TRNGle: Chaos-based True Random Number Generator

Ananya Rao, Matthew Burke, Siddhanta Shrestha, Stephen Thimothe

Abstract: The goal of TRNGle was to create a True Random Number Generator with a chaotic circuit as a source of entropy. We would make use of a particular implementation of a chaotic circuit, known as Chua's circuit, which would generate a bitstream using the amplified noise generated from the circuit. Upon capturing the bitstream data, we then pass the bits into an hashing algorithm to protect against bit manipulation and other potential environmental factors. The hashing algorithm we chose to work with was SHA-256 due to its popularity in cryptocurrency (used for transaction verifications by Bitcoin) and in TRNG. This secure algorithm takes in the data from the circuit, conducts bit manipulation in order to produce a 256 bit output hash value with low collision probability. As such, the hashed output can be further used as a key for encryption. The hashed output is fed into the NIST 800-22 SP test suite to evaluate the randomness of the generated non-deterministic bitstreams. This test suite is the industry standard to validate the randomness of immense binary streams produced by pseudorandom and true random number generators for applications including cryptography, IoT, communication or networking. Though it may not be completely perfect in determining the randomness, it is one of the best practices available. Though the circuit implemented does not perform entirely as intended, we show that it is very likely chaotic, and that it has potential to act as a high-quality TRNG if improved upon.

Introduction

Random number generators are very useful in cryptography because they provide unpredictability and limited vulnerability to several attack models against a cryptographic system. Brute-force and seed manipulation attacks against them are inefficient because they could be time consuming and difficult to execute. Not all random number generators produce truly random outputs however, thus it is important to distinguish between a pseudo random (PRNG) and true random number generator (TRNG). A pseudo random number generator uses mathematical equations to produce numbers, resulting in deterministic outputs that are easier to reproduce. TRNGs have no direct correlation between the input and output, meaning that the same input will not yield the same output. Additionally, slight variations to the input produces a different output making TRNGs very difficult to attack.

For this project we implemented a chaotic circuit to amplify noise sources because such a circuit produces non repeating sequences and is extremely sensitive to changes in its input [1]. The output of the RNG is then fed into an SHA-256 hashing algorithm to produce an irreversible unique digest. This makes the output more robust against environmental changes and malicious manipulation [2].

The resulting RNG should produce an output that passes the National Institute of Standards and Technology (NIST) test SP 800-22 [3]. Doing so would indicate that the resulting RNG has randomness properties, though it would not guarantee that it is truly random since TRNGs are not possible to prove using a finite number of statistical tests [4]. Image encryption is a great use case for our project since chaos-based random number generators have previously

been used for this purpose, and is one of the “three most common cryptographic primitives” along with Block ciphers and hash functions [5]. The output 256 bits of the hash can be used as a key for encryption algorithms when encrypting images.

This project allows us to gain in-depth knowledge of the requirements and validation process of true random number generators. It also grants the opportunity to learn how TRNGs are applied in cryptographic systems, how to implement a hash in such a system, and how a hash can be integrated with a TRNG to enhance its cryptographic qualities.

The rest of this report is structured as follows: Section 2 describes papers that we referenced for this project. Each paper contributes to our overall project and was used to gain an understanding of how others have approached similar projects. Section 3 divides our projects into different components where we provide insight on how each was implemented. It is split into three subsections: hardware which focuses on the chaotic circuit, software which focuses on retrieving the data from the circuit and the hashing portion, and the testing portion where we discuss the specifics of how we determined the output shows randomness properties. Section 4 describes how tasks in this project were assigned to the individual team members, and the evolution of the assignments over the timeline of the project. Section 5 documents the experimental methods used in this project, and the results and data of those tests. Section 6 includes reflections upon lessons learned from this project relative to the class as well as potential oversights and errors, and predictions of work that could have been done with additional time. Section 7 documents our conclusions from our work, and proposes areas of further exploration.

Related Work

The authors of [1] describe a chaos-based TRNG they designed by creating a loop between the first and final stages in an ADC pipeline with 1.5 bit stages. When configured this way, [1] shows that an ADC pipeline behaves as a discrete-time chaotic circuit, and requires no additional input since its analog components provide the noise necessary for it to act as a TRNG. With this circuit as a TRNG, [1] claims an output of up to 100 Mbits/sec for 0.18 um technology, with an output capable of passing all tests for a TRNG in the NIST suite. This paper provided a great deal of information on the potential use of chaotic circuits in TRNGs, and served as the original design for our chaotic circuit. However, we quickly discovered that it was quite difficult to find hardware compatible with such a design, necessitating a new method for a chaos-based TRNG[1]. Thus, the chaotic circuit component of our proposed TRNG comes from the design given by Wang et al. [6].

Wang et al. propose a method of creating multi-scroll chaotic circuits by using multiple nonlinear resistors in an op-amp high-pass filter implementation of Chua's circuit. Though a Chua's circuit would typically rely upon an inductor as well as resistors and capacitors [7], replacing the inductor with an op-amp circuit would be highly preferable for many applications of a TRNG, since it would allow the whole circuit to be placed on an integrated circuit. As well as providing a method of designing these chaotic circuits, [6] also demonstrates via simulation that an implementation of their design is capable of being used as the source of key bits for a one-time-pad (OTP) cipher. Though we were not without issues in securing proper circuit

[1] Credit to 'Max' Chen and Wayne Burleson for observing this.

components (as described later), the op-amp centric design of this circuit meant that the hardware needed for this was much more accessible than the chips needed to implement the circuit of [1]. Moreover, if implemented on an IC, the circuits of [6] would likely be more lightweight than those proposed by [1], since the chaotic pipeline would rely on multiple stages, each with multiple op-amps and gates[2].

While the rate at which we sample data from our implementation of Wang et al.'s circuit is much less than that of [1]- as will be explained later- the relatively lightweight and abundant components of the op-amp Chua's circuit would be advantageous for a key-generating TRNG. Moreover, the simulated circuit of [6] was sampled for the OTP at the same rate as that of the chaotic ADC (1 microsecond). Even if the Chua's circuit must be sampled at a lower rate in a hardware implementation, it should still be sufficiently high as to make the faster output of the chaotic pipeline circuit irrelevant (assuming the hardware maintains random properties comparable to the simulated values). Like the chaotic pipeline, the Sallen-Key Chua's circuit consists of analog components, each with their own noise, so no additional input is needed; the only necessary supply to the circuit is the voltage to the positive and negative rails of the op-amps [1],[8]-[10].

The idea behind using the output bitstream from a RNG and hashing it using SHA-256 to produce keyes for a stream cipher comes from the paper [11]. This paper used a MEMS based device to produce a stream of numbers, which showed randomness properties when sampled at low sampling rates. This was then fed into an SHA-512 hashing algorithm which acted as a conditioning stage in order to whiten the raw sensor data. The SHA algorithm family is often

[2] Credit to Wayne Burleson for observing this during conversation.

used as a whitener, thus when incorporating this with low sampling rates allows the RNG to pass all of the NIST tests. The proposed algorithm was implemented on a Xilinx Virtex 7 FPGA and when compared with other chaos-based cipher systems, this proposed system was able to achieve higher performance and lower vulnerability.

There have been chaos based random number generators used as primitives for image encryption. The authors of [5] propose an image encryption algorithm that uses SHA-2 based image cryptosystem to construct a mask for image encryption. This algorithm consists of 4 stages, the first two have substitution and diffusion processes while the last two only have diffusion. For substitution, a fourth of the image pixels are replaced with the S-box of the corresponding AES cipher (AES-128 for SHA-256, AES-192 for SHA-384, AES-256 for SHA-512). The diffusion process modifies the pixel value sequentially in an attempt to spread this change in pixel values of others. The authors conducted multiple tests on the proposed algorithm to analyze its performance. Histograms of the images post encryption showed near uniformity. This coupled with the analysis of adjacent pixels demonstrates higher performance compared to other methods proposed in other papers.

This NIST SP test suite would evaluate and verify the randomness of the output as it is a set of statistical tests for randomness verification. Various statistical tests can be applied to a sequence to evaluate the sequence as a truly random sequence. The NIST test suite is used so that we understand the output in relation to industry set uniform standards. Since the output generated must be unpredictable in the absence of knowledge of the inputs, the NIST test serves as a characteristic and recommended statistical test. If the seed is unknown, the next output number in the sequence should be unpredictable in spite of any knowledge of previous random numbers in

the sequence. No correlation between a seed and any value generated from that seed should be evident. [NIST]

Methodology and Tools

Chaotic Circuit Design

The design of the chaotic circuit was copied closely from [6], which was an inductor-free implementation of Chua's circuit. This design relies solely upon op-amps, resistors and capacitors, which was desirable since we sought a design that would be compatible with an integrated circuit. Every effort was made to match the resistive and capacitive values used by [6], and the same op-amps were used as well. However, where their circuit was implemented on a PCB with variable resistors, ours needed to be implemented on a breadboard with fixed-value resistors due to limited time and resources. Since simulations of the circuit in [6] were shown to be capable of producing a one-time-pad (OTP) output, we decided that this would likely be suitable for our purposes. While we were not intending to implement an OTP, we decided that this project would be a good opportunity to examine the circuit's cryptographic capability by evaluating it for key generation with our hash.

The circuit we replicated from [6] used a Sallen-Key high pass filter in place of the classic inductor of Chua's circuit. Additionally, both circuits used a series of op-amp based nonlinear resistors. Where a traditional Chua's circuit usually uses one nonlinear resistor to produce a double-scroll chaotic attractor [7], the system of multiple nonlinear resistors used by [6] produces multi-scroll patterns, which should enhance the statistical properties of the circuit by creating a wider range of possible internal phases. These multi-scroll patterns are generated

because the additional op-amp resistors produce additional breakpoints in the i-v characteristics of the nonlinear resistor, while a single such op-amp resistor would only produce two breakpoints [6]. These additional breakpoints allow for more equilibrium points in the circuit's dynamics, which in turn produce patterns with additional scrolls in its phase diagrams.

Sampling

To sample the circuit and generate a bitstream, we used Arduino Uno with an ATmega328P microcontroller. The ATmega328P on the Arduino has an ADC with a 16MHz clock [12] divided by a prescaler set to 128 setting the clock frequency to 9600Hz. This would have been sufficient as far as sampling goes if not for the delay caused by Arduino function calls which drastically slow down max sampling rate to 125Hz. Due to having to generate one million bit inputs for the NIST test, being limited to 125 bits a second would lead to the process taking roughly 21 minutes per input, with a total of 55 inputs taking about 19.25 hours. A larger problem is that the voltage range for the ADC is 0-5V[12], but the circuit requires at least 6V. This could lead to poor resolution and saturation with our ADC, so for future projects I'd recommend an ADC with a larger voltage range.

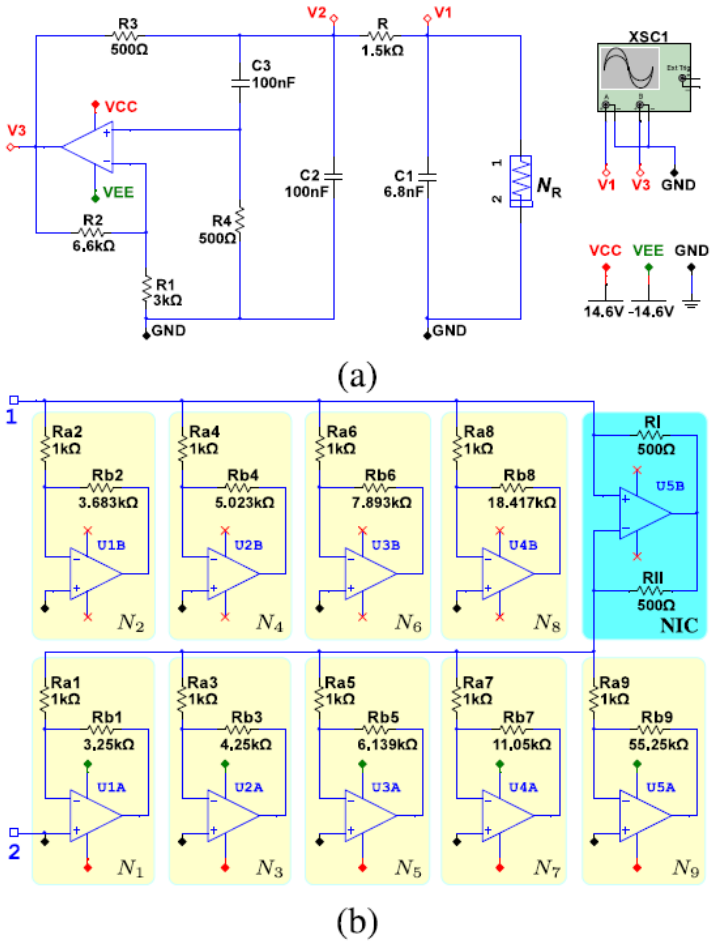

Fig. 1: a) Diagram of op-amp based chua's circuit with block for location of nonlinear resistors (N_R), b) diagram of all op-amp based nonlinear resistors in the circuit [6]. V3 and V1 will be plotted in our oscilloscope phase diagrams.

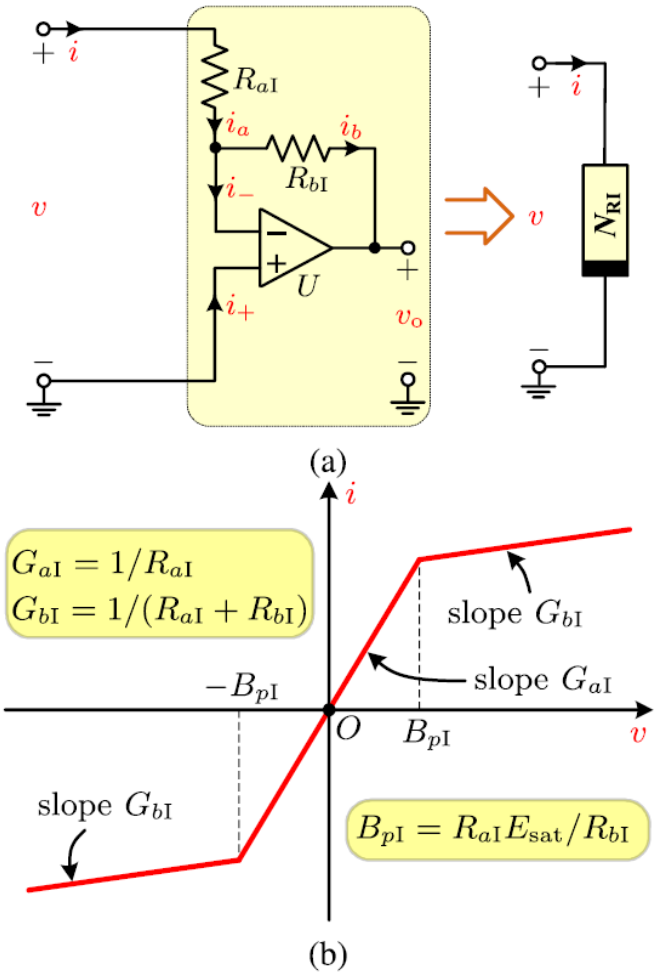

Fig. 2: [6] The 'type-I' nonlinear resistor used by Wang et al. (a), with its piecewise-linear i-v characteristics (b).

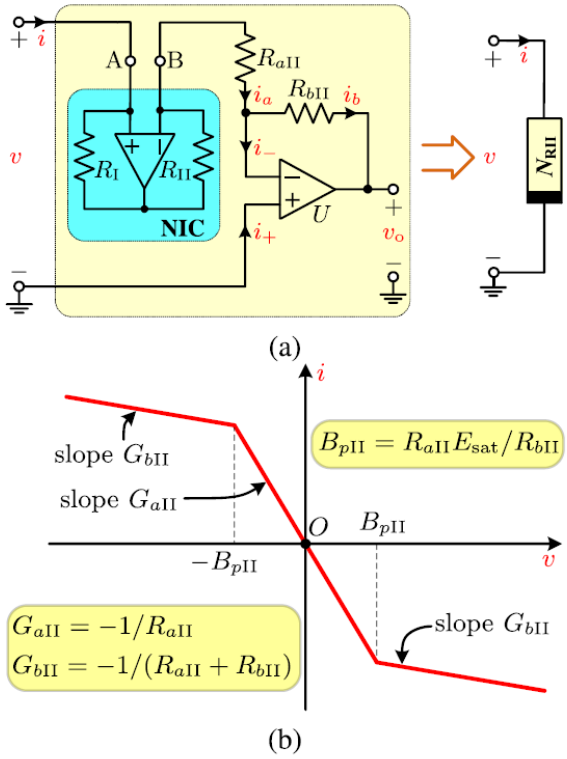

Fig. 3: [6] The 'type-II' nonlinear resistor used by Wang et al. (a), with its piecewise-linear i-v characteristics (b).

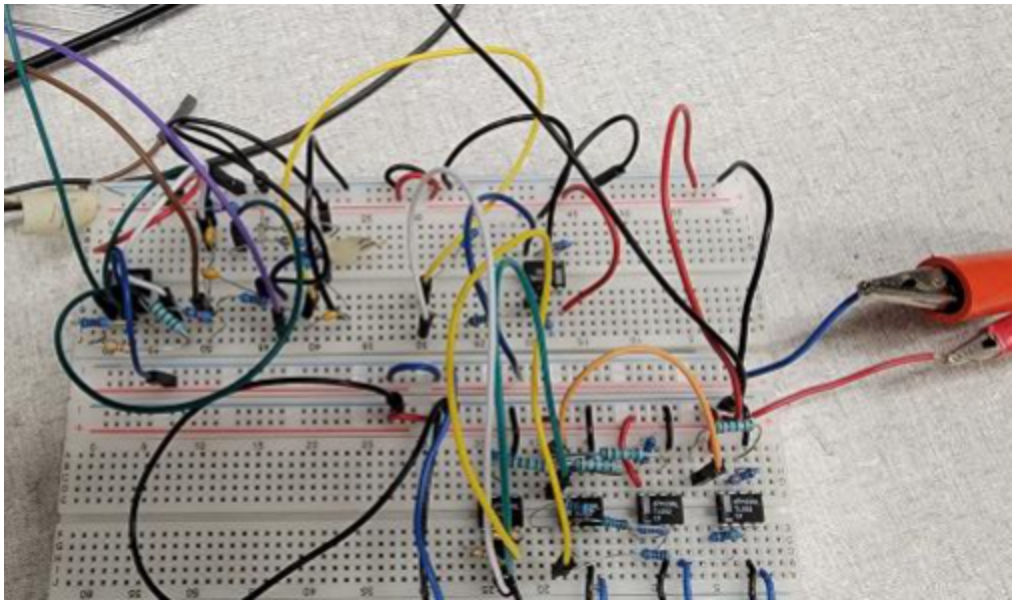

Fig. 4: Our breadboard implementation of the Chua's circuit schematic in fig. (1); the Sallen-Key based circuit is in the upper left part of the board, with one of the nonlinear resistors in the top right and the rest of the op-amp resistors in the bottom right.

Hashing

Similar to the paper on [11] we used an SHA hashing algorithm for our conditioning stage. We chose to use SHA-256 because it is a mathematically secure hashing algorithm with a low collision probability (2^{256} possible values). The reason why we went with 256 instead of the 512 was because it is much more efficient. It is smaller and requires less memory/processing power for computation. In our implementation, we use the algorithm to hash the values we obtained from the chaotic circuit. Because the SHA-256 works on data in 512-bit chunks, any input data that is less than 512 bits would need to be padded with zeros. Although it is impossible to ensure that no two input values produce the same hashed output due to the range being 256 bits, it is computationally infeasible to produce two inputs to do so. As such, we are able to use this hashing algorithm to hash the output of our chaotic circuit. These values have random properties where minute changes to the input results in major changes in the output. Feeding this into the hashing algorithm results in a very secure output with an extremely low collision probability. The resulting 256 bit hash value can then be used as a key for encryption algorithms. An example of such would be image encryption discussed in [5] where they used the corresponding AES cipher shown in figure [5]

Terms	SHA-1	SHA-256	SHA-384	SHA-512
Size of hash value(Bit)	160	256	384	512
Complexity of the best attack	2^{80}	2^{128}	2^{192}	2^{256}
Equivalently secure secret-key cipher (Bit)	-	AES-128	AES-192	AES-256
Message size (Bit)	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Message block size (Bit)	512	512	1024	1024
Word size (Bit)	32	32	64	64
Number of words(Bit)	5	8	8	8
Number of digest rounds	80	64	80	80

Fig. [5]. Characteristics of four different hashing algorithms

Testing

The focus of testing the stream of bits is the verification of randomness that is required for cryptographic purposes. A set of statistical tests for randomness is described in the cited document that was used to conduct and run the NIST 800-22 SP test suite on the bits that were hashed. The goal of this analysis would be that if the seed is unknown, the next output number in the sequence should be unpredictable in spite of any knowledge of previous random numbers in the sequence. No correlation between a seed and any value generated from that seed should be evident.

Partitioning of Work and Schedule

As for the partitioning of work, the group was divided into two halves, hardware and software, in which Matt and Stephen worked on the former and Siddhanta and Ananya worked on the latter. As for hardware, Matt was responsible for finding an implementation of Chua's circuit to use, as well as understanding the general dynamics of the circuit and designing the tests for chaotic behavior. Stephen was responsible for finding out how to generate a bitstream from the circuit to pass to the hashing algorithm, and aided in placing the circuit components on the breadboards. Similarly, Matt assisted with the ADC processing, in particular with advising Stephen on the sampling rate and debugging the hardware of the Arduino ADC. Stephen later took responsibility for the DIEHARD tests as well. For software, Siddhanta was tasked with finding a suitable and well tested hashing algorithm to use and Ananya was tasked with learning

how to operate the NIST test suite to analyze our output for randomness. Ananya downloaded the NIST test suite and ran the list of numbers provided by Sidd to check for its randomness.

Initially, Matt and Stephen were intended to be responsible for implementing the chaotic ADC pipeline of [1] on an FPGA, while Sidd and Ananya were tasked with placing the hash on the same FPGA; the whole group would have been responsible for the NIST tests after completion of the previous assignments. However, after the first meeting with Professor Burleson and 'Max' Chen, it was decided that an FPGA implementation of the pipeline/hash RNG would be out-of-scope for this project. Simulating the hash and pipeline system was briefly considered, with Matt and Stephen writing an HSPICE simulation for the pipeline, and Sidd and Ananya figuring out how to feed the output from that simulation into a C implementation of SHA-256. As with the first iteration of the project plan, all four team members would collaborate on the NIST test upon completion of the previous steps. Upon a brief meeting between Stephen, Matt and Professor Burleson, the ADC design as a whole was deemed too complex for the project. A breadboarded chaotic circuit was settled on, and the circuit of [6] was deemed ideal for the project by the group.

Like the previous iterations, Stephen and Matt were assigned to oversee the construction of the chaotic circuit, and Sidd would lead the hash implementation. Ananya, however, was assigned to begin the NIST tests with sample data while the rest of the project was underway, at Professor Burleson's desire for someone to specialize in those tests during the early stages of the project. Sidd's primary task was to find a hash implementation in C and modify it so that it would read the data outputting from the circuit and then hash it. Two types of algorithms were considered: CHACHA20 encryption and SHA-256 hashing. The latter was chosen for our

implementation primarily because a majority of the papers we had already referenced used SHA implementation.

The project was done over a period of 6 weeks. For the hardware design, Stephen and Matt spent two weeks studying the chaotic ADC pipeline and gathering components for it, before moving on to the op-amp Chua's circuit. It took a total of 3 weeks to complete the circuit and run satisfactory tests on it. Early oscilloscope measurements were made within a week of building the first circuit (at which time only one nonlinear resistor had been built), but it was decided that the whole circuit from [6] should be built once we observed inconclusive oscilloscope measurements. The remainder of the time was used by Stephen and Matt to extract the data via the Arduino, then by Stephen to attempt the DIEHARD tests.

Experiments and Results

Chaotic Circuit Testing

As with the design of the circuit, the methodology used for testing the circuit for chaotic behavior was closely replicated from that used in [6]. In order to verify that their circuit was chaotic- and possessed multiple equilibrium points- the authors of [6] relied upon oscilloscope measurements of their circuit at nodes V1, V2 and V3 of figure (1) a). They created phase diagrams of those nodes by plotting measurements of V3 v. V1 on an oscilloscope, and looked for multi-scroll plots (V2 v. V1 plots were shown from simulated values, but not from hardware-based values). These plots would be indicative of a chaotic system with multiple equilibrium points, which would be the expected result of a Chua's circuit with multiple

piecewise-linear resistors (if arranged and designed properly). We obeyed this technique, and made the same phase diagram plots on an oscilloscope using probes in the same locations.

While the phase diagram plots captured on our oscilloscope do not match the multi-scroll patterns generated by [6], they do appear to possess a striking similarity to a known double-scroll chaotic pattern of Chua's circuit shown in figure (7)[7]. It is also important to note that, while we were able to build the Sallen-Key Chua's circuit and the first nonlinear resistor (the one with resistors Ra2 and Rb2 from figure [1]) with parts practically identical to those prescribed by [6], we had a very limited selection of resistors at our disposal and were unable to replicate the exact values used for the remainder of the nonlinear resistors. Thus, we suspect that, while we were able to get a proper piecewise-linear resistor and Chua's circuit for double-scroll chaotic behavior, we were simply ill-equipped to create the i-v breakpoints required for the multi-scroll behavior observed in [6].

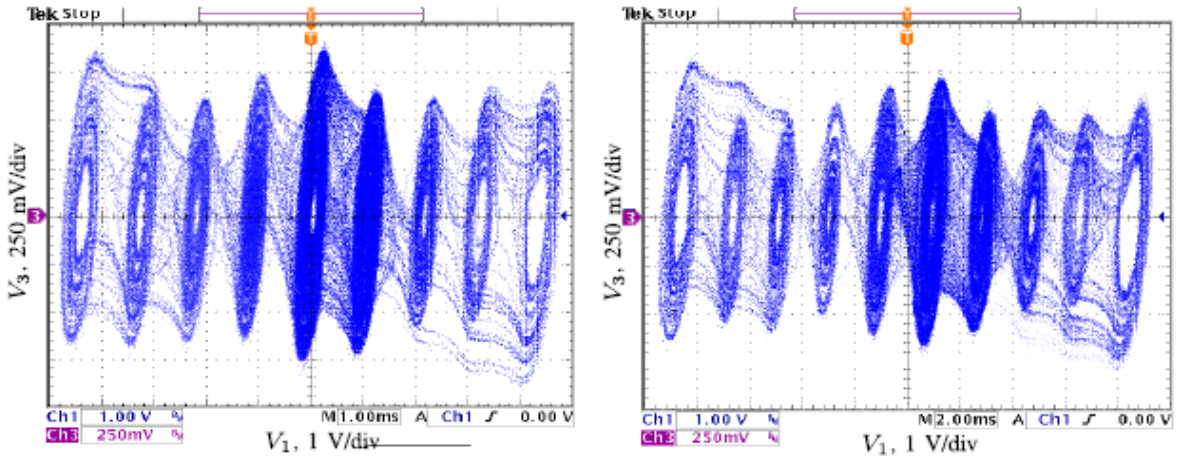

Fig. 6: The V_1 v. V_3 phase diagrams captured on an oscilloscope in [6] *for a 9-scroll Chua's circuit (left) and a 10-scroll Chua's circuit (right).*

In addition to the phase diagram plotting prescribed by [6], we made our own oscilloscope measurements of V3 and V1 as a function of time, as shown in figures (9) and (10).

Both of these plots appeared to be showing that the circuit was outputting oscillations with random amplitude at the time scale used. Moreover, the output of V1 had lower maximum amplitude than the output from V3, which matches our intuition that there must be a voltage drop due to resistor R in figure (1). Regardless of these plots, further testing and mathematical analysis should be done to confirm that our circuit is experiencing chaotic behavior, but we can say with reasonable confidence that our circuit has produced a chaotic output.

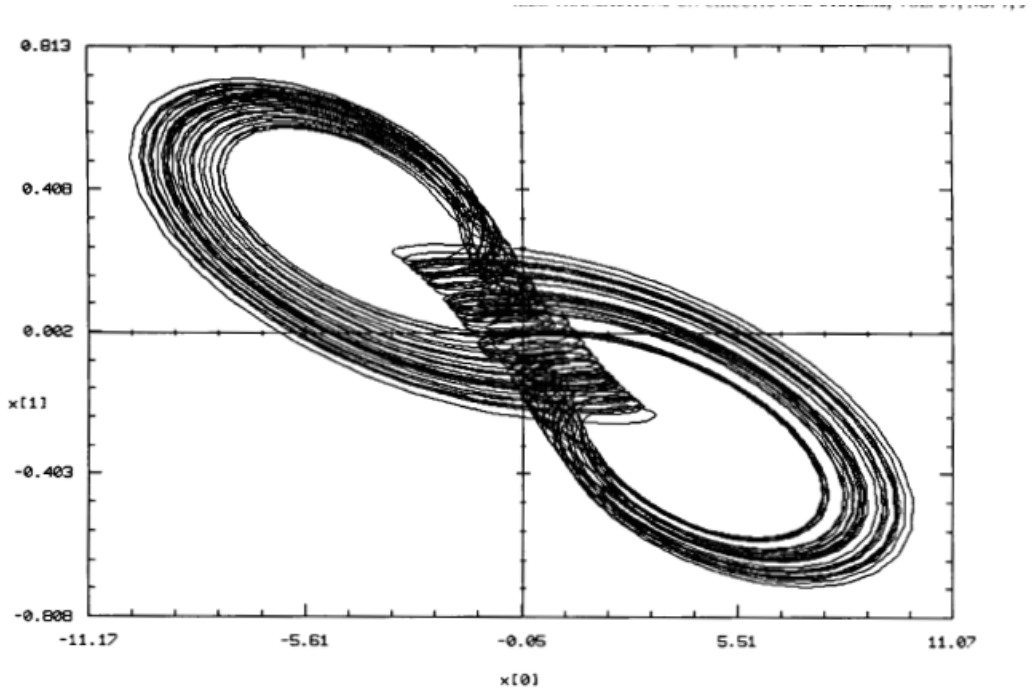

Fig. 7: a known chaotic voltage phase diagram from Chua's circuit [7] that appears very similar to the phase diagrams captured from our circuit.

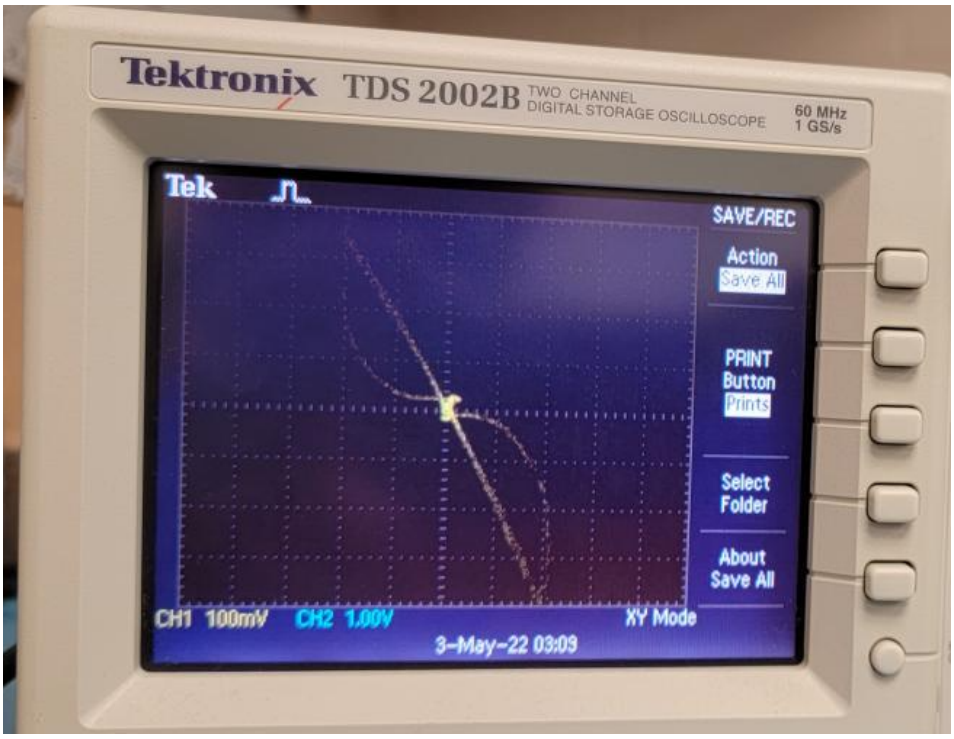

Fig. 8: The oscilloscope image of the V_1 v. V_3 phase diagram from our circuit. Notice how it possesses a very similar figure-8 pattern to figure (3), with a similar dense oscillation about the center. V_3 is on the y-axis with 1.00 V/division, and V_1 is on the x-axis with 100 mV/division.

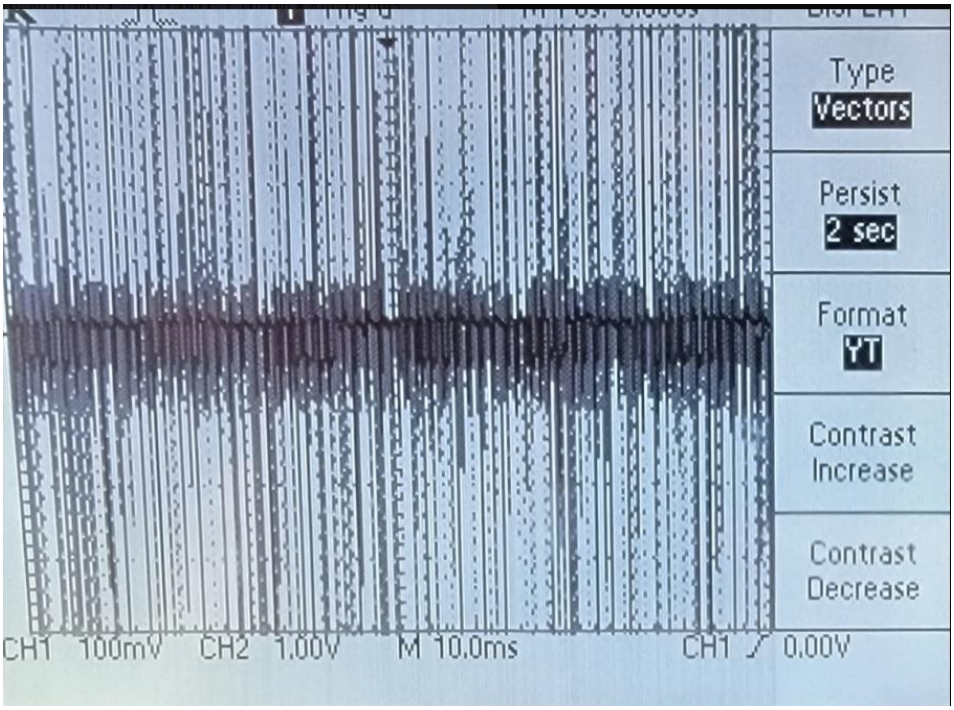

Fig. 9: Oscilloscope image of V3 plotted against time. The oscillations in V3 appear to be random, with low-order noise.

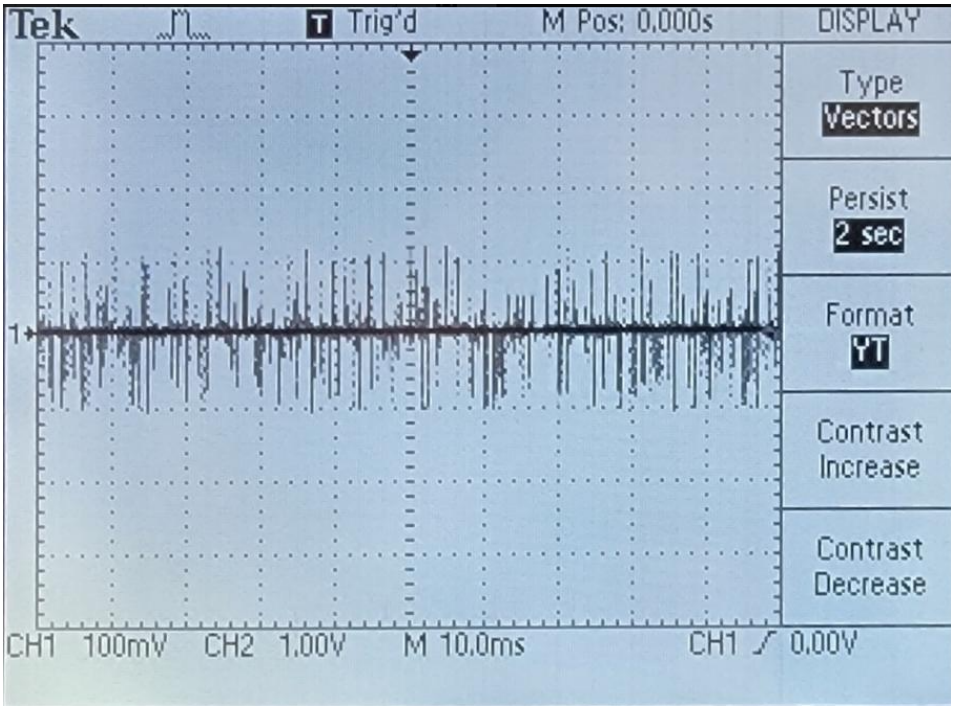

Fig.10: Oscilloscope image of V1 plotted against time, with the same x and y scales as those in figure (7). As with figure (7), the oscillations appear random with low-order noise. As expected, the amplitude of oscillation is lower than that of figure (6).

Demonstration:

The demonstration of our project can be found using the following link.

Analysis and Reflections

From this project we were able to learn that true random number generators are extremely hard to verify since there is no known method to prove a sequence is true random. TRNGs built with analog components typically produce higher-quality outputs and have more well-known mathematical properties, but often suffer from inferior embeddability than their digital-based counterparts [1]. As we have experienced, chaotic systems can be difficult to build and verify in

practice, and thus are difficult to implement in a TRNG system, though their intrinsic dynamics make them excellent candidates for providing the needed statistical properties to noise sources.

If we were given one more week to work on the project we definitely would have been able to deliver better results on sampling and testing. As for sampling, a problem we had run into was that the ADC we had chosen to use for the circuit was fairly poor and we would have simply found an alternative solution to sample with instead. In fact, the ATmega328P has a decent ADC, but the Arduino limits its capabilities due to hard coded pre scalars and slow function calls that can't keep up with the ADC. We also would have looked into using level shifters or transistors in order to step down the voltage to get around this problem. If we were to look into alternative ADC's we would look for external ADCs rather than integrated ones which are likely to have more suitable parameters such as larger voltage ranges to avoid poor or faulty outputs. As for testing we simply didn't have time to fully understand the DIEHARD test suite, or other alternative test suites while Ananya was sick. We probably would have solved it or picked a better documented test suite within an extra week to get around this but whether or not our analysis of the outputted p-values would have been to par is also in question. Otherwise, we'd have relied on Ananya to get the NIST test working properly and would have had a better idea about the information the outputs from each test were giving. Other than the problems we ran into with sampling and testing our circuit, the other problems would most likely require more time

Given another month, we'd likely be able to either rebuild the circuit with variable resistors, or perhaps fine tune the resistors we have to match the output shown in [6]. We weren't able to determine the cause of our discrepancy but went on to assume that because our resistor

values were different in the non-linear Chua's diode section of the circuit, that we wouldn't be able to replicate their results. We also would have had time to run simulations of the circuit in [6] and ours in spice to verify our speculations about values. Simulations would have been a vital way to debug the circuit as well since we can only spend so much time looking for faulty wiring or hope that the parts used weren't faulty; they would have also let us predict which configurations would produce optimal outputs for a TRNG.

Additionally, we would have likely been able to integrate our system on the Arduino, and feed the ADC-processed values of our chaotic circuit directly into hash via the Arduino. Currently the output bits are manually fed into the hash and, while this may have been done in around one weeks time, getting the ADC to work well with the circuit could take roughly a week as previously mentioned so we'd definitely be able to integrate with an additional week or so. Also, due to health complications, it was necessary for us to look into and implement an alternative test suite other than the NIST test in a very short amount of time so a thorough analysis of our test results wasn't available. We had tried to implement another test suite such as DIEHARD but the documentation on using it was greatly lacking in detail so in the time allotted we weren't able to get it working.

Skills and expertise that would have been useful to have for this project would have been more knowledge about the behavior of non-linear resistors and chaos theory to help verify and debug the output of our circuit. We have reason to believe that one of the greatest discrepancies in phase diagrams that we had with [6] was due to resistor value variations. Due to the lack of knowledge in non-linear resistors, we only speculated that it was due to the non-linearity of the Chua's diode section that the small offset in resistor values led to the drastically different output.

Most of us didn't have much knowledge regarding chaotic circuits so we had to actively learn about them while trying to implement one. More knowledge on chaotic circuits would have definitely helped with finding an implementation and would have saved us time we had lost when trying to build the circuit from [1].

Conclusions and Future Work

It is important to note that the results we present in this paper are primitive. As previously mentioned, though we can claim that our Sallen-Key Chua's circuit is very likely chaotic, it is a far cry from the chaotic behavior shown in [6] that we had hoped to replicate. Further work needs to be done with this circuit to confirm that it is chaotic, or adjust it so that it may exhibit chaotic behavior. Moreover, the nonlinear resistors should be rebuilt with either resistors more precise to those used by [6], or variable resistors tuned to the appropriate values.

Moreover, our restrictive time frame for this project left us little room to experiment with varying sampling rates for the collection of data from our circuit. If revisited, data should be sampled by the ADC at a wide range of frequencies. Additionally, the same time restrictions prevented us from experimenting with taking the low-order bits only from our ADC-processed values. Hypothetically, these values would have greater variation than the higher-order bits, and thus provide more desirable statistics.

References

[1]F. Pareschi, G. Setti, and R. Rovatti, “Implementation and Testing of High-Speed CMOS True Random Number Generators Based on Chaotic Systems,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 12, pp. 3124–3137, Dec. 2010, doi: 10.1109/tcsi.2010.2052515.

[2]Sunar, B. (2009). True random number generators for cryptography. *Cryptographic Engineering*, 55–73. https://doi.org/10.1007/978-0-387-71817-0_4

[3] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, and M. Vangel, “A statistical test suite for random and pseudorandom Number Generators for Cryptographic Applications.” [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf. [Accessed: 06-Mar-2022].

[4]A. Kamadi and Z. Abbas, “Implementation of TRNG with SHA-3 for hardware security,” *Microelectronics Journal*, vol. 123, p. 105410, 2022.

[5]S. M. Seyedzade, S. Mirzakuchaki and R. E. Atani, "A novel image encryption algorithm based on hash function," 2010 6th Iranian Conference on Machine Vision and Image Processing, 2010, pp. 1-6, doi: 10.1109/IranianMVIP.2010.5941167.

[6]N. Wang, C. Li, H. Bao, M. Chen, and B. Bao, “Generating Multi-Scroll Chua’s Attractors

via Simplified Piecewise-Linear Chua's Diode," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 12, pp. 4767–4779, Dec. 2019, doi: 10.1109/tcsi.2019.2933365.

[7]L. O. Chua and G.-N. Lin, "Canonical realization of Chua's circuit family," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 7, pp. 885–902, Jul. 1990, doi: 10.1109/31.55064.

[8]W. T. Holman, J. A. Connelly, and A. B. Dowlatabadi, "An integrated analog/digital random noise source," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 44, no. 6, pp. 521–528, Jun. 1997, doi: 10.1109/81.586025.

[9]G. Hanson and A. Der Ziel, "Shot Noise in Transistors," *Proceedings of the IRE*, vol. 45, no. 11, pp. 1538–1542, 1957, doi: 10.1109/jrproc.1957.278349.

[10]A. Der Ziel, "Thermal Noise in Field-Effect Transistors," *Proceedings of the IRE*, vol. 50, no. 8, pp. 1808–1812, Aug. 1962, doi: 10.1109/jrproc.1962.288221.

[11]M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, and S. Celma, "Application of a MEMS-Based TRNG in a Chaotic Stream Cipher," Sensors, vol. 17, no. 3, p. 646, Mar. 2017, doi: 10.3390/s17030646.

[12] Atmel, "8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash", ATmega328P datasheet, / Rev.: 7810D–AVR–01/15 ,(accessed May 7, 2022)

[13]Bassham, L. E., Rukhin, A. L., Soto, J., Nechvatal, J. R., Smid, M. E., Barker, E. B., Leigh,