CS773-2025, Mini-quiz-I, Feb-3, 2025, 75 minutes

Please note the question paper has covert answers for some of the questions asked in this exam. Please use a high bandwidth and highly accurate covert channel for the answers

Q1 to 12: Answer with T/F without any justification. Each question carries 0.25 marks.

(F) Q1. Prime+Probe attack through last-level cache can lead to privacy breach whereas Flush+Reload attack is a security threat.

(T) Q2. Prime+Probe attack can be expediated by replacing prime step with a flush step, where an attacker flushes its own data mapped to a particular cache set.

(F) Q3. Cache occupancy attacks can only be mitigated by randomized caches with proper remapping rate.

Covert channel :- Q13 ☺

(F) Q4. Side channel attacks are faster than covert channel attacks in terms of bandwidth.

(T) Q5. If a cache provides same latency for hit and a miss then all timing channels related to caches will be mitigated.

(F) Q6. Reverse Engineering the last-level cache involves understanding the distribution of access latency of the last-level cache.

(T) Q7. Copy on write facilitates all shared memory-based attacks like Flush+Flush.

(T) Q8. Way-based partitioning is limited by the associativity of the last-level cache.

(T) Q9. Huge pages make the life of an eviction-based attacker easy.

(F) Q10. A new **x86** instruction called *cldemote* is in market, which demotes caches lines from L1 to LLC, but does not flush it to the DRAM. If you want to mount a timing channel based on cldemote, then LLC hit/mis latency should be used for calibration.

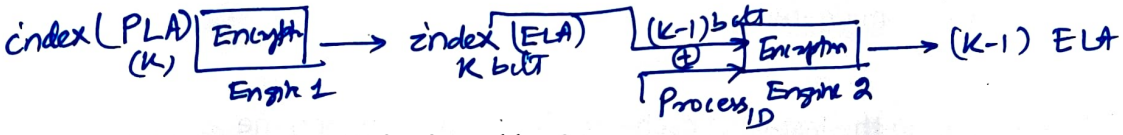
(F) Q11. X86 ISA does specify about timing channels but not the ARM based ISAs.

(T) Q12. It is possible to write programs in such a way that data dependent side-channels can be mitigated.

Real exam begins now

Q13. Hari wants to design a randomized cache that can mitigate all kinds of cache occupancy-based attacks. However, Biswa, in the last lecture, mentioned that randomized caches do not mitigate occupancy-based attacks. Please help Hari in designing a modified randomized cache for a 2MB cache (16 ways, 64B cache lines) shared by 2-cores. You can use the process ID or core ID in your encryption function. Please do not generalize it for an n-core system. Your design should be specific to a 2MB cache shared by 2 cores. You can use a maximum of two encryption units. You are free to use different inputs for different encryption units. The inputs of interest for the encryption unit are the physical index bits and the process/core ID. You are allowed to change the number of index bits you want to use for both the encryption units (or a specific encryption unit). Please note that while designing this circuit, you should also guarantee that conflict based attacks are mitigated as per the in-class discussion with a remapping rate of 16 accesses to each cache line (a cache line is remapped after every 16 accesses). You are not supposed to assume anything else that is not mentioned in this question.

[10 (5+5) marks, 5 marks for the details of each encryption unit. Zero marks if only one encryption unit is used for conventional randomization that is discussed in the class.]



Q14. A 64MB LLC is shared by 64 cores. The cache is a 16-way cache with a cache line size of 64 bytes. The OS wants to use page/cache coloring so that all 64 cores will be isolated at the LLC and cannot cause conflict or occupancy-based attacks. The LLC uses a 64-bit physical address.

(i) What will be the capacity of each color (core) that is assigned by the OS if the OS page size is 4KB **[5 marks]**

(ii) What will be the capacity of each color if the page size is 1GB? **[2 marks]**

(iii) A 64MB LLC with 16-ways, 64B cache lines with 64-bit physical address is used by an OS with page size of 4KB. If the OS uses page coloring to provide complete security, how many unique processes it can support and what is the size (capacity) of each color? **[3 marks]**

(i) 10 bits for coloring, $(16+6)-12=10$, for 64 cores, 2^6 colors
(index bits − page size)

$\frac{64\text{MB}}{64\text{ colors}} = 1\text{ MB per color}$

(ii) Only one color possible of size 64MB, (iii) 2^{10} colors, 1024 processes, 64KB/process

Q15. Please re-write this pseudo-code so that the functionality of the function *exponent* remains the same. However, a side channel attack through the cache will be difficult to mount. You are not supposed to assume anything else about functionality. The functionality should be the same as shown in this pseudo code. **[10 marks]**

```
1  function exponent(b, e, m)
2  begin
3    x ← 1
4    for i ← |e| − 1 downto 0 do
5      x ← x^2
6      x ← x mod m
7      if (e_i = 1) then
8        x ← xb
9        x ← x mod m
10     endif
11   done
12   return x
13 end
```

Same

$x \leftarrow x \bmod m$
$x' \leftarrow xb$
$x' \leftarrow x' \bmod m$
if $(x_i == 1)$
$x = x'$

Covert answer in T/F

Q16. Please suggest two pertinent points that can improve the learning out of CS773, provided you want to learn, attend lectures etc. Adhoc suggestions will lead to -2 marks from your total marks. So attempt this question only if you are serious about learning. It is completely fine if you are not interested in this course but still want to credit for k number of reasons. **[2 marks]**

Have a Trustworthy exam. Make sure your answers are not leaked through side/covert channels.