Exploring Operating System Mechanisms for Practical and Resilient Confidential Computing

Background. Cloud computing underpins critical services in healthcare, finance, and AI. Yet, it faces a constant threat of cyberattacks and data breaches. In 2024, these cybercrimes exposed over 280 million records in the US's health sector alone. Data breaches are not only financially devastating for the cloud providers hosting customer services, costing up to $4.4 billion in remediation on average in 2025, they also erode customers' trust in cloud providers.

To regain trust, cloud providers are now strengthening the security of their own infrastructure and isolating their customers' systems from it to minimize the risk of compromises in their platform from permeating to customers. In the latter case, particularly, providers are increasingly adopting specialized hardware from compute vendors (e.g., Intel, AMD). These are new processors with secure primitives built into the silicon that enable the creation of a secure "bubble" on the platfom, called a Trusted Execution Environment or TEE, which protects sensitive data and programs, even if the rest of the system, even the cloud administrator, are compromised.

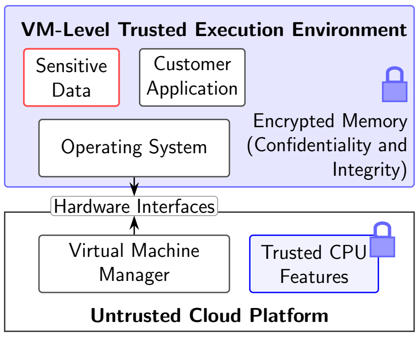

Figure 1: Modern TEE design

After the initial exploration of TEE designs, the community has settled on the idea of isolating an application along with its complete execution environment, including libraries and supporting software, within a single TEE. This design has helped migrate many applications into TEEs with reduced performance and engineering costs. Especially, companies have been deploying cutting-edge workloads such as large language model inference into TEEs for their protection with only trivial adaptation.

Challenges of modern TEEs. This direction in TEE design introduces two new security challenges that I plan to tackle in my postdoctoral studies. The first challenge lies in the huge amount of code now being placed inside a TEE, on which the security guarantees for the customer's application and data depend. This code (called the trusted computing base or the TCB of an application) includes the operating system, often consisting of several million lines of code and known to have thousands of vulnerabilities that the adversaries could exploit to compromise the sensitive application. Thus, making TEE code resilient against compromises, ensuring that vulnerabilities have minimal impact on the security guarantees offered to cloud customers, is a key research challenge for the community.

Second, as a component of TEEs, the role of operating systems must now be rethought. Unlike normal applications, the operating system's primary task is to manage system resources, such as memory and files, by utilizing its access to low-level hardware mechanisms that provide fine control over software running on the system. Importantly, the operating system's high degree of control not only enables transparent resource management but also offers powerful tools for building practical in-TEE security mechanisms. The work conducted during my doctoral studies demonstrated the promise of this approach, leveraging OS capabilities to implement practical TEE self-defense mechanisms, particularly against side-channel attacks. Building on that foundation, I plan to tackle the challenge of exploring a broader class of OS-supported techniques that address the issues in practical TEE deployments.

Toward taming large TCB. Securing the application TCB is an essential challenge given the complexity of modern TEEs. Over the years, research has sought to tackle this challenge in three ways. The first approach proposes comprehensively testing the code placed inside the TEE to detect mistakes. However, applying this to a large codebase, such as the OS, would require an insurmountable effort and is unlikely to find all the bugs. The second approach is compartmentalization, which splits a large codebase into smaller, isolated components. The intuition is simple: if each component is placed in a "cage" where its failures cannot spread, then the system as a whole becomes more robust to attacks. Still, the decision of which components to place these cages into is largely unexplored. The third approach is TCB reduction, which seeks to minimize the code running inside the TEE, based on the belief that code size correlates with security risk. While this principle holds some truth, an exclusive focus on code size is misleading. It can sacrifice system performance by removing features, and more importantly, it overlooks where the risky code actually resides – the means by which trusted and untrusted worlds communicate, or their interfaces.

My proposal focuses on securing the TCB of TEEs around the interfaces. These interfaces, often inherited from legacy systems, were rarely designed with strong adversarial models in mind; yet, they are easy targets for attacks. For instance, recent research has shown that the interfaces where the TEE communicates with the outside world expose critical vulnerabilities that leak sensitive data. Based on this observation, I plan to pursue research in two strategies. First, building on my experience in software compartmentalization, I plan to explore methodologies that compartmentalize code interacting with the interfaces, thereby containing the impact of the untrusted platform on security. Second, I will revisit legacy interface design choices, rethinking interfaces from the ground up with security as a first-class goal, rather than an afterthought. By focusing security decisions at the interfaces where they matter, TEE designs can manage complexity in a principled manner without sacrificing functionality.

Exploration of OS as a security component. My research toward this area will focus on two main challenges with modern TEEs: (i) side-channel mitgation and (ii) private data processing.

Focus 1: Practical Side-channel attacks mitigations. *Side channels* are a critical yet persistent class of vulnerabilities at the software-hardware boundary of TEEs. They are indirect signals – such as the time difference measured when accessing a memory location – through which an attacker can infer private information. Such attacks are widely acknowledged as a formidable threat to CCC (AWS, Azure, IBM, and Alibaba). Efforts to completely remove side channels often run into roadblocks: some side channels are buried deep in the processor's internal circuitry, requiring massive hardware redesigns to eliminate; others come from everyday cloud resource management operations, and removing them would impair flexibility.

A more practical, less intrusive approach is to mitigate these side-channels by obscuring side-channel signals from programs. However, current state-of-the-art mitigation techniques fall short; systems such as Klotski and Obelix are complex to adopt as they require developers to rewrite or recompile their entire software. Even when applied, these systems can slow programs by hundreds or even thousands of times. Worse, to remedy the severe performance hit, security trade-offs are often made, but they are ad hoc and lack a sound theoretical basis.

At UBC, my research will tackle these challenges under the supervision of Prof. Aastha Mehta, who has deep expertise in both systems security and OSes. I will also enrich this research with perspectives from UBC's researchers, especially systems and cloud computing researchers at Systopia Lab and CIRRUS Lab. Plans for these collaborations will be detailed in the remainder of this proposal.

My postdoctoral work will tackle the limitations of previous research from an OS vantage point. I aim to extend the limited prototype of my doctoral research implemented on a minimal OS by incorporating side-channel mitigation into Linux, an OS most widely used in cloud data centers. Also, I plan to explore theoretical models to enable principled trade-offs that enable better performance.

This direction synergizes with the collective expertise at UBC. Primarily, my supervisor, *Aastha Mehta*, with profound experience in side-channel attacks on TEEs and formal models for side-channel defenses, would provide invaluable feedback to the research. Further, the work of userspace memory management by *Margo Seltzer* and *Alexandra Fedorova* will play an essential role in making the side-channel mitigation technique using memory management practical in a complex OS such as Linux. The large TCB of commodity OSes also makes side-channel elimination particularly challenging – it requires tracking potential leaks across millions of lines of code. To this end, the study of cross-system *information* *flow tracking*, pioneered by *Margo Selter* and *Thomas Pasquier*, will make it a tractable problem.

Focus 2: Private data processing. A common scenario for CCC deployments is *private data processing* involving multiple stakeholders, as shown in Table 1. For instance, companies like 23andMe provide personal health data analytics as a cloud service. Its users upload private data to the service's server, which runs a health analytics program on the data and returns the results to the user. The service must handle users' personal data carefully – this is, while mandated by regulations like GDPR, often not trusted by users. A dilemma arises: On the one hand, the user wants to verify that the service is using their data responsibly, for example, never giving it to third parties. On the other hand, to prove this, the service must publicize its data processing to users for verification, which might contain trade secrets.

Stakeholder	Sensitive Asset	Security Interest
Service users	Personal data	Ensure data confidentiality
Service providers	Proprietary data processing	Keep trade secrets confidential

Table 1: Representative private data processing stakeholders, and their assets and interests.

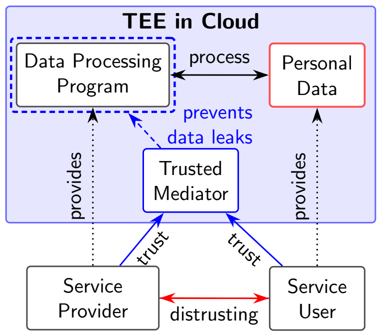

Figure 2: How CCC resolves stakeholders' conflicts.

To resolve this *conflict of interests*, existing deployments employ a mediator in a TEE, which is trusted by both service users and providers (Figure 2). The service provider first sends the data processing program to the mediator, trusting it to keep the program secret. Similarly, the users provide personal data without needing to examine the program. On users' data, the mediator now runs the service provider's program in a way that strictly maintains the confidentiality of users' data.

Unfortunately, most research prototypes addressing this challenge fail to introduce a deployable solution. Recent solutions like PAVE and Erebor achieve security by placing the program inside a sandbox. The sandbox enforces strict information flow control on the program, for instance, preventing it from saving personal data to a file or communicating with other programs. As a result, this approach severely limits flexibility and compatibility with many workloads, especially complex ones and ones involving cooperating programs.

My proposal looks to incorporate a trusted mediator into the OS to improve usability. This is achieved through dynamic information flow tracking, a common technique in OS research. Using this, the mediator would prevent only the point at which sensitive data is about to be extorted, while allowing benign file accesses and network operations. Thus, the private data processing system would be compatible with a wide range of cloud workloads. I envision that whole-system information flow tracking expertise from *Margo Selter* and *Thomas Pasquier* will be of tremendous help. Scaling the trusted mediator design across cloud machines would also require distributed cloud computing expertise, which I plan to employ the help of *Mohammad Shahrad* from the ECE department.

1 Appendix: Guide (REMOVE IN FINAL)

The proposal should be written in clear, non-technical language that allows a non-specialist to comprehend the overall content and importance of the work. Members of the Killam Postdoctoral Fellowships and Prizes Committee are from abroad range disciplines and may not have expertise in your area of study.

Although the fellowship may be used to extend or expand upon doctoral work, it must be made clear that you are not intending to use the award to wrap up a thesis. While it is expected that a postdoctoral fellow will be taking the next step beyond the PhD thesis, you must differentiate clearly between the postdoctoral project and the thesis research. Feel free to include hyperlinks to provide links to additional information.

As applicants must make UBC their base, it is important – particularly for applicants whose primary research materials are elsewhere – to indicate what travel is involved, to where, and for how long. You should describe how you will deal with the remoteness of the primary materials.

The adjudication committee is very interested in “fit” with the selected UBC department or unit and the university’s research programs. You must provide information on how your research relates to that of specific campus programs and advisors. If a colloquium is envisioned, a possible title should be proposed. If you will visit in classes, please suggest which ones. Since inter-disciplinarity is often a valuable dimension (the Killam Trusts declares that a candidate shall not be “a one-sided person”), specific details on proposed inter-departmental connections are welcome.