Summary

This paper presents a virtualization techniques based implementation of TrustZone on ARM architecture. TrustZone is a hardware feature of ARM architecture and is widely used when implementing a trusted execution environment (TEE), which hosts security sensitive services like secure payment, Digital Rights Management (DRM). The author argues that virtualization techniques could be used to achieve the same goal since it is superior to the hardware solution in terms of flexibility and security and TrustZone could be removed from the chip, saving die space to add other hardware features. Some concerns about this proposal are addressed as well. One is performance, another is some features of TEE not supported by the virtualization solution and the third is backward compatibility.

In order to convince the readers that virtualization based TEE is a better solution. The author states that it can support multiple isolated TEEs in contrast to TrustZone, which only support one TEE. Virtualization also makes it possible to put different security services component into different VMs and allows for sandboxing of untrusted kernel modules. A modified version of Xen called passive Xen hypervisor aka pXenas is used as the underlying hypervisor. This modified version of hypervisor is invoked only when the guest operating system makes an explicit hyper call. By doing so, vCPU scheduling is no longer needed and the performance of the guest operating system is close to native execution. To support this claim, the paper presents the results of LMbench micro-benchmark. Security features such as secure I/O and secure memory are inherently supported using virtualization. Secure boot and cryptographic keys storage are not supported directly but the author claims that there are workarounds. They also argue that backward compatibility is not an issue at all because one could run existing TEE on top of the hypervisor.

Pros

- The idea of using virtualization to implement trust execution environment

Cons

- The benchmark has nothing to do with the performance of virtualization based trust execution environment
- Does not address the issue of how to support secure boot and cryptographic keys storage properly
- Does not provide enough detail of how to ensure backward compatibility
The trusted code base is huge
Many assumptions are flawed and not convincing. For example, it claims that IPI will incur significant overhead without clear evidence.

Criticism

This paper is of extreme low quality. The work it presents has nothing to do with the claim and the design decisions are unsound. It is valueless except for its idea of passive hypervisor.

It is true that in the context of TEE, no vCPU scheduling is needed. TEE should only take control when the guest operating system explicitly invokes a security service. Consequently, disabling it might reduce the unnecessary overhead and improve performance. However, the results of LMbench micro-benchmarks is useless because it does not involve trust execution environment at all. The trust execution environment will provide security sensitive services, which for sure does not exist in LMbench. All the tests involve the guest operating system and the hypervisor only. Moreover, LMbench, written in 1996, is not suitable for modern architectures since the working set of LMbench is too small. Even in such a loose environment, the overhead of context switches is still not negligible. According to table 2, pXen incurs a significant 55% overhead. The second technique to eliminate vCPU scheduling by disabling idle domain will have a negative impact on the battery life of mobile devices. WFI instruction will cause the device entering low-power standby state. After deprivileging it, the device will never enter low-power standby mode, wasting battery and doing nothing.

Using super pages does not necessarily improve performance. Admittedly, increasing page size results in high TLB hit ratio and small page tables, all of which might improve performance. However, doing so will increase page fault latency since more contents need to be read from disk. To support this claim, the paper should have provided the macro benchmark results of using different page size. I doubt that a 2M page is a better choice on mobile platforms. The micro-benchmark does not justify the claim since tests of memory read/write, file read/write are sequential access which favors large page size.

The paper does not provide clear evidence that Inter-processor Interrupts (IRI) is the culprit of high overhead. To support this claim, the author should have done an in-depth analysis instead of drawing such a hasty conclusion.

The paper fails to explain how to support secure boot and cryptographic keys. It merely mentions that these features could be added by modifying hardware together with software layers without providing details of how to achieve it. As for backward compatibility, there exists the same problem. No work has been done in these two parts but they have been mentioned at the beginning of the paper.

The fixed overhead of memory accessing in hardware-based TEE does not necessarily exceed that of a virtualization-based TEE. The author simply claims that
the memory access latency could be mitigated with an effective TLB algorithm, which is not convincing at all since a page table lookup requires permission check as well. A through comparison should be done to support such claim. Personally, I think the hardware-based memory check is superior to software-based solution.

**Benchmark**

The most significant flaw in the benchmark is that the results have nothing to do with the performance of a virtualization-based trust execution environment because it does not involve TEE at all. Besides, without macro-benchmark, readers could not tell the actual performance in real-world cases. From the huge overhead of context switches, personally, I think the performance of pXen is unacceptable.

Moreover, the number of runs of the micro-benchmark is not given, impairs the reliability of the standard deviation. The CPU utilization is also unknown and thus the result of the overhead is also unconvincing. pXen might incur even more overhead than presented.

**Conclusion**

The paper is valueless. It lacks a large portion of the benchmark results to support its claim. Besides, it also fails to address some issues proposed at the very beginning.