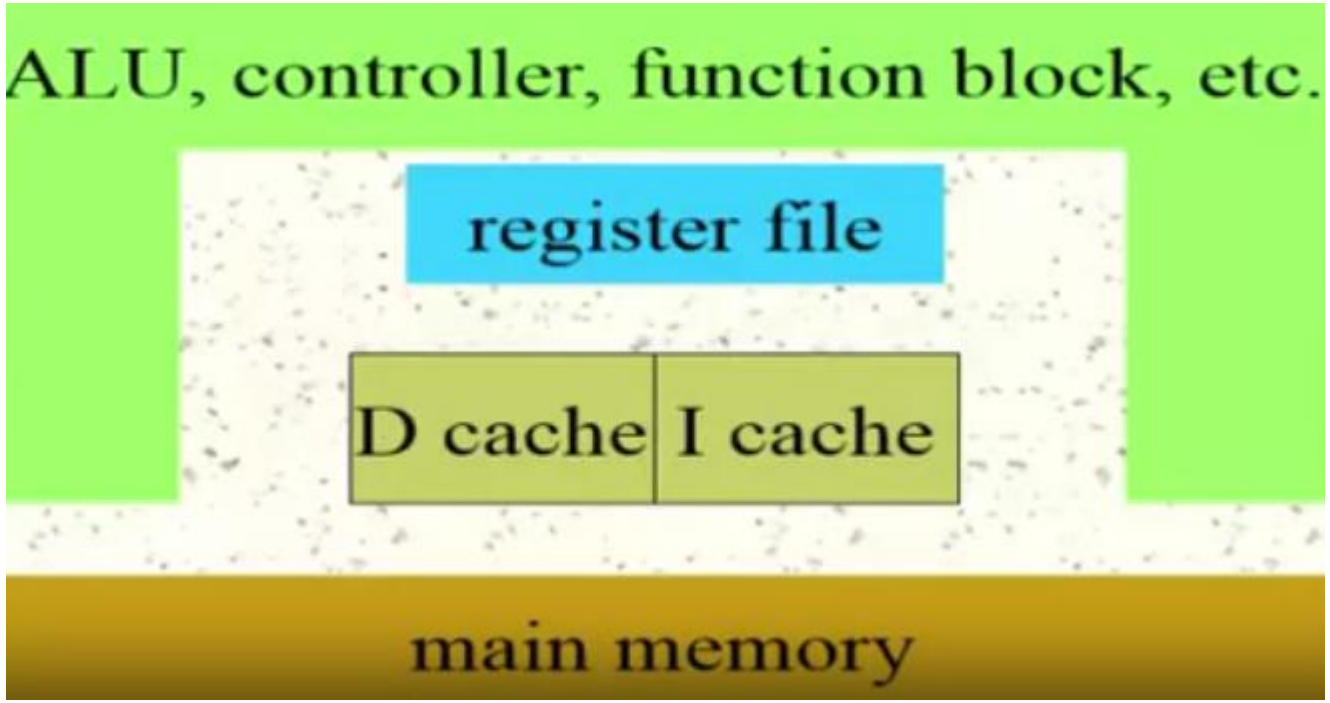
Memory Vulnerabilities and Cache Attacks

Execution and Vulnerabilities

Execution flow
Assume that secret data is stored in register file during execution

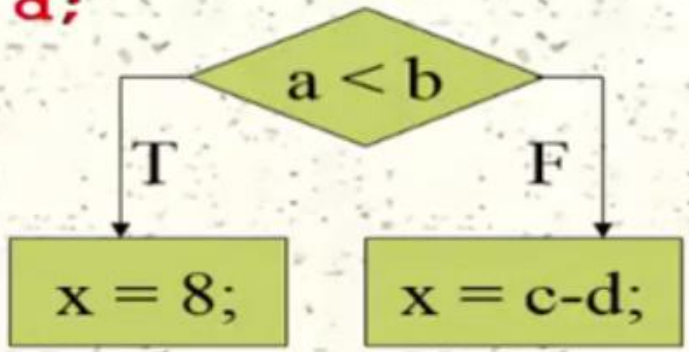
Vulnerabilities
- Memory load
- Memory store
- A & L Operations
- Control Flow



Example C → Assembly

C:

```
if (a < b) {x = 8;} else x = c - d;
```

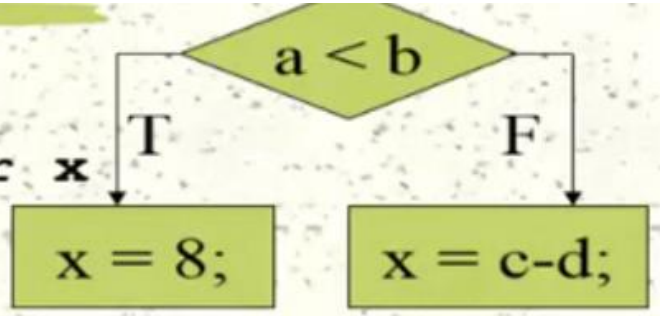


Assembly:

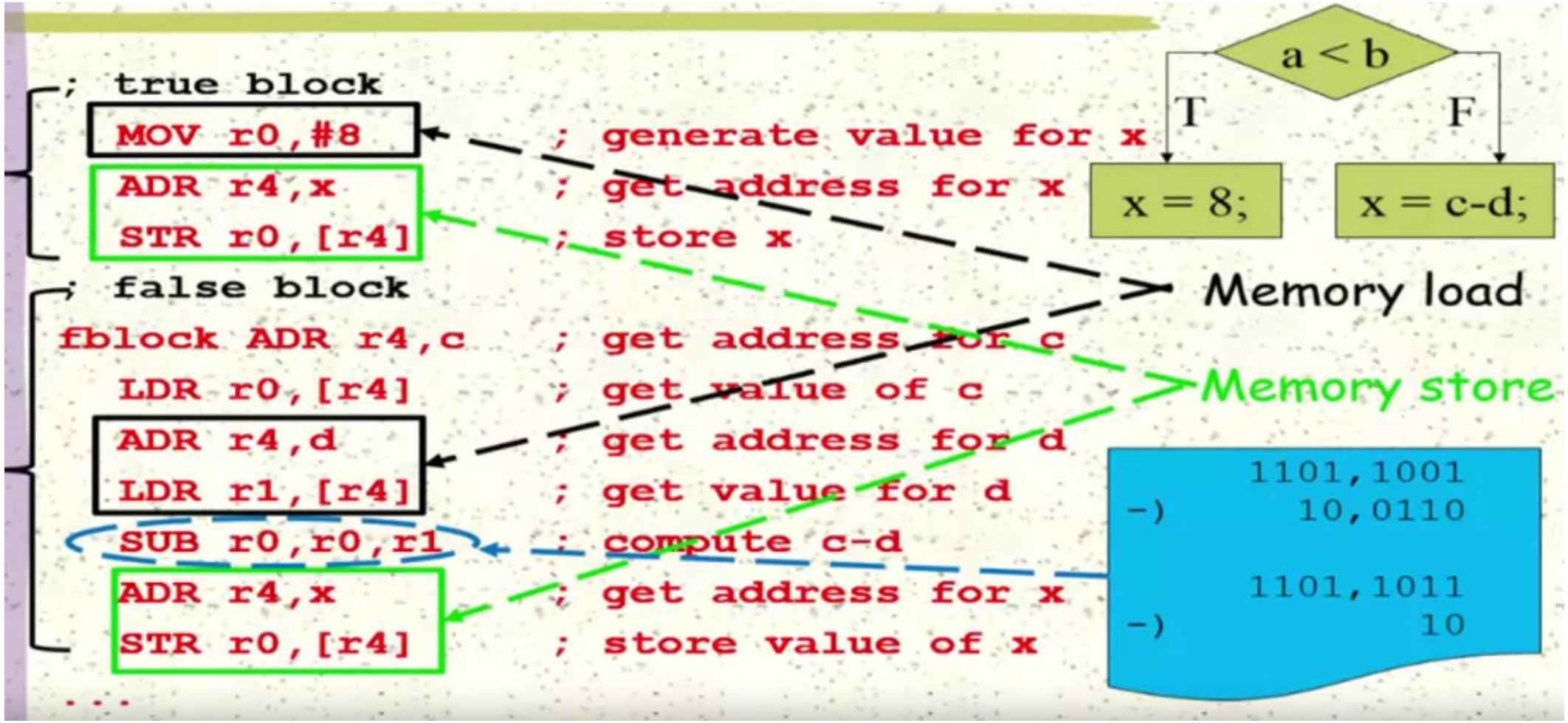
```
compute and test condition
ADR r4,a          ; get address for a
LDR r0,[r4]       ; get value of a
ADR r4,b          ; get address for b
LDR r1,[r4]       ; get value for b
CMP r0,r1         ; compare a < b
BGE fblock        ; if a >= b, branch to false block
```

Example C → Assembly

```
; true block
   MOV r0,#8          ; generate value for x
   ADR r4,x           ; get address for x
   STR r0,[r4]        ; store x
; false block
fblock ADR r4,c       ; get address for c
   LDR r0,[r4]        ; get value of c
   ADR r4,d           ; get address for d
   LDR r1,[r4]        ; get value for d
   SUB r0,r0,r1       ; compute c-d
   ADR r4,x           ; get address for x
   STR r0,[r4]        ; store value of x
...
```



Example C → Assembly



Access-driven Cache Attacks

- Access-driven attacks on cache memory are a class of side-channel attacks that exploit the patterns of cache access by different processes to infer sensitive information.
- In these attacks, an attacker monitors cache usage to detect which cache lines are accessed by a victim process, indirectly revealing information such as cryptographic keys or other sensitive data.

Key Mechanisms of Access-Driven Cache Attacks

Cache Sets and Lines:

- Cache memory is organized into sets and lines. Multiple memory addresses map to the same cache set. When multiple addresses map to the same set, they compete for the same cache lines.
- Access-driven attacks exploit this by monitoring which cache sets are used by the victim, allowing the attacker to make inferences about the memory access patterns of the victim.

- Observing Cache Behavior:
- Attackers can infer information about what a victim is doing by observing whether specific cache lines are accessed or evicted during the victim's operation.
- This indirect observation gives attackers the ability to infer secrets such as encryption keys or passwords, especially in cases where memory access patterns depend on sensitive data.

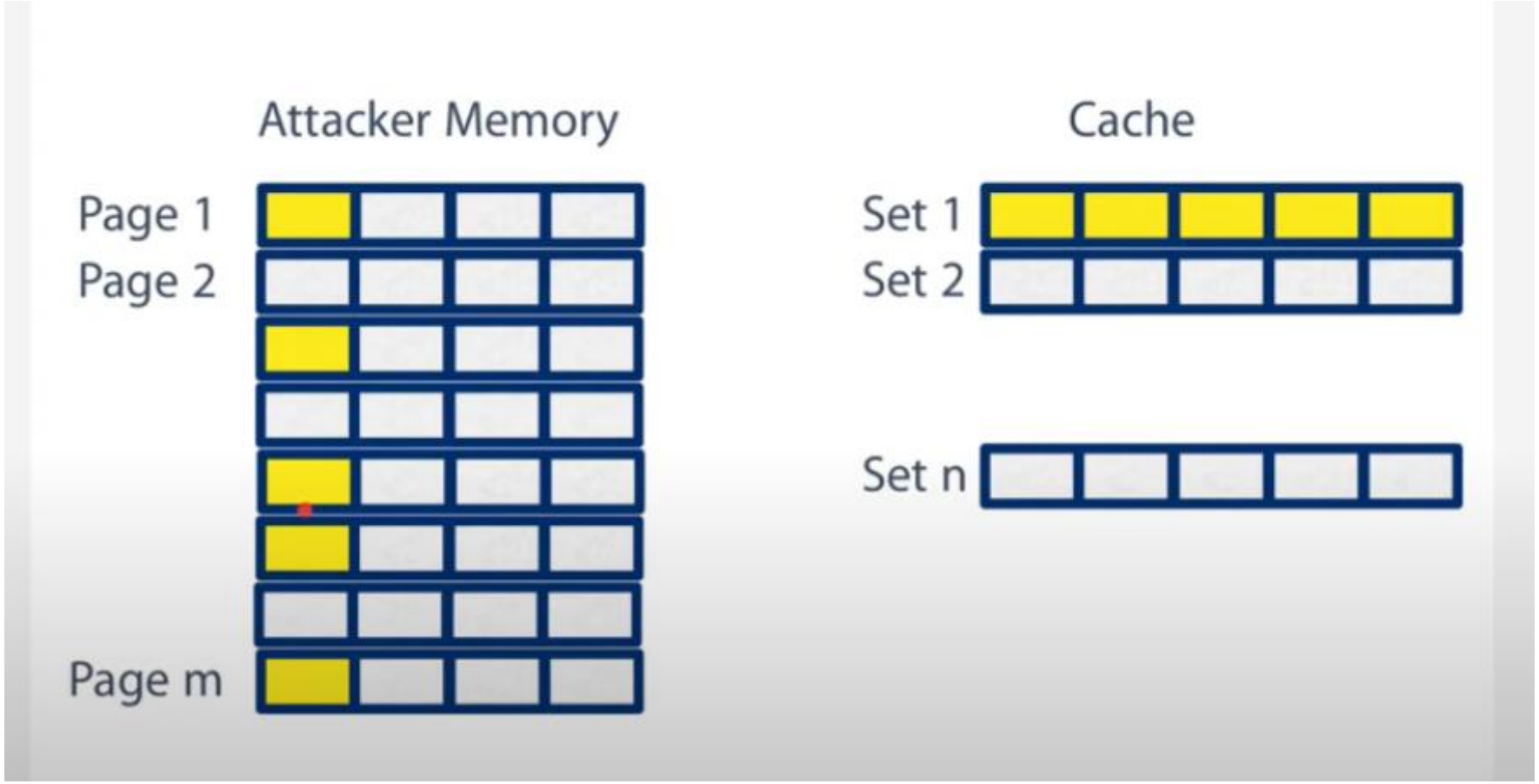
Types of Access-Driven Cache Attacks

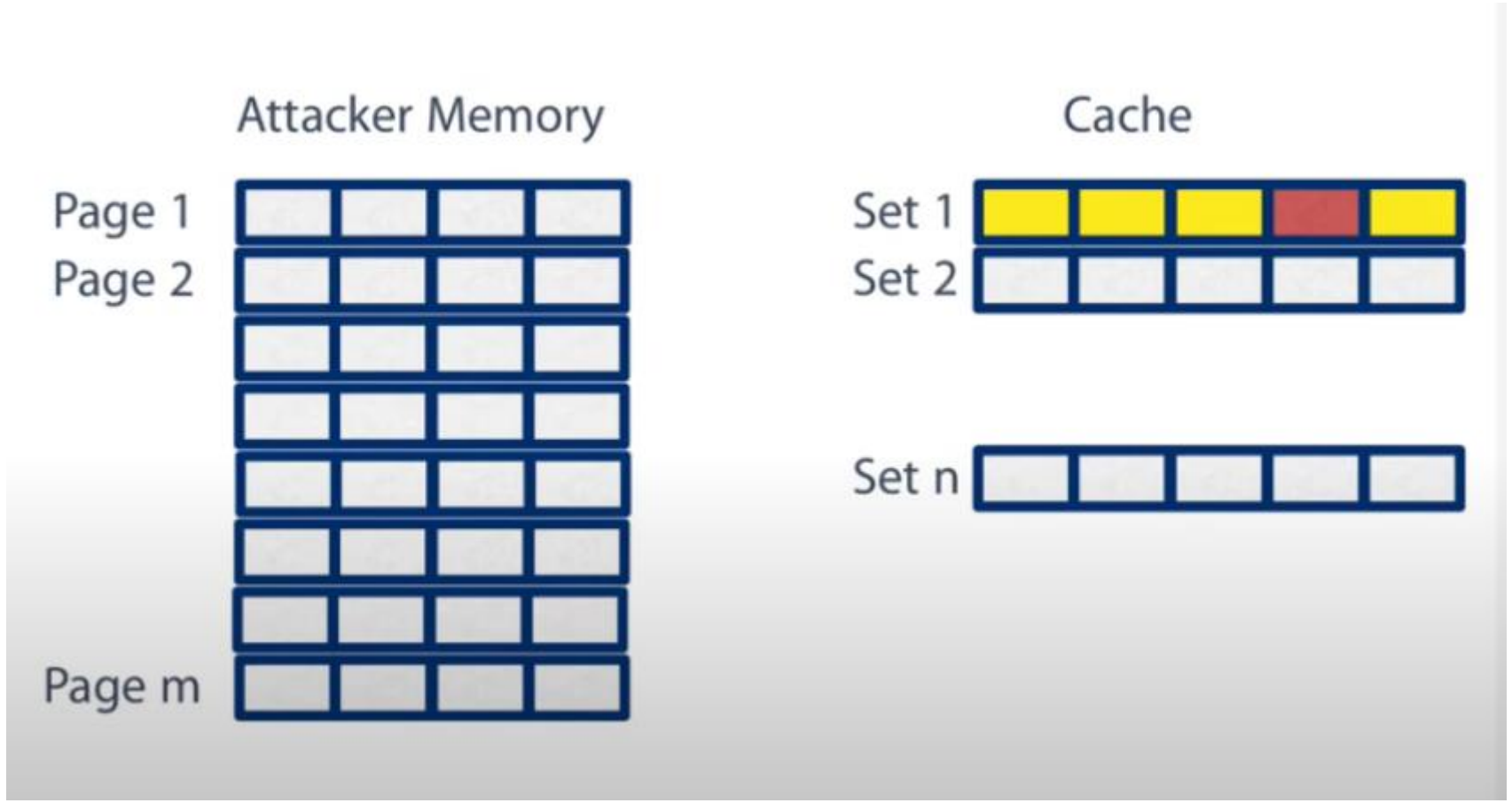
- Prime and Probe Attack
- Flush and Reload Attack

Prime and Probe Attack

- **Prime:** The attacker fills the cache with their own data by accessing memory addresses that map to the same cache lines as the victim's data.
- **Victim Execution:** The victim performs operations that may evict some of the attacker's cache lines by using the same memory addresses or cache sets.
- **Probe:** The attacker then reaccesses the same memory addresses and measures the access times. A slower access indicates a cache miss (which implies that the victim used the same cache set and evicted the attacker's data), while a faster access indicates a cache hit.

Prime and Probe Attack




Attacker Memory
Page 1
Page 2
Page m
Cache
Set 1
Set 2
Set n

Flush and Reload Attack

- **Flush:** The attacker uses specific instructions to flush shared cache lines, ensuring that data is evicted from the cache.
- **Wait**: The attacker waits for a short period while the target process runs. If the target process accesses the flushed memory location, it will reload the data into the cache.
- **Reload**: The attacker then reloads the same cache line and measures the time it takes. If the target process accessed the memory during the wait, the data will already be in the cache, and the reload will be fast. If not, the reload will be slower.
- **Use Case**: Requires shared memory between the attacker and the victim, such as in a shared library scenario (e.g., in virtualized environments or cloud services).

Practical Attack Scenarios

Attacking Cryptographic Algorithms:

- **AES (Advanced Encryption Standard):** AES implementations that use lookup tables (such as the T-Tables) can be vulnerable to cache attacks. By monitoring which cache sets are accessed during encryption, an attacker can recover the secret key.
- **RSA and DSA (Digital Signature Algorithm):** These algorithms often involve operations that result in cache accesses that depend on private key bits, enabling attackers to recover parts of the key by analyzing access patterns.

Cross-VM Attacks in Cloud Environments

- In multi-tenant cloud environments, where multiple virtual machines (VMs) share the same physical hardware, access-driven cache attacks can be used by one tenant (attacker) to observe the cache behavior of another tenant (victim).
- This can lead to sensitive data leakage, such as cryptographic keys or user data.

Defense Strategies Against Access-Driven Attacks

- Cache Partitioning:
- By partitioning the cache, different processes or virtual machines can be prevented from sharing the same cache sets, reducing the ability of an attacker to observe cache activity.
- Constant-Time Cryptographic Algorithms:
- Cryptographic implementations should be designed to access memory in a way that is independent of secret information. This makes it impossible for an attacker to infer sensitive data based on memory access patterns.

- Randomized Cache Mapping:
- Randomizing the cache-to-memory mapping makes it harder for attackers to predict which cache sets their victim is using, mitigating cache attacks.
- Cache Flushing:
- Some systems use cache flushing techniques to clear sensitive data from the cache before switching between processes or virtual machines. This can reduce the effectiveness of access-driven attacks.

Thankyou