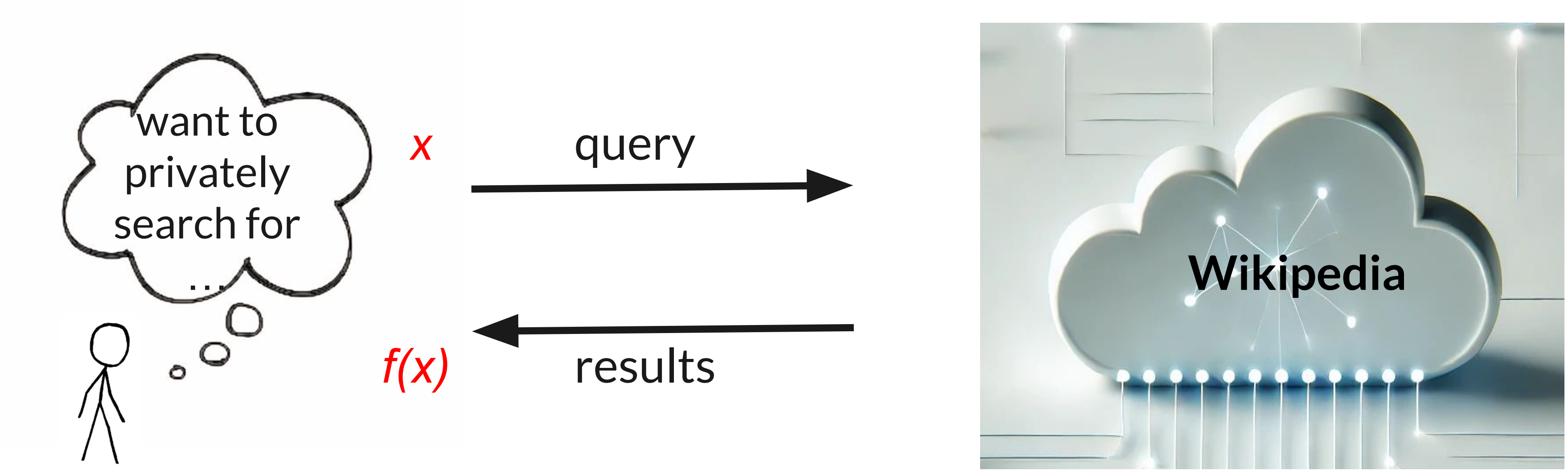
Alcatraz: Secure Remote Computation via Sequestered Encryption in Hardware Security Module

Albert Lu

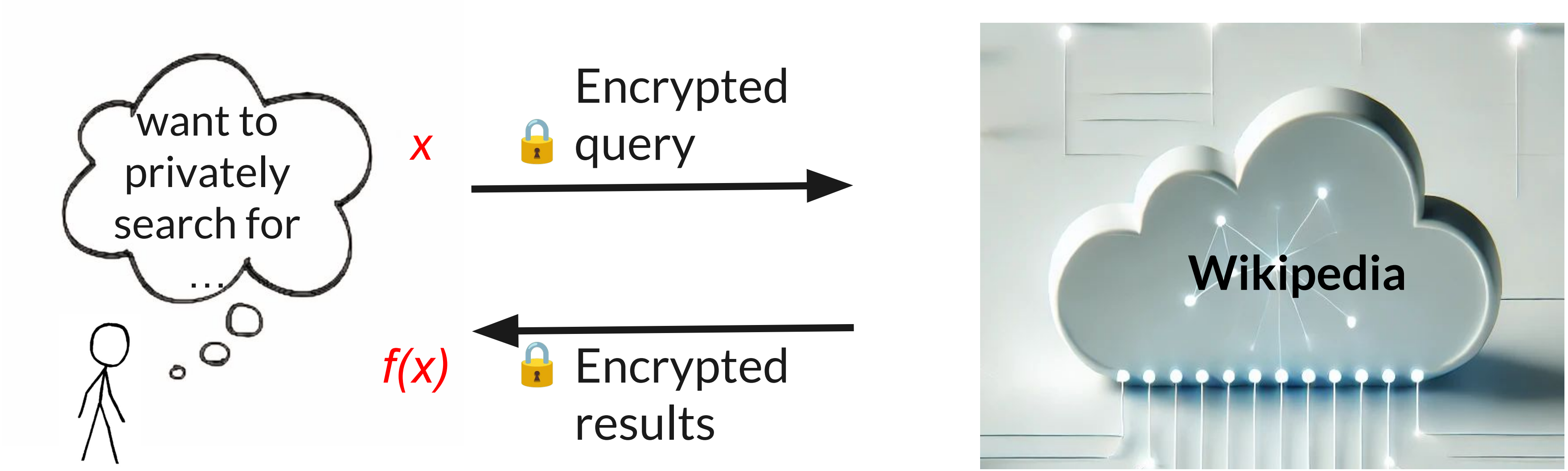
Mentors:
Jules Drean and Sacha Servan-Schreiber

October 12th, 2024
MIT PRIMES October Conference

Secure Remote Computation



Secure Remote Computation



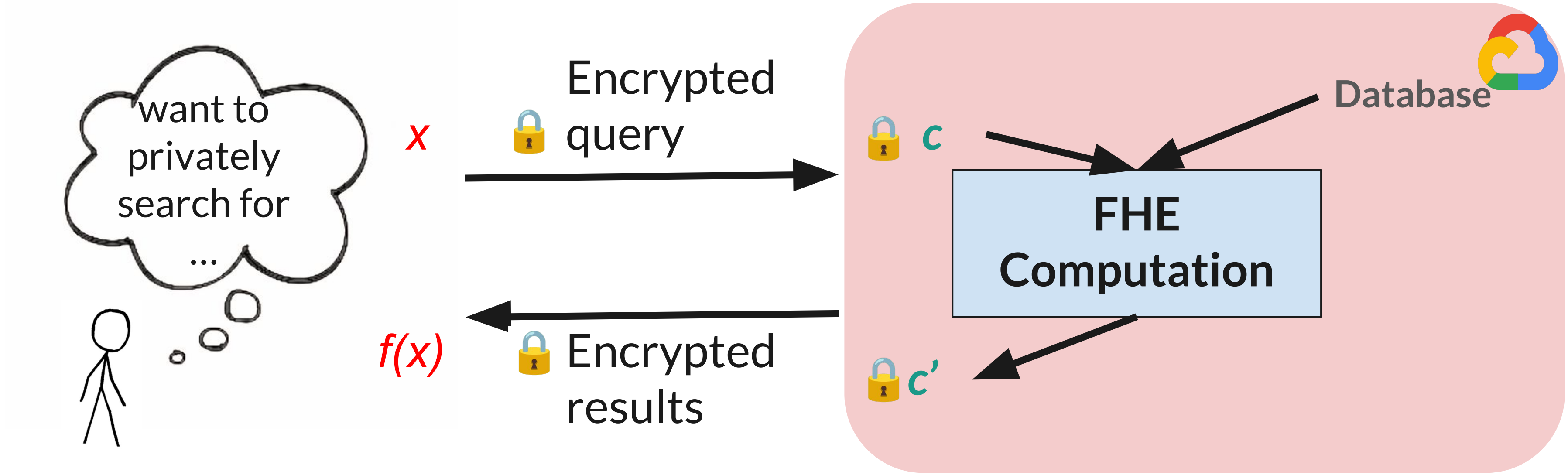
Task: *x* and *f(x) are* sensitive data. Can you query database without revealing what you're searching and your search results?

Private Information Retrieval (PIR)

Secure Remote Computation

	Ideal Solution: Fully Homomorphic Encryption (**FHE**)	More Practical: Trusted Execution Environment (**TEE**)	Our Solution: **Alcatraz** (inspired by both)
Security	Based on strong cryptographic assumption	Based on empirical hardware security; Vulnerable to side channels	Minimal trusted hardware; Protected against side channels
Efficiency	Slow	Fast	Fast
Expressivity	Only compute Logical Circuits	Can run programs	Only compute Logical circuits

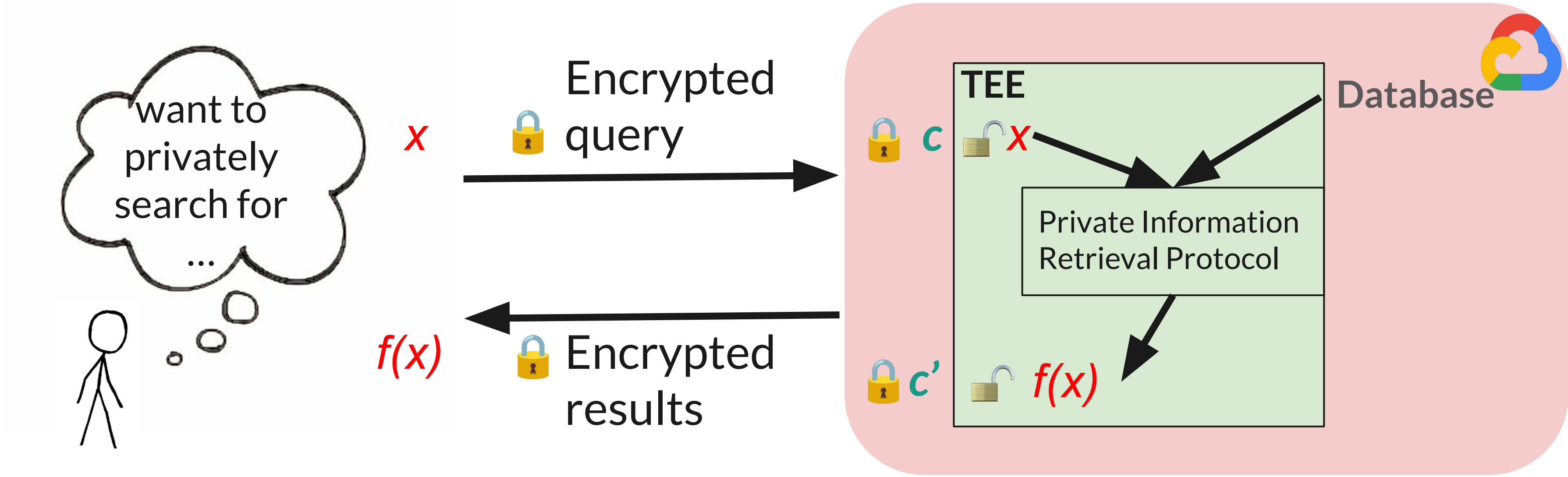
Fully Homomorphic Encryption (FHE)



Task: x and *f(x)* are both sensitive data. We want the cloud to compute *f(x)* *securely* without knowing x and *f(x)*

FHE computes *on ciphertext c (x is never exposed)* ⟹ too slow

Trusted Execution Environment (TEE) - Trusted Hardware



Task: *x* and *f(x)* are both sensitive data. We want the cloud to compute *f(x)* *securely* without knowing *x* and *f(x)*

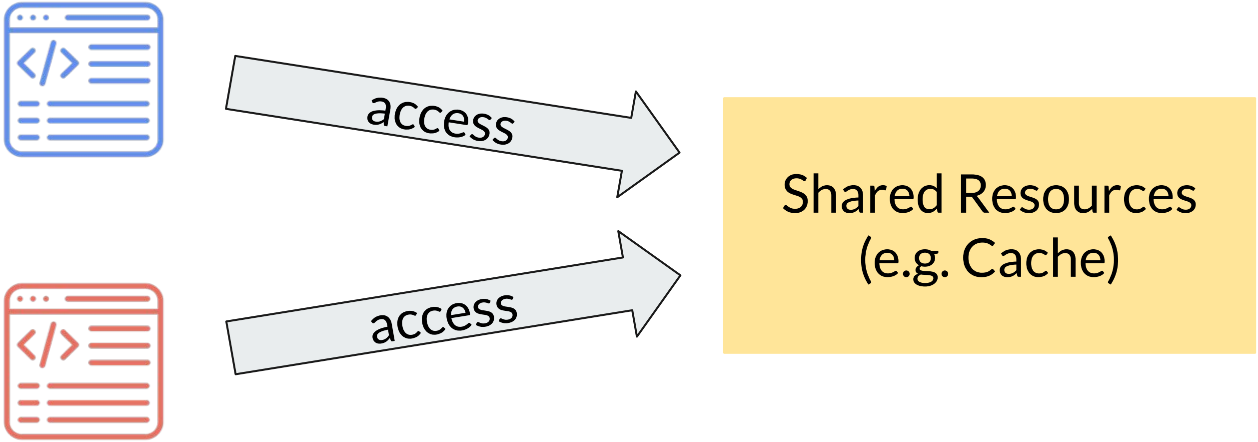
We trust TEE so operation is done on unencrypted info ⟹ faster

Problem: leads to large attack surface, subject to side-channel attacks

What are Side Channels?

Code in TEE

Attacker



Attacker can observe

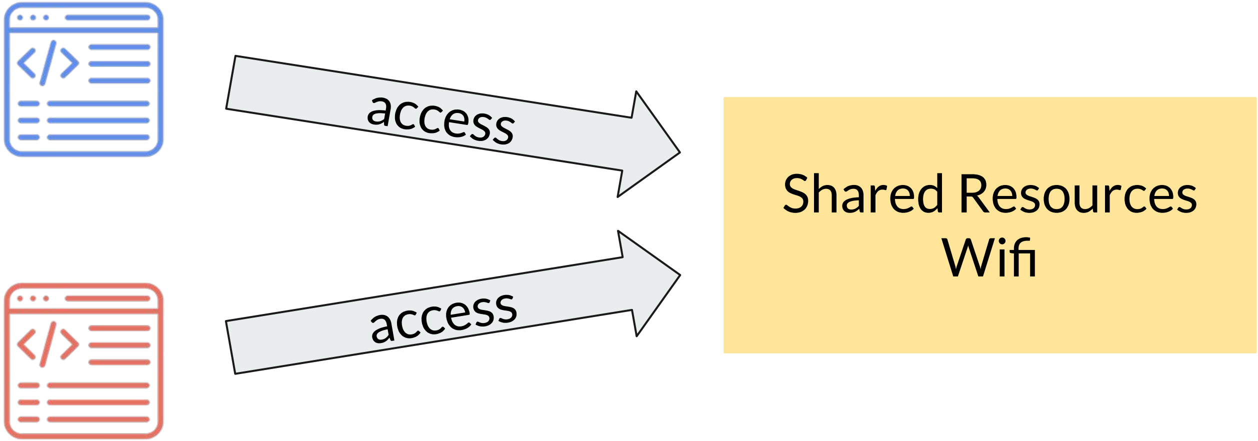
“SECRETS”

One program can exploit shared resources to spy on another

Example of Side Channels

Family using the internet

You watching a movie



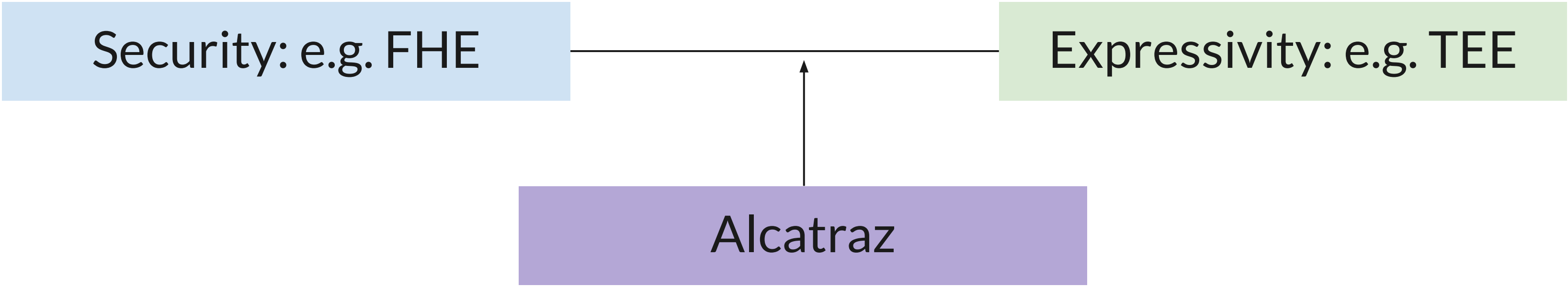
You observe “movie lags a lot”

“family is also using wifi”

Lots of different side channel attacks against Trusted Execution Environments!

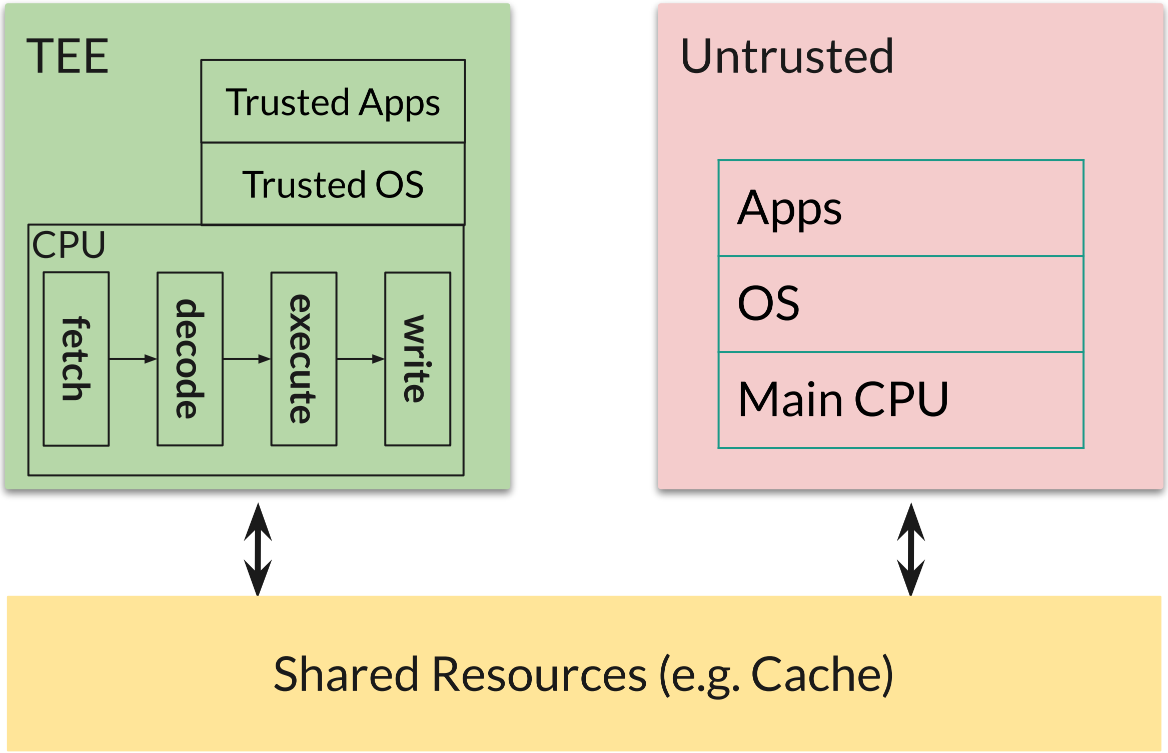
Our Solution to Secure Remote Computation

- Take inspiration from Fully Homomorphic Encryption (FHE) and Trusted Execution Environment (TEE)
- Based on trusted hardware
- BUT reduce our “trusted area” as much as possible
    - Key idea: **reduce expressivity (only compute circuits)**
- Result: **mitigate side channel attacks**

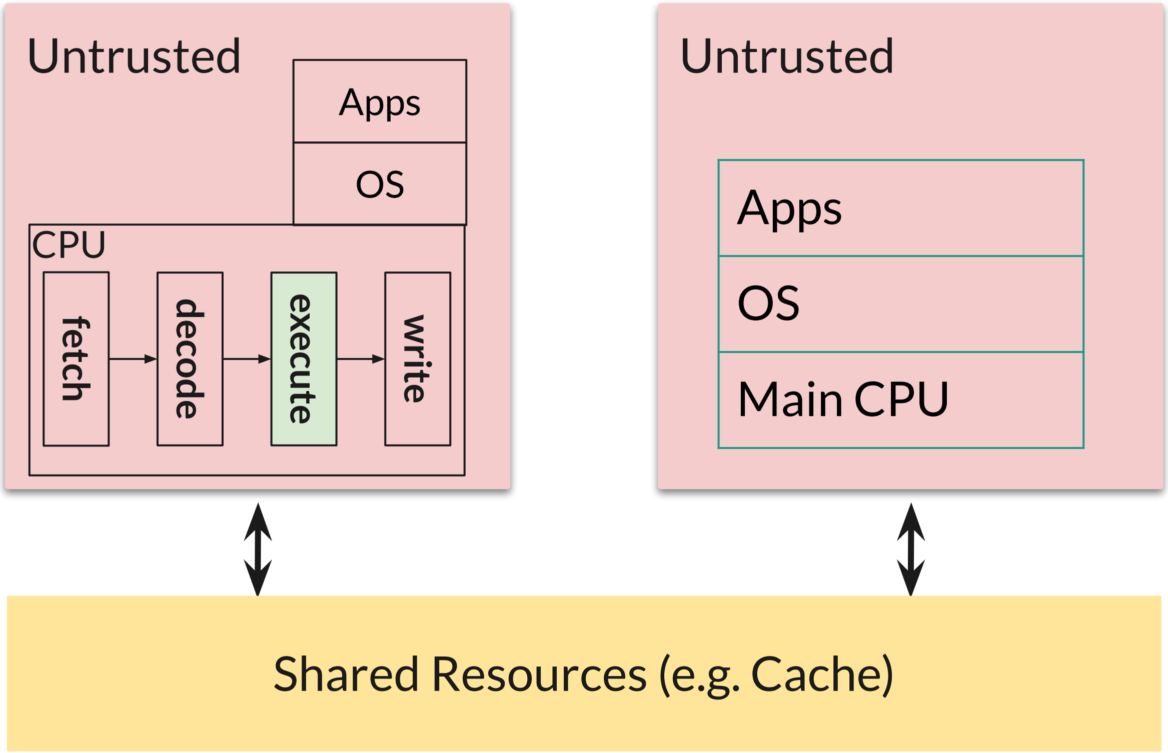


Reducing Area of Trust

Trusted Execution Environment and Shared Resources

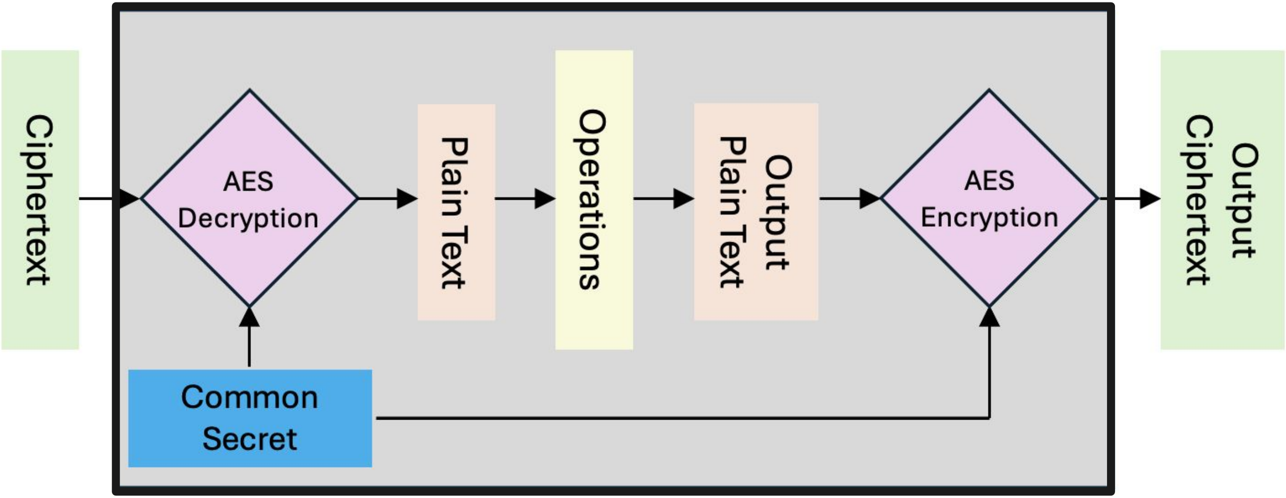


No More TEE

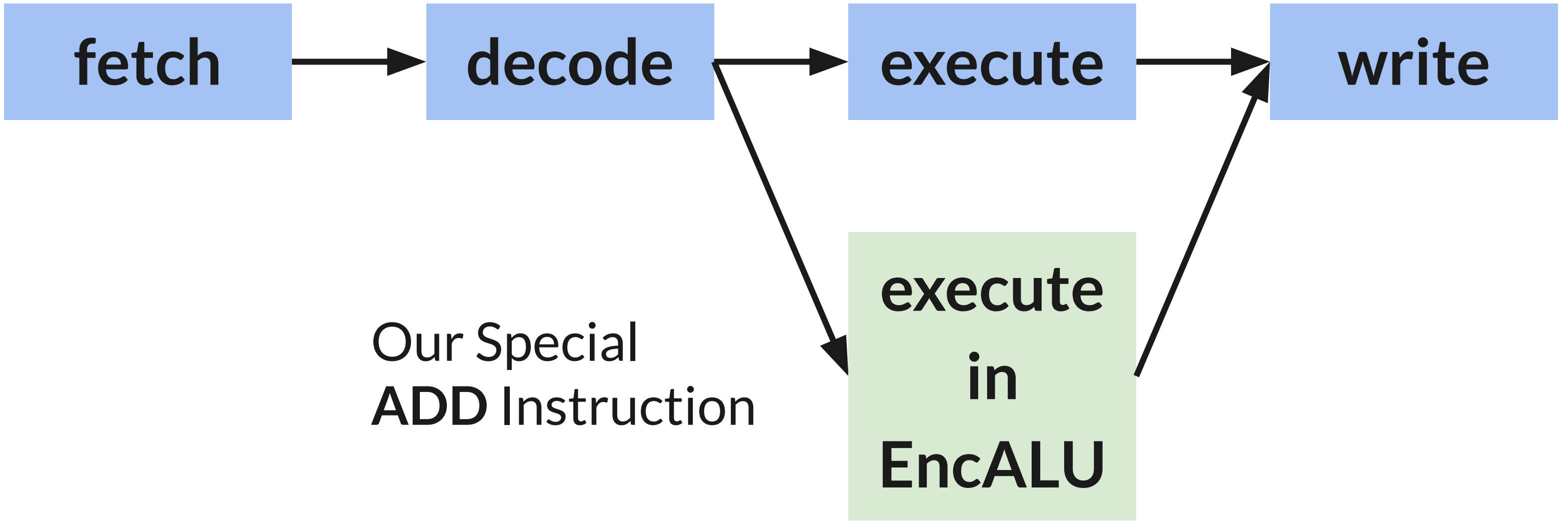

Untrusted
Apps
OS
CPU
fetch
decode
execute
write
Untrusted
Apps
OS
Main CPU
Shared Resources (e.g. Cache)

Encrypted ALU via Sequestered Encryption

- ALU (Arithmetic Logic Unit) operates at the **execution** stage of CPU pipeline
- Alcatraz introduces an Encrypted ALU, which sandwiches ALU operations between an encryption and decryption



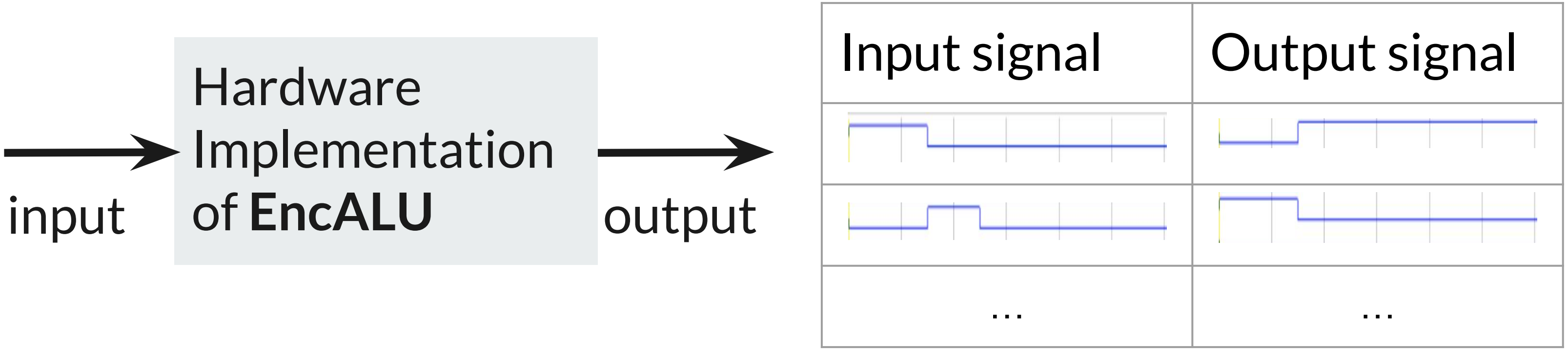
Extended Instructions are Dispatched to Encrypted ALU



Proving Security of Encrypted ALU Against Timing-Based Side Channel Attacks

Formal Verification

- We want to prove our hardware module is secure against **all** possibilities of timing-based side channel attacks

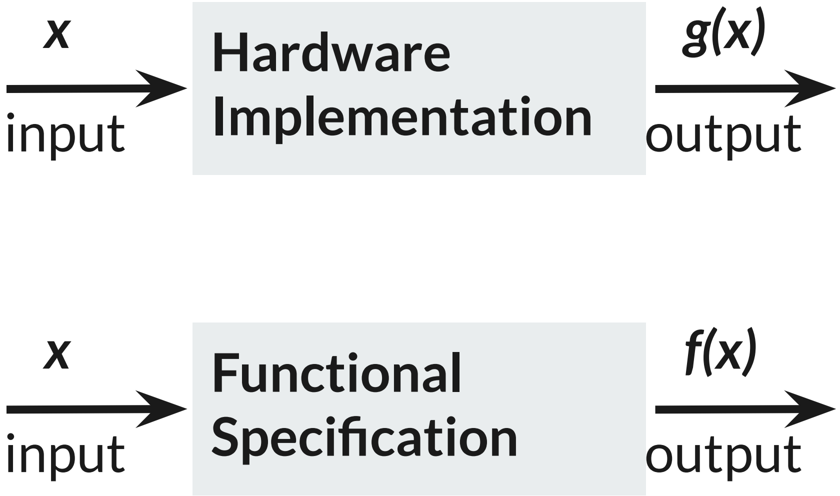


- **Infeasible** to try all types of input signals one by one
- Instead, we use “symbols” to represent the input signals (similar to algebra)

Knox Framework

Real world

Ideal world (correct and secure)



- ***g(x)***: symbolic computation result following the **hardware implementation**
- ***f(x)***: symbolic computation result following the **functional specification**

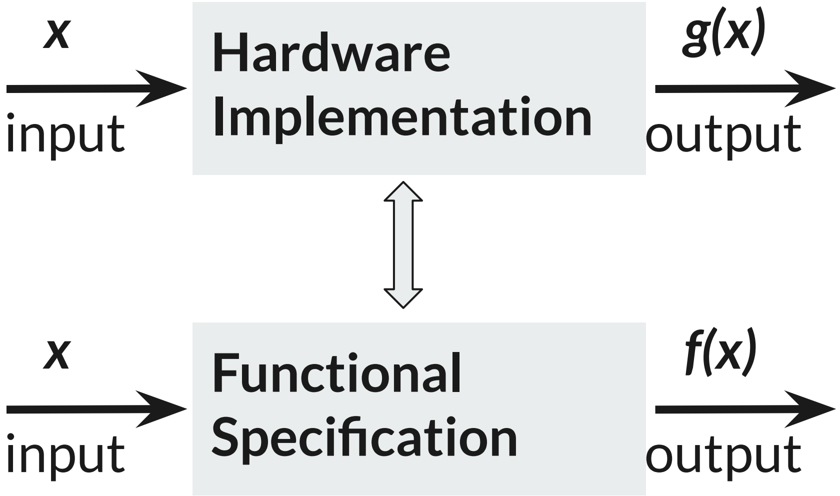
Athalye et al. "Verifying Hardware Security Modules with Information-Preserving Refinement" (OSDI 2022)

Knox Framework

Real world

We want to prove these two are indistinguishable

Ideal world (correct and secure)



- Formulate the problem as proving the formula **"*g(x)* ≠ *f(x)*" is unsatisfiable**
- Use **Satisfiability Modulo Theories** (SMT) solvers to prove the formula
- Successfully applied to identify timing side channels in hardware security modules

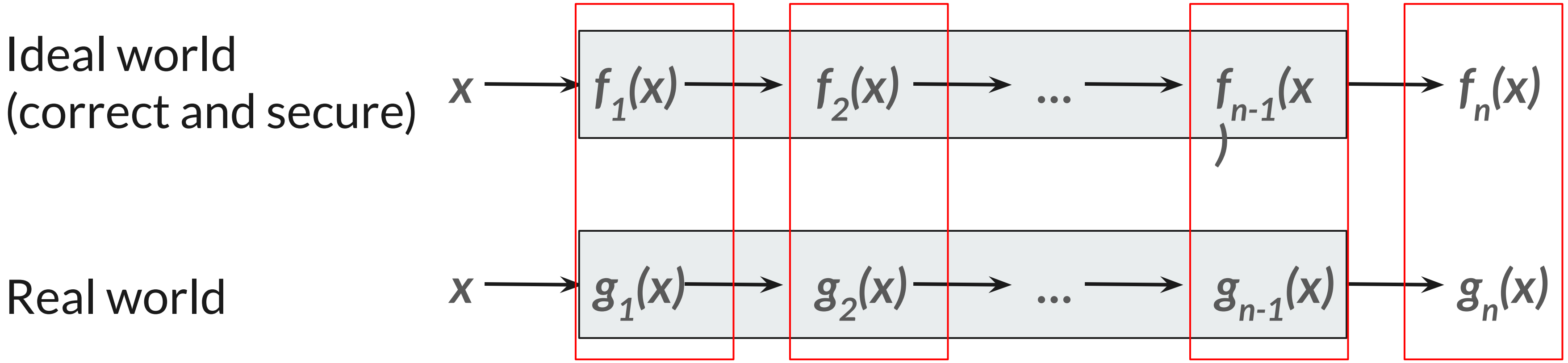
Athalye et al. "Verifying Hardware Security Modules with Information-Preserving Refinement" (OSDI 2022)

Challenges in Applying Knox to Our Problem

- It is still very challenging to apply the approach to our problem, because both *f(x)* and *g(x)* are **extremely large** and **complex** terms.
- SMT problem is typically NP-hard. There is no efficient algorithms to solve the general case.
- Given the size and complexity of our problem, we need to give guidance to the SMT solver to speed up the proof
  - We need to **break** big problems into smaller problems
  - We need to create **customized hints** for the SMT solver

Speeding Up Verification

- Technique 1: Break down
  - Break down the problem using states in the finite state machine
  - First, prove $f_1(x)$ and $g_1(x)$ are indistinguishable
  - Step by step, prove $f_k(x)$ and $g_k(x)$ are indistinguishable
- Technique 2: Add customized hints to speed up at each step



Performance Results

Implementation

- We implemented the encrypted ALU in Verilog
  - Created correctness and security proofs in Knox
- Integrated the encrypted ALU with an open source RISC-V core (Ibex) and vector coprocessor (Vicuna)
- Encoded the customized instructions using inline assembly
- Microbenchmark done in simulation (used Verilator with synthesis by Vivado)
  - Synthesis target: Digilent Nexys Video board (Artix-7 FPGA)

https://ibex-core.readthedocs.io/en/latest/
https://vicuna.readthedocs.io/en/latest/

Results

- We measure the efficiency using the performance counter in the RISC-V core
- Alcatraz completes 1 multiplication in roughly 250 clock cycles

	Alcatraz	Agrawal et al.	Shivdikar et al.
Operation	Multiplication	FHE multiplication	FHE multiplication
Hardware	50 MHz (estimate)	300 MHz FPGA	GPU acceleration
Performance	5 microseconds	28 microseconds	464 microseconds
LUTs and FFs	<10k LUT, <9k FF	1012k LUT, 1936k FF	N/A

Agrawal, et al. "HEAP: A Fully Homomorphic Encryption Accelerator with Parallelized Bootstrapping." ISCA 2024.
Shivdikar, et al. “GME:GPU-based Microarchitectural Extensions to Accelerate Homomorphic Encryption” (2023)

Acknowledgements

My Mentors

Jules Drean

Sacha Servan-Schreiber

Author of Knox (PRIMES Alum)

Anish Athalye

Prof. Srini Devadas, Dr. Slava Gerovitch, and MIT PRIMES for making this possible!

References

- *Athalye et al. “Verifying Hardware Security Modules with Information-Preserving Refinement” (OSDI 2022)*

- Agrawal, Rashmi, Anantha Chandrakasan, and Ajay Joshi. "HEAP: A Fully Homomorphic Encryption Accelerator with Parallelized Bootstrapping." *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2024.

- Shivdikar, et. al. “GME:GPU-based Microarchitectural Extensions to Accelerate Homomorphic Encryption”, arXiv:2309.11001 [cs.CR].

- Biernacki, et. al. “Sequestered Encryption: A Hardware Technique for Comprehensive Data Privacy”, 2022

Thank you!