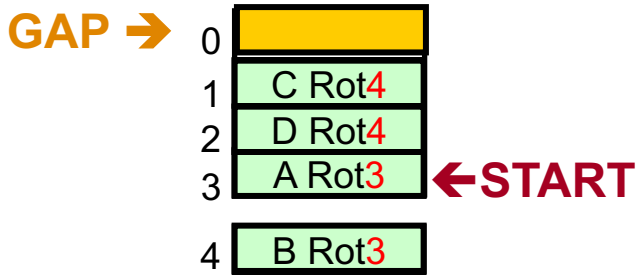
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling
- Horizontal Wear Leveling—Intra-line leveling



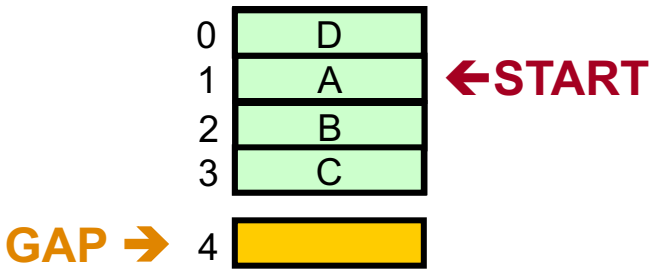


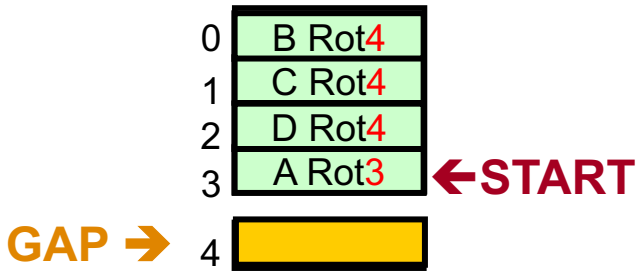
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling
- Horizontal Wear Leveling—Intra-line leveling



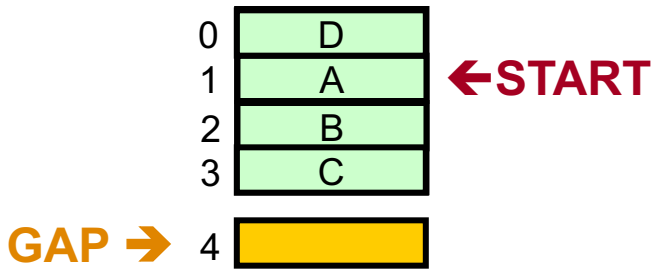


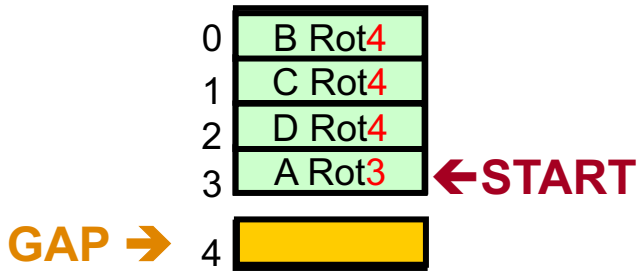
Vertical Wear Leveling

Horizontal Wear Leveling

IMPROVING PCM WEAR LEVELING

- Vertical Wear Leveling (Start-gap)—Inter-line leveling
- Horizontal Wear Leveling—Intra-line leveling





Vertical Wear Leveling

Horizontal Wear Leveling

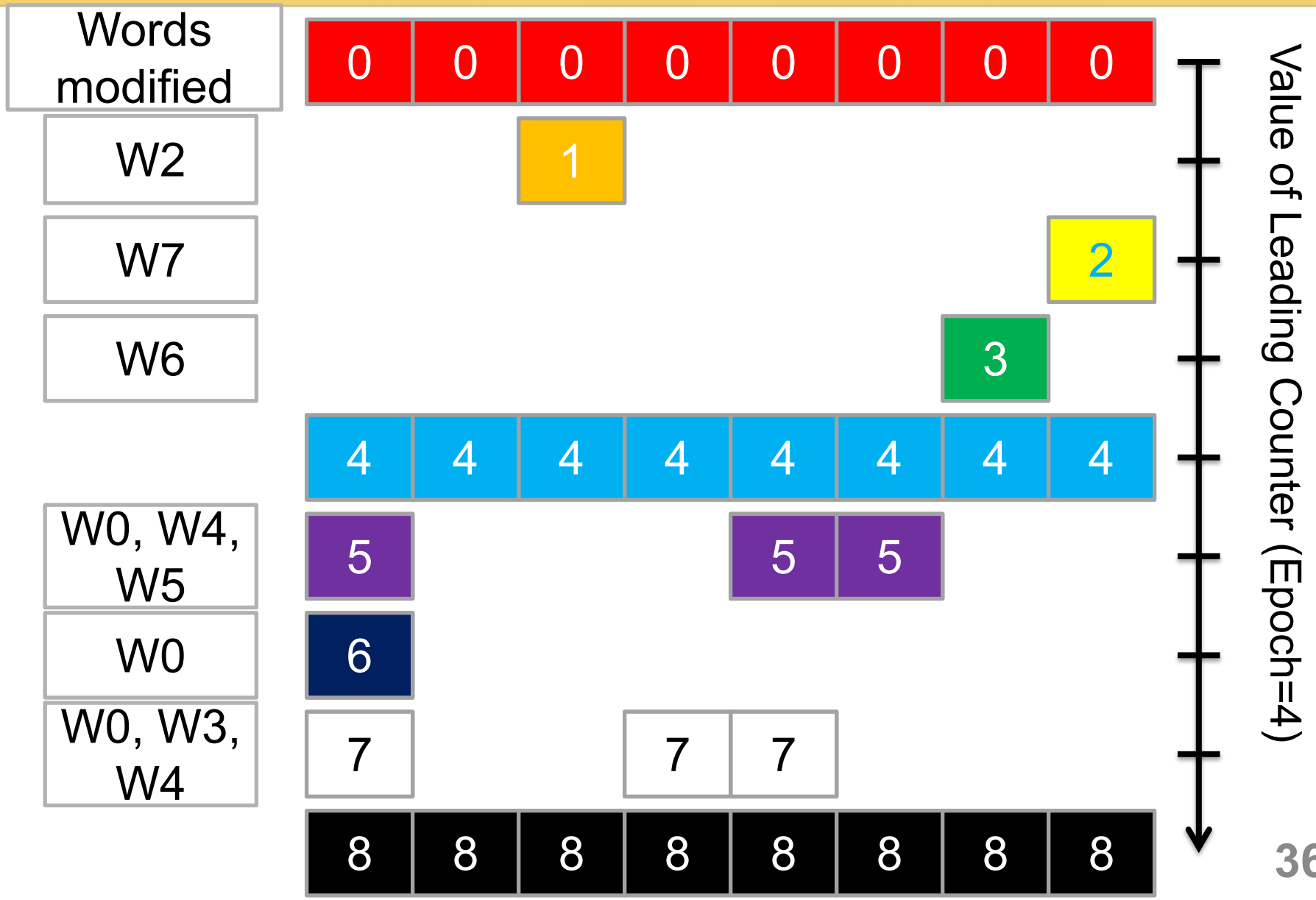
Re-use vertical wear leveling to level inside line

QUESTION: IS IT SECURE?

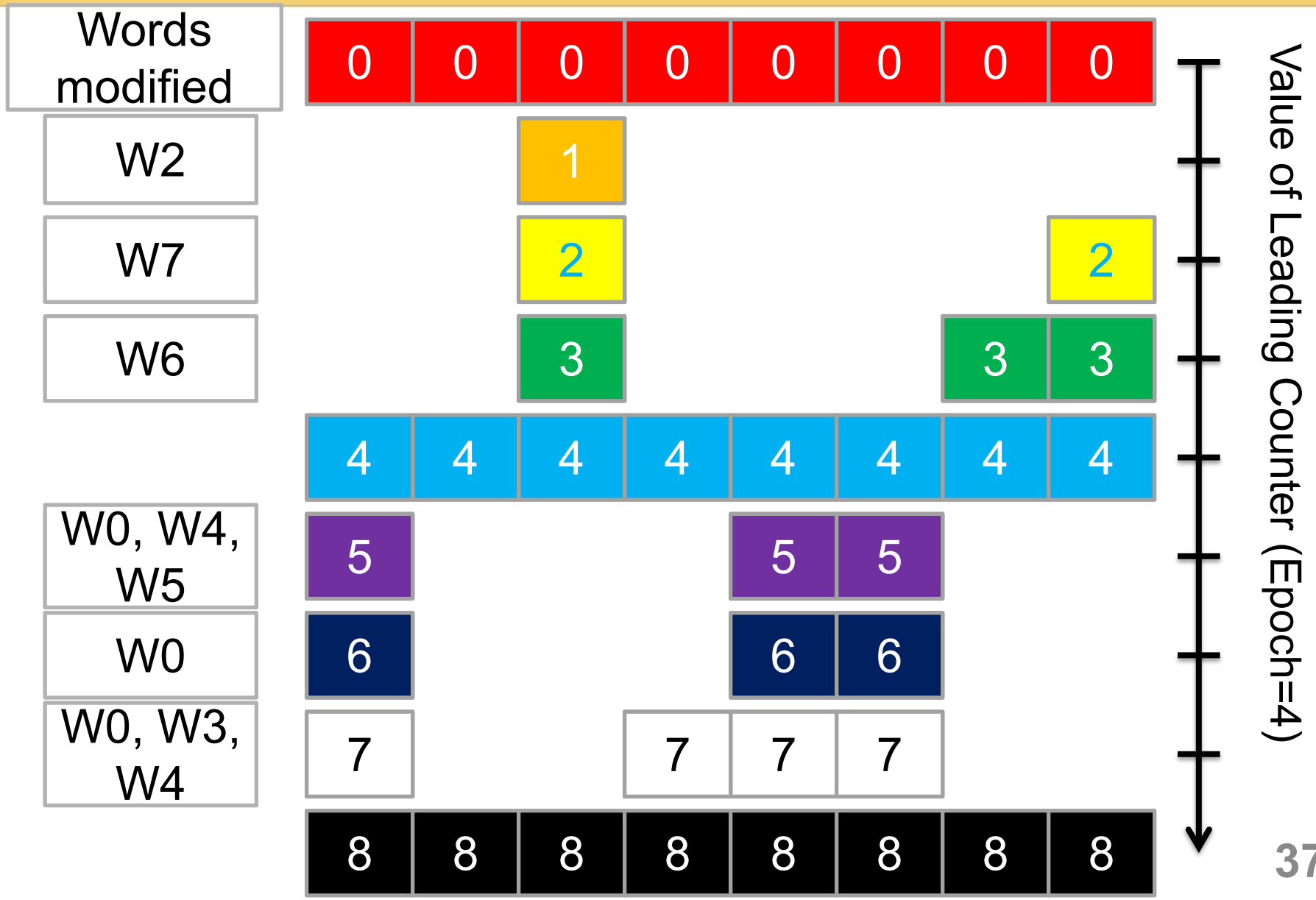
- Aren't you re-using the pad?
- AES-CTR-mode
  - Unique full pad per ever write.
- DEUCE
  - Epoch start, uses unique full pad
  - Between epochs, uses a unique full pad per write, to re-encrypt modified words. Unmodified words simply not touched

  ➔whole line is always encrypted, and pads are not re-used

MINIMUM RE-ENCRYPT



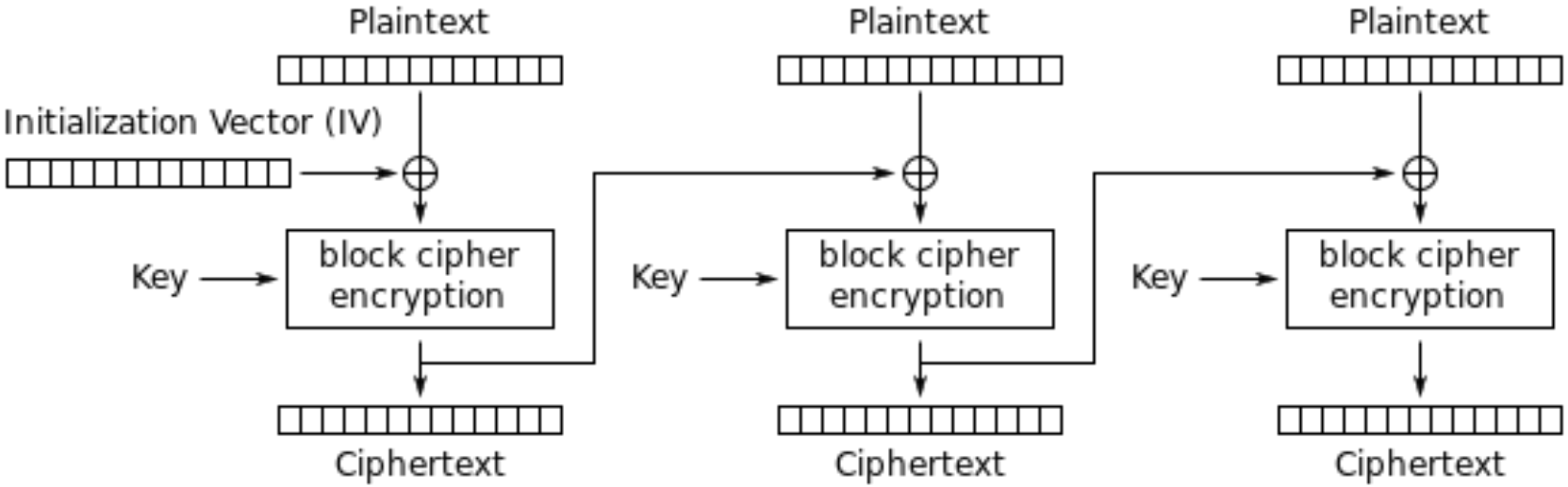
DEUCE RE-ENCRYPT



QUESTION: IS IT SECURE?

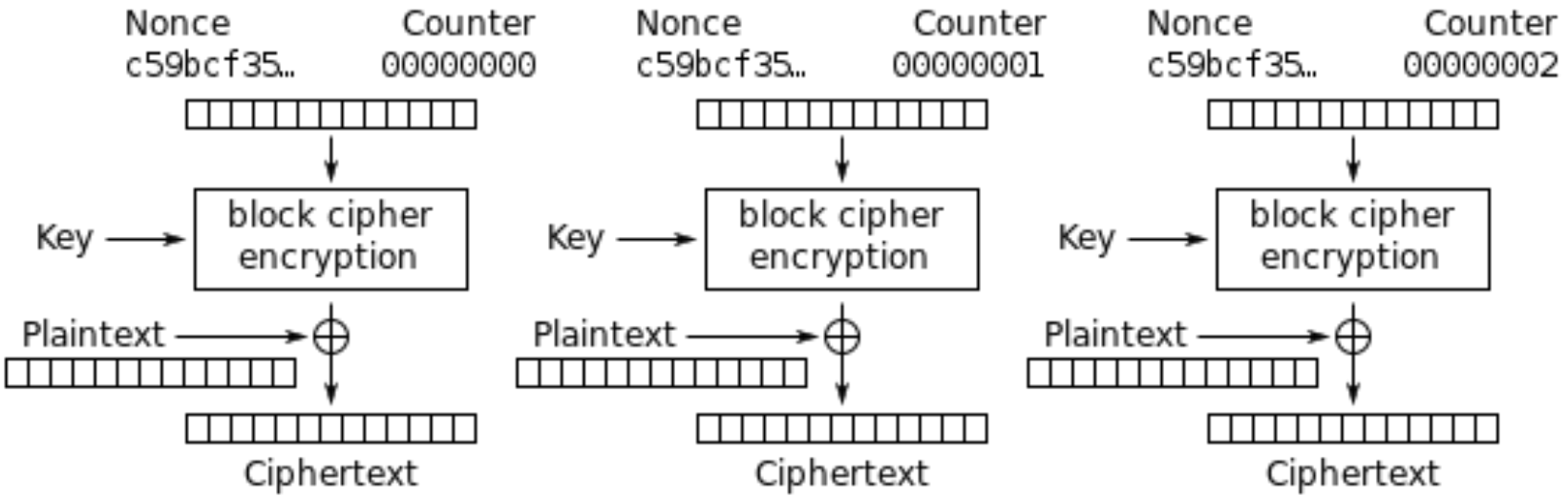
- What about information leak?
- AES-CTR-mode
  - Can tell when a line is modified
- DEUCE
  - Can tell when a line is modified, and
  - Can sometimes tell when a word is modified
  - Similar information leakage

CIPHER BLOCK CHAINING



Cipher Block Chaining (CBC) mode encryption

COUNTER-MODE ENCRYPTION

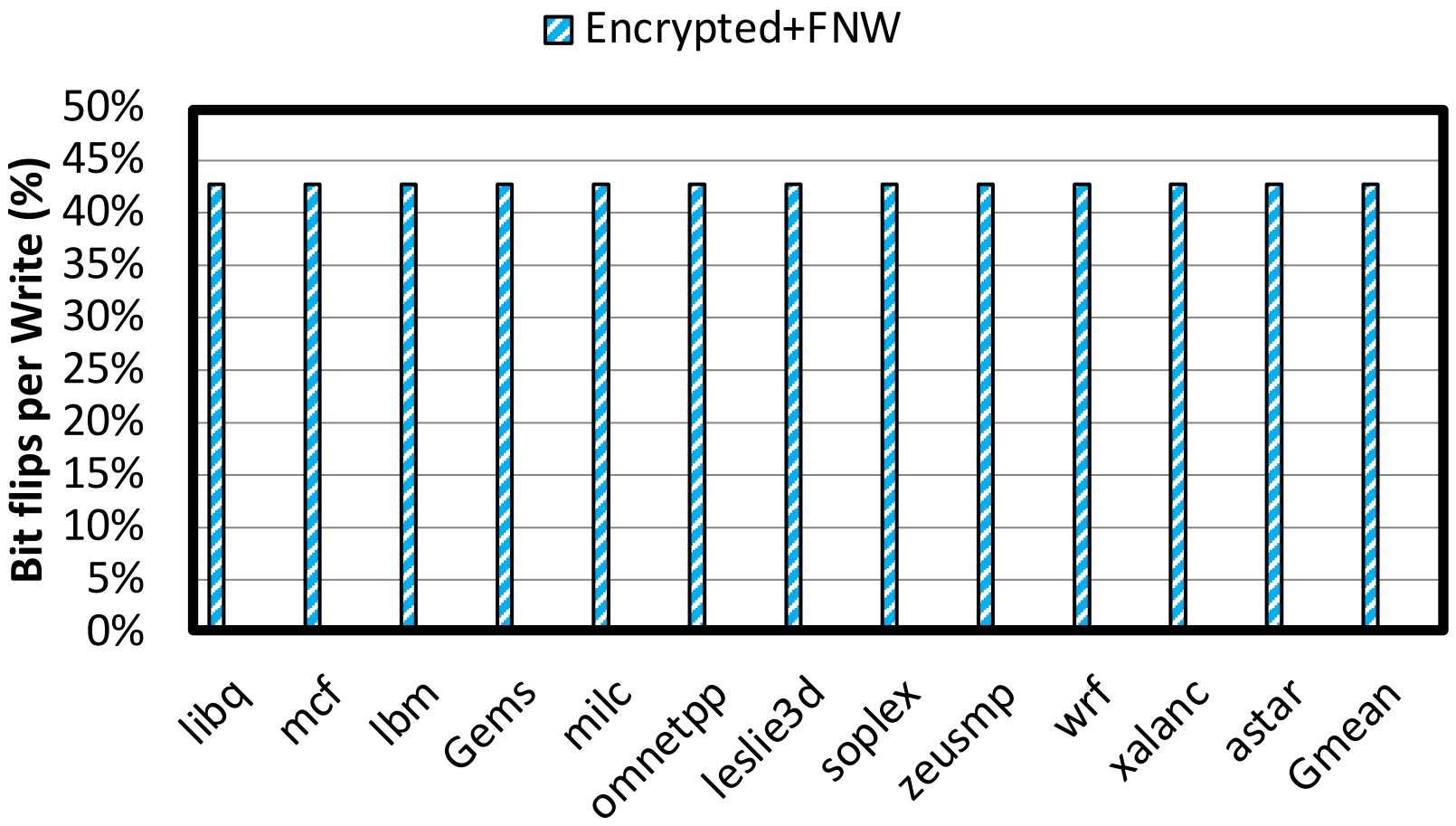


Counter (CTR) mode encryption

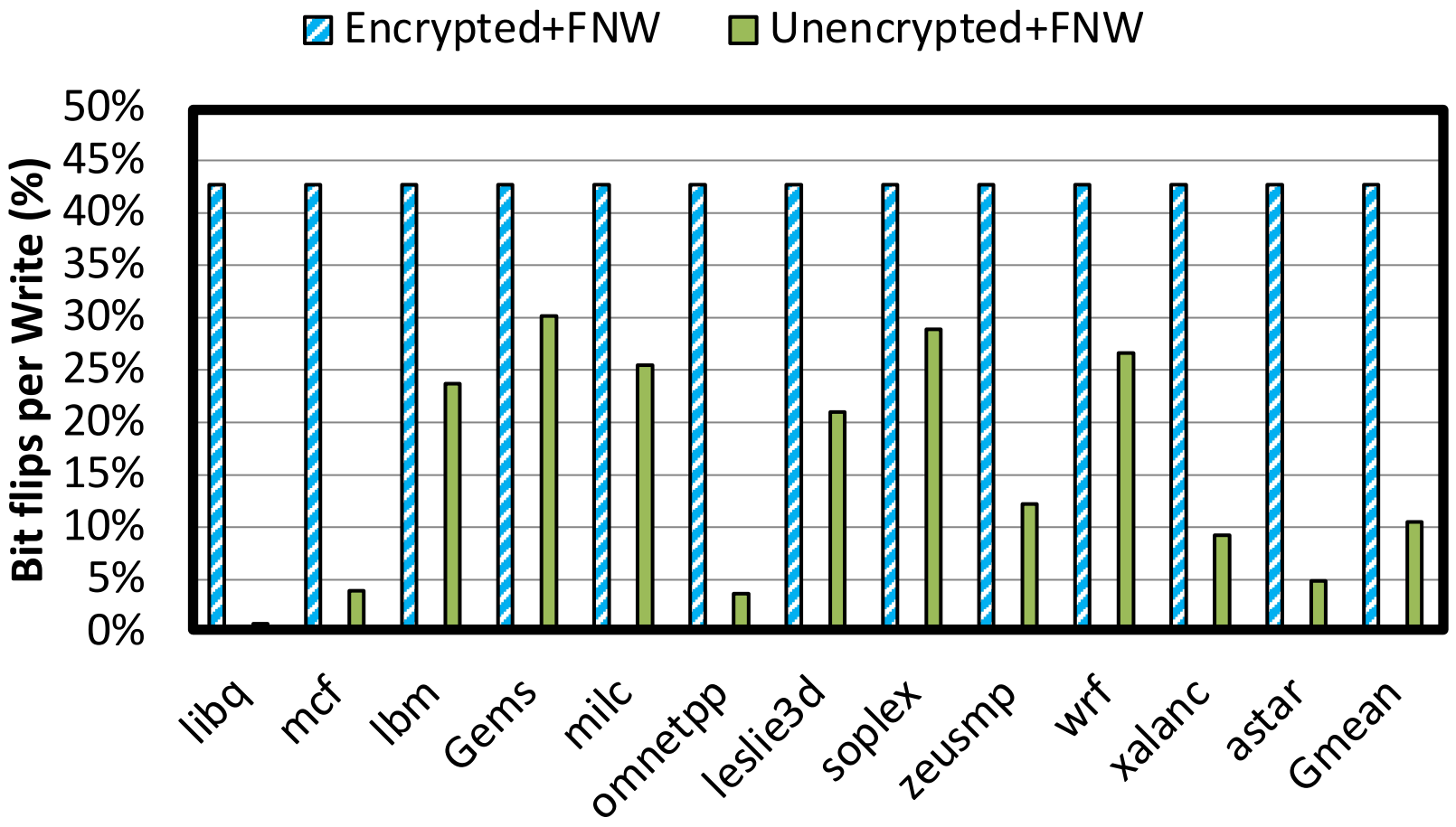
Security bounds no worse than CBC (NIST-approved)

Weakness to input is accepted to due to the underlying block cipher and not the mode of operation

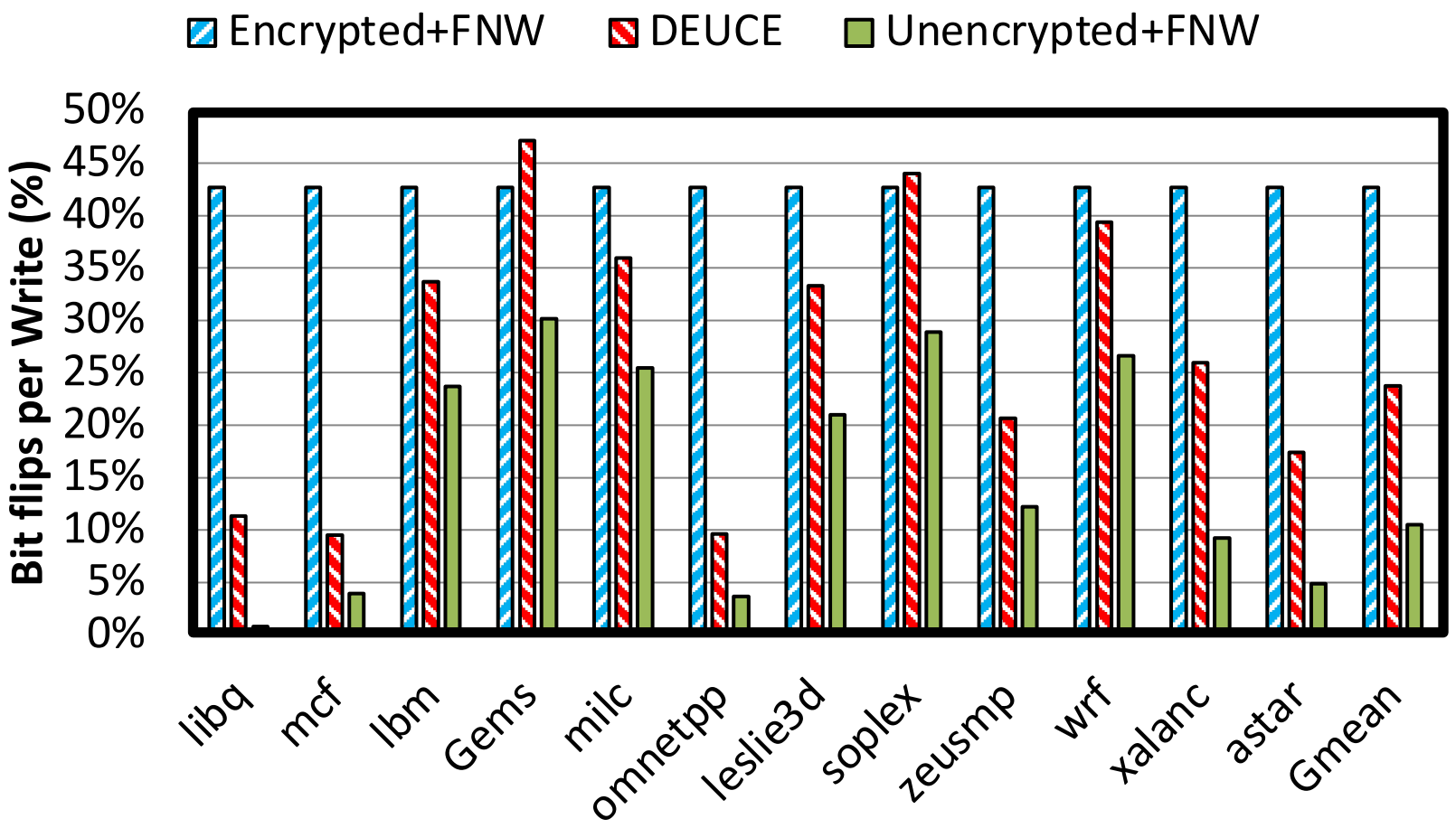
RESULTS: BIT FLIP ANALYSIS



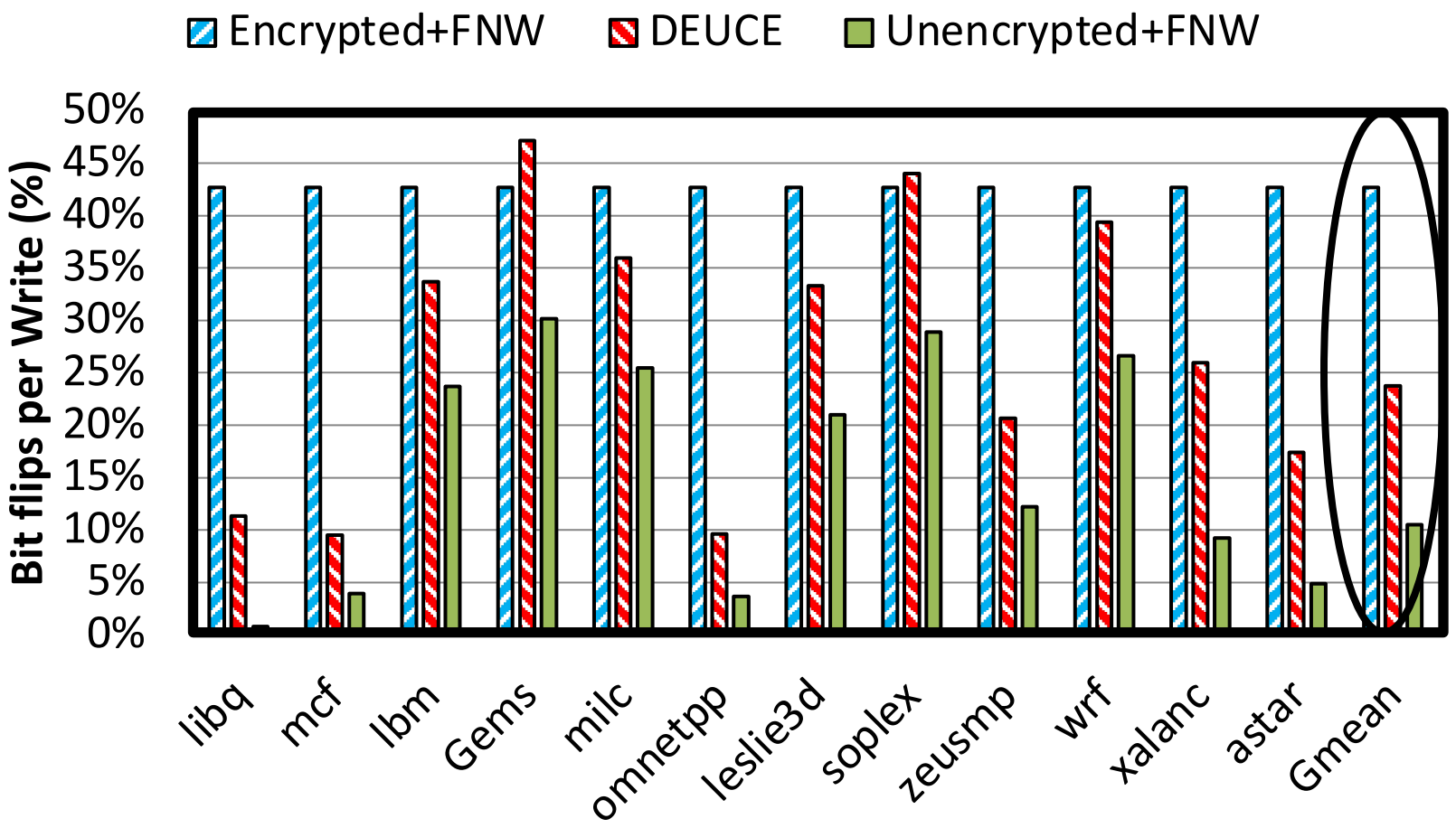
RESULTS: BIT FLIP ANALYSIS



RESULTS: BIT FLIP ANALYSIS

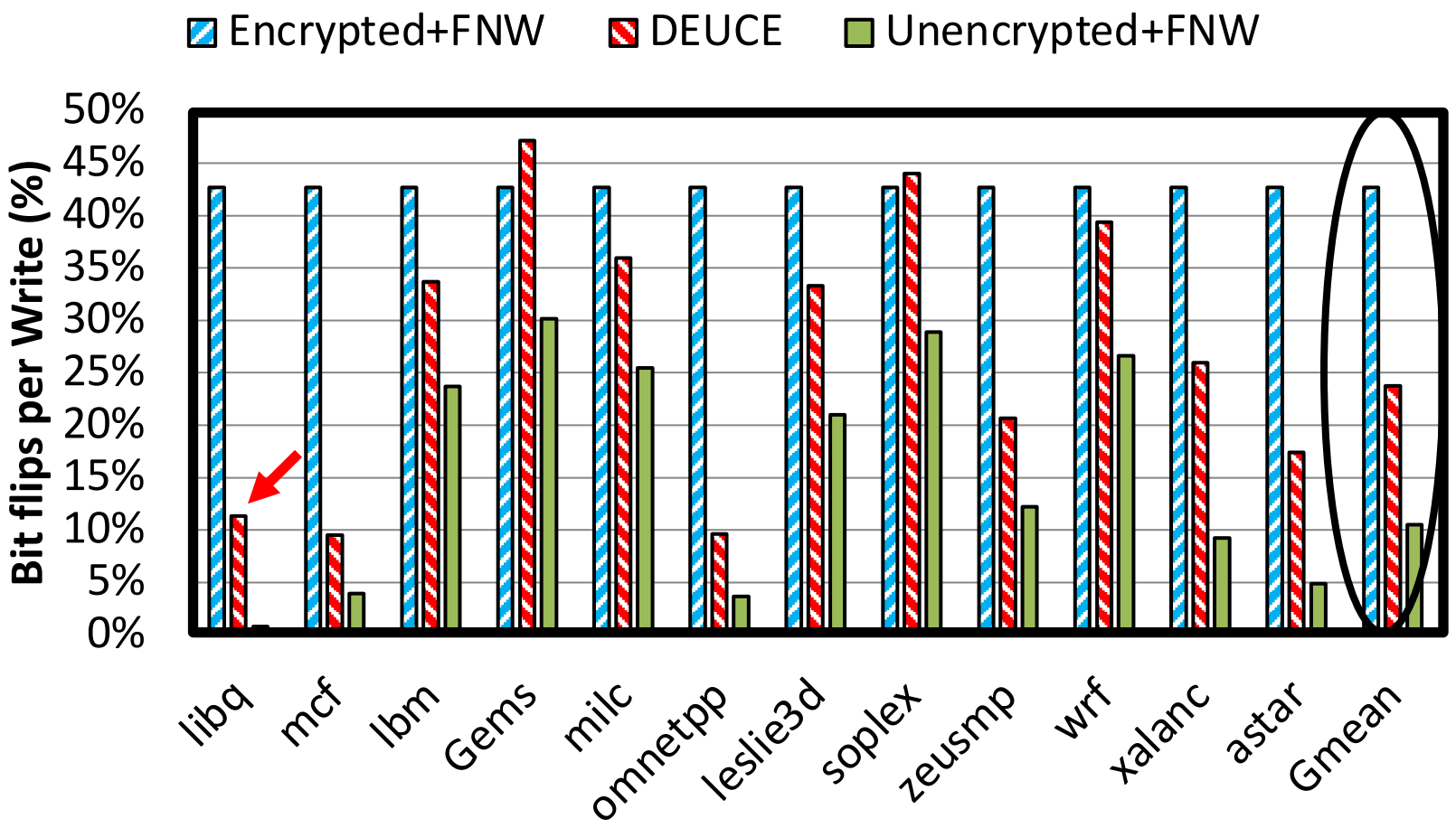


RESULTS: BIT FLIP ANALYSIS



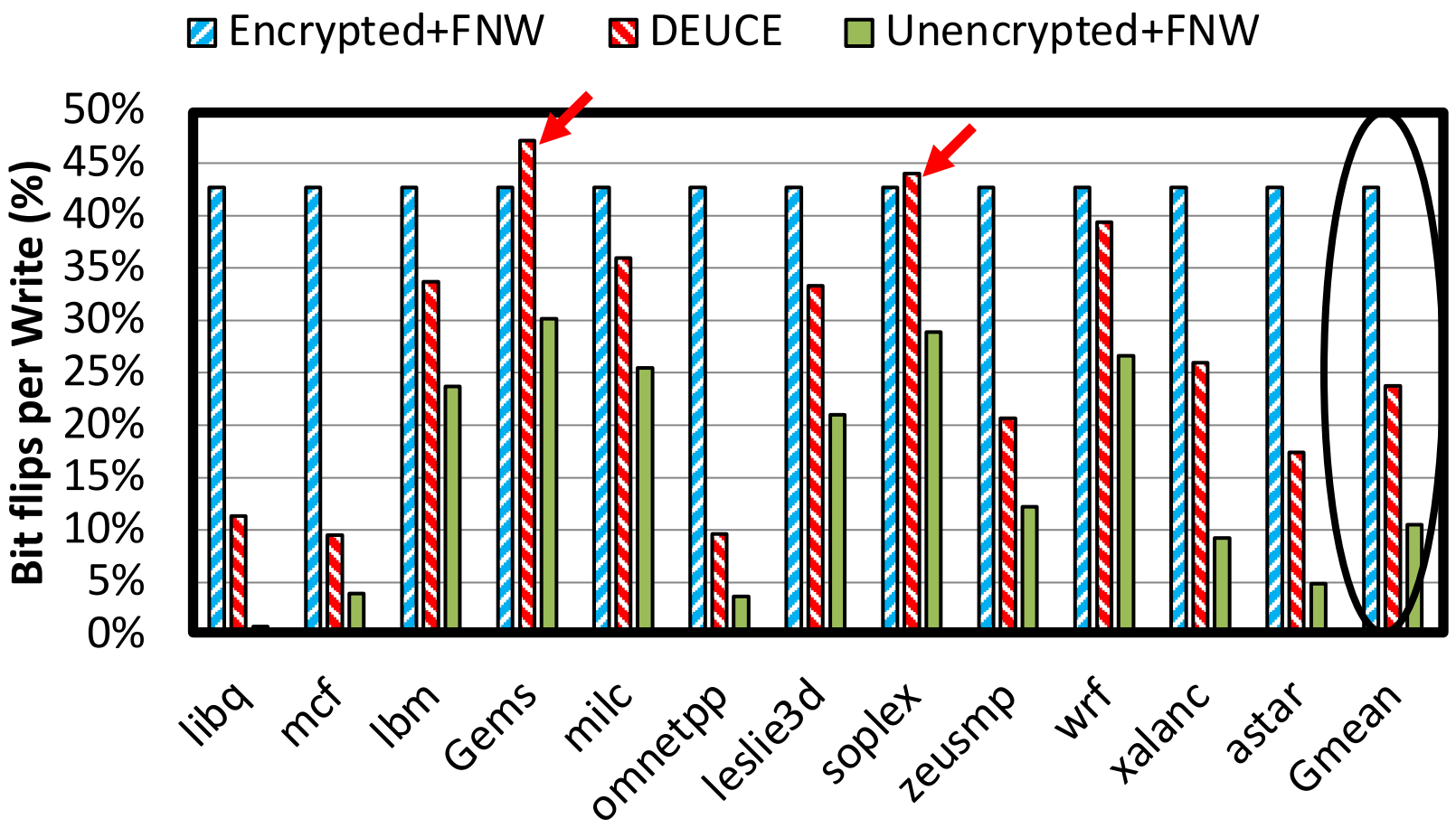
DEUCE eliminates two-thirds of the extra bit flips caused by encryption

RESULTS: BIT FLIP ANALYSIS



DEUCE eliminates two-thirds of the extra bit flips caused by encryption

RESULTS: BIT FLIP ANALYSIS



DEUCE eliminates two-thirds of the extra bit flips caused by encryption