CS 5341

Oscar Galindo

**Enabling Design Space Exploration for RISC-V Secure Compute Environments**

This paper begins by exploring the importance of software simulators for architectural simulations. As the paper points out “Considering the importance of security in computing architectures, it is imperative to enable architectural simulation of hardware for accelerating security such as Trusted Execution Environments (TTEs).” The paper further elaborates that such “microarchitectural simulations” are fundamental to assess and consider the tradeoff between security and performance on future architectures. As the paper mentions, to perform architectural simulations two types of tools exist, on one hand those that provide functional analysis but lack detailed timing/cycle-by-cycle analysis, on the other extreme there exists that so-called RTL or cycle-exact models which provide a detailed cycle-by-cycle analysis of the execution but lack performance and modifiability.

The paper indicates that the focus of the work is to develop the need extensions and models to simulate an already existing, customizable TEE called Keystone on the well-known gem5 arcitectural simulator. On this last matter the paper offers as a main-contribution the implementation of PMP (Physical Memory Protection) hardware simulation on gem5 to simulate Keystone on a RISC-V architecture. In addition, the paper also offers to show Keystone-execution benchmarks with similar performance as those presented by Lee et al. Logically, the paper begins by offering background information about RISC-V’s security. For example, it defines the privileged execution modes U, S, and M – user-mode, supervisor mode, and machine mode-, PMP (Physical Memory Protection), and Virtual Memory Management. Beyond the security enforcement mechanisms of RISC-V, the paper also provides background information about the Keystone Security Monitor (SM), which is a part of the system that guards access to resources. The paper further explains the uses of SM and PMP mechanisms through SM and PMP’s intervention in the lifecycle of *enclaves* – user level applications with access to the TEE’s resources- which are creation, execution, and destruction. Finally, the paper offers background information about gem5’s support for different kinds of simulations including those that support a full-on operating system (FS or Full System) and those with a more focused look at calls within the system once a program runs (SE or System Emulation).

The paper then continues the narrative by outlining the interactions between the PMP, the MMU (Memory Management Unit), and the ISA of the processor. The paper provides some explanation about the arrangement of those elements in the context of Keystone Memory management and directs the interested to the link <https://github.com/darchr> to find the implementation of the needed PMP to simulate Keystone. With such implementation the paper proceeds to perform validation experiments. Among the experiments are simple memory access using the Linux Busybox software, simple functionality tests, and benchmark operations. As observed, similar “slowdowns” are shown on benchmark operations as those slowdowns shown in the work by Lee et al. Also, it is shown among other results that the throughput of instructions in a trusted system (i.e., having Keystone) is like those untrusted (i.e., not implementing Keystone) if the class of CPU is not changed (i.e., in-order vs. in-order and single-cycle vs. single-cycle). Furthermore, the paper provides an expected result which is that trusted simulations take longer to execute that their untrusted simulations.

Finally, the paper offers some use cases for the simulations that can be performed once the PMP implementation is utilized in conjunction with gem5. Among such uses cases the paper points out the analysis of microarchitectures to understand performance, which the paper shows as result of the analysis at the microarchitecture level that the “slowdowns” in trusted executions are caused by a bigger number of instructions. A secondary use case is that of performance analysis of memory encryption, to which the paper shows that even with difference encryption latencies the systems behave similarly in terms of time taken per enclave processed to that of an unencrypted system. In addition, the paper shows that for independent loads and stores of different sizes of buffers the maximum slowdown in comparison to no encryption is around 15% - decent. To further test the results for the second use case, the paper presents an experiment in which the authors vary the degree of encryption with a fixed latency of 30 cycles and conclude once encryption is determined to be admissible it should be complete since the time taken to process enclaves is similar between all configurations of encryption even with full encryption. The last use case presented is that of micro-architecture impact on performance of trusted execution. The experimentation on this case is done by simulating a single-cycle system and an in-order system, each with three different possible configurations of memory and cache subsystems with either a trusted execution or an untrusted execution. As shown in the paper, the results are that in all cases, either with an extreme or simple configuration of memory resources, executions are similar in terms of performance penalties to those of similarly configured untrusted executions.

As usual the paper offers a conclusion to the work and a discussion section. The discussion section is more a best wishes letter hoping people are now intrigued to continue this line of research. Lastly, the conclusion is a reiteration of what the paper presented, not much to analyze in both cases.

My opinion of this paper has several elements. I believe the paper does a great job at explaining and naturally presenting the motivations to perform such a work to develop a workable PMP simulation. In fact, up to figure 6 the results are undisputable and logical. Nevertheless, I believe that when it comes to figures 7- 10 the use of the term “normalized” I feel uneasy about the normalization methodology as it is obscure and not explained thoroughly. Secondly, I believe the last experiment of the 12 configurations running the benchmark experiments was very illuminating as to how untrusted executions are not as performance effective in comparison to trusted executions. Furthermore, through the analysis given to trusted environments and engrained encryption I think the paper makes very reasonable the argument of including protections for enclaves and for memory, which honestly, I think is a great idea for industry. Lastly, I believe that we could extend the analysis to other applications, of course time is short, but I believe the benchmarks are important to be tested with more intensive applications.