Can Secure architecture perform?

Motivated by Yale's talk

Machine Learning and Quantum Computing; Is there anything else worth working on?

Avi Mendelson Technion, Israel

avi.mendelson@technion.ac.il

Agenda

- Intro and motivation
- New approach
- Conclusions and next steps

Introduction

1. For many years we design systems to achieve maximum performance
   - So we created a power problems
2. We start balancing power and performance
   - Aims to minimize the performance lost as a result of power considerations.
3. Power and performance may result in information leak.
   - Current systems are loosing much power and performance in order to be immune against information leaks.

So far, with very limited success

Security of high performance speculative cores

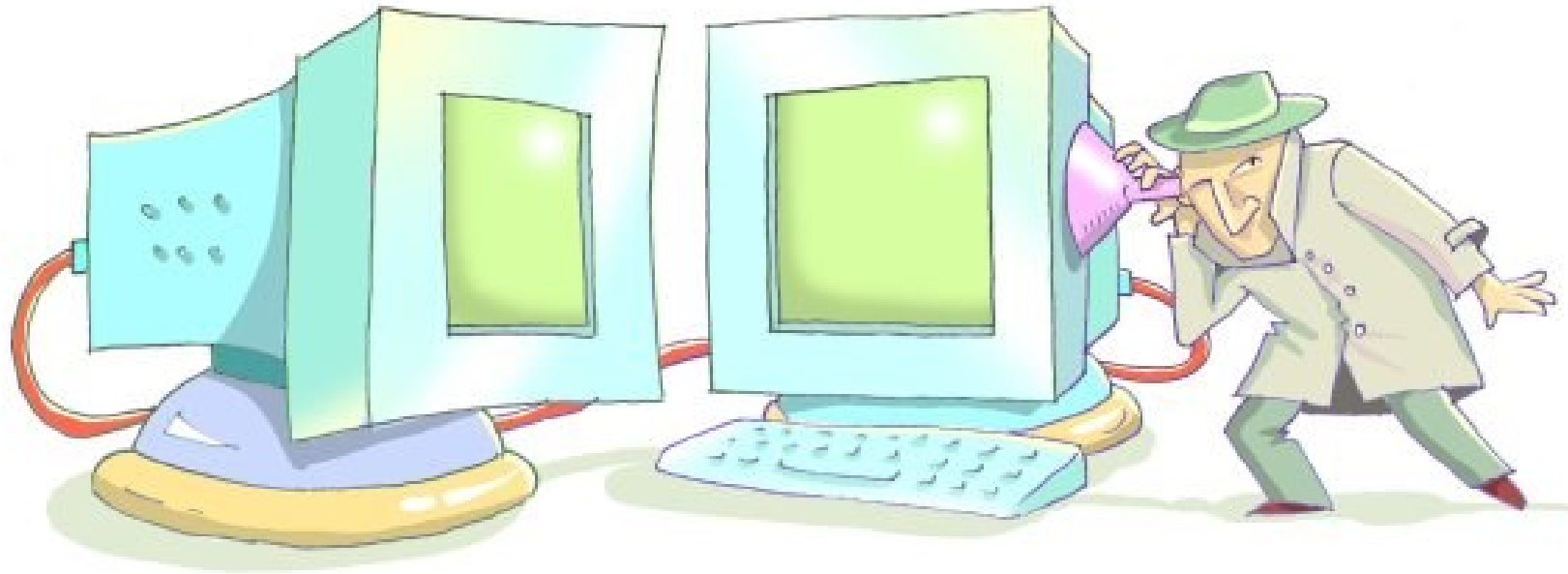
It leaks and most likely will continue to leak

UNLESS we do something *different*

Security of modern systems

- Speculation is one of the most noticeable means to gain performance
- Speculative execution leaks information since it creates “observable dependencies” with the data it executes (to be extended)
- Systems are becoming extremely sophisticated and complicated so the probability for bugs cannot be ignored
- Security attacks are becoming a “profitable business” and so, individuals, companies and even armies/countries are spending much effort (and money) to “improve” it.

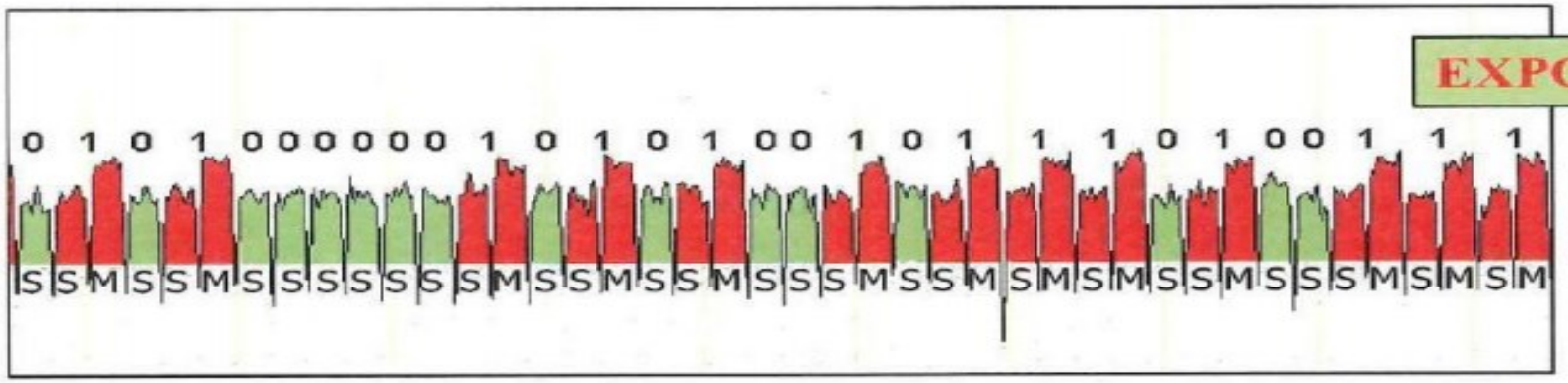
Side-channel attacks



- The goal is to find dependencies between “measurable parameters” and “secretes”
- We can inject faults to help the process

Notice – in order to “break a code”, it is enough to reduce the “solution space” to a size we can exhaustively search in a reasonable time

Simple power analysis -- RSA Conditional Branch




```
Algorithm MODULAR_EXPONENTIATION (exponent e, base b, modulus n)
      Let A = 1
      Let X = b mod n
      Let k be the number of bits in e.  Then,
      For i = k downto 0, do
            A = (A * A) mod n
            Let e_i be the i'th bit of exponent e        ; bit 0 is LSB
            if (e_i == 1), do
                  A = (A * X) mod n
            Done
      Done
      Return A
End Algorithm
```

LOOP

SQUARE

CONDITIONAL MULT

"White box attack"

DPA – Differential Power analysis

- DPA is mainly used for “black box” model where you know little (if at all) on what the system is executing (but you may have a good model on the hardware itself.)
- We may observe (or guess) that on-average, the power of the system when execute some values (e.g., “0”) is different than the power when execute other values (e.g., “1”)
- We are using distribution of many samples
- We care of the average behavior not on understanding a single measurement; e.g., when using OOO cores, each execution may consume different amount of power. Still the average of executing some values may be different than the average of executing other values

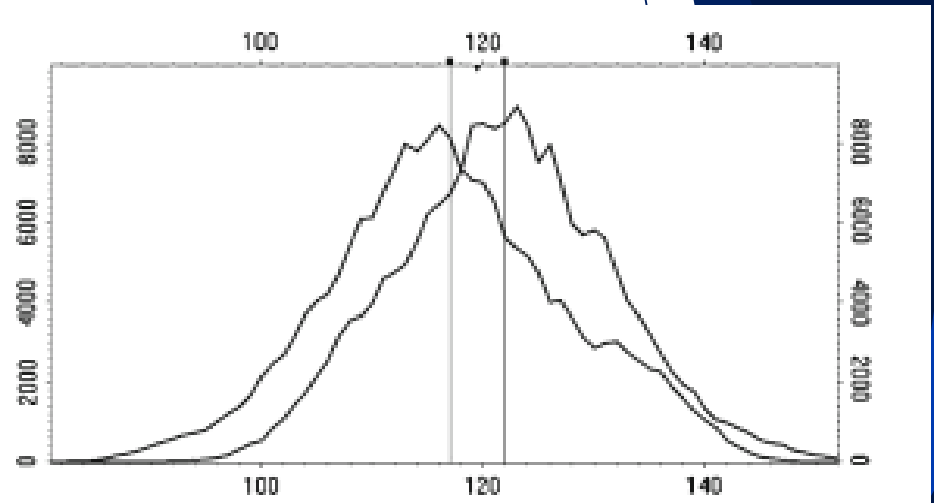
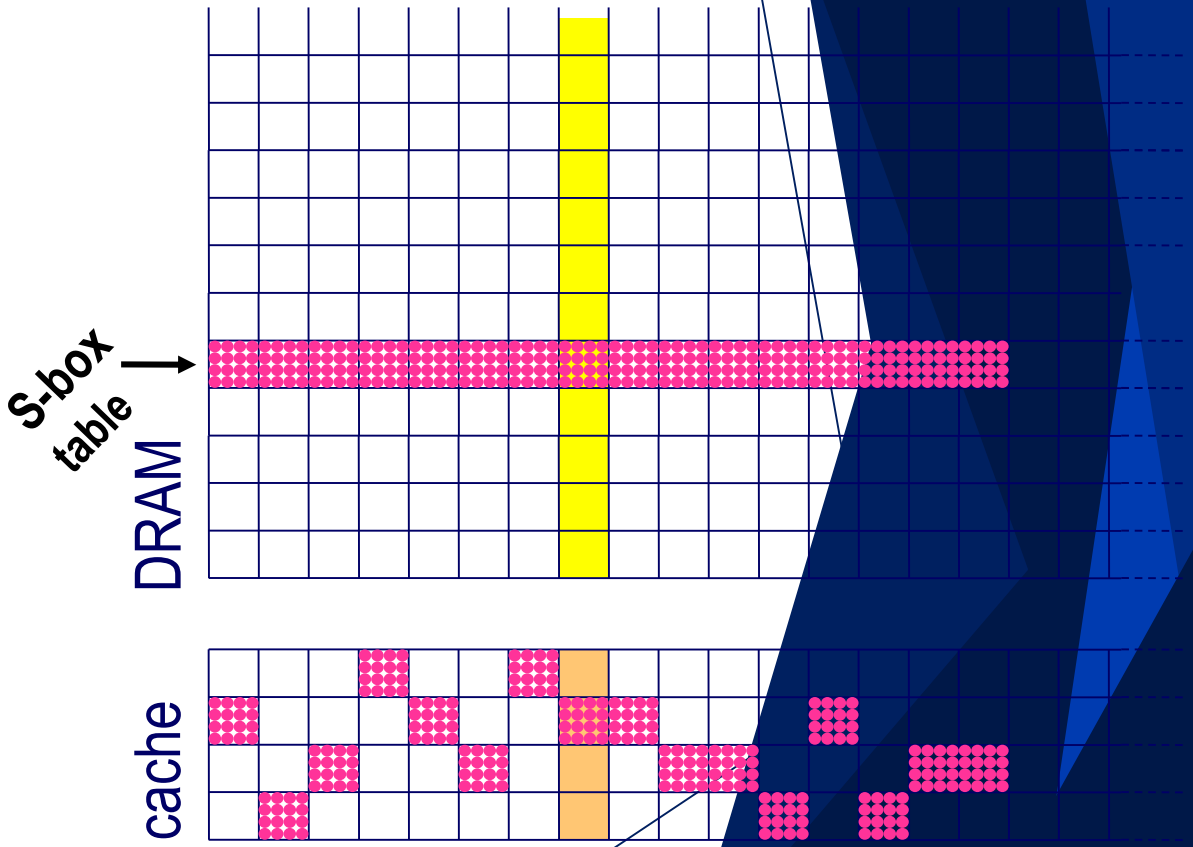


Fig. 5 Distributions of power consumption measurements for traces with the LSB of the output of the first S-box being 1 (left) and 0 (right)

Can be “Black box” attack

Timing based attacks - simple example

- Cache is great for performance but problematic for security since “keep traces”
- Here, two threads
  - Thread 1 fill out the cache
  - Allows Thread 2 to operate
  - Thread 1 check which of his cache lines were replaced.
    - HOW? Access the data and observe how long it take you to retrieve the data (cache hit or miss)
  - Does it expose the secret key ?
    - Not directly
    - It reduces the search space to a manageable size that can be exhaustively searched.



Exclusive summary - Spectre and Meltdown

- **Spectre** and **Meltdown** are vulnerabilities that allow user level code (unprivileged) to indirectly read data from all physical pages (including privilege pages and pages belonging to other processes.)
- Speculative execution of out-of-order cores leave traces in processor state since “roll back” impact internal structures such as caches

- Based on Branch side effects created during miss speculation of branch prediction

Families of attacks – several generations already exist e.g., was extended for exposing vulnerabilities in SGX (Intel’s extension for Security)

Root-cause and implications

- Processors are designed with best performance in mind (or best performance per given power).
- Modern architectures uses speculative execution as much as possible, since it gains performance
- Most of computer architects are NOT aware of security in general and side channel affects in particular.
- “patches” to ease the impact of security issues were proven to cost much performance
  - In some cases, “patches” were disabled since uses prefer performance over security

Intel stock value after the security bugs announced

Agenda

- Intro
- New approach(*)
- Conclusions and next steps

(*) Some parts of the new approach are under provision

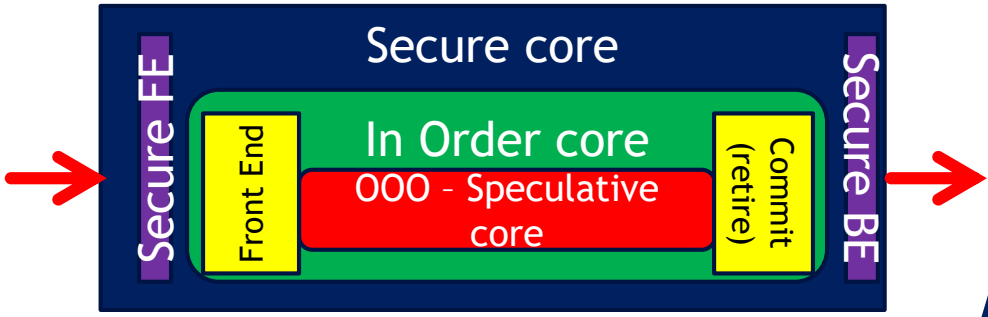
Back to basics – in-order vs. out-of-order

- Modern systems are “hiding” the speculative execution with in a simple, in-order, non speculative cores
- From the external observer point of view the “committed state” of the system is identical for both systems.
- We design to OOO part of the system so that we could imposes the “in-order view”

Secure cores

- I don't believe we can take a core and make it secure (this is what we are currently try to do)
- We need to "design for security" (resembles, design for testability)
- From an external observer point of view, the system should looks like secure, but integrally it may differ; e.g.,
  - Internally, I different execution paths may consume different amount of power
  - We need to develop a mechanisms that for an external observer, the execution power will not depends on the values

- New optimization point → maximum security with minimal cost in terms of power and performance



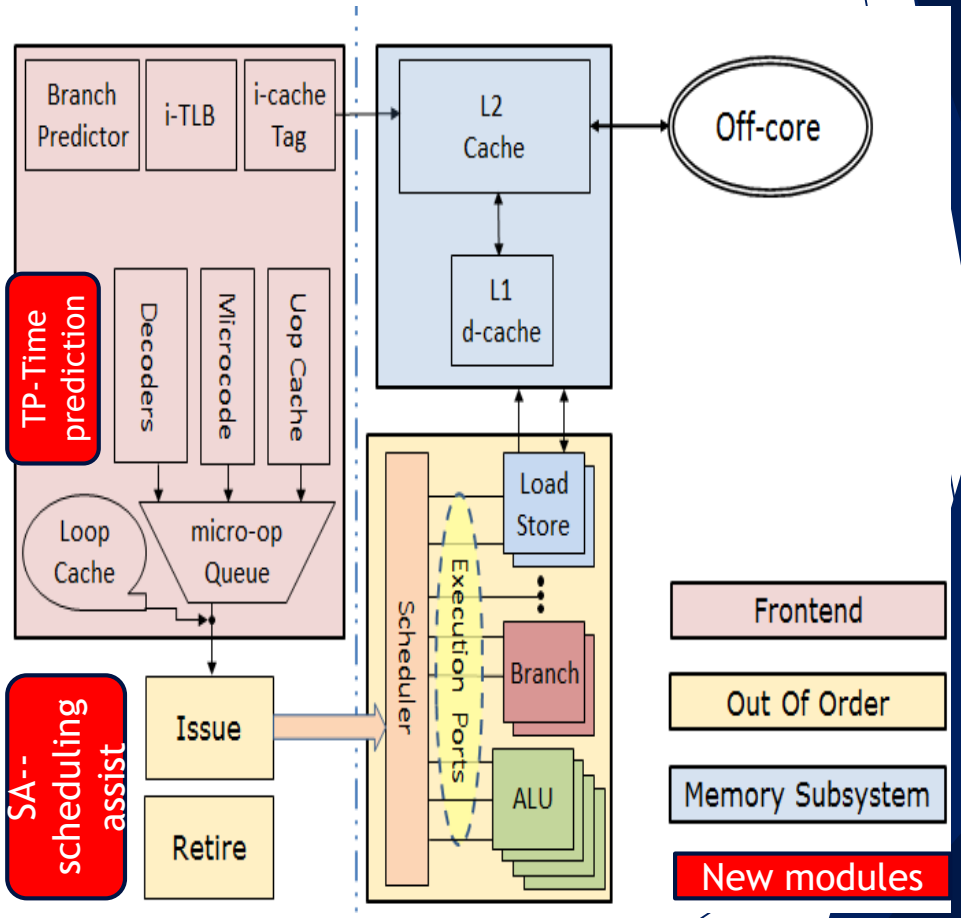
please note that current systems try to balance all execution paths, here we only request that the execution paths will not be dependent on the values

A possible secure core architecture(*)

- SPA resistant mechanism
- In order to balance all execution paths, we can impose (by hardware) that commit will take the same time for all execution paths.
- Please note that

1. Instruction are executed OOO
2. The impact of commit time on performance is minimal (assuming that kilo-instruction ROB)

In this case we can make the external observer to measure same execution time to all paths, w/o significantly impact performance (power)



(*) More details to appear in the SOCC paper (soon)

Agenda

- Intro
- New approach(*)
- Conclusions and next steps

(*) Some parts of the new approach are under provision

Conclusions and next steps

- In order to build a secure core we need to develop a “design for security” methodology
- In this presentation I show one possible alternative to implement such a core
  - Preliminary simulations indicates that with minimal cost in performance (<5%) and power (<1%) we can build an high performance secure core.
- The fun have just began, so the answer to the Yale’s challenge “**Machine Learning and Quantum Computing; Is there anything else worth working on?” --** is


THaNK
YOU