Ashley Kurian

akurian@ncsu.edu | http://www.linkedin.com/in/ashley-kurian-383043208 | Raleigh, NC

EDUCATION

Ph.D. in Electrical & Computer Engineering **Aug 2022 - present**
North Carolina State University**,** Raleigh, NC GPA: 3.833 / 4
Course work: Micro-Processor Architecture, Architecture of Parallel Computers, Operating System Design, Embedded System Design, Compiler Optimization and Scheduling, Cryptographic Engineering and Hardware Security

M.Tech. Research in VLSI Design **Aug 2020 - Jul 2022**
National Institute of Technology, Karnataka, India GPA: 9.67 / 10
Course work**:** CMOS VLSI, Logic Synthesis Techniques, Digital Design using FPGAs

B.Tech. in Electronics and Communication Engineering **Aug 2015 - July 2019**
APJ Abdul Kalam Technological University, Kerala, India GPA: 9.16 / 10

RESEARCH EXPERIENCE

Graduate Research Assistant, North Carolina State University **Aug 2022 - present**

- Performed the **first side-channel attack** for model stealing **on Google's Edge TPU**, a commercial deep **neural network accelerator** in collaboration with Google under the guidance of Dr. Aydin Aysu.
- Demonstrated the recovery of model architecture and hyper-parameters (number of nodes, activation function, kernel size, etc.) on multiple neural network architectures using EM side channel. The attack methodology is generic and can be extended to any neural network.
- Automated the hyperparameter attack workflow so that the attack could be extended to any real-world model.
- Developed a python framework for side channel analysis as an alternative to Riscure's Inspector software.

Research Assistant, National Institute of Technology, Karnataka **Aug 2020 - Jul 2022**

- Studied the Novel Posit arithmetic that has the capability to surmount the drawbacks of widely used IEEE-754 standard.
- Conducted a study on the resource utilization and timing requirements for Posit arithmetic implementation on RISC-V core as an accelerator as well as within the core pipeline. The design was synthesized for an Artix-7 FPGA (xc7a200tfbg676-1) device using Xilinx Vivado. [Publication: https://doi.org/10.1007/978-981-19-6634-7_40]
- Developed sub-interfaces using Rocket Custom Co-processor (RoCC), for communication between the Posit accelerator and the Floating-Point unit and investigated different Type casting implementations on hardware.

SKILLS

- Good understanding on Hardware security, **Side-channel attacks**, fault attacks, defenses against side-channel attacks
- Knowledge in **Machine learning algorithms and deep learning**
- Understanding of Computer Architecture, Real-Time Operating Systems, Multi-core systems
- Programming Languages: C, Python, Verilog, C++
- Software Tools and Devices: Riscure Inspector, TensorFlow, MATLAB, Linux, Oscilloscope, icWaves, Transceiver

PROJECTS

Cryptographic Engineering and Hardware Security course projects (python, verilog)
- Developed cryptographic hardware for Ring-Binary-Learning-with-Errors encryption scheme.
- Extracted secret key of AES using Differential Power Analysis.

Architecture of Parallel Computers course projects (C++)
- Developed a simulator to compare Modified MSI and Dragon Coherence protocols.
- Implemented Scheduling of GEMMs in OpenMP Parallel Programming Model.

Microprocessor Architecture course projects (C++, Micro-processor Architecture)
- Created a Dynamic Instruction Scheduling simulator in C++ for an out-of-order super scalar microprocessor.
- Designed a branch predictor simulator (gshare, bimodal and hybrid)
- Built a Cache simulator in C++ to simulate the flow of data for L1, L1 + L2 cache.

Operating Systems Design course projects (C, Xinu, x86 architecture)
- Developed a fork system call in Xinu and implemented cascading termination of user processes.
- Incorporated lottery scheduling and MLFQ scheduling algorithm support into Xinu's kernel.
- Implemented spinlocks, priority inheritance protocol, virtual memory system, and interrupt handling in Xinu.

Compiler Optimization and Scheduling course projects (C, LLVM, Flex & Bison).
- Implemented code optimizations (Dead code elimination, common subexpression elimination, load-store optimization, Global value numbering).
- Utilized function inlining heuristic to examined performance enhancements on the benchmarks.