Lecture Note

Chapter 8: Side-Channel Threats and Protections

CYENG 225: Microcontroller Essentials for Cyber Applications

Instructor: Dr. Shayan (Sean) Taheri
Gannon University (GU)

Chapter 8 Overview

- **Major Items**
  - It presents side channel threats, and protection approaches, for secure processor architectures.
  - It introduces side and covert channels, and how they can lead to side- and covert-channel attacks.
  - It presents various processor features and how they contribute to information leaks.
  - It presents a simple classification of side and covert channels.
  - It discusses information leakage bandwidths due to various attacks.
  - It discusses defense techniques, and especially highlights various secure cache designs.
  - It presents arguments for use of side channels as means to detect attacks.
  - It provides discussion of side channel threats assumption made by secure processor architectures.

Side and Covert Channels

- These channels are a means for communication of information.
- They are called side- and covert-channel attacks if the communication leads to some secret information leaking out.
- The attacks are often on confidentiality, i.e., the goal is to leak information out.

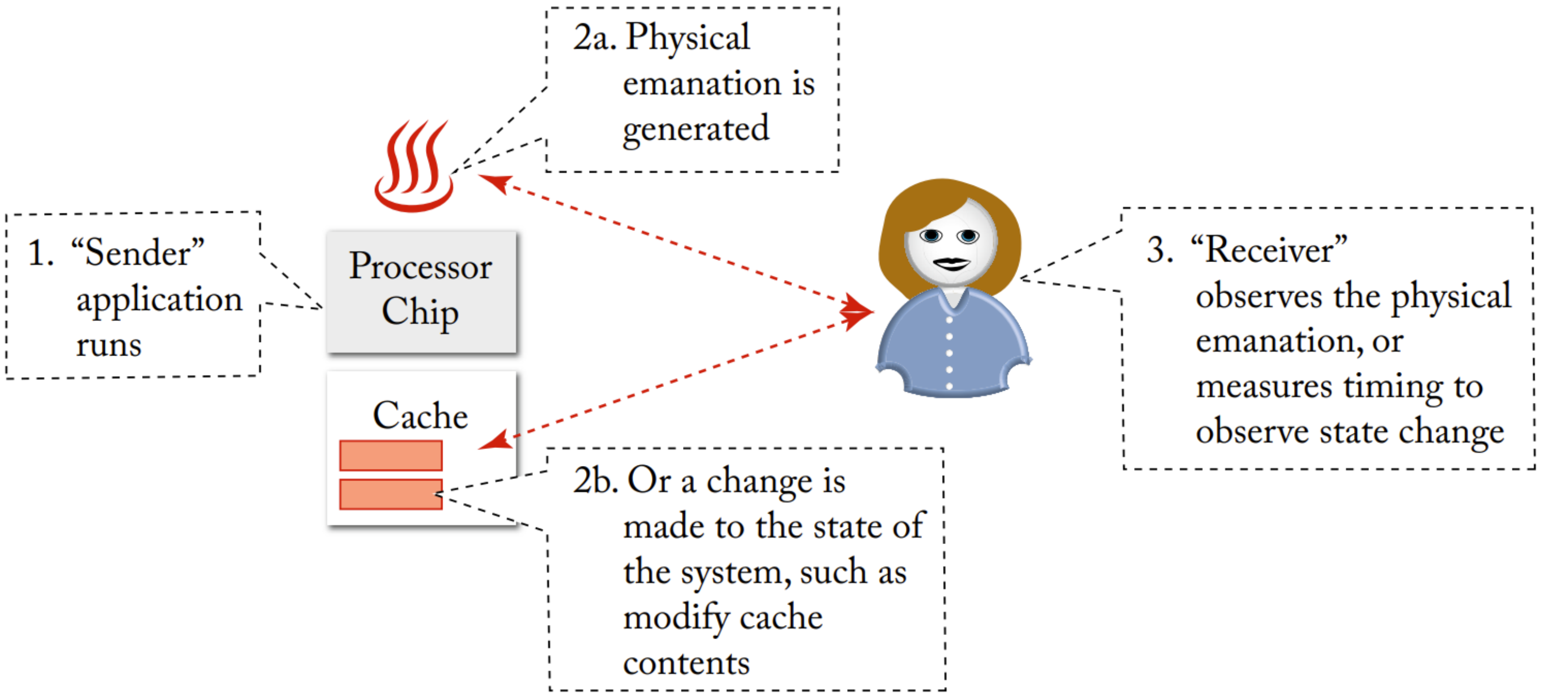
Covert Channel

- A covert channel is an intentional communication between a sender and a receiver via a medium not designed to be a communication channel.
- Covert channels typically leverage unusual methods for communication of information, never intended by the system's designers.
- They typically leverage observable changes in: timing of execution of some instructions or programs, power consumption of the system, thermal emanations of the system, electro-magnetic (EM) emanations generated as the system operates, or even acoustic emanations.
- Covert channels are easier to establish since both the sender and the receive are working together.
- Establishing a covert channel is a precursor to being able to establish a side channel (defined below).
- Covert channels are a means for communicating information.
- A covert-channel attacks is an attack on a system's confidentiality, where the sender and receiver use a covert channel to leak out sensitive information.

Covert Channel - Steps of a Typical Attack (Cont.)

- The sender runs some application that attempts to leak secret information to the receiver.
- Execution of the application or instructions can generate some emanations or cause processor state to be changed; the sender and receiver know how the change in emanations or processor state is used to encode secret information that is to be transmitted.
- The receiver is able to observe the emanations or change of the processor state (usually through timing changes), and deduce the secret information that was communicated by the sender.

Threats Against Main Memory


1. “Sender” application runs
Processor Chip
Cache
2a. Physical emanation is generated
2b. Or a change is made to the state of the system, such as modify cache contents
3. “Receiver” observes the physical emanation, or measures timing to observe state change

Schematic of a covert-channel attack. Side-channel attacks are similar, but the sender in the covert channel is an unsuspecting victim in the side channel.

Side Channel

- In a side channel, the “sender” in an unsuspecting victim and the “receiver” is the attacker.
- A side channel is similar to a covert channel, but the sender does not intend to communicate information to the receiver, rather sending (i.e., leaking) of information is a side effect of the implementation and the way the computer hardware or software is used.
- Just as covert channels, side channels can be created using: timing, power, thermal emanations, EM emanations, or acoustic emanations.
- The goal of a side-channel attack is to extract some information from the victim.
- Meanwhile, the victim does not observe any execution behavior change nor is aware that they are leaking information.
- This way confidentiality can be violated as data, such as secret encryption keys, is leaked out.
- Substitution: The “sender” from the figure in the victim and the “receiver” from the figure is the attacker.
- Interestingly, side-channel attacks can work in “reverse.”
- A side channel can also exist from attacker to victim.
- In a reversed attack, the attacker’s behavior can “send” some information to the victim.
- The information, in the form of processor state change for example, affects how the victim executes, without the victim knowing there is a change.

Side and Covert Channels in Processors

- From a processor architecture perspective, there is an intrinsic connection between the side and covert channels and the characteristics of the underlying hardware.
- Two key features need to be present for communication to exist.
- First, for communication to exist there has to be a channel.
- In processors, the channels are based on the shared hardware used by the different processes—because of spatial and temporal sharing of processor functional units among different programs, behavior of the hardware can be modulated and observed by the different programs.
- Second, for communication to exist there needs to be means of modulating the channel.
- In processors, many decades of processor architecture research have resulted in processor optimizations which create fast and slow execution paths, and which result in stateful functional units that are influenced by the instructions that execute on them.
- Taking the fast or slow path, or affecting state of functional units, which can be later observed, are some of the means of modulating information onto the processor hardware "communication channel."
- The channels can be classified based on whether logical (software-only) or physical presence is needed.
- With software-only channels, the sender (attacker in covert channels and victim in side channels) and the receiver (attacker in both cases) are only utilizing microarchitectural features of the processor.
- Meanwhile with physical presence, the attacker can use physical means to probe the processor chip, requiring more resources and physical access to the target processor.
- Any defenses need to consider the performance, power, and area overheads they impose on the whole computer system.

Processor Features and Information Leaks

- The sole act of executing an instruction and affecting one or more of processor’s functional units’ state can lead to a side or a covert channel.
- This is because there is an intrinsic relationship between processor microarchitectural features which allow today’s processors to efficiently run various programs, but at the same time optimizations which lead to the side and covert channels.
- Because of the sharing of functional units among different programs, programs can in general observe, directly or indirectly, timing of the operation of the functional units.
- Knowing the design of the different functional units, timing in turns reveals whether the fast or slow execution path was taken.
- Finally, knowing one’s own operations (i.e., the attacker program), or victim’s program’s operations, and the timing, an information leakage can be observed.

Processor Features and Information Leaks (Cont.)

- There are five characteristics of modern processors and their design which lead to microarchitectural-based information leaks that can form a basis for a side or covert channel.
  1. Variable Instruction Execution Timing—Execution of different instructions takes a different amount of time, thus by observing how long some program takes to execute, some information can be learned about which instructions it is executing.
  2. Functional Unit Contention—Sharing of hardware leads to contention, whether a program can use some hardware or not depends on the scheduling and what operations other programs are doing, leaking information about other programs trying to use the same functional unit.
  3. Stateful Functional Units—Program behavior can affect the state of the functional units and output of the functional units or timing of instructions which depend on these functional units is related to state of the functional units, allowing for information leaks to occur.
  4. Memory Hierarchy—Memory hierarchy is composed of many components designed to improve the average performance of a program, such as caches. However, memory accesses can be slow or fast depending on the state of the memory hierarchy, e.g., if data is in the cache or not, leading to many timing side-channel attack possibilities.
  5. Physical Emanations—Execution of programs affects physical characteristics of the chip, such as thermal changes, which can be observed with on-chip sensors and lead to information leak.

Processor Features and Information Leaks (Cont.)

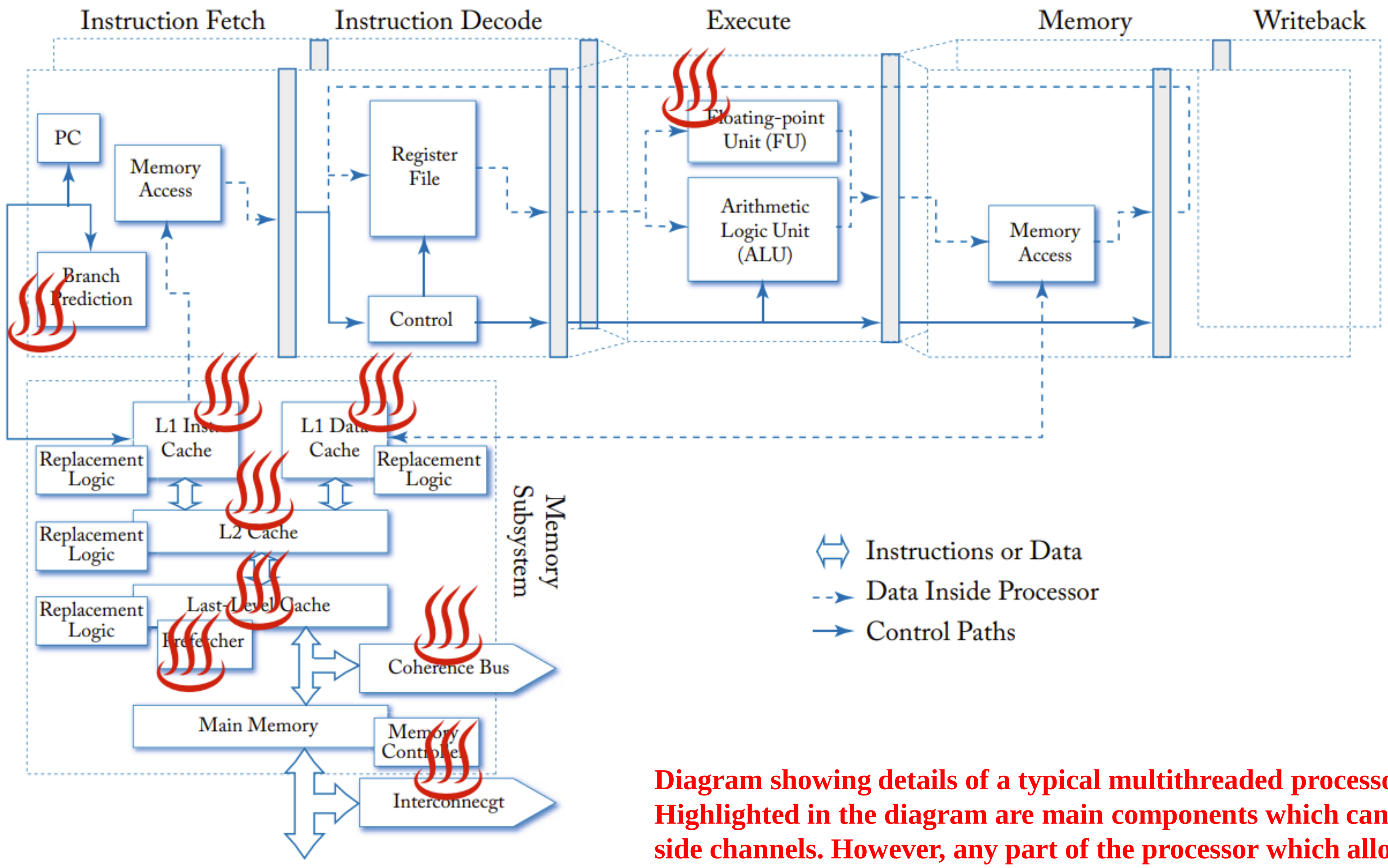


Diagram showing details of a typical multithreaded processor core. Highlighted in the diagram are main components which can contribute to side channels. However, any part of the processor which allows for sharing (spatial or temporal) of resources among different applications can potentially be a source of a side channel, even if no example attack using that component exists today.

Memory Hierarchy

- The processor memory hierarchy has some of the biggest impacts on the programs' performance, and information leaks.
- **Caches**. The memory subsystem is composed of different layers of caches.
- Each L1 cache is located closest to the processor, it is smallest in size, but accessing data in the L1 cache takes about 1–2 processor cycles.
- There are separate L1 caches for instructions and data.
- The L2 cache is a larger cache, but at the cost of taking about 10 processor cycles to access data in the cache.
- The L2 cache can be per processor core or shared between multiple processor cores.
- L3 cache, also called Last Level Cache (LLC), is the biggest cache in size up to few MB, but accessing data in L3 takes tens of cycles.
- There is the main memory, sized in GBs, but requiring 100 cycles or more to access data.
- Processor designers use the cache hierarchy to bring most recently and most frequently used data into the cache closest to the processor.
- This ensures that when there is memory access or instruction fetch, it can be fetched from one of the caches, rather than requiring going all the way to memory.

Memory Hierarchy (Cont.)

- Unfortunately, the fastest caches closest to the processor are also smallest, so there needs to be some policy of which data to keep in the cache.
- Often, the policy is some variant of the least recently used policy (LRU) that kicks out least recently used data or instructions and keeps most recently used ones.
- As programs execute on the processor and perform memory accesses, they cause processors to bring into the caches new data, and kick out least recently used data back to lower cache or eventually to the main memory (DRAM).
- Keeping track of least recently used data in the whole cache is not practical, thus caches are broken down into sets, where each memory location can only be mapped into a specific set.
- Multiple memory addresses are mapped to a set. A cache typically has two or more ways, e.g., in a two-way set associative cache, there are two locations that data from specific set can be placed into.
- The LRU policy is kept for each set. For example, if memory addresses 0 X 0, 0 X 2 and 0 X 4 are accessed in that order, then when 0 X 4 is accessed, it will evict 0 X 0 from the two-way set associative cache as 0 X 0 was least recently used in that set.
- Such design of the caches lends itself easily to contention and interference, which in turn leads to information leakage that is typically due to timing.
- The leakage can reveal whether some data is in the cache or not. Accessing data in L1 takes 1–2 cycles, while data in memory can take up to 100 cycles.
- Eliminating such leakages is difficult.

Memory Hierarchy (Cont.)

- Ideally, the cache replacement logic could search whole cache for the least recently used data, rather than just within a set.
- The Z-cache is one step toward that, however, its complexity has prevent it from being implemented.
- Proposals for randomized caches have also been put forward.
- However, currently if caches are removed, each memory access could take 100 or more cycles; it is not realistic to eliminate the caches form today’s processors’ memory hierarchy.
- Again, execution of different (memory) instructions takes different amount of time leading to potential for side or covert channels.
- **Prefetcher**. Another component of the memory hierarchy is the prefetcher which is used in microprocessors to improve the execution speed of a program by speculatively brining in data or instructions into the caches.
- The goal of a processor cache prefetcher is to predict which memory locations will be accessed in near future and prefetch these locations into the cache.
- By predicting memory access patterns of applications the prefetcher brings in the needed data into the cache, so that when the application accesses the memory, it will already be in the cache or a stream buffer, avoiding much slower access to the DRAM.
- Hardware prefetchers attempts to automatically calculate what data and when to prefetch.

Memory Hierarchy (Cont.)

- The prefetchers usually work in chunks of size of the last level cache (LLC) blocks.
- Sequential prefetchers prefetch block x + 1 when block x is accessed.
- An improvement, which is most often used in today’s processors, is the stride prefetcher which attempts to recognize sequential array accessed, e.g., block x, block x + 20, block x + 40, etc..
- Because hardware stride prefetcher fetches multiple blocks ahead, it will sometimes bring in data that the application is not going to use.
- However, depending on the physical memory allocation, that pre-fetched data may actually be used by another application.
- When the application accesses memory and measures timing, the blocks which were prefetched based on pattern detected for the other application will be accessible more quickly.
- In addition, if the exact details of the prefetcher algorithm are known, it is possible to trace back which addresses and how many addresses were accessed by the other application.
- Like for other parts of the memory hierarchy, prefetcher removal is not an easy option to defend against potential information leaks.
- Prefetcher removal would have largest penalty for applications with very regular memory accesses, but others will be negatively affected as well.

Memory Hierarchy (Cont.)

- **Memory Controller**. The memory controller and the DRAM controller in the main memory are responsible for managing data going to and from the processor and the main memory.
- The memory controller contains queues for request from the processor (reads and writes, usually coming form the last level cache), it has to schedule these request, and arbiter between different caches making request to DRAM, which can cause contention.
- The memory controller, which is a shared resource, becomes a point of contention.
- For example, two processor cores are each connected to the same memory controller and memory chip.
- Requests from each processor core need to be ordered and queued up for processing by the memory.
- Dynamically changing memory demand from one processor core will affect memory performance of the other core.
- While the memory controller attempts to achieve fairness, it is not always possible to balance out memory traffic from different cores.
- In particular, today's DRAM is typically divided up into pages and data from within DRAM is first brought into a row buffer before being actually processed (reads sent data back to the requesting processor from the buffer, or writes update it with incoming data).
- Many of today's chips use open-page policy that gives some preference to reads or writes to currently opened page (i.e., on in the row buffer).
- Memory accesses with lots of spatial locality may get preference as they hit in the open page—giving overall better performance as opening/closing new pages is expensive in terms of energy and time.
- Because of such optimization, again shared hardware leads to contention which in turn can be basis for side or covert channels.

Memory Hierarchy (Cont.)

- **Interconnect**. Modern processors have replaced a typical bus that connected multiple processors and memory with more advance interconnects, such as Intel’s Quick Path Interconnect (QPI).
- The interconnect is used to send data between processors and also for memory accesses in NUMA where main memory is divided up and separate DRAM chips and memory controllers are located near each processor.
- Such an arrangement gives each processor fast access to local memory, yet still large total system memory.
- However, timing of memory accesses can reveal information, such as accessing data in the DRAM chip close to the processor is faster than accessing remotely located DRAM at another core.
- In addition, locking and atomic operations can lock down the interconnect making memory accesses stall.
- Thus memory access timing can reveal state of the interconnect and leak information about what other processes are doing.

Side and Covert Channel Classification

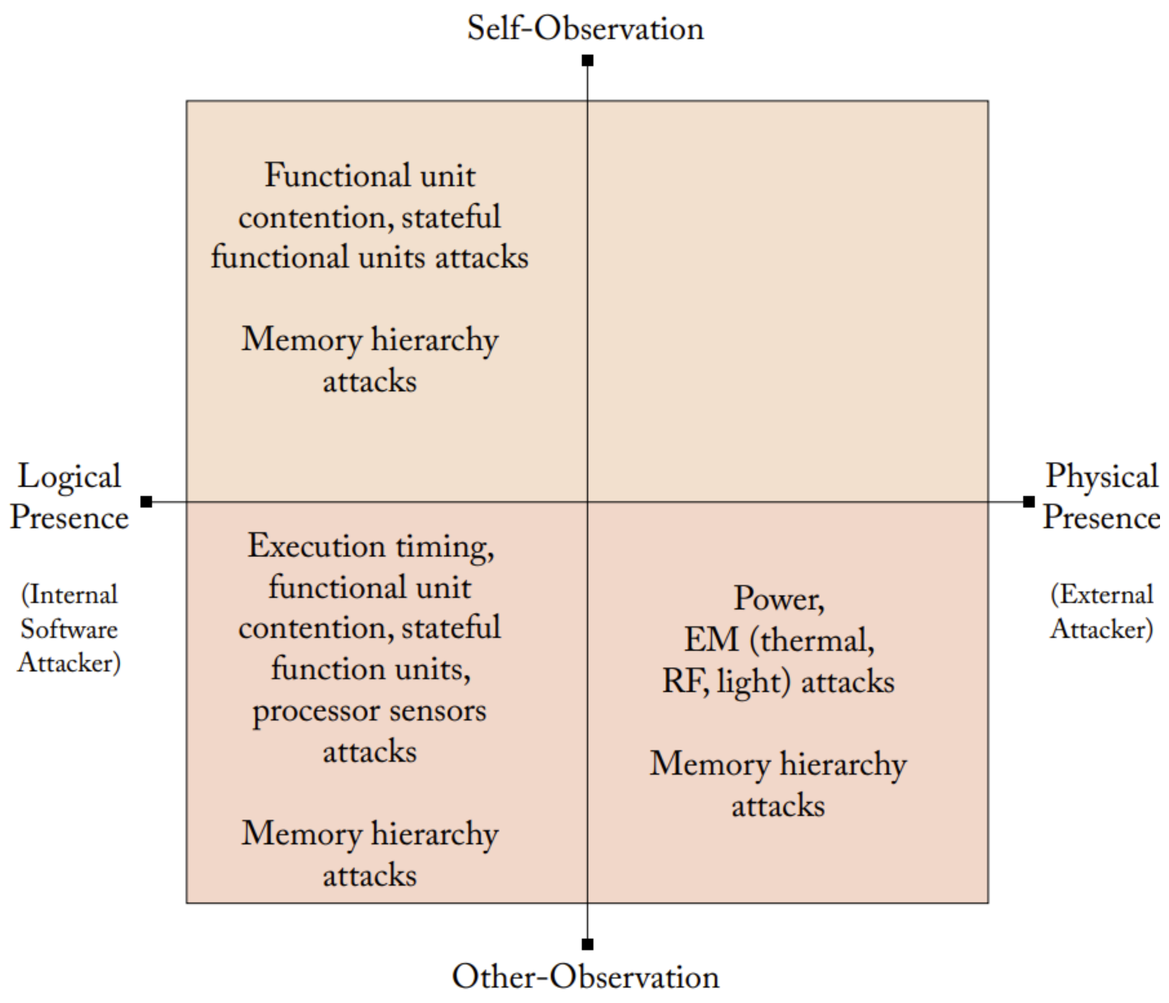
- In context of the processor, the information leaks can be turned into side or covert channels by attackers who are either running software on the same processor as the victim, or who are external to the processor.
- The distinguishing features are the logical vs. physical presence, and whether the attacker needs to observe the victim, or just themselves.
- Most of the microarchitectural attacks (and defenses) that are considered by the architects are at the left-hand side of the figure where software-only attacks are considered (i.e., logical presence).
- Architects do not usually control the exact physical features of the processor chip, such as a special package that may be added to protect from EM radiation leaks.
- They do, however, design and control the types of functional units, their sharing and behavior, thus have some control over potential information leaks.
- Nevertheless, many attacks in the right-hand side where physical presence by the attacker is required.
- Because contention in the processor exists among different programs, attacker can observe the behavior of the victim, but also the victim can affect he behavior of the attacker.

Side and Covert Channel Classification (Cont.)

- The memory hierarchy is a contributor to attacks in most cases.
- Meanwhile, self-observation with physical presence, top-right of the figure, does not make sense, thus no attacks are listed there.
- In terms of memory and cache attacks, in so-called internal timing attacks, the attacker measures its own execution time.
- Based on knowledge of what it (the attacker) is doing, e.g., which cache lines it accessed, and the timing of its own operations, the attacker can deduce information, e.g., which cache lines were being accessed by other applications on that processor.
- In the so-called external timing attacks, the attacker measures execution time of the victim, e.g., how long it takes to encrypt a piece of data; knowing the timing and what the victim is doing the attacker can deduce some information, e.g., was there addition or multiplication done during the encryption, potentially leaking bits of information about the encryption key.
- External timing channels often require many iterations to correlate timing information, however, basic principle is the same that attacker observes a victim’s timing and the victim’s timing depends on the operations it performs.
- The so-called trace-based attacks require tracking exact memory accesses, which is most easily done with physical presence.
- Requirements of physical presence or only virtual presence (software-only attacks) impacts the types of attacks that are possible.
- Nevertheless, the boundary is fluid as introduction of new sensors onto processor chips, changes the requirements and makes new attacks possible with only software.
- Addition of new sensors contributes to functionality, but may create new avenue for side and covert channels.

Side and Covert Channel Classification (Cont.)

A classification of side- and covert-channel attacks based on whether physical presence is required, or attacker only needs to run some software on same processor as the victim; and whether attacker observes the victim ("other-observation"), or attacker only observes its own behavior ("self-observation"). Some combinations do not make sense, such as physical (external) attacker observing its own behavior, and are left blank.


Self-Observation
Functional unit contention, stateful functional units attacks
Memory hierarchy attacks
Logical Presence
(Internal Software Attacker)
Physical Presence
(External Attacker)
Execution timing, functional unit contention, stateful function units, processor sensors attacks
Memory hierarchy attacks
Power, EM (thermal, RF, light) attacks
Memory hierarchy attacks
Other-Observation

Estimates of Existing Attack Bandwidths

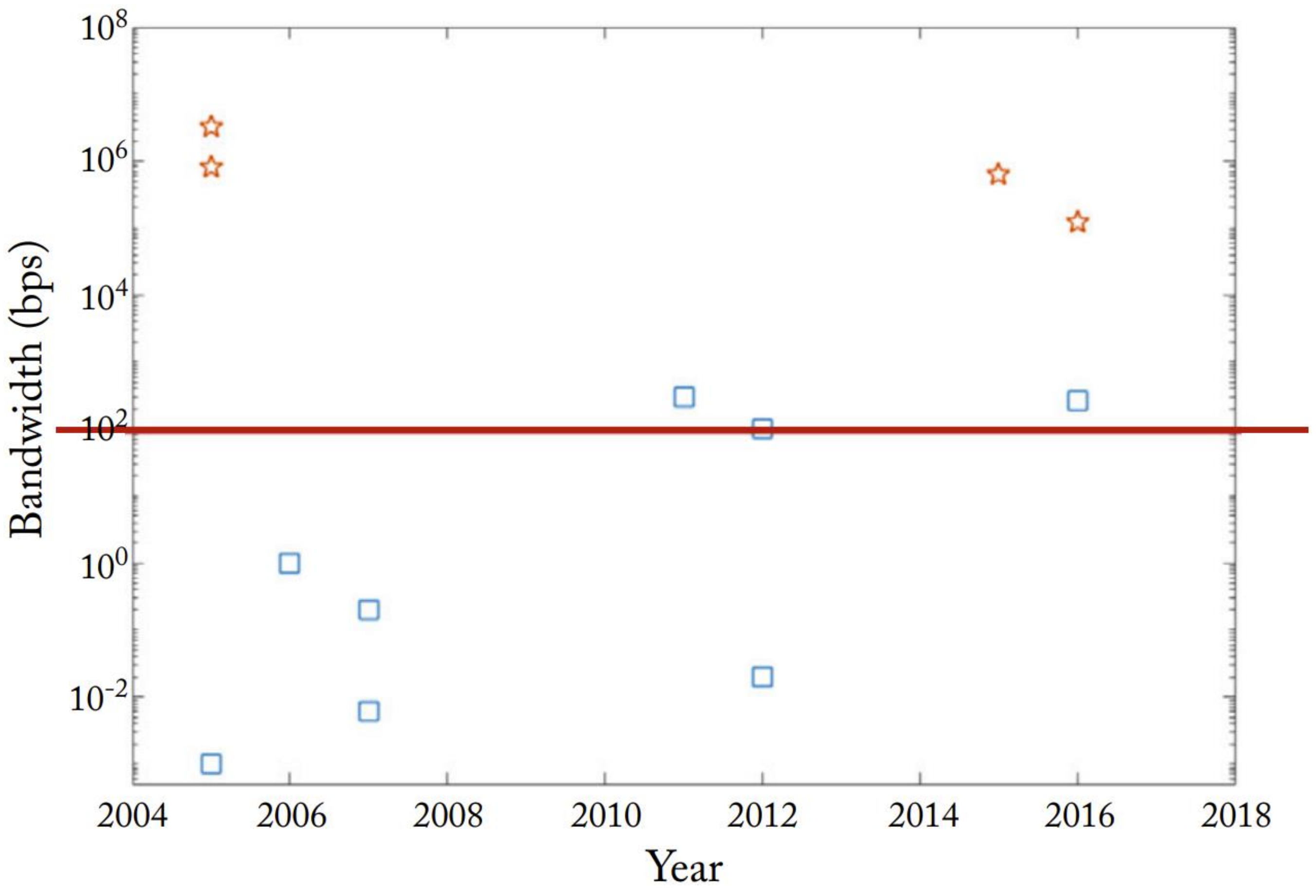
- Most of the presented attacks in the past had bandwidths for information leakage in ranges of kilobits per second (Kbps) in optimal or idealized setting, and in bits per second (bps) or less in more practical settings.
- New attacks can reach Mbps.
- The attack bandwidths have continued to increase over recent years.
- One of the first side-channel attacks was the 0.001 bps Bernstein's AES cache attack using L1 cache collisions.
- Around same time, Percival reported attacks with about 3200 Kbps using L1 cache-based covert channel, 800 Kbps using L2 cache-based covert channel, which reduce to few Kbps when they were done in a realistic setting.
- Besides caches, a 1 bps proof-of-concept channel due to branch predictor was presented.
- The work has since been updated and latest results show about 120 Kbps channel.
- Further units inside the processor that have been exploited are the thermal sensors and recent work has shown 300 bps covert channel that leverages these on-chip sensors that can be used to measure physical properties of the chip (i.e., temperature) without physical presence.
- In the realm of cloud computing, researchers have focused on virtualization and virtual machines.
- The attacks are only a subset of all attacks, and they only cover attacks without physical presence.
- These attacks leverage variety of functional units in the processor, and it should be expected that most processor features could potentially be used as source of information leak leading to side or covert channels.
- Designers of new processor features need to expect that any additional features may be leveraged in a malicious way and should consider how the new addition impacts system security.

Attack Bandwidth Analysis

- The Trusted Computer System Evaluation Criteria (TCSEC), also commonly referred to as The Orange Book, sets the basic requirement for trusted computer systems.
- The Orange Book specifies that a channel bandwidth exceeding a rate of 100 bps is a high bandwidth channel.
- Many attacks have long surpassed the boundary set by TCSEC.
- The cache-based attacks are highest in bandwidth as potential attackers are able to affect specific cache sets by executing memory accesses to particular addresses that map to the desired set.
- Other functional units let potential attackers affect the units state less directly.
- When considering idealized attacks which are on the order of 100s Kbps, the “high bandwidth” boundary has long been passed in their case.
- These attacks, however, are usually specific to a single program, which often is the AES encryption algorithm.
- The attacks tend to be also synchronous, where the attacker and victim are executing synchronized (e.g., attacker and victim alternate execution on same processors).
- The synchronous attacks tend to have better bandwidth.
- Dedicated attacker can thus come up with clever ways of improving bandwidth by having more synchronous attacks, approaching the idealized attack scenarios and further improving attack bandwidths.
- In summary, bandwidths for many side channel attacks today have already surpassed the bounds set by TCSEC for high bandwidth channels.
- Designers need to account for these potential attacks when considering new designs or analyzing existing ones.

Attack Bandwidth Analysis (Cont.)

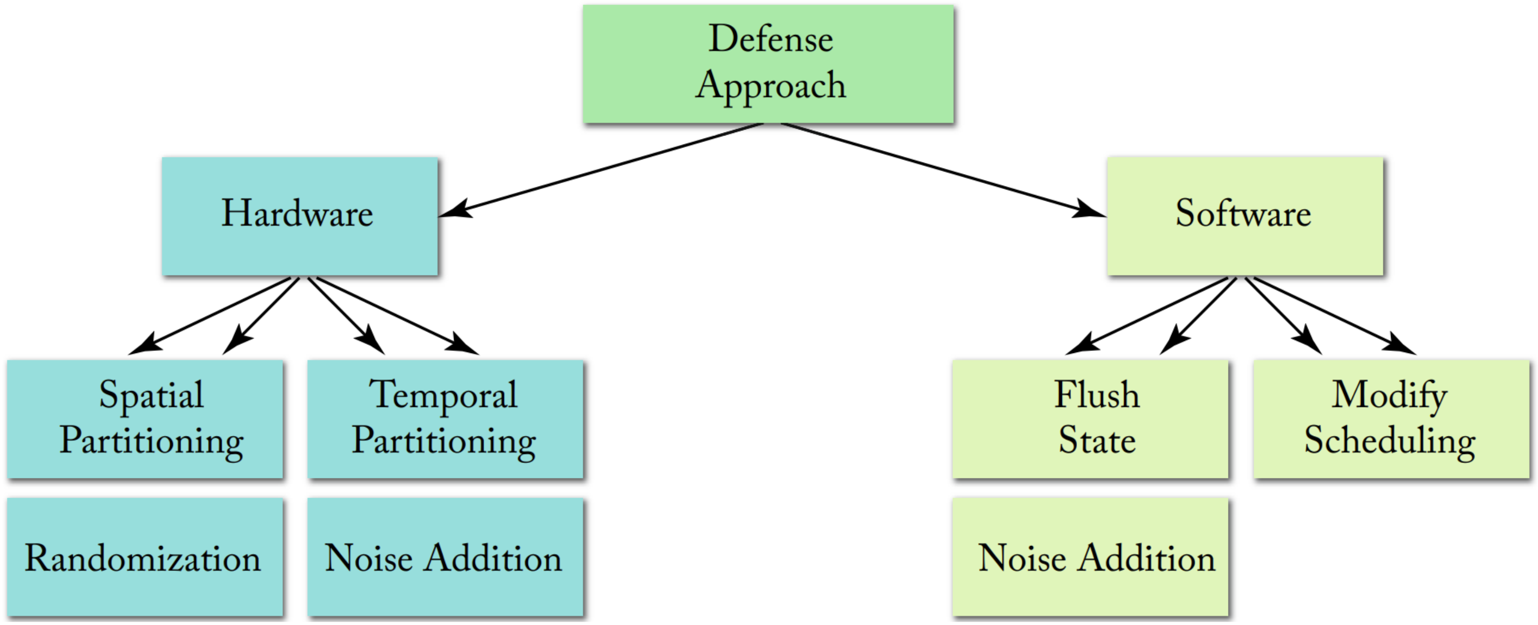
Scatter plot of bandwidths of select attacks, orders of magnitude bandwidth increase can be seen over last years for the non-idealized attacks (squares). Idealized attacks (pentagrams) show much greater bandwidths, but these are for specific cases, such as attacks on particular table lookup based AES software implementation or a particular RSA implementation. Today, some attacks are even better, reaching Mbps data rates.



Defending Side and Covert Channels

- Since the side and covert channels depend both on the hardware, and the software that is running on that hardware, the defenses can be based on both hardware approaches and software approaches.
- To mitigate side and covert channels, hardware architectural changes have been proposed including partitioning or time-multiplexing caches, which have since been improved.
- Such approaches essentially reserve a subset of the cache for the protected program.
- Other programs are not able to interfere with these reserved cache blocks.
- This prevents internal-timing, but external-timing attacks are still possible since measuring the protected program’s timing from outside still can reveal some patterns about memory it is accessing.
- In addition, other applications cannot use the reserved cache blocks, effectively cutting down on cache size and the performance benefits it brings.
- One of the best proposals focuses on new type of randomized caches.
- Today’s commodity hardware has caches where the mapping between memory and cache sets is fixed an same for all applications.
- Randomized caches in effect change this mapping for each application.
- In general, increasing associativity helps defend attacks, and number of caches aim to simulate highly associative cache without actually requiring implementation of a highly associative cache (which may have poor performance and use a lot of power).
- Both inside and outside the computer system needs to be considered even when focusing on microarchitectural channels.

Defending Side and Covert Channels (Cont.)



Hardware and software approaches to microarchitectural side and covert channel defense.

Side Channels as Attack Detectors

- While side and covert channels are mostly considered as a negative aspect of a system, side channels can be actually used to detect or observe system operation.
- Measure timing, power, EM, and other behavior can be used to detect unusual system behavior and potential attacks.
- This approach is similar to using performance counters, but attacker does not know measurement is going on, or even how the measurement is being made.
- Consequently, a tension between side channels as attack vectors vs. as detection tools exists.
- Side channels are mostly used for attack today.
- But if the channels are someday fully eliminated, then their use as attack detectors will be eliminated.
- Thus, a middle ground needs to be found.

Side Channel Threats Assumption

- There is typically one assumption related to side channel threats in secure processors.
- The protected software assumes that the TEE is side-channel free.
- The TCB hardware and software should clean up processor state (both ISA-visible and microarchitectural) to remove any side channels.
- Memory hierarchy should especially defend protected software from side-channels attacks.
- Despite protections from attacks by other software or hardware, TEE software still needs to defend against internal interference-based channels.
- In particular, software's own memory accesses interfere with each other and create timing differences that external attackers can observe when interacting with the TEE software.

Assignment

- **Reading Assignment:**
  - Zferer, J., 2018. **Principles of secure processor architecture design**, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 9048, pp.1-175.
    - ✓ "Chapter 8: Side-Channel Threats and Protections", Pages 85-102.

Questions?