Attackers who have physical access to mobile phones can potentially uncover private data pertaining to the original user. They can use the authentication credentials that are cached in the phone by the operating system or individual applications. Even without cached credentials, physical access admits a variety of well-known attacks, some of which can result in root access. A determined attacker may even inspect the memory of a running machine using operating system interfaces or hardware probing. Many existing encryption-decryption algorithms like Advanced Encryption Algorithm (AES) have used parallelism to improve the implementation of cryptographic modes such as Cipher Block Chaining (CBC) and Interleaved CBC (ICBC).

Modern GPUs are attractive for parallel processing because these architectures have hundreds of processing cores and high bandwidth, delivering up to a teraflop (1 trillion calculations per second) of computing power from the same silicon area as a comparable microprocessor and using only a fraction of the power per calculation [4]. These high-performance and low-energy advantages are a result of the GPU's support for hiding latency in memory transactions through massive multi-threading with low context switch overhead. Another advantage is that instruction processing in the thread contexts is based on the Single Instruction Multiple Data (SIMD) paradigm which therefore makes them suitable for algorithms that can expose a high degree of parallelism.

In many modern mobile devices, hardware manufacturers have adopted a system-on-chip (SoC) architecture where the CPU, GPU, and other computing nodes share the same system bus for accessing main memory, thereby limiting memory bandwidth [2]. Further constraints imposed on GPU design in comparison to their desktop counterparts are (1) dependence on battery power, (2) lack of cooling, (3) limited instruction set, and (4) low clock frequency.

Recently, frameworks have appeared for performing general-purpose parallel programming on GPUs (GPGPU), such as OpenCL and CUDA [5]. OpenCL provides a model for programming whereby a host processor (in this case, a CPU) delegates work to be performed in parallel by computing nodes (GPUs) [5]. The system has a high-latency large global memory for managing OpenCL context, whereas the compute nodes share a small but low-latency local memory for executing jobs [5].

Knowing when to program for the CPU or GPU to meet computational time and power trade-offs is made difficult by several factors. First, a GPU algorithm may require high-level changes that reduce memory traffic at the expense of additional computations [1]. Second, achieving speed gains from GPU parallelism first requires offsetting the expense of additional memory traffic.

The Nexus 4 phone from Google uses a Qualcomm Snapdragon S4 chipset which supports the OpenCL Embedded Profile. The chipset model used by the Nexus 4 (APQ8064) has an Adreno 320 GPU [3]. We wish to investigate the trade-offs in computational time and power consumption that can be achieved on the Nexus 4 when comparing a software implementation of the AES encryption algorithm to an OpenCL GPU implementation. Further, we wish to provide a framework through which regions of physical memory belonging to processes on the Nexus 4 can be encrypted when they are in an idle state, and to investigate the overhead paid in battery consumption.

| Date | Milestone |
|---|---|
| October 18th | • Determine the status of OpenCL access to the Adreno 320 GPU on the Nexus 4 Android platform.<br>• Create user-level programs for interacting with the Adreno 320 GPU through OpenCL to determine if it is stable. Such programs might include reading/writing bytes to the GPU memory buffer, XOR encryption (simply flipping the bits of a memory region).<br>• Find an existing user-level AES encryption implementation that makes use of OpenCL. If one is not available, find a similar implementation (e.g. CUDA), and port it to OpenCL. Also, find a user-level AES encryption algorithm.<br>• Benchmark the performance of both algorithms on the Nexus 4 for battery consumption (amplitude during execution) and throughput (rate of encryption). |
| October 23rd | • **First progress report** |
| November 8th | • Determine the best way to provide user-level encryption of processes, so that we can use the OpenCL implementation of the AES encryption algorithm.<br>• Provide a privileged process access to physical memory (which it can access by using the mmap system call on /dev/mem), and provide the process with physical address ranges in memory to encrypt through a kernel module |
| November 13th | • **Second progress report** |
| November 22nd | • Encrypt a process and benchmark battery consumption compared to no encryption |
| November 27th | • **Class presentation** |
| December 13th | • **Final report** |

[1] Akenine-Moller, Tomas, and Jacob Strom. "Graphics processing units for handhelds." *Proceedings of the IEEE* 96.5 (2008): 779-789.

[2] Cheng, Kwang-Ting, and Yi-Chu Wang. "Using mobile GPU for general-purpose computing–a case study of face recognition on smartphones." *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*. IEEE, 2011.

[3] Google Inc., *Nexus 4,* http://www.google.com/nexus/4/.

[4] Kothapalli, Kishore, et al. "CPU and/or GPU: Revisiting the GPU Vs. CPU Myth." *arXiv preprint arXiv:1303.2171* (2013).

[5] Wang, Guohui, et al. "Accelerating computer vision algorithms using OpenCL framework on the mobile GPU-a case study." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. 2013.