Research Plan **Category: Systems Software**

Ikey Croog

Using RGB Pixel Data to Generate a Stream of True Random Numbers for Encryption

A. Rationale:

Random number generators (RNG) have a critical use in the generating of encryption keys. These encryption keys are important in the encryption and protection of data on computers. Most of these RNGs rely on numbers that are generated to be only seemingly random. These are called pseudo random number generators (PRNG) and come with their own challenges. PRNGs aren't truly random and depending on the PRNG it can fail many statistical tests if not built correctly [7][8]. PRNGs also have the problem of repetition in the generation of numbers over several iterations and can therefore be predictable. In order to give pseudo-randomness, PRNGs utilize an initial value called a seed. This seed is processed to produce a pseudo-random number and another number put back into the PRNG to produce additional numbers [3]. An example of a simple PRNG is as follows: (Ks + c) mod X; where X is some number, K is some multiplier, c is some increment, and s is a seed [2]. The seed is important in the generation of numbers different from any other. If someone were to find the seed they would be able to generate the same exact numbers [4]. Most PRNG will rely on true random numbers (TRNG) to generate the seeds [3]. TRNGs use random processes that generate efficiently random numbers. An example of this would be the generation of random numbers through the random properties of light and quantum mechanics [9]. TRNGs come with their own problems. Some of these include: the reliance on existing hardware, the difficulties of implementation, and the speed of TRNG compared to PRNG which can be significantly slower [3]. Because of this TRNG are mainly used for seed generation and not used for generating streams of random numbers. Another problem is even random processes according to statistics will sometimes generate similar seeds which can be problematic [1]. A proposed solution to the speed and hardware restrictions of TRNGs and the lack of randomness of PRNGs is to generate random numbers through individual RGB (red, green, blue) values of pixels in a JPEG image. Although the idea of generating true random numbers through images has existed, it is only a one time generation of a random number either through the hashing of the image or the processing of RGB channels [6]. These methods of generation require a stream of images and therefore requiring existing hardware and take time to generate. A better method of generation is to combine the RGB values of several individual pixels in an image together instead of using the entire image to generate one random number. The proposed method uses the following equation: (RGB_1 * $RGB_{2\ldots}$ + c) mod X; with X being some number, and c being some increment. The data from an individual pixels RGB values in a random image can be averaged or multiplied together to give a random number. The algorithm can then move to a new pixel mod of the first pixels multiplied RGB value. This next pixel's RGB values can also be multiplied together and then multiplied with the original pixels averaged number. This process can be repeated several times and then the mod X of the number can be taken with some increment. To generate another random number, another pair of RGB values are selected and combined. Unlike PRNGs there is no initially seed, instead two or more truly random RGB values are multiplied together to give a random number. Because this method of generation does not rely on a seed, one number put in the generator will have no correlation to the last number put in. This method not only gets rid of hardware restrictions but also generates true random numbers with speed and efficiency.

B. Engineering Goal/Expected Outcome:

- Through the multiplication of two random RGB values in a given JPEG image, it is expected that a truly random number can be generated without the requirement of having an initial number to act as a seed.

C. Procedure:

- An x and y value, some value (o) to move by, and increment (c), and a max value (m) will be initially selected
- The RGB value of a pixel with the value of [x, y] will be selected
- The individual red, green, and blue values of the given pixel will be multiplied together to give a given number (v)
- The algorithm will then move mod o of the multiplied RGB value
- The new pixel will then have its red, green, and blue values multiplied
- This new value will multiplied together with v to get a new number
- This will process will repeat several times
- Once done a sufficient amount of times the final number will be added with c
- This final value will be taken to the mod of m to produce a final value

Data Collection/Analysis:

- Using a python script, NIST tests of randomness will be performed on the set of numbers [8]
- If a significant correlation is found between any of these numbers through these tests then an error will be printed to the console

D. Bibliography:

Cook, John D. “Random Number Generator Seed Mistakes.” *John D Cook*, 29 Jan. 2016, www.johndcook.com/blog/2016/01/29/random-number-generator-seed-mistakes/. [1]

Downham, D. Y. “Multiplicative Congruential Pseudo-Random Number Generators.” *The Computer Journal*, vol. 10, no. 1, 1967, pp. 74–77., doi:10.1093/comjnl/10.1.74. [2]

Jun, Benjamin, and Paul Kocher. “THE INTEL® RANDOM NUMBER GENERATOR.” *Http://www.cryptography.com/*, 22 Apr. 1999, pdfs.semanticscholar.org/10be/4f31b2b5bd7c51dd38b7339543ff46d51a2a.pdf. [3]

Lee, Kyungroul, et al. “TRNG (True Random Number Generator) Method Using Visible Spectrum for Secure Communication on 5G Network.” *IEEE Access*, vol. 6, 2018, pp. 12838–12847., doi:10.1109/access.2018.2799682. [4]

Lewis, P. A. W., et al. “A Pseudo-Random Number Generator for the System/360.” *IBM Systems Journal*, vol. 8, no. 2, 1969, pp. 136–146., doi:10.1147/sj.82.0136. [5]

Li, Rongzhong. “A True Random Number Generator Algorithm from Digital Camera Image Noise for Varying Lighting Conditions.” *SoutheastCon 2015*, 2015, doi:10.1109/secon.2015.7132901. [6]

O'Neill, M.E. “Specific Problems with Other RNGs.” *PCG, A Better Random Number Generator*, 31 Aug. 2014, www.pcg-random.org/other-rngs.html. [7]

Rukhin, Andrew, et al. “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.” 2000, doi:10.6028/nist.sp.800-22. [8]

Stefanov, André, et al. “Optical Quantum Random Number Generator.” *Journal of Modern Optics*, vol. 47, no. 4, 2000, pp. 595–598., doi:10.1080/09500340008233380. [9]

NO ADDENDUMS EXIST