OLD DOMINION UNIVERSITY

Security and Privacy for Client-Hosted Agents via Trusted Hardware

Exploiting Out-of-Enclave Memory for AI Integrity and Confidentiality

Jonathan Takeshita
Department of Computer Science and School of Cybersecurity, Old Dominion University

Oct. 30, 2026

About the Speaker

Assistant Professor @ ODU CS/CYSE. Research areas: AI security, private bioinformatics and federated learning

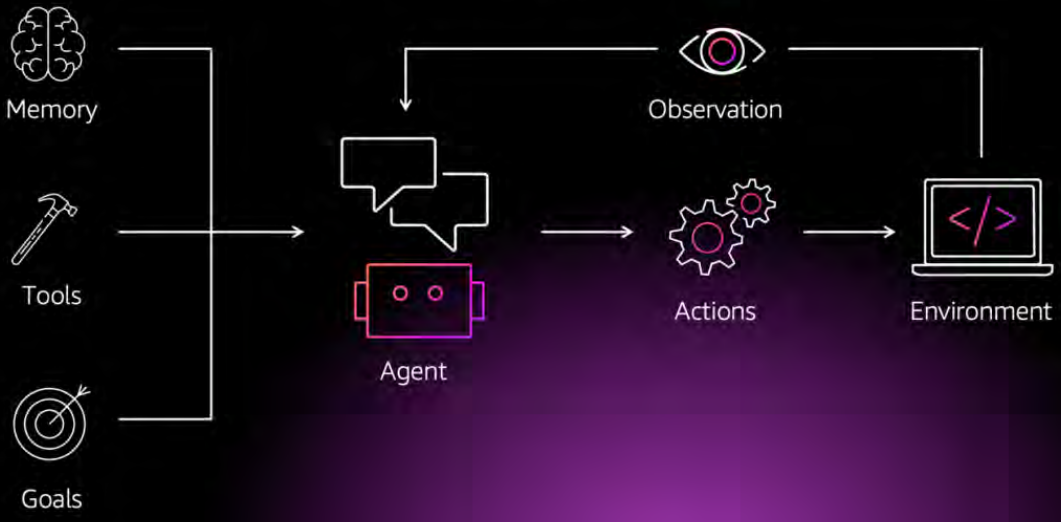
My research philosophy:

Bespoke solutions to specific problems yield general insights and innovation

My tools:

- Homomorphic encryption: mutate data while it remains encrypted
- Private Stream Aggregation: collect and compute privately on Internet-scale data streams
- **Trusted Hardware: processor instructions providing security against a hostile host system**

What is agentic AI?



Secure/Private Agents

Agent execution can take place at the client or remotely - need to protect model/inputs from a *hostile host system*

What are the tools and challenges?

Secure/Private Agents

Agent execution can take place at the client or remotely - need to protect model/inputs from a *hostile host system*

What are the tools and challenges?

- Tools: trusted hardware
- Challenges: agents use very large models (e.g., LLMs)

Our goal: both security and scalability for an agent's AI model components

Trusted Execution Environments

- TEEs use hardware isolation, memory encryption, etc., to protect code/data in an *enclave* from the host OS
- Some TEEs (e.g., AMD SEV, AWS Nitro, Intel TDX) have large memory arenas but weaker security [1]
- **Intel SGX** has the strongest security, but limited memory (100s of MB), and heavy performance penalties for memory access

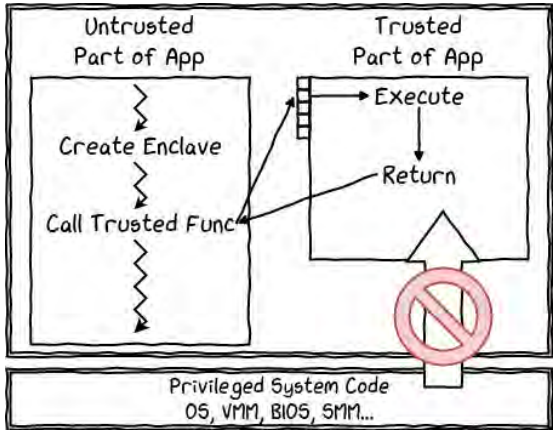
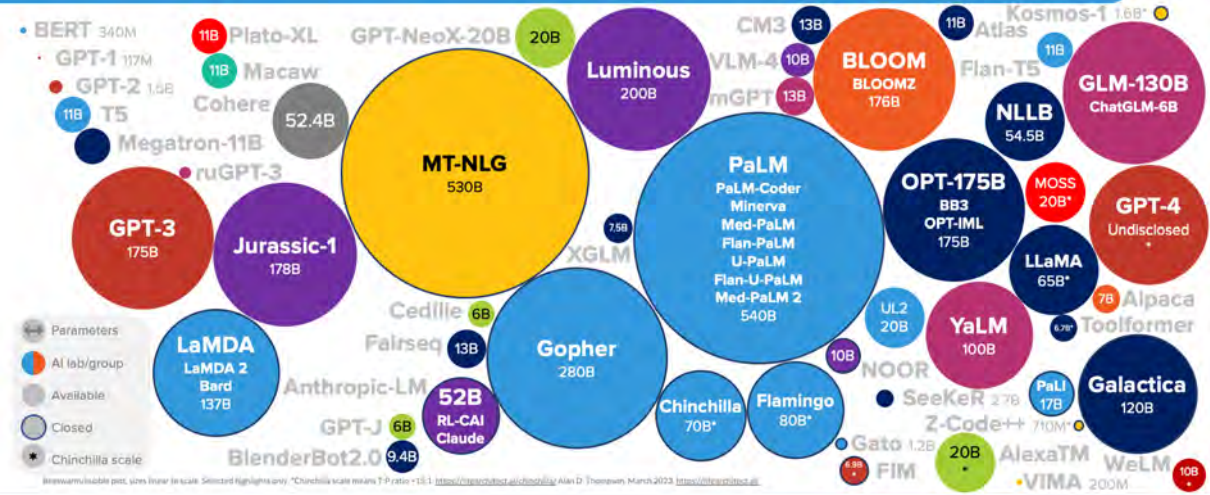


Figure 1: SGX Overview

LANGUAGE MODEL SIZES TO MAR/2023



Arrows/unstable plot, sizes linear to scale. Selected highlights only. *Chinchilla scale means T-P ratio = 19:1 https://lifearchitect.ai/chinchilla/ Alan D. Thompson, March 2023. https://lifearchitect.ai/

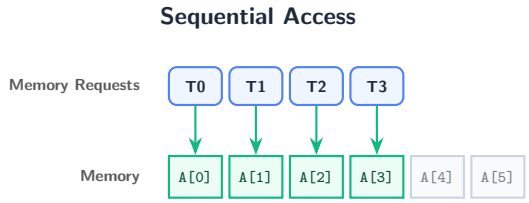
LifeArchitect.ai/models

Problem

We wish to efficiently protect AI inference (model, input, and intermediate values) at an untrusted node, as if it were all in the enclave.

Key insights:

- SGX protects *any* memory accesses - inefficient at scale
- AI inference isn't an arbitrary memory access pattern - predictable objects (matrices), viewed in a predictable order
- SGX can also access untrusted memory *outside* its enclave, much more efficiently!



Method

Problem: Modern AI models cannot fit into the small enclaves of high-security TEEs with multiple concurrent tenants

Solution: Keep large model matrices A outside the enclave in untrusted memory, and use the enclave to store some (smaller) bookkeeping information to ensure security

Method

Problem: Modern AI models cannot fit into the small enclaves of high-security TEEs with multiple concurrent tenants

Solution: Keep large model matrices A outside the enclave in untrusted memory, and use the enclave to store some (smaller) bookkeeping information to ensure security

Confidentiality through masking:
Store $A'_{i,j} = f_k(i,j) \oplus A_{i,j}$ in untrusted memory, and keep the random function key k safely in the enclave

Integrity through signatures:
Whenever A' is read from untrusted memory, validate $\sigma_A = \sum A'_{i,j} \cdot g_k(i,j)$ is the same as the last time A' was read

System Diagram

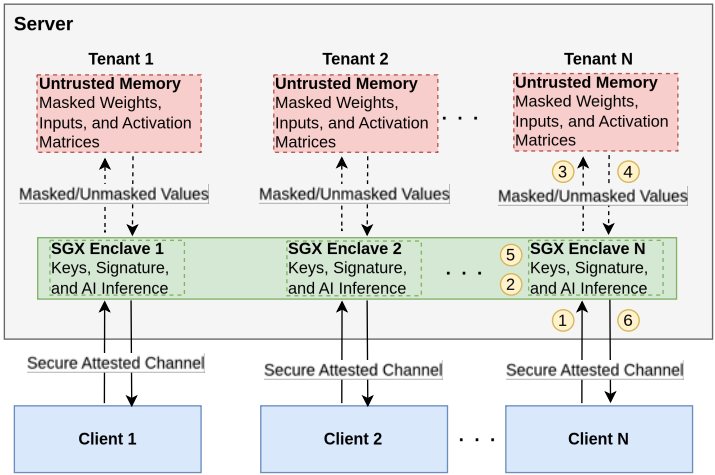


Figure 2: Multi-tenant deployment on a single cloud server. Untrusted memory arenas can be much larger than SGX enclave memory.

Sample Algorithm

```
def matrix_scalar_multiply(A, f_A, g_A, current_sig_A, c):
  sig_A = sig_B = 0
  B = Matrix(A.length());
  f_B = new_random_function(); g_B = new_random_function()
  for i in range(A.length()):
    A_val = A[i] - f_A(i) # Read element of A into enclave from
    untrusted memory, and unmask
    sig_A += A_tmp * g_A(i) # Update signature for A
    B_val = c * A_val # Perform per-element scaling
    sig_B += tmp_B * g_B(i) # Update signature for B
    B[i] = B_val + f_B(i) # Mask element of B, and write it to untrusted
     memory
  assert(sig_A == current_sig_A) # Check the signature matches the last
    one seen
  return (B, sig_B, f_B, g_B)
```

Listing 1: Matrix Scaling

Listing 1 shows $B = c \cdot A$ for a matrix A stored in untrusted memory. (For simplicity, we use a 1-dimensional matrix.) This strategy can be adapted to matrix multiplication and other AI building blocks.

Experimental Results (single-tenant)

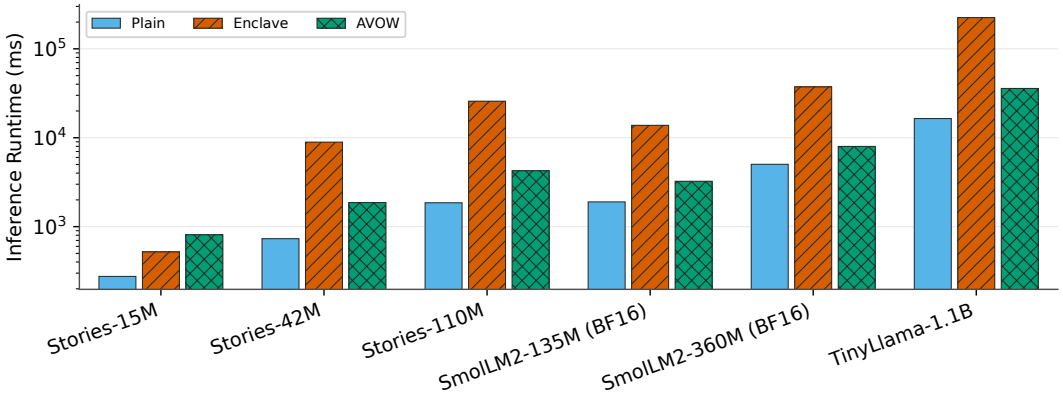


Figure 3: Inference comparison between our method (AVOW) and all-in-enclave naive method, with $N = 1$ tenant.

Experimental Results (multi-tenant)

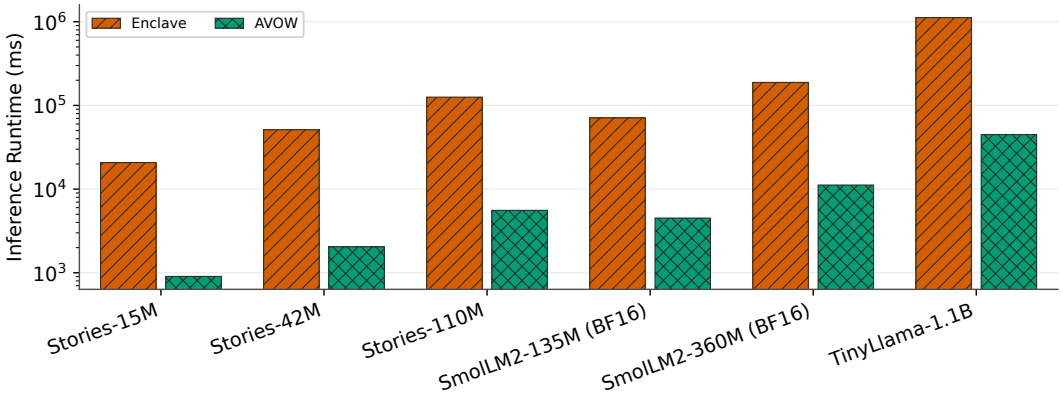


Figure 4: Inference comparison between our method (AVOW) and all-in-enclave naive method, with $N = 8$ tenants.

Acknowledgments and Questions

Special thanks to...

- COVA CCI for funding this research (and the invitation to speak)
- Collaborators at Tokyo Tech (Yang Cao, Hao Du) and William and Mary (Qun Li, Binghao Yan)
- My Ph.D. students and lab (Bikash Thapa, Xuezhengwu Rao, Bongiwe Mkwananzi, Md Shakhawat Hossen)




OLD DOMINION
UNIVERSITY
THANK YOU!
ANY QUESTIONS?

Extra: Related Work

- All-in-enclave: [2, 3, 4, 5]
- Partitioning/offload: [6, 7, 8, 9]
- Expanding trusted domain: [10, 11, 12]
- GPU TEEs: [13, 14]
- Outside-enclave: [15, 16, 17, 18, 19]

Our work keeps data outside the enclave, but with security protections to prevent unauthorized reading or mutation.

References I



P.-L. Aublin, M. Mahhouk, and R. Kapitza, "Towards TEEs with large secure memory and integrity protection against HW attacks," in *SysTEX'22: 5th Workshop on System Software for Trusted Execution, February 28, 2022, Lausanne, Switzerland*, p. 24, 2022.



O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 619–636, USENIX Association, Aug. 2016.



T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," 2018.



D. L. Quoc, F. Gregor, S. Arnautov, R. Kunkel, P. Bhatotia, and C. Fetzer, "secureTF: A secure tensorflow framework," in *Proceedings of the 21st International Middleware Conference*, Middleware '20, (New York, NY, USA), p. 44–59, Association for Computing Machinery, 2020.



K. Grover, S. Tople, S. Shinde, R. Bhagwan, and R. Ramjee, "Privado: Practical and secure DNN inference with enclaves," 2019.

References II

D. Li, Z. Zhang, M. Yao, Y. Cai, Y. Guo, and X. Chen, "TEESLICE: Protecting sensitive neural network models in trusted execution environments when attackers have pre-trained models," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 6, pp. 1–49, 2025.

A. Frikha, N. Walha, R. Mendes, K. K. Nakka, X. Jiang, and X. Zhou, "ObfuscaTune: Obfuscated offsite fine-tuning and inference of proprietary LLMs on private datasets," Tech. Rep. arXiv.2407.02960, arXiv, 2024.

Q. Li, Z. Shen, Z. Qin, Y. Xie, X. Zhang, T. Du, S. Cheng, X. Wang, and J. Yin, "TransLinkGuard: Safeguarding transformer models against model stealing in edge deployment," in *Proceedings of the 32nd ACM International Conference on Multimedia*, p. 3479–3488, 2024.

Z. Lin, S. Zhang, X. Wang, Y. Su, Y. Wang, R. Hou, and D. Meng, "LoRATEE: A secure and efficient inference framework for multi-tenant LoRA LLMs based on TEE," in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2025.

P.-C. Cheng, W. Ozga, E. Valdez, S. Ahmed, Z. Gu, H. Jamjoom, H. Franke, and J. Bottomley, "Intel TDX demystified: A top-down approach," *ACM Computing Surveys*, vol. 56, no. 9, pp. 1–33, 2024.

References III

AMD, "AMD SEV-SNP: Strengthening VM isolation with integrity protection and more," Tech. Rep. 70366, AMD, 2020.

A. Lutsch, C. Franck, M. El-Hindi, Z. István, and C. Binnig, "An analysis of AWS nitro enclaves for database workloads," in *Proceedings of the 21st International Workshop on Data Management on New Hardware*, pp. 1–8, 2025.

Q. Wang and D. Oswald, "Confidential computing on heterogeneous CPU-GPU systems: Survey and future directions," *ACM Computing Surveys*, vol. 58, no. 9, pp. 1–35, 2026.

NVIDIA, "NVIDIA confidential computing." `https://www.nvidia.com/en-us/data-center/solutions/confidential-computing/`, 2026. Accessed: 2026-04-14.

M. Taassori, A. Shafiee, and R. Balasubramonian, "VAULT: Reducing paging overheads in SGX with efficient integrity verification structures," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 665–678, 2018.

References IV


M. Orenbach, P. Lifshits, M. Minkin, and M. Silberstein, "Eleos: Exitless OS services for SGX enclaves," in *Proceedings of the Twelfth European Conference on Computer Systems*, EuroSys '17, (New York, NY, USA), p. 238–253, Association for Computing Machinery, 2017.


T. Kim, J. Park, J. Woo, S. Jeon, and J. Huh, "Shieldstore: Shielded in-memory key-value storage with SGX," in *Proceedings of the Fourteenth EuroSys Conference 2019*, EuroSys '19, (New York, NY, USA), Association for Computing Machinery, 2019.


W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Opaque: An oblivious and encrypted distributed analytics platform," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, (Boston, MA), pp. 283–298, USENIX Association, Mar. 2017.


T. Lee, Z. Lin, S. Pushp, C. Li, Y. Liu, Y. Lee, F. Xu, C. Xu, L. Zhang, and J. Song, "Occlumency: Privacy-preserving remote deep-learning inference using SGX," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019.