An Investigation Regarding Pseudorandom Number Generation with Xor Shifting

October 1, 2025

1 Introduction

This investigation covers Pseudorandom Number Generation (PRNG) with the assistance of Xor Shifting. Xor shifting is a common way of finding various ways to encrypt data or provide encrypted data for users to protect their accounts via the hosting company in cybersecurity. The way Xor Shifting works is by selecting a random variable of bytes from a 16-byte input, extracting them from the original input, and then shifting them to the front of the 16-byte input and returning it back as the output value. That process can continue from input to input and is guaranteed to give you a different number every time without any correlation to the previous number (Arobelidze). I picked this topic because I have a personal connection to some insecure processes surrounding encrypted data. For instance, in 2019 my garage that had a combination lock got broken into when we were out of the house due to the minimal layers of security the lock was equipped with. If that data was better secured within the system and there were additional layers of security in place, this could have easily been avoided. I will start off by communicating the Key Terms that are necessary to understand for a clear understanding of this investigation, then move towards mathematical representation and personal engagement surrounding this topic. Next, I will cover some possible questions surrounding my topic, then I will connect to a global scale to show the real-world usages of this investigation. Then I will introduce some data and go into the reasonable math surrounding it as well as the conclusion. Finally, I will move into an evaluation, the use of mathematics, and finally the raw math surrounding this investigation. To model this study, I will show a basic byte-shifting model, a very simple computer science algorithm that models this investigation in Java, and a histogram of my data to show the accomplishment of the aim at the end. By the end of this investigation, I will explain the implications of perfecting a PRNG model with Xor Shift and how it can positively impact billions of people across the world. Below is a list of terms that will be beneficial to the reader throughout the course of this investigative paper:

Pseudorandom Number Generation, Generating a sequence of numbers that have zero correlation to each other

Xor Shifting, Linear feedback shift registers

Byte, A unit of digital information

2 Explaining the Problem at Hand

Cybercrime is at an all-time high, and it is not unavoidable in any capacity. The Identity Theft Research Center reported that 422 million individuals were impacted by a combination of cybercrime and identity theft in 2022 alone (). The FBI reported that in the United States in 2022 alone, over $10.2B USD was lost due to cybercrimes and identity thefts (). Finally, the Federal Trade Commission identified that password-related fraudulent activities accounted for 67% of all cybercrimes. Cybercrimes are rapidly becoming a growing threat and harming hundreds of millions of people in my country, as well as billions of innocent people in our world today. As a community, we need to be more concerned about the security of our people and take action to make our internet a safer place for the 5,300,000,000 people who are on it on a day-to-day basis (contributors). Cybersecurity is a sector that is just now getting a lot of attention and supporters trying to crack down and make it more powerful than ever. One of the ways people have identified that makes a strong case for being the pinnacle of cybersecurity is a concept called Pseudorandom Number Generation, where every time you generate a number, you never generate the same number twice, ever. The reason this has been identified as a way to stop a lot of cybercrimes is that PRNG is easily compatible with things like account and password security, specifically through Two-factor Authentication (2FA). Moreover, despite the large amount of cybercrime in places like the United States of America, Europe, and larger places in South America, the world really needs to shift its focus to protecting more third-world countries in continents like Asia and Africa from cybercrimes. May third-world countries suffer from heinous cybercrime every day due to the lack of development on their internet infrastructures, which causes billions of people to suffer from identity theft, cybercrimes, impersonation, etc. Concepts like PRNG with Xor Shifting and other methods of cybersecurity are so easy to implement anywhere in the world,

however it is only the places like America and Europe where big technology companies are housed where all the attention for a good cybersecurity infrastructure is emphasized. if we implement these basic security models in these third-world countries we can help billions of people stay safe in the vast void of the internet, not only in the present, but make a huge impact on communities in an increasingly digital era for decades to come.

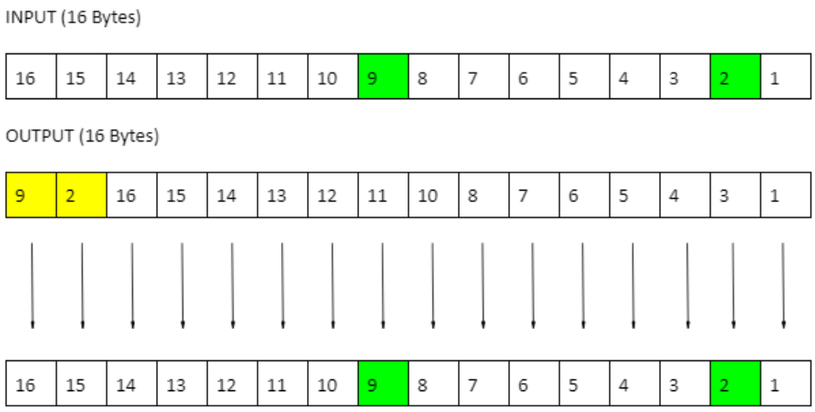
3 Motivation and Aim

This investigation aims to cover the processes of PRNG and how it works specifically within the medium of Xor Shifting, and then further analyze it through a mathematical lens to see where it functions in society today. Computers and their systems have always come naturally to me, seeing my dad and my family friends' parents thrive in the computer science division of the world has been really inspiring to me since I was a little kid. I participated in a research program in collaboration with various professors at Arizona State University where I was able to understand how important keeping certain things secure was, pushing me towards this topic. Moreover, in 2019 my house was broken into via a hacking of my garage's number lock. Understanding that this could have been stopped by a simple addition of a few layers of security infuriated me and I knew I could do better than the people who designed that locking system. Since then, two of my friends from neighboring high schools and I have been working towards launching our own variation of the popular digital combination lock. In the future, I plan to mix a few of my interests and pursue a Business Analytics pathway that will let me continue to stay close to computers in the future. As I have mentioned before, this investigation connects to my interests so deeply. Being into computers since I was a kid makes this investigation a little easier to do for me since I have the general knowledge necessary to carry this out. Morally and ethically, this investigation goes towards helping the 5,300,000,000 people who use the internet yearly (contributors) stay safe in the ever-growing world of the internet, where most of the data that helps the world work is stored. With the concepts shared in my investigation, the world could benefit greatly in the race for maximum internet safety. I will know that I accomplished my aim for this investigation if my standard deviation is high enough to make sense for my data set, this will effectively prove that my derived equation works for PRNG. If the number I get for the standard deviation of my randomly generated data is over the approximated value of 13,000, then the data is good and I have succeeded. However, if the number is under 13,000, I have effectively failed at modeling PRNG in Xor Shifting. Keep in mind that the number 13,000 is relative to a data set size of 20,000 random numbers.

4 The Mathematics Behind Pseudorandom Number Generation with Xor Shifting

A fair warning to readers, this concept of PRNG is heavily dependent on computer science functions and principles, many of the things you will see in this section have been fully derived from computer science topics and functions that relate to PRNG with Xor Shifting specifically.

The first step to deriving the standard equation for PRNG is finding the bytes we want to shift, and then figuring out what the shifting actually looks like, which I have modeled here:



What this model shows is how you can shift bytes from anywhere within an input and put them in front of it to completely change the number at hand. To take this one step closer to the mathematical equation, I will show how this is modeled with a basic computer science function.

```
public short getNextRandomShort () {
      short newBit = (short) (((seed >> 1) ^ (seed >> 9)) & 1);
      seed = (seed * 31) ^ (newBit << 15);
      return (short) seed;
}
```

This code essentially tells the computer that it needs to pick a random number from the inputted bit and factor in the bytes that were actually moved. In the example, the first and ninth bytes were the ones that were moved from the original input, so those are the bytes that are going to undergo the Xor function here.

Now, to actually translate this mathematically, we need to classify certain types of shifting in mathematical terminology.

Left Shift (x, y) =

$$2^y x \tag{1}$$

Right Shift (x, y) =

$$|\frac{x}{2^y}| \tag{2}$$

Total Shift (x, y) =

$$\lfloor 2^y x \rfloor \tag{3}$$

Now that we have classified all our mathematics for shifting, we can put it into a generalized equation that shifts any number and turns it into another random number. This equation looks like this:

$$f(x, y_1, y_2, ...y_n) = \lfloor \frac{x}{2} \rfloor + 2^{15}((\lfloor \frac{x}{2^{y_1}} \% 2) \rfloor)) \oplus ... \oplus (\lfloor \frac{x}{2^y_n} \rfloor)) \tag{4}$$

where

$$x = \text{the initial inputted number}$$

$$y = \text{the number of the bit that is being shifted}$$

This equation allows the user to put a number and the bytes from a range of 1-16 inside it and then evaluate it to find a random number.

I will perform a sample calculation with the number 3, to show how a random number is derived from this equation.

$$f(3, 1, 9) = \lfloor \frac{3}{2} \rfloor + 2^{15}((\lfloor \frac{3}{2^1} \% 2) \rfloor)) \oplus (\lfloor \frac{3}{2^9} \rfloor)) \tag{5}$$

$$f(3, 1, 9) = \lfloor \ [1.5] \rfloor + 32,768((\lfloor \ [1.5] \% 2) \rfloor)) \oplus (\lfloor \frac{3}{512} \rfloor)) \tag{6}$$

$$f(3, 1, 9) = 1 + 32,768(1) \oplus (0.005859375) \tag{7}$$

$$f(3,1,9) = 1 + 32,768 \oplus (0.005859375) \tag{8}$$

$$f(3,1,9) = 32,768 \oplus (0.005859375) \tag{9}$$

$$f(3,1,9) = 32,771 \tag{10}$$

In a perfect system, you would ideally use the output number as the next x value to generate a random number after this, and you would repeat that system over and over again which ensures that you would get a random number every time. Which I will demonstrate here:

$$f(3,1,(32,771)) = \lfloor \frac{32,771}{2} \rfloor + 2^{15}((\lfloor \frac{32,771}{2^1} \% 2) \rfloor)) \oplus (\lfloor \frac{32,771}{2^9} \rfloor)) \tag{11}$$

$$f(3,1,(32,771)) = \lfloor\ [16,385.5] \rfloor + 32,768((\lfloor\ [16,385.5] \% 2) \rfloor)) \oplus (\lfloor \frac{32,771}{512} \rfloor)) \tag{12}$$

$$f(3,1,(32,771)) = 16,385 + 32,768(1) \oplus (64) \tag{13}$$

$$f(3,1,(32,771)) = 16,385 + 32,768 \oplus (64) \tag{14}$$

$$f(3,1,(32,771)) = 49,153 \oplus (64) \tag{15}$$

$$f(3,1,(32,771)) = 49,217 \tag{16}$$

We can see that a completely random number has been generated, once again, by the equation I derived earlier. Moreover, if we analyze the two outputs, 32,771 and 49,217, no matter how you analyze these two generated numbers, you cannot find a clear and clean pattern between them, which proves the efficacy of the derived equation. Next, I will further prove the efficacy of this equation using statistical modeling.

4.1 Proving the Success of Pseudorandom Number Generation with an Xor Shifting Model

To really test if my mathematical modeling works, I need to test the standard deviation condition on my data.

If we can classify variance as

$$\sigma^2 = \sum_{i=1}^{n} (x_i - \mu)^2 \cdot p_i \tag{17}$$

where

σ^2 is the variance of the random variable.
$(x_i - \mu)^2$ represents the squared deviation of x_i from the mean.
p_i is the probability of X taking the value x_i ,

The standard deviation is the square root of the variance, then:

$$\sigma = \sqrt{\sigma^2} = \sqrt{\sum_{i=1}^{n} (x_i - \mu)^2 \cdot p_i} \tag{18}$$

where

σ is the standard deviation of the random variable.
$(x_i - \mu)^2 \cdot p_i$ is the weighted squared deviation from the mean.

Now I will perform standard deviation on all 20,000 thousand numbers from my data set, however for the convenience of the reader, I will show it for 5 numbers and then just give the value of the standard deviation for the 20,000 number set after.

First I will solve for the mean, which is represented as μ

$$\mu = \frac{23,497 + 44,741 + 55,138 + 27,569 + 46,552}{5} \tag{19}$$

$$\mu = \frac{197,947}{5} \tag{20}$$

$$\mu = 39,589.4 \tag{21}$$

Next, I will calculate the squared differences from the mean, multiplied by the probability, for this to work we need to assume there is an equal probability each number could have been pulled. This will be represented as $(x_i - \mu)^2(p_i)$

For 23,947:

$$(23,947 - 39,589.4)^2(\frac{1}{5}) \tag{22}$$

For 44,741:

$$(44,741 - 39,589.4)^2(\frac{1}{5}) \tag{23}$$

For 55,138:

$$(55,138 - 39,589.4)^2(\frac{1}{5}) \tag{24}$$

For 27,569:

$$(27,569 - 39,589.4)^2(\frac{1}{5}) \tag{25}$$

For 46,552:

$$(46,552 - 39,589.4)^2(\frac{1}{5}) \tag{26}$$

To make these squared differences usable in the next section, I will call each squared difference x, x_1, x_2, x_3, and x_4 respectfully.

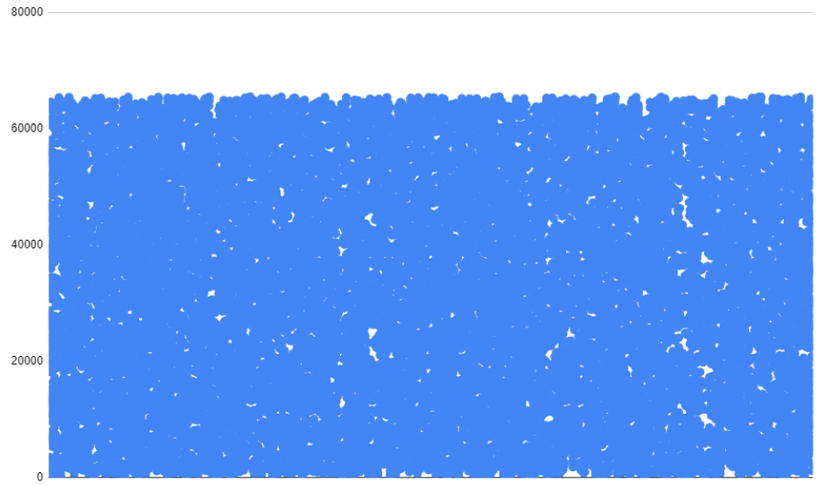
Now, we can finally take the square root of the summation of all these numbers and find our standard deviation for the data set.

$$\sigma = \sqrt{\sum_{i=1}^{5}(x_i - \mu)^2 \cdot p_i} \tag{27}$$

$$\sigma = \sqrt{x + x_1 + x_2 + x_3 + x_4} \tag{28}$$

$$\sigma = 10,450.5 \tag{29}$$

Evaluating my data set using this standard deviation model attains a value of approximately 18,857.39. The randomness of the data is represented in the chart below.



As we can see in this chart, the lack of white spaces tells us that the data is so random it spans over almost the whole of that 65,000 value cap I had placed on the generator. This chart in collaboration with the standard deviation value of 18,857.39 I achieved from my data set shows that the aim of this investigation has been successfully accomplished.

5 Limitations

Even though this investigation has been proven successful by the value of the standard deviation from the section above, this isn't the fullest extent this investigation could have gone to, and there were many limitations within the small investigation that I was able to do. Some ways I was limited within my own investigation are as follows. First, I had to use a generated set of 20,000 numbers and try and perform tests on them. As a human, performing standard deviation by hand on 20,000 numbers was nearly impossible given the time frame that I had to complete this investigation, which meant I had to trust a Rust computer science program I made with the data and the calculation of the standard deviation value I got in the previous section, which means there is always a slight chance something might have gone wrong within my data. Secondly, I am a human, which means there are places where something is bound to go wrong during the course of this investigation, which limits the value of the output of my investigation. Now, more than the investigation itself, the entire scope of PRNG isn't represented in my study, which in itself is a limitation of this investigation, as there could be many ways to attain similar, or even better results, that weren't covered in my study. Thirdly, there is some limitation in the perception of this study if the reader doesn't fully understand what is going on, as there isn't really a great way to visually model this investigation and its data besides the chart provided above, which means the math and the writing needs to be clear enough for the perception of this study to be successful, but as a human, there are many places where something could not have flown properly or felt right to read.

6 Conclusion and Real World Applications

In all, we can see that my aim was met and that the PRNG system that I modeled mathematically will always generate a new random number when given a new, unique input. This was achieved by using mathematical techniques such as flooring, Xoring, and standard deviation, which in the end proved why my model worked for the situation at hand. Despite the limitations that were previously addressed, we can now see that this model is applicable to many things in the world pertaining to cybersecurity. A place we commonly see this model is when you are trying to log in to a Google account on a new, untrusted device, and it sends you random numbers and asks you to select the right ones on an already logged-on device. This is a prime example of the usage of Xor shifting in the real world and shows us places where it can be made of use. Now that we know there is a working model out there, and we now know how to use it, we can revisit the infrastructure of underdeveloped nations, implement this system, and save billions of people from being victims of cybercrimes and digital attacks as a whole. If I were to further investigate this, I would like to test out other

methods and see if they really are more efficient than Xor shifting, or if Xor shifting still remains kind in cybersecurity today. As we take in all the information and use cases for Xor shifting, we can see it has done the world a great service by providing a lot of fundamental building blocks to protecting our data in the vast and scary world of the internet, proving the importance of having systems like this in our world.

A Bibliography

Works Cited

III. www.iii.org/fact-statistic/facts-statistics-identity-theft-and-cybercrime.

IC3. www.ic3.gov/Media/PDF/AnnualReport/2022_IC3Report.pdf.

Arobelidze, Alexander. Random Number Generator: How Do Computers Generate Random Numbers? June 2021. www.freecodecamp.org/news/random-number-generator/.

contributors, Wikipedia. List of Random Number Generators. Nov. 2023.