Parallelizing Password-Hash Search for Security Testing: A Comparative Study of CPU, Multi-thread, and GPU Approaches

Course: Parallel and Distributed Computing Project (J. Tian, Fall 2025)
Group 14: Noah Meduvsky (solo)
Prior Work: Extension of Security and Privacy in Computing project

PROBLEM STATEMENT

Password-hash search is a very parallel task. Security teams use it to test how strong their password policies are, but it is not obvious how much benefit parallel execution really brings on typical hardware. I will build on my Security and Privacy in Computing project to create a small benchmarking setup that compares a single-threaded CPU program against a multi-threaded CPU version, with a lightweight GPU kernel as an optional stretch goal. I will run every experiment on synthetic or consented password hashes.

PROPOSED APPROACH

I will write a Python harness that creates workloads, keeps timing data, and loads different execution backends. The required backends are: (1) a single-threaded C/C++ version for baseline measurements and (2) a multi-threaded C/C++ version that splits the candidate list across a thread pool. If time allows I will add (3) a basic CUDA or OpenCL kernel that launches one hash attempt per thread without heavy optimization. The harness will handle brute-force ranges and dictionary lists. I will only use hashes that I generate or that come from sources with explicit permission.

EVALUATION PLAN

For each workload I will record hashes per second, total runtime, and speedup relative to the single-threaded baseline. I will test scalability by sweeping the number of CPU threads. If I finish the GPU backend, I will compare its raw throughput to the CPU results. I will keep the metrics simple so I can stay focused on demonstrating concurrency instead of spending time on deep micro-optimizations.

RESOURCES & TIMELINE

Hardware: I will use a multi-core CPU workstation and try to schedule time on an NVIDIA GPU either on campus or through a cloud account. Software: I will rely on Python for the harness, C/C++ for CPU code, and CUDA/OpenCL only if I reach the stretch goal. My planned 3-week sprint:

- Week 1: finish the serial CPU baseline, set up timing, and clean up the earlier project assets.
- Week 2: implement the multi-threaded backend, tune workload partitioning, and collect first scaling runs.
- Week 3: run benchmarks, write the report, and, if time remains, add the simple GPU kernel for comparison.

RISKS & MITIGATIONS

Scope creep is the biggest risk. I will lock in the CPU serial and multi-threaded goals first and treat the GPU backend as bonus credit. GPU access may be limited, so I will reserve lab time early or use cloud credits if needed. To avoid ethical issues, I will document how I generate datasets and include an ethics statement in the repository and the final report.

EXPECTED CONTRIBUTION

I expect this project to produce a reproducible framework that demonstrates how much parallelism helps password-hash search on a standard CPU, with an optional GPU comparison. In my final report I will highlight the measured speedups, the simplicity of the metrics, and the practical trade-offs I faced while building a parallel security tool under a tight timeline.