# ACACES 2018 Summer School

# GPU Architectures: Basic to Advanced Concepts

**Adwait Jog, Assistant Professor**

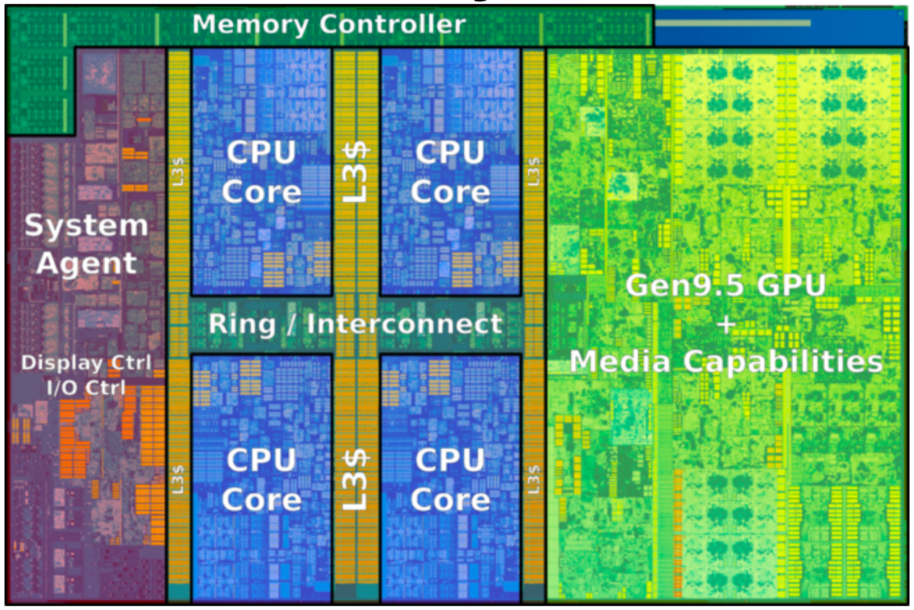
College of William & Mary
(http://adwaitjog.github.io/)

## Course Outline

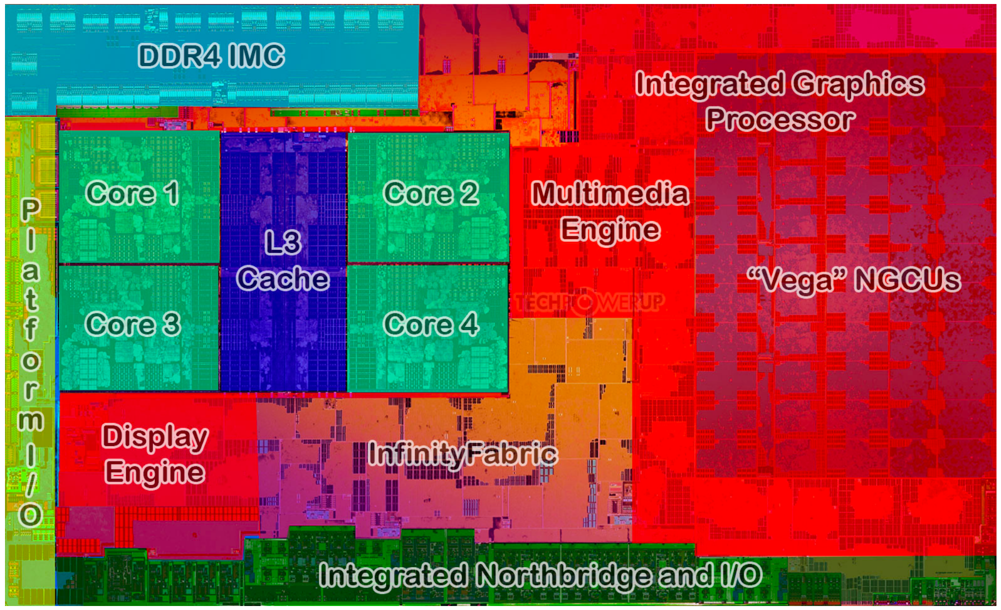
- Lectures 1 and 2: Basics Concepts
  - Basics of GPU Programming
  - Basics of GPU Architecture
- Lecture 3: GPU Performance Bottlenecks
  - Memory Bottlenecks
  - Compute Bottlenecks
  - Possible Software and Hardware Solutions
- Lecture 4: GPU Security Concerns
  - Timing channels
  - Possible Software and Hardware Solutions

# Era of Heterogeneous Architectures

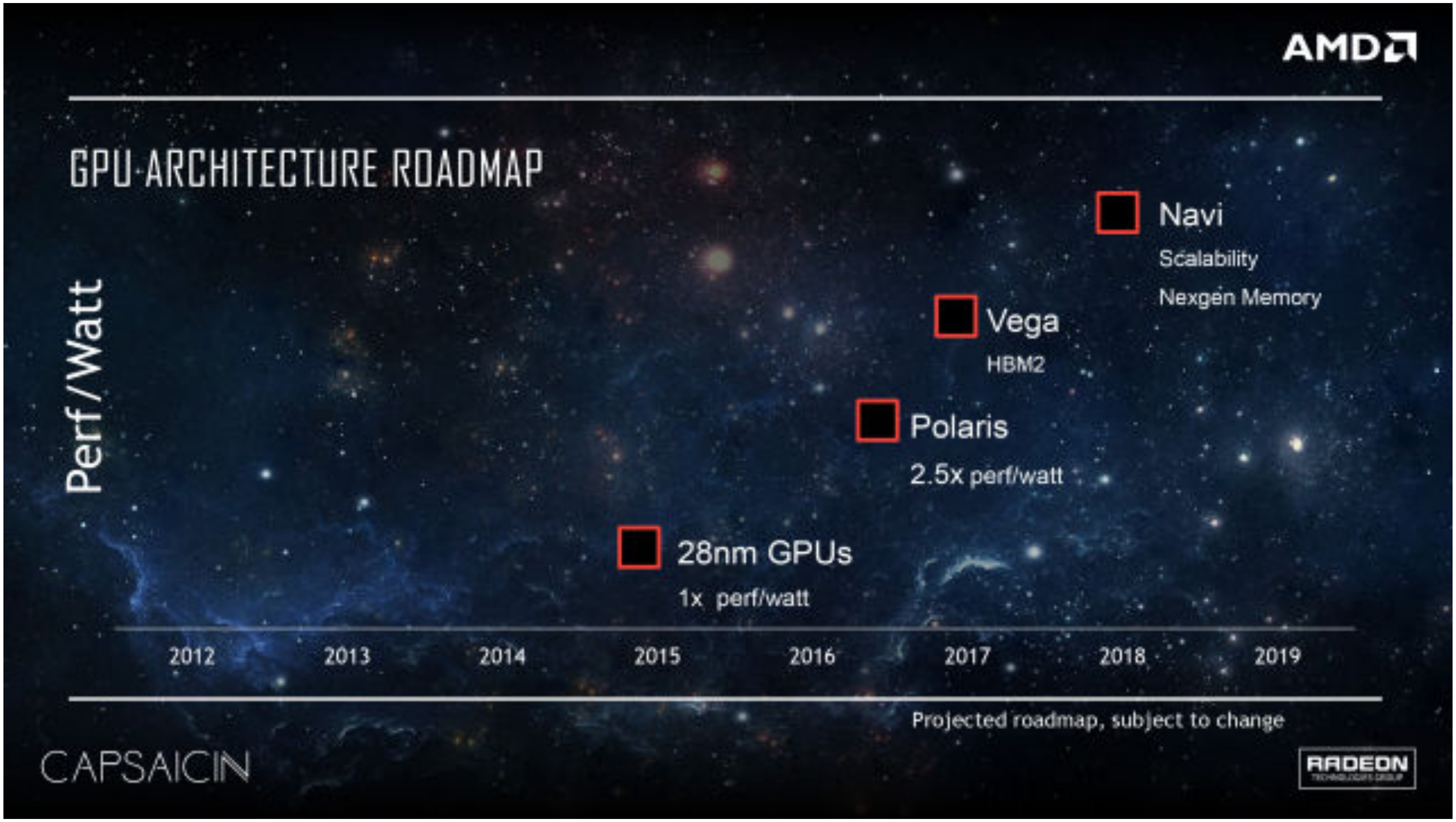
**Intel Coffee Lake and Kaby Lake**



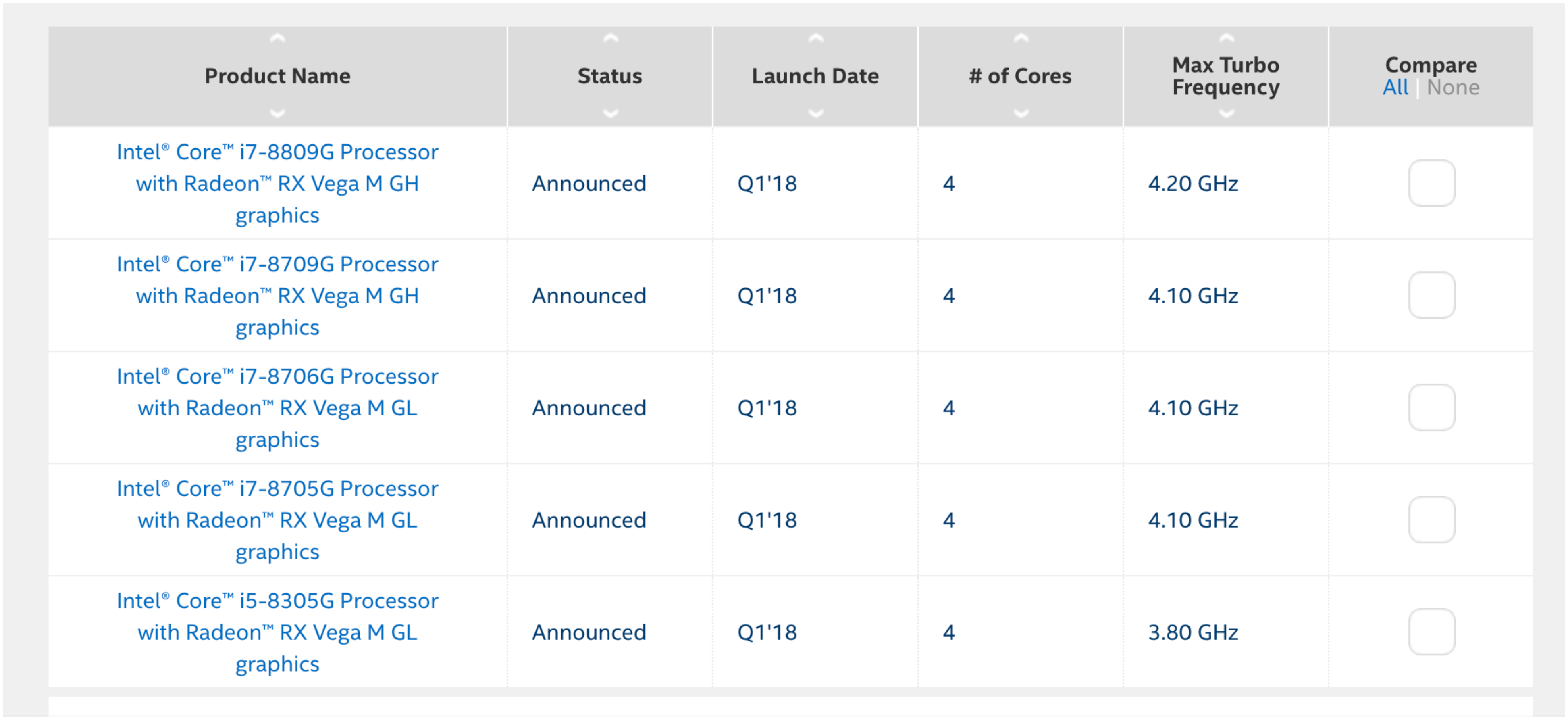
**AMD Raven Ridge**



# Discrete GPUs



# Discrete GPUs + Intel Processors



| Product Name | Status | Launch Date | # of Cores | Max Turbo Frequency | Compare All \| None |
|---|---|---|---|---|---|
| Intel® Core™ i7-8809G Processor with Radeon™ RX Vega M GH graphics | Announced | Q1'18 | 4 | 4.20 GHz | ☐ |
| Intel® Core™ i7-8709G Processor with Radeon™ RX Vega M GH graphics | Announced | Q1'18 | 4 | 4.10 GHz | ☐ |
| Intel® Core™ i7-8706G Processor with Radeon™ RX Vega M GL graphics | Announced | Q1'18 | 4 | 4.10 GHz | ☐ |
| Intel® Core™ i7-8705G Processor with Radeon™ RX Vega M GL graphics | Announced | Q1'18 | 4 | 4.10 GHz | ☐ |
| Intel® Core™ i5-8305G Processor with Radeon™ RX Vega M GL graphics | Announced | Q1'18 | 4 | 3.80 GHz | ☐ |

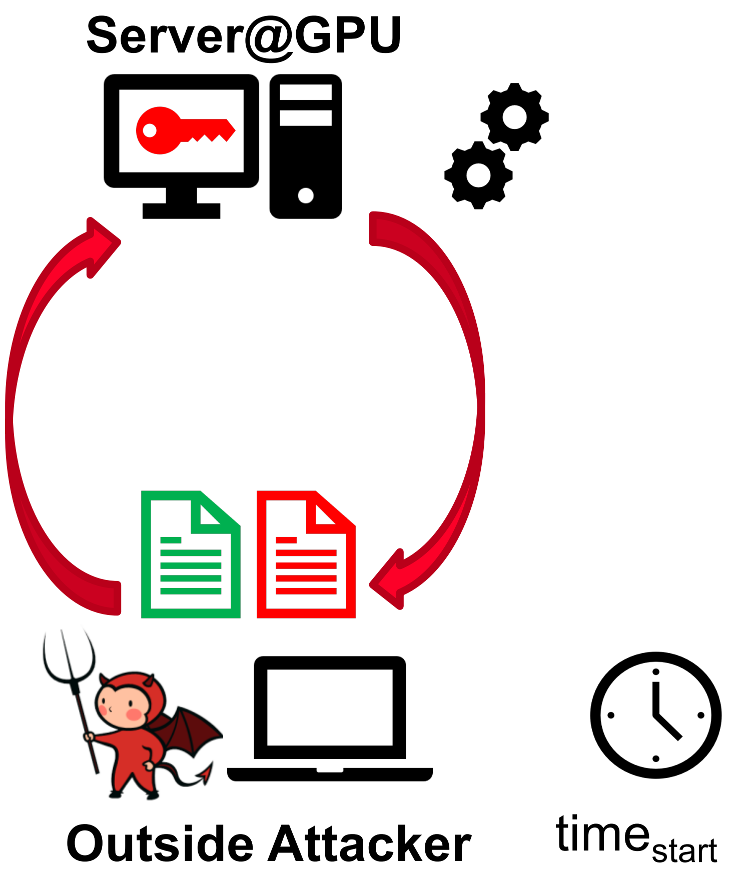
## Security Concerns

- GPUs may be accelerating applications that are using user-sensitive data (e.g., genomics, financial)
- GPUs may be accelerating cryptographic applications (e.g., AES, RSA etc.) and authentication algorithms on-behalf of CPUs
- *Given the popularity of GPUs, it is imperative to keep GPUs secure against a variety of side-channel attacks and other security vulnerabilities.*

## Security Attacks

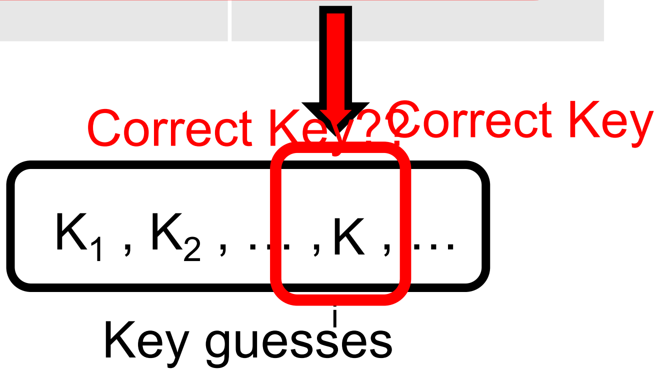
- User's web activity on GPU can be tracked by the malicious attacker who is co-located on the same card [Oakland'14]
- AES private keys can be recovered by correlation timing attacks [HPCA'16]
- Accelerating attacks via GPUs [Oakland'18]
  - Glitch: Accelerating row hammer attacks

# Correlation Timing Attacks

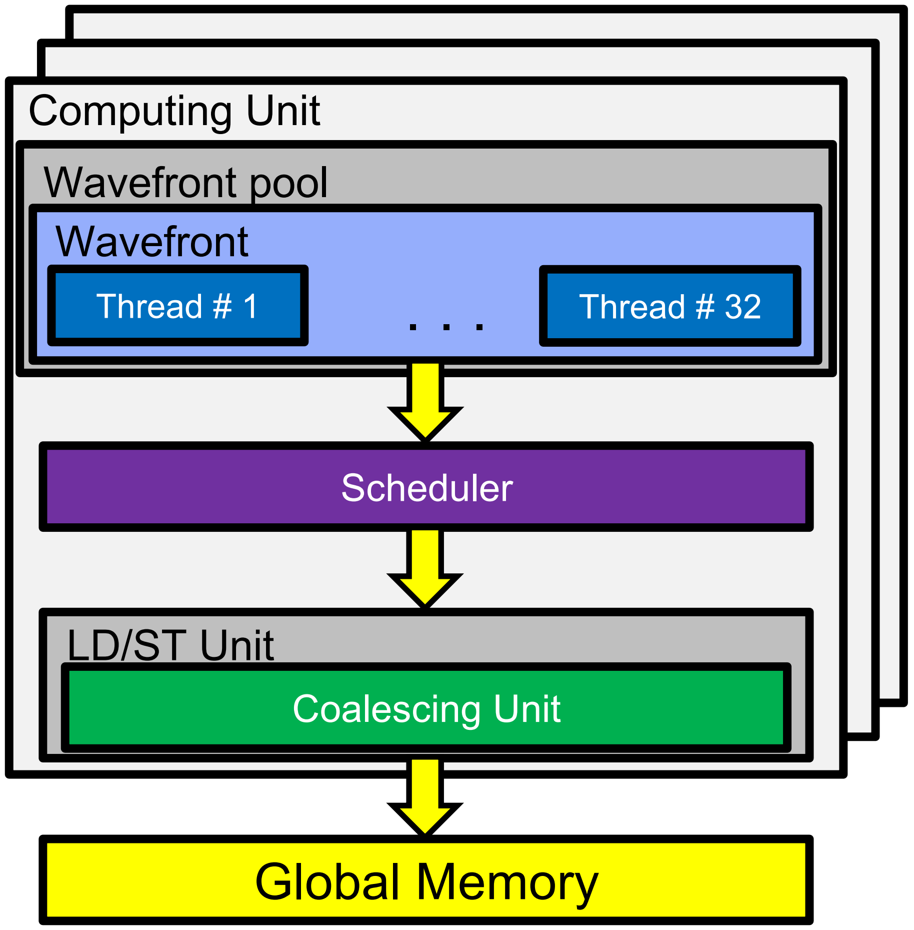


| Plaintexts | Ciphertexts | Time duration |
|---|---|---|
| Plaintext # 2 | Ciphertext # 2 | $time_2$ |
| Plaintext # 3 | Ciphertext # 3 | $time_3$ |
| … | … | … |
| | | |

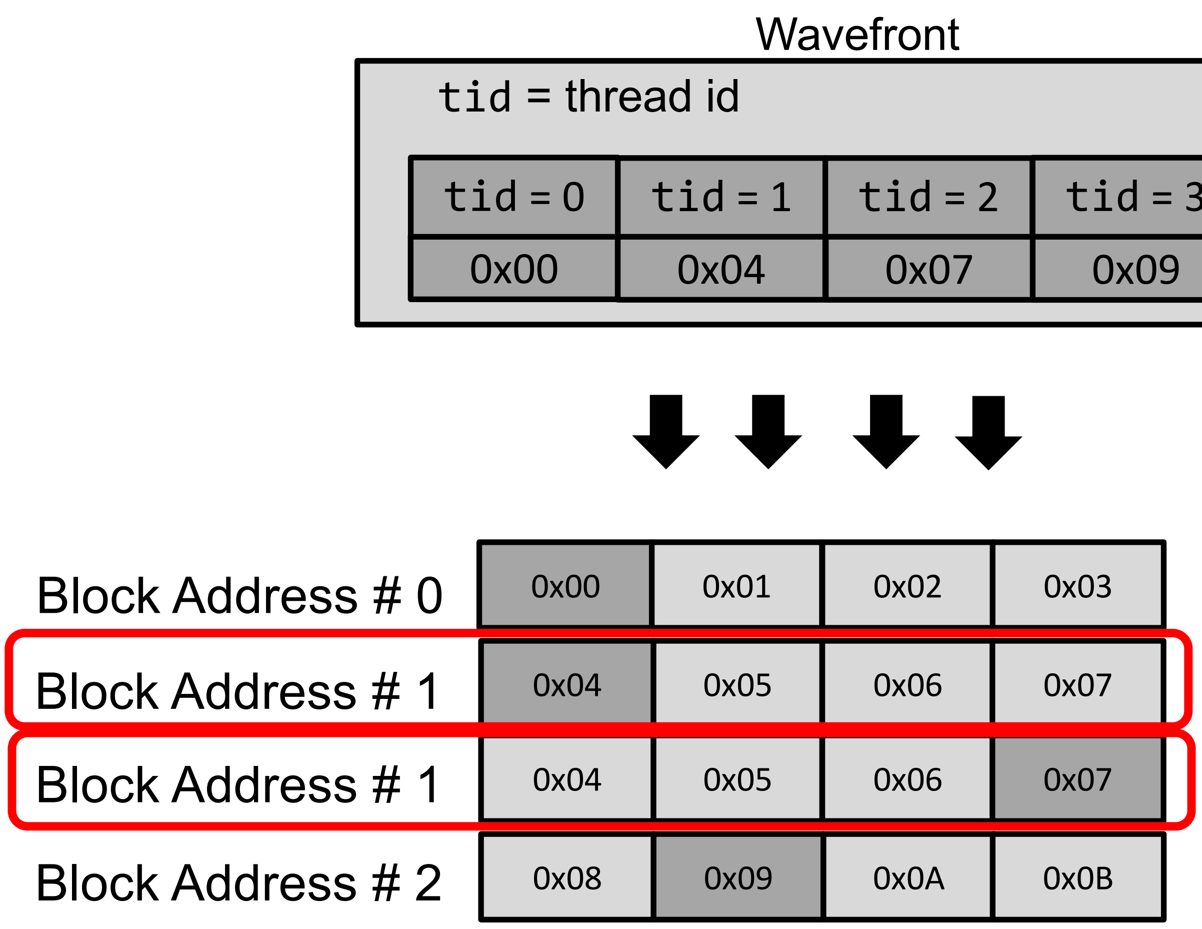
$time_{start}$ - $time_{stop}$ = $time_1$



# Memory Access Coalescing in GPUs

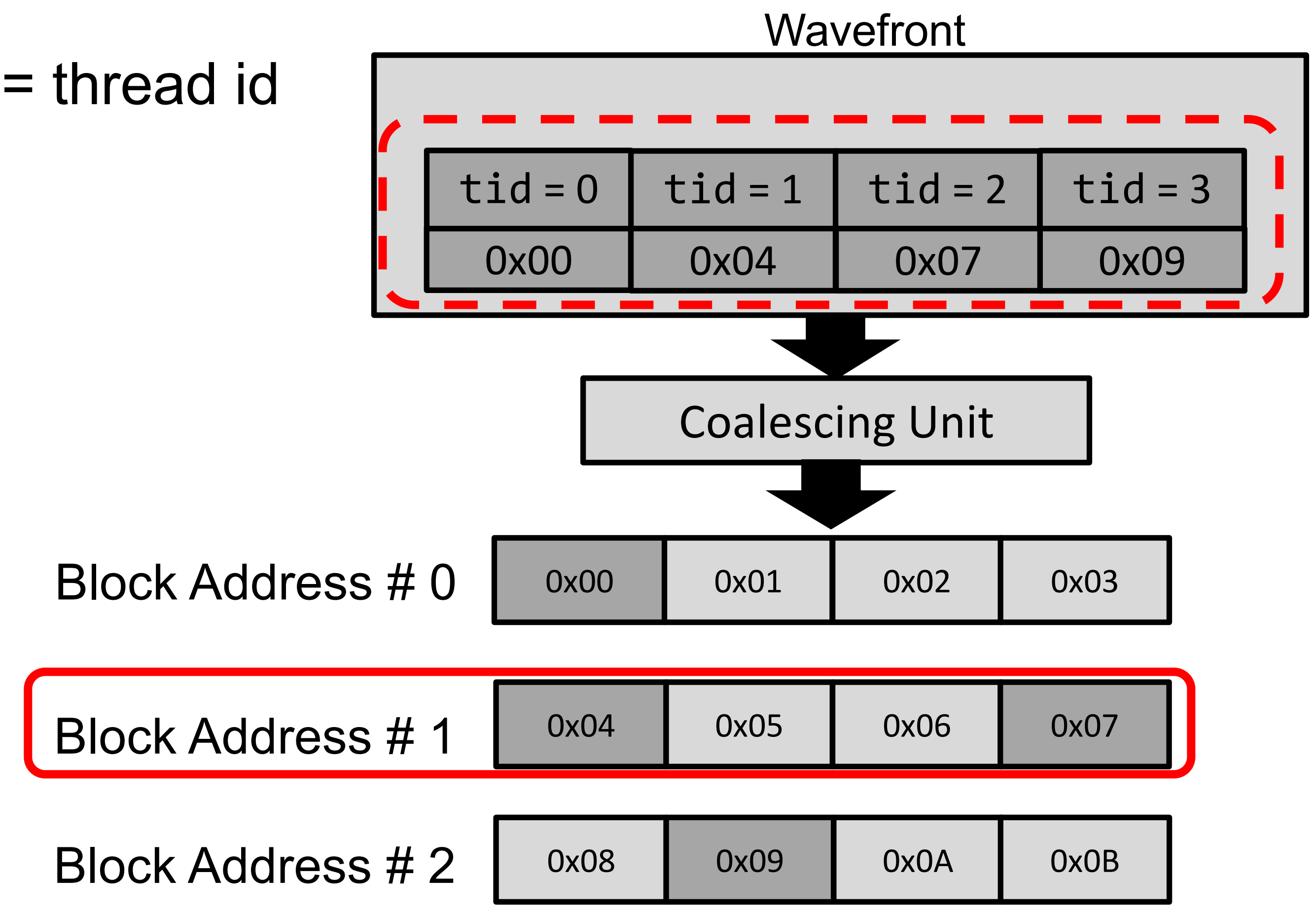


## Memory Access Coalescing in GPUs



# Memory Access Coalescing in GPUs

tid = thread id



## AES implementation on GPU

- Symmetric Encryption with 128-bit key and 10 rounds.

- S-box implementation involves table lookups.

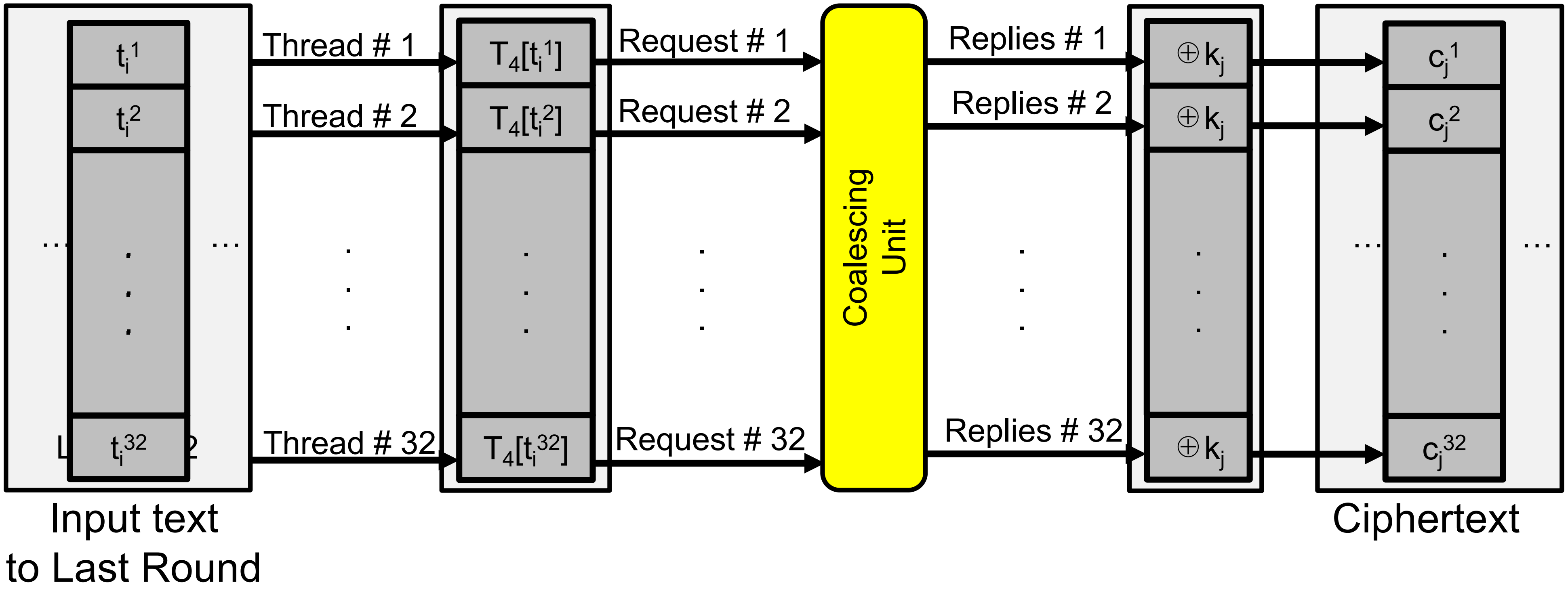
- [Jiang/Fei/Kaeli, HPCA'16] demonstrated that the last round is vulnerable.

## Last Round of AES on GPU

$$\boxed{c_j^{tid}} = \boxed{T_4[t_i^{tid}]} \boxed{\oplus \; k_j}$$

## Last Round of AES on GPU

$$c_j^{tid} = \boxed{T_4[\boxed{t_i^{tid}}]} \boxed{\oplus \; k_j}$$



## Correlation Timing Attack on GPU

- Goal of the attack: Recover the AES Key (byte-by-byte)
- Last Round of AES is ***vulnerable***

$$c_j^{tid} = T_4[t_i^{tid}] \oplus k_j$$

- Last Round is ***invertible***

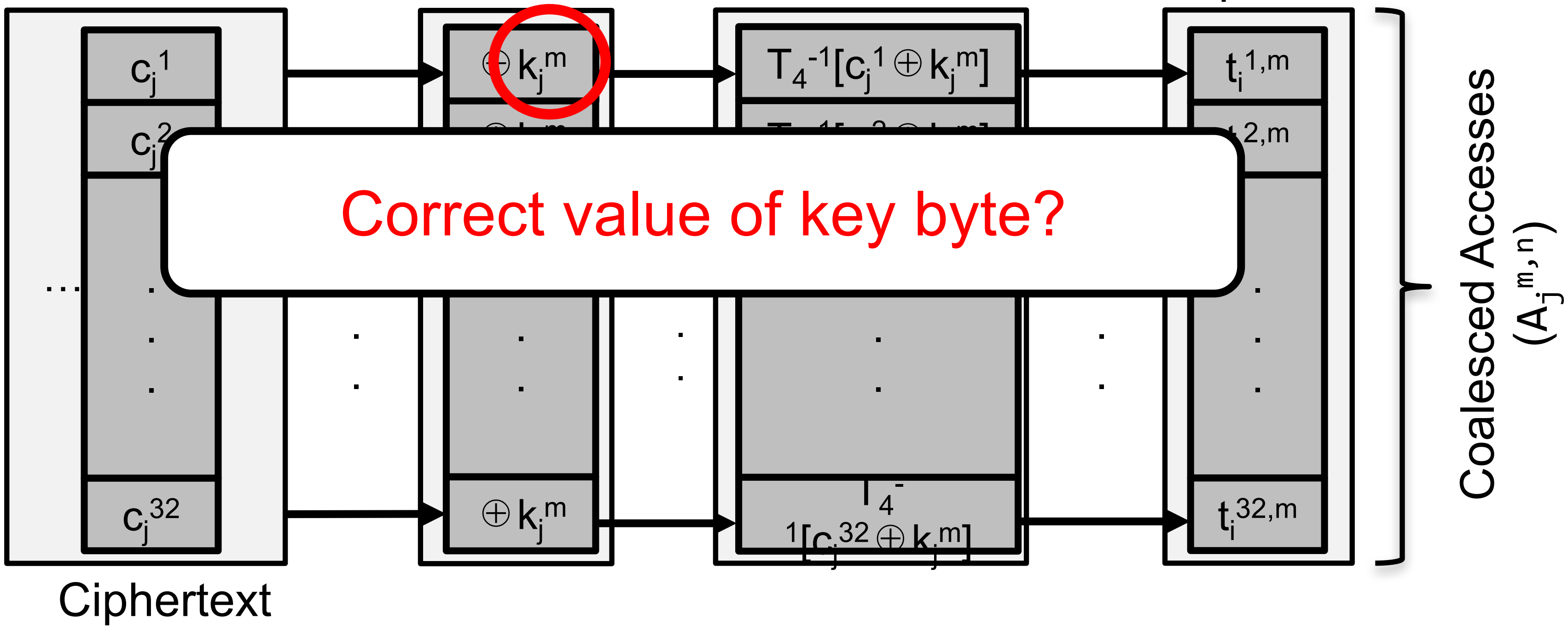
Memory access of thread *tid*

$$\boxed{t_i^{tid}} = T_4^{-1}[c_j^{tid} \oplus \boxed{k_j}]$$

How an attacker can calculate the number of coalesced accesses?

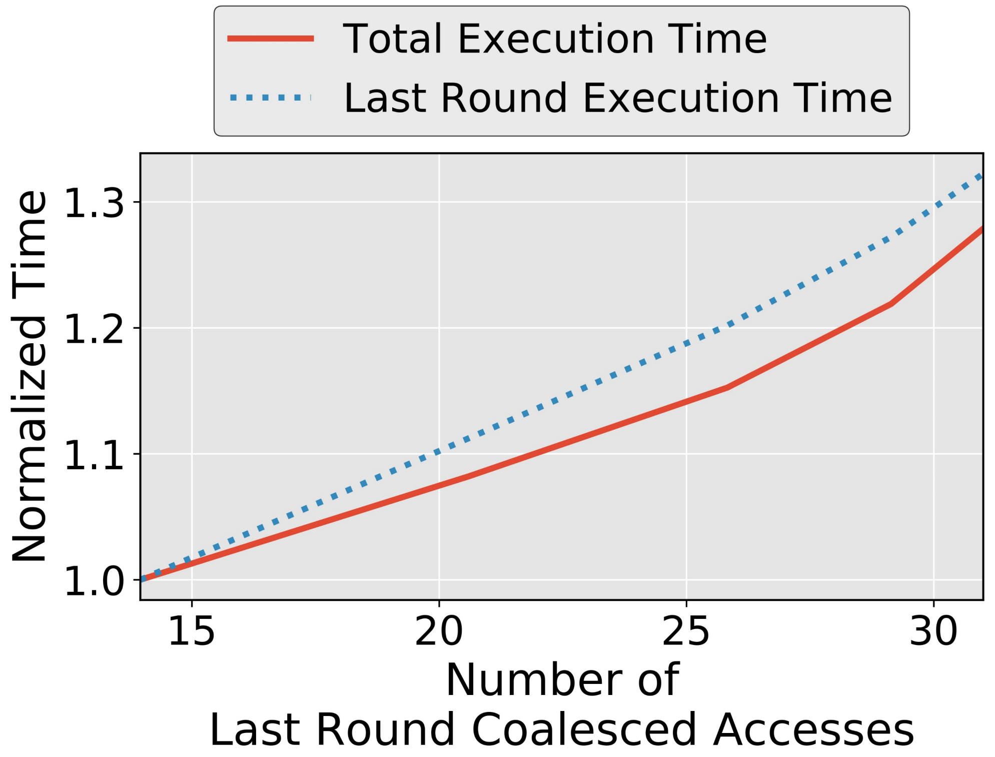
# Attacker calculates the # of coalesced accesses

$$t_i^{tid} = T_4^{-1}[c_j^{tid} \oplus k_j]$$



Correct value of key byte?

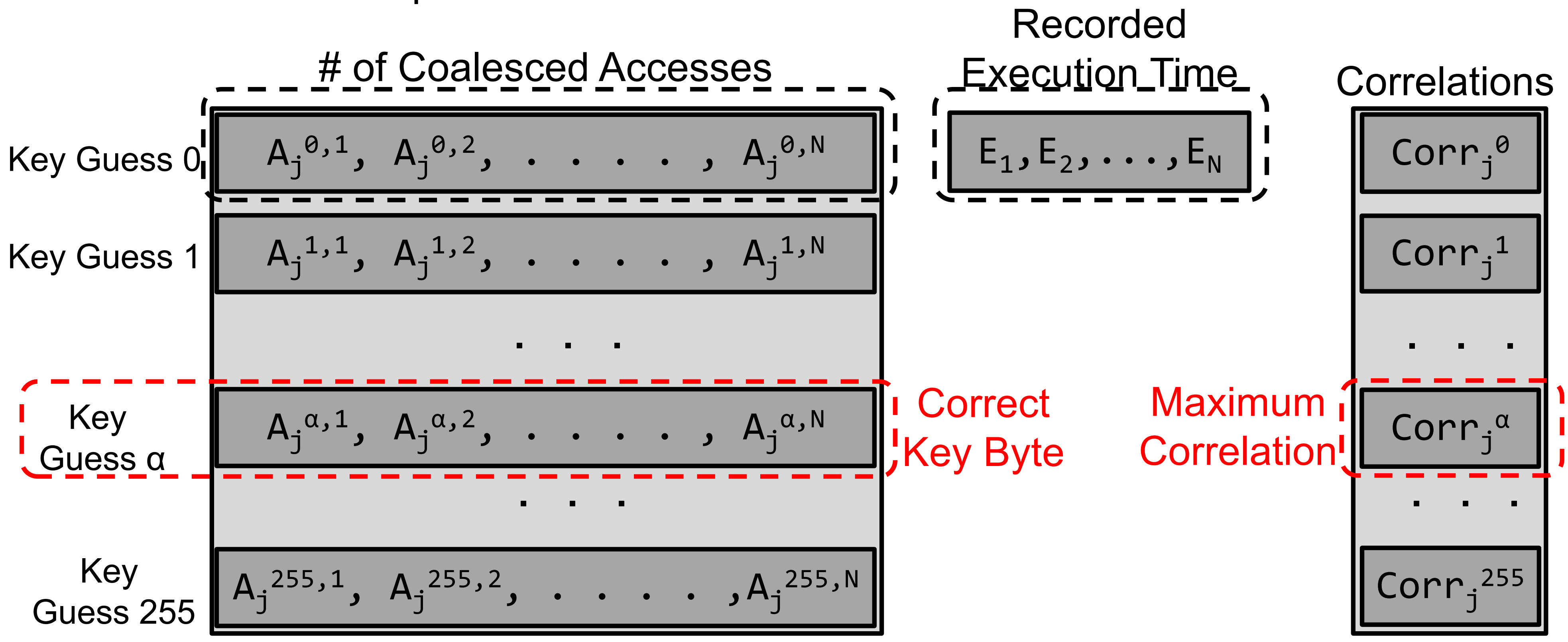
# Coalesced Accesses and Execution Time



Associate the number of coalesced accesses with execution time

## Finding the Correct Key Value

- Attacker encrypts ‘N’ number of plaintexts over server
  - Records Ciphertext and Execution time



## Simulating Timing Attack on our Set-up

How to mitigate Correlation Timing Attacks on GPU?

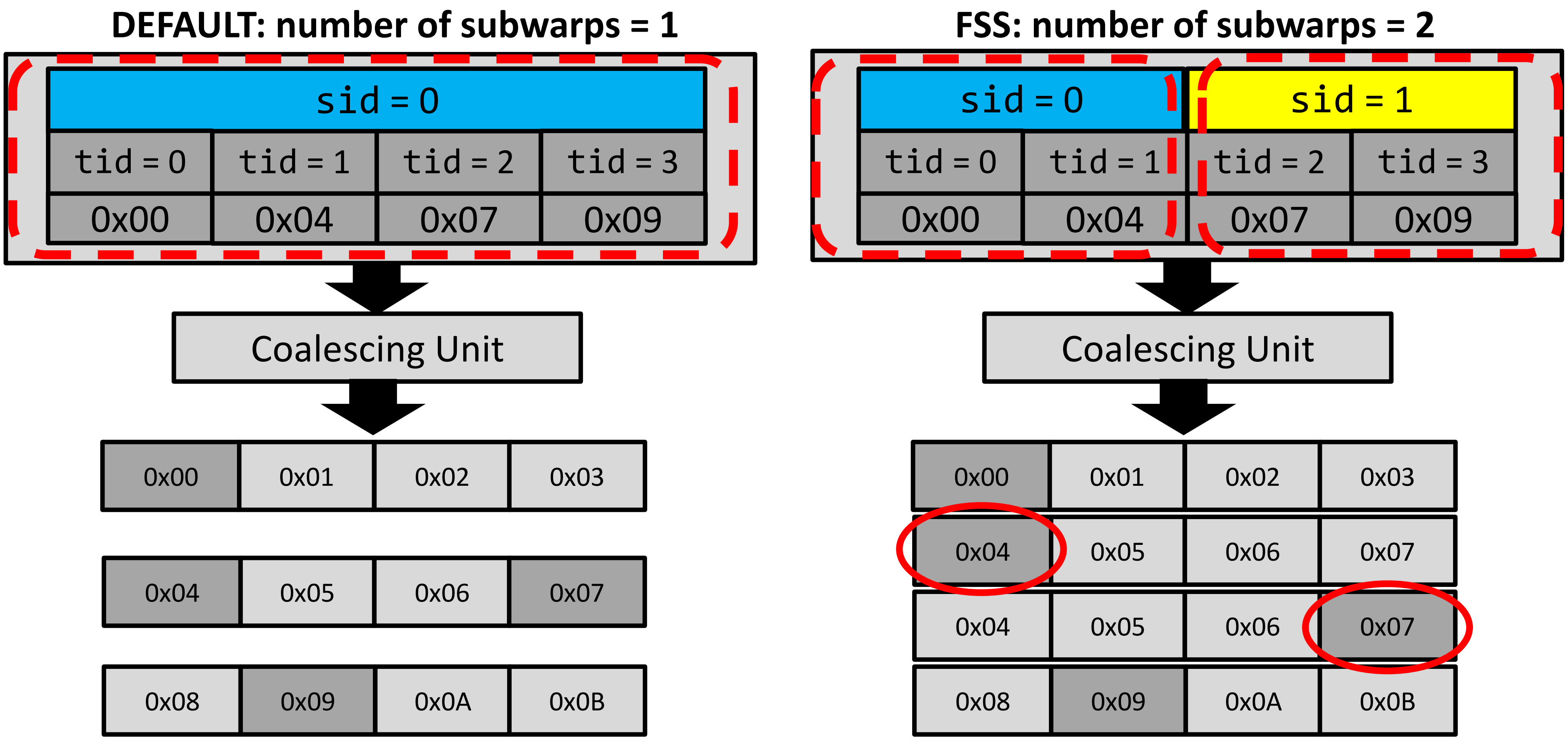
Answer: By making it harder for the attacker to correctly calculate the number of coalesced accesses

## Naïve Solution

**RCoal** to mitigate the correlation timing attacks

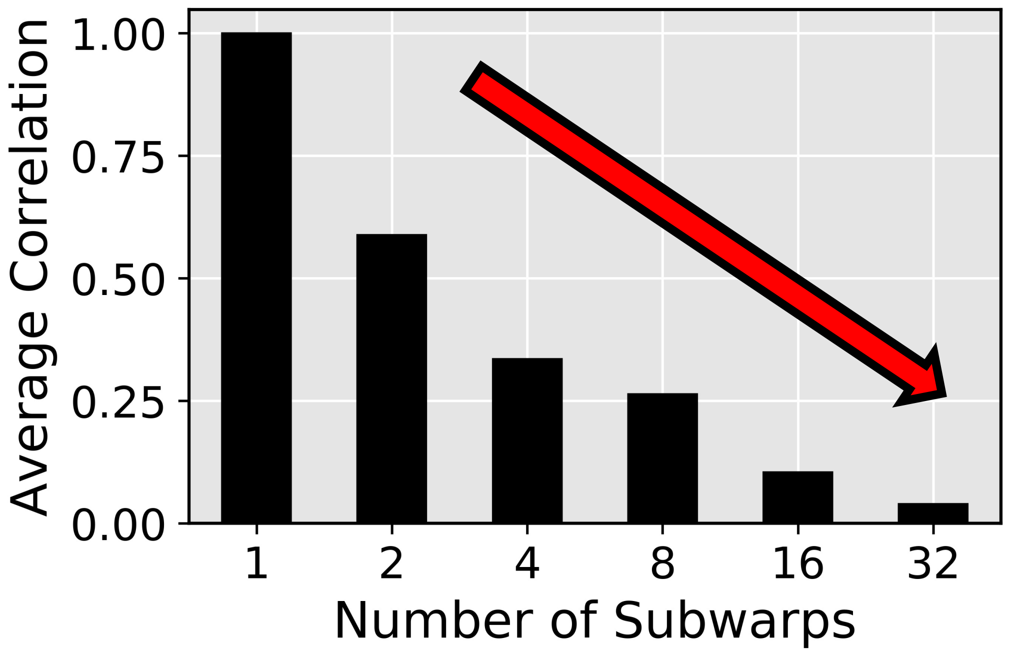
- Targets the ***deterministic*** nature of the coalescing mechanism
  - Fixed number of subwarps (or subwavefronts)
  - Fixed sizes of subwarp (or subwavefronts)
  - Deterministic mapping of the thread elements to subwarps (or subwavefronts)

# RCoal: Fixed Sized Subwarp (FSS)



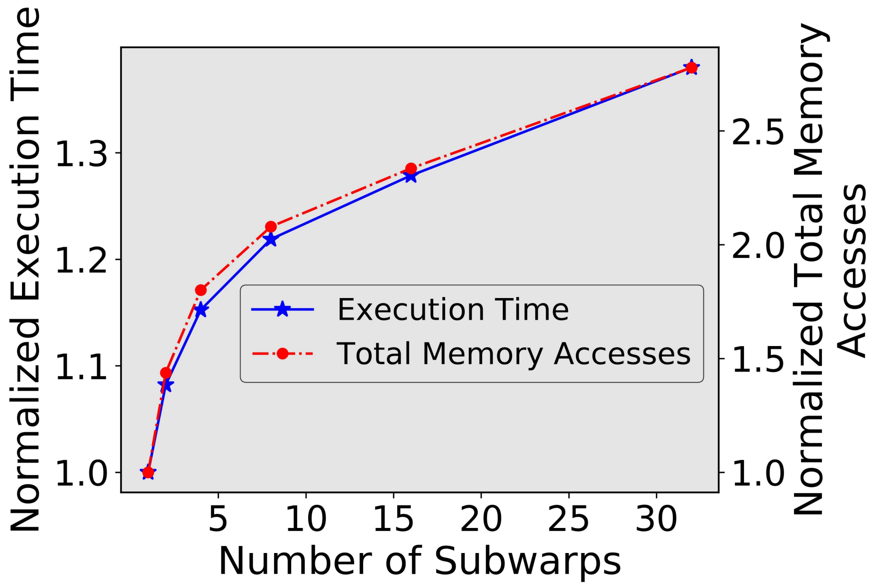
## FSS Security against Baseline Attack

- Correlation between the number of coalesced accesses and the execution time **drops**
- Correct key byte is **harder to find**
- **Improved** security



# FSS Performance

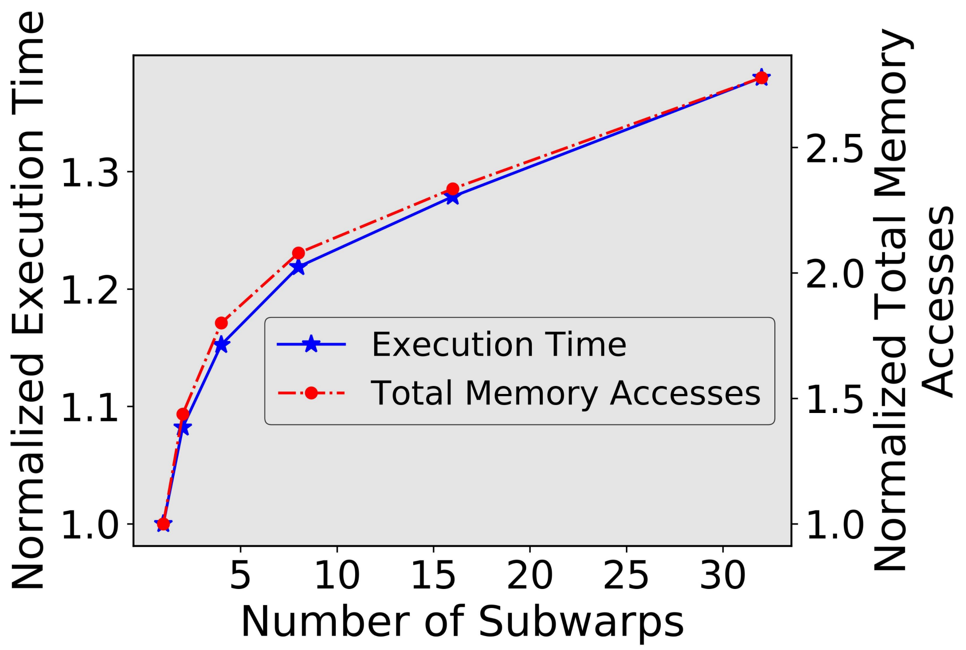
- Memory accesses **increase** with number of subwarps
- Execution time **increases** with number of subwarps
- Performance **degrades** as number of subwarp increase



Can attacker still recover the AES key?

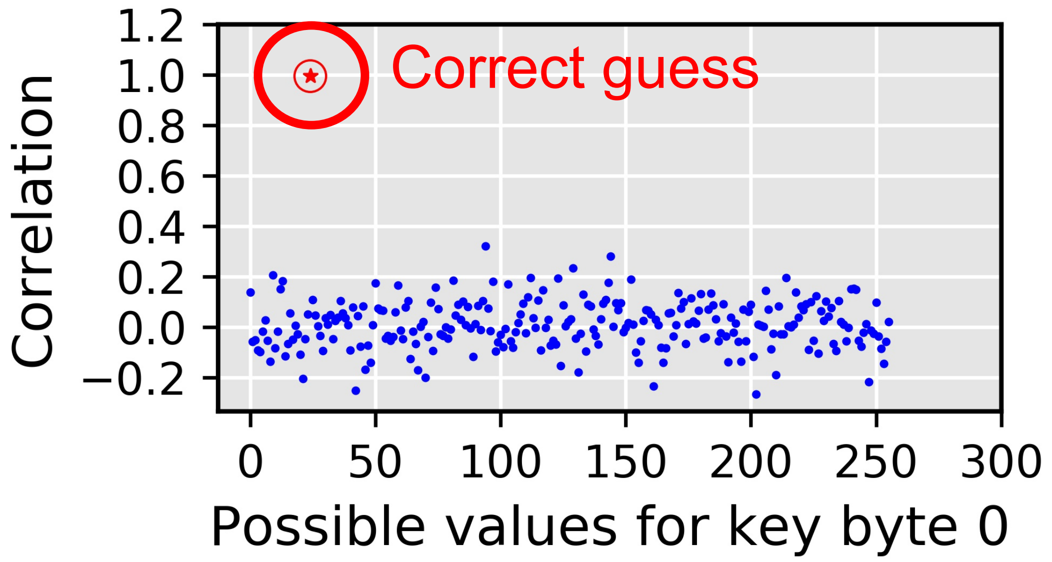
# FSS against FSS attack

- Attacker can figure out the number of subwarps



# FSS against FSS attack

- ❑ Attacker can figure out the number of subwarps
- ❑ Attacker can calculate per subwarp accesses



## FSS against FSS attack

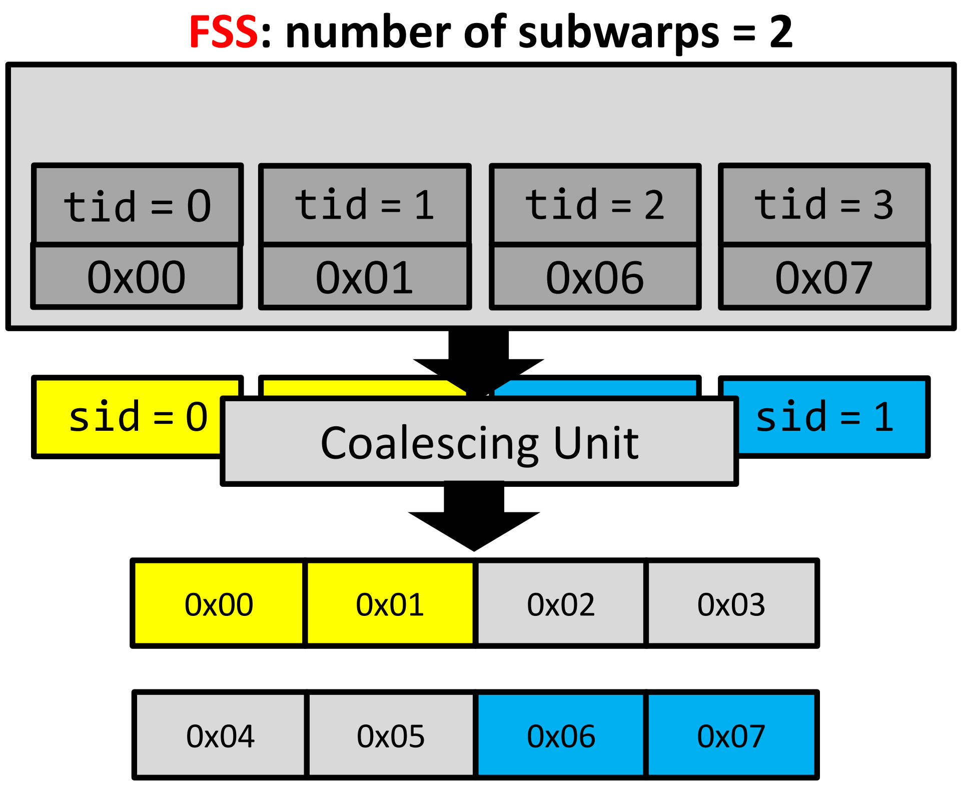
**RCoal** to mitigate the correlation timing attacks

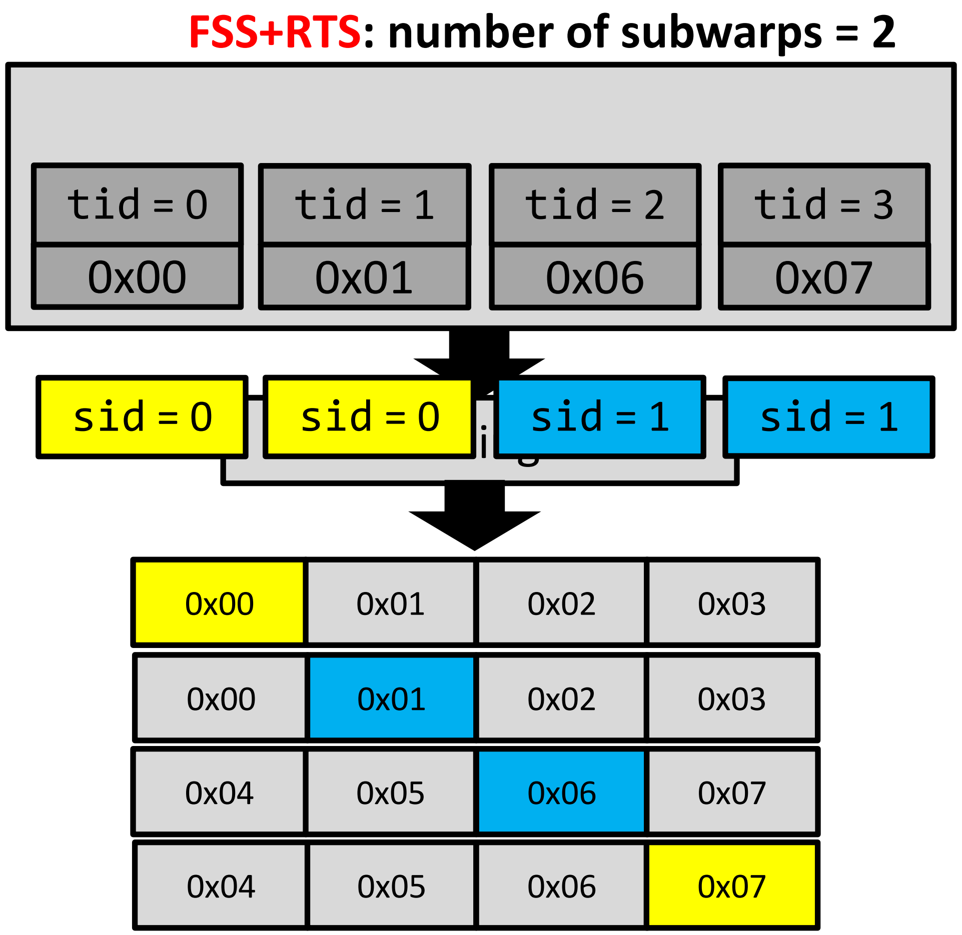
- Targets the ***deterministic*** nature of the coalescing mechanism
  - Fixed number of subwarps
  - Fixed sizes of subwarp
  - Deterministic mapping of the thread elements to subwarps

## **RCoal** to mitigate the correlation timing attacks

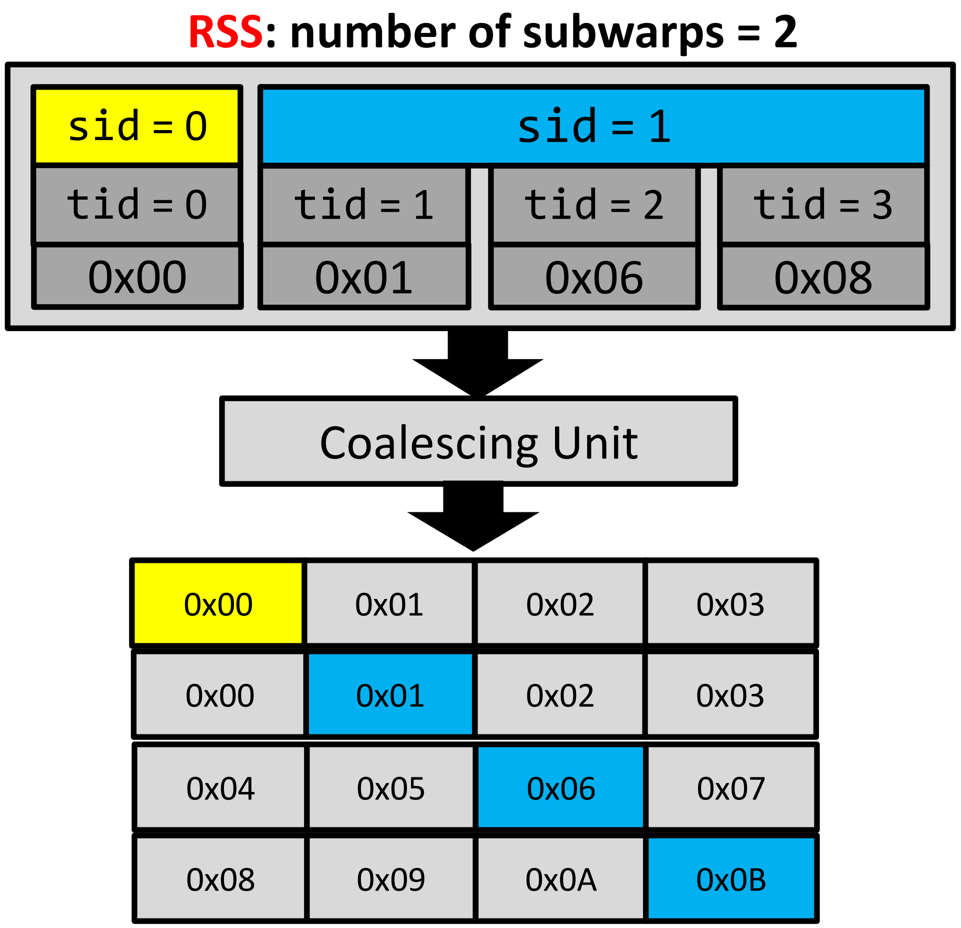
- Targets the ***deterministic*** nature of the coalescing mechanism
  - Fixed number of subwarps
  - Fixed sizes of subwarp
  - Deterministic mapping of the thread elements to subwarps

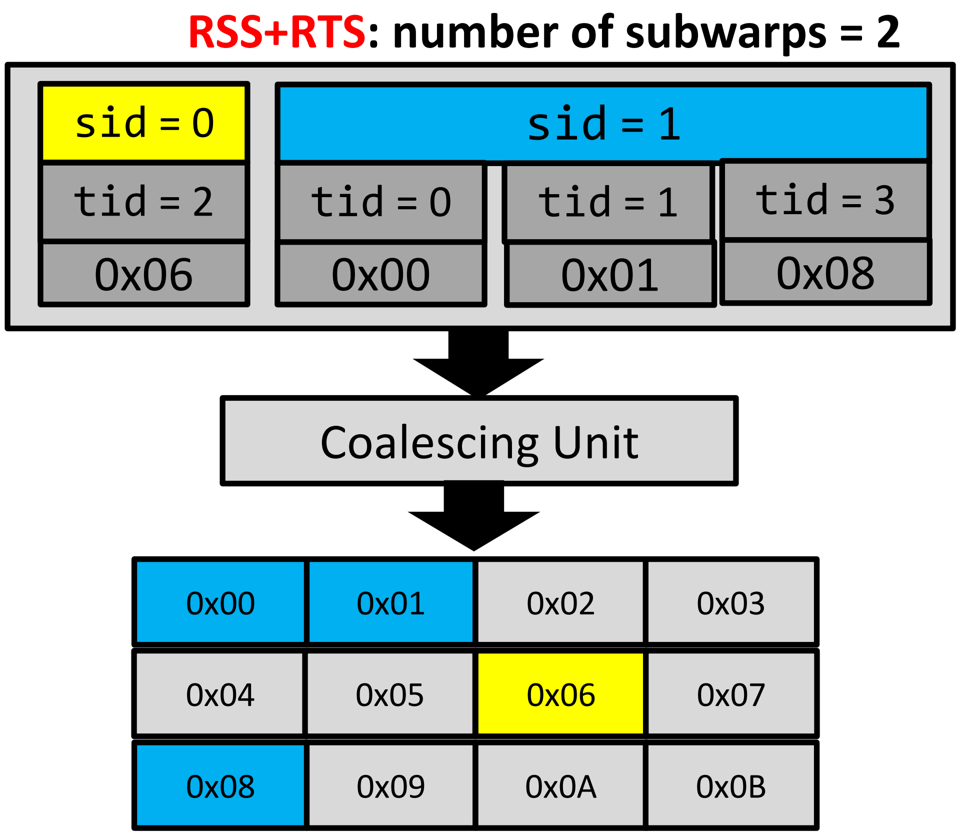
# RCoal: Random-Threaded Subwarp (RTS)





# RCoal: Random-Threaded Subwarp (RTS)



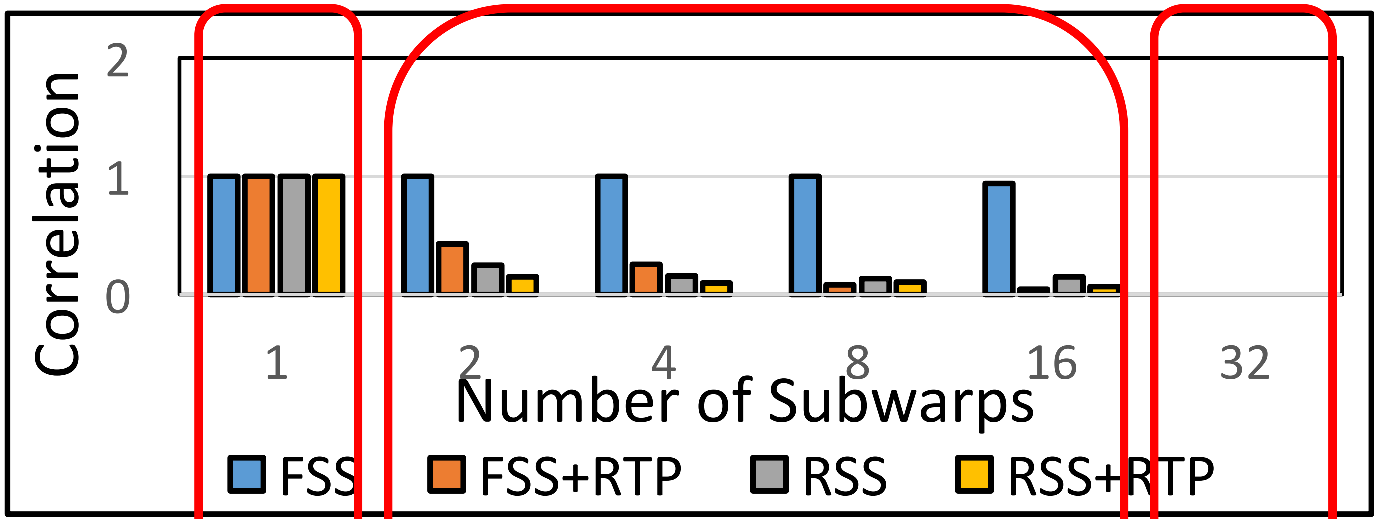


## Evaluation Set-up

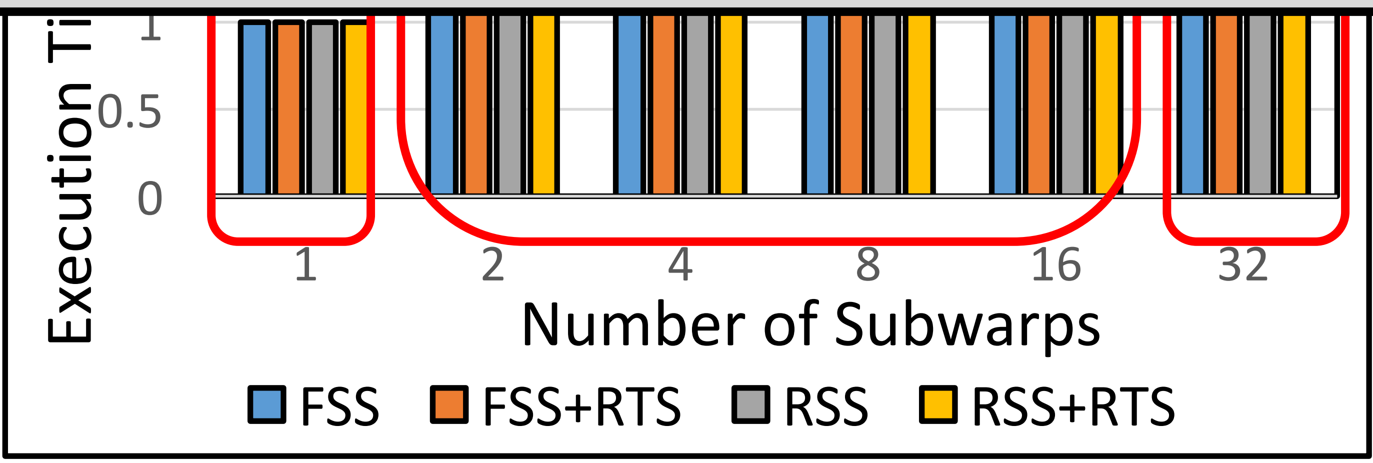
- AES-128
- Plaintext with 32 lines
- GPGPU-SIM
  - 15 SMs, 32 threads/warp, one subwarp per coalescing unit (base case)
  - GDDR5 Memory with 6 MCs, 16 DRAM-banks, 4 bank-groups/MC
- Enhanced Attack Algorithms
  - Corresponding Attacks

## Performance/Security Trade-off

Security
(Lower the better)



(Lower the better)



**Offers Security/Performance Trade-off**

## Conclusions

- We discussed RCoal, a set of three novel defense mechanisms
  - To mitigate the correlation timing attacks
  - Randomizes the memory access coalescing
  - Scales with the plaintext size (analysis in paper)
  - Theoretical analysis in the paper

- RCoal offers a trade-off between security and performance and improves security at a modest performance loss.

## Food for thought

- **Improving security at lower performance cost**
  - *Can we randomize logic at other parts of the memory hierarchy?*
    - GPU Cache Management
    - GPU Bandwidth Management (e.g., MSHRs)
    - GPU Prefetching and Memory Scheduling
  - *Can we leverage software-driven hints?*
    - Only randomize when “security-critical” sections of the code are executing
    - How do we identify “security-critical” sections? If yes, can we automate the process?

## References

- RCoal: Mitigating GPU Timing Attack via Subwarp-based Randomized Coalescing Techniques, HPCA’18
- A Complete Key Recovery Timing Attack on a GPU, HPCA’16
- Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU, Oakland’18