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, and in 2025, they cost up to $4.4 billion in remediation on average, worldwide. This is not only financially devastating but also severely eroding public trust in the cloud.

To restore trust in cloud computing systems, cloud providers have started to isolate their customers' data from their own platforms by adopting Trusted Execution Environments (TEEs), built on top of modern CPU security features. TEEs protect customers' data against a strong threat model, resistant to even attacks from a cloud administrator. TEEs form the foundation of Cloud Confidential Computing (CCC), a paradigm shift toward trustworthy private data processing in the cloud.

The latest TEEs can now fully isolate virtual machines (VMs), made possible by technologies like AMD SEV (Figure 1). This improves usability by safeguarding existing workloads with minimal modifications, while also enhancing their performance. However, this means that TEEs must now protect the entire software stack of the VM – referred to as its Trusted Computing Base (TCB). In this picture, the Operating System (OS) is particularly problematic for its huge code size. As of 2025, the Linux kernel contains around 40 million lines of code – seeing over 3,500 vulnerabilities in 2024 that attackers can leverage. Furthermore, the OS's complex interactions with the underlying hardware – now being virtualized by the untrusted platform – complicate efforts to secure CCC.

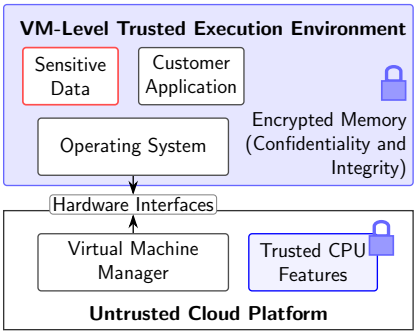

Figure 1: VM-Level TEEs in Cloud Confidential Computing

Proposal overview. This proposal seeks to make CCC more robust and practical, building on the observation that the industries' shift toward VM-level TEEs also exposes a golden opportunity: the OS can now actively shape the security of CCC. My doctoral studies demonstrated the potential of this direction by designing INCOGNITOS, a side-channel mitigation technique for CCC that harnessed OS abstractions, such as memory management and task scheduling, showing that they can be employed for robust and deployable security. Building on that foundation, I will explore a broader class of OS-supported techniques that address the practical challenges of CCC deployments.

At UBC, I will pursue this agenda under the supervision of Prof. *Aastha Mehta*, who has deep expertise in both systems security and OS. 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, along with the focuses of my postdoctoral research: *side-channel attacks* and *private data processing*.

Focus 1: Side-channel attacks. *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.

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.

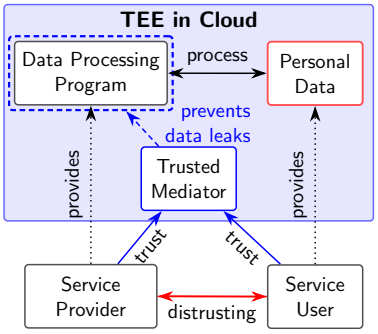

Figure 2: How CCC resolves stakeholders' conflicts.

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 exorted, 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.