Determining The Laziest Way to Force Other People to Generate Random Numbers For Us Using IoT Vulnerabilities

mathmasterzach
Villanova University
zachdestefano[at]gmail.com

StarChar
Villanova University
mholmblad13[at]gmail.com

Wednesday, March 32, 2020 Anno Domini

Abstract

The idea of this project is simple. Hack into a nearby IoT device and pull some random numbers off of it. This device can be located either in the same room as you, or your next-door neighbor's house. Anything from the password of their lock screen to the neighbor's baby monitor. The goal is to take privacy-violating hacks on IoT devices and use them just for random number generation in a legally dubious way while also being as lazy as possible.

1 Introduction

In the past decade, the number of IoT devices in the world has been increased at an exponential rate [1]. There has been a similar exponential increase for generating random numbers for use in cryptography, Monte Carlo simulations, digital simulations of the Monte Carlo Casino, RPGs, generating hashes, lottery tickets, picking people to send off to war, and grading papers.

These random numbers are secure against side-channel attacks because the best defense is a good offense. Also, if someone else had the same access to your IoT device that we have using this framework, then you should have bigger concerns about security than someone knowing your random numbers [10].

2 Background

In order to understand this new discovery. The reader needs to understand a couple simple information theoretical concepts of high importance. We strongly recommend a full understanding of the following concepts before approaching this paper: reading and the implications of the asymptotic equipartition property for discrete-time finite-valued stationary ergodic sources.

2.1 Random Numbers

Two, ten, a million, one, sixty-three, four, four, four. These are some random numbers. Rolling a die is considered truly random. Creating a good random number generator (RNG) is a popular theoretical problem that was originally formally addressed by Donald Knuth. [4] RNGs require a seed in order to generate numbers. This isn't a seed that is used to grow a plant, but a seed for creating hundreds of thousands of numbers. It is addressed later on how our random numbers are generated.

Figure 1: An example of older random number generation technology

The goal of generating random numbers is always for them to be truly random. Unfortunately, many attempts at true randomness have often ended up being the creation of significant pseudo-random number generators (pRNG). There is a well-known statistical test suite made by Robert G. Brown known as Dieharder [3]. Many pRNGs have been tested using this test suite. However, it is still unknown whether a true RNG actually exists.

2.2 IoT Vulnerabilities

An IoT (Internet of Things) device is a device which has at least one sensor and is connected to the internet. These devices are useful for various levels of home automation; however, we have discovered an alternate, more powerful set of features implicit in these devices. None of the manufacturers that we surveyed list this feature in their device capabilities, so we have taken on the task of making these "hidden features" more known to the public while demonstrating the extent and effectiveness of these features. As a result of our research, we have discovered that most IoT devices have a built-in random number generator that anyone can use with proper hacking. We are currently unsure about why these truly marvelous features are not listed anywhere by the manufacturers.

3 Methods and Results

We will now demonstrate various theoretic frameworks for using IoT devices to get random numbers from other people without them knowing. We will discuss how to get random numbers from other people's babies while they sleep, from their phone passcodes, and from their general habits (particularly those involving toasters). We will assess each of these sources for their information theoretic properties to determine the rate of information production by these stochastic sources so we might make proper comparisons between their information entropy [6].

Due to budget cuts and time constraints, any proper calculations of entropy have been replaced with approximations by using a large sample of data gathered from the source and compressed with 7-Zip (7-Zip implements the Lempel-Ziv-Markov chain-Algorithm). Using

the Lempel–Ziv theorem which states that: the algorithmic entropy of a file is bounded by its Shannon entropy (or something like that more or less), we can assume that our experimental method of zipping large quantities of data suffices to produce a reasonable ballpark estimation of the entropy of our sources. [2]

To quantify our results and for a proper comparison, entropy per bit is only one part of the formula. We must also quantify the amount of work/time it takes to acquire this entropy so that we can determine the optimal way to get entropy with as little effort as possible. We have designed the following formula to help us calculate and ultimately maximize the number of shannons per joule, the amount of entropy per bit as a function of personal energy.

$$\frac{\frac{bits\,of\,entropy}{bits\,of\,information}}{\frac{gathering\,work}{time} + activation\,energy}$$

Given that there have been no prior references to this important, shannons/joules unit, we have named this new important unit lazy-lokis which will be abbreviated in this paper as zk.

3.1 Secretly Using People's Babies for Random Numbers

A baby is a... wonderful noise maker. They make a lot of different sounds at different volumes and frequencies. This is perfect when converted to numbers. Thus, the baby's cry is a successful RNG, because it is practically impossible to guess what numbers the baby will produce next when they cry.

Figure 2: Example of a baby in a sophisticated rig for measuring information theoretic properties

In order to assess the information theoretic properties of babies as noise sources for random number generation, we must appraise and quantify them as sources of entropy. Unfortunately, we were told that, "under no circumstances," would our Institutional Review

Board approve using human subjects for this project, but in order to advance science, we undeterred by this bureaucratic hurdle, downloaded an open dataset of baby cries from GitHub to estimate the information theoretic properties of babies.

The "donate a cry corpus" is a dataset of baby cries which separates all baby cries into a subset of the Dunstan Baby Language categories for cries (hungry, needs burping, belly pain, discomfort, tired) and further qualifies the cries into sex (male, female), and age buckets (0 to 4 weeks, 4 to 8 weeks, 2 to 6 months, 7 month to 2 years, more than 2 years) [9]

Our results are documented in the following table with the units being the fraction of the number of bits of entropy in each bit of data:

Data	0-4wk		4 - 8wk		2 - 6mo		7mo - 2yr	
	M	F	M	F	M	F	M	F
Hungry	0.7176	0.6618	0.6576	0.6904	0.7257	0.7211	0.7038	0.6378
Needs Burping	0.6920	N/A	0.9034	0.7285	0.8900	0.7337	N/A	0.7295
Belly Pain	0.8153	N/A	0.5767	N/A	0.7062	0.6948	0.6571	N/A
Discomfort	0.3763	N/A	0.6964	0.6719	0.7068	0.8231	0.8524	N/A
Tired	0.7609	0.6787	0.6685	0.6383	0.7189	0.6436	0.7230	N/A
Average	0.6724	0.6702	0.7005	0.6823	0.7495	0.7233	0.7341	0.6837

Assuming that all five states that a baby can be in occur with equal frequency, we can assume that the information entropy of a baby's cry is approximately .7020. That is to say that we can expect to get .7020 bits of entropy on average for every 1 bit we siphon from baby's cries. If we have the fortune of selecting a specific baby for random number generation, we should prefer to choose a male baby 2 months or older, although this does not give a major increase in entropy over other categories.

Future research in this area can involve provoking babies with light and sound features on the cameras to see if this increases the entropy of their cries, and expanding our dataset to include categories for which there was no data available to us.

3.2 Secretly Using People's Habits for Random Numbers

Linux uses various sources such as timing in between user interaction events as entropy which is stored in /dev/random to be used later, but when this runs out of entropy it is a blocking operation while it waits for more entropy. This is relatively limited because Linux machines are often lifeless servers connected to the internet running from within a dark room for decades at a time with minimal human interaction.

These machines are deprived of the warmth and love required to create random-ish numbers, but it is possible to, with the power of modern technology, allow these machines to experience the joy of an "endless" stream of random numbers without any need for warmth or love. This process involves taking devices that individuals use directly more habitually such as smart toasters, smart water bottles, smart fridges, and smart crockpots, and collecting and transferring the human interaction information so it can be used for random number generation.

Now perhaps you might begin to protest and say, /dev/urandom is universally preferred over /dev/random and neither offers better randomness because they are ultimately run through CSPRNG but /dev/urandom is better because it doesn't have the blocking issue and thus breaking into other people's toasters is a wildly unnecessary and ethically dubious

way to get random numbers. We answer that strings of true random numbers are like beautiful heartfelt poems and thus you are a joyless husk of a human being to believe that the pseudo-random numbers that a lifeless computer generates are in any way of the same caliber.

Human interaction with these devices on the macro level is largely predictable, ex. toaster used at breakfast same time Monday-Friday within a 10-minute window with the time dial set to about 1 minute or fridge opened at precise 5-minute windows during the day to get drinks, snacks, and frozen pizzas, but at a more precise level, the variations within these patterns is completely random. The number of milliseconds above or below normal modulus some desired small value has an entropy value that approaches 1 bit per bit on analog toaster input. There is the first issue for us that this exceedingly high-quality randomness is only found in a smaller quantity; however, this is not the biggest issue with this method. Many IOT devices in this area have shifted away from allowing this error through digital interfaces. This poses a potential risk, but for now, it appears to be safe to use this method in production.

3.3 Someone's Lock Screen Password

Encryption often involves a salt, which is

Definition 3.1 (Salt)**.** Random data that is used as an additional input to a one-way function that hashes data, a password or passphrase.

Salts are generally small numbers. Frequently on the iPhone, a user's password for their lock screen is 4 numbers. The iPhone is the most common kind of phone. This is a perfect salt for an encryption algorithm. Thus, the lock screen code is a very useful set of random numbers. Plus, outside of general cases and permutations of the classic 1234 password, phone passwords have an extremely wide variety.

The issue arises when we discuss how to get these passcodes. Let's say we're inside the comfort of our home. Then the easiest way to get the passcode is through the neighbor's WI-FI or picking up the signal off a nearby cell tower, which is a ton of work. An easier method is to go to the nearby coffee shop across the street because everyone connects to that coffee shop's free WI-FI capabilities. The process is simple, walk in the coffee shop, buy a coffee, sit at the corner table and act like you're writing the next great novel like most people. Meanwhile, lots of phones and people come through, and we harvest lots of phone passwords. As optimal as this is, lots of people visit the same coffee shop. Thus, to keep the randomness, we need to frequent many different coffee shops. The amount of zks that this will cost is exponential.

Considering the cost, these must be incredibly random numbers to be worth our time, and as it turns out, using a collection of passcodes sorted by frequency [5] we calculate 13.2627 bits of entropy per symbol where symbols are the set of integers 0 through 9 which when the divided by the number of bits per symbol, gives us .9981 bits of entropy per bit of data.

4 Method Comparison

With the values above, we now believe ourselves to be adequately prepared to ask the question of which aforementioned source is best for random number generation. For comparison,

we will not use the raw entropy values, but instead, lazy-lokis to account not only for the optimality of the entropy of the source but also to account for the effort required to gather quantities of this data. The exact calculation of these values can be trivially done with the data provided and our formula above, so this calculation has been left as an exercise to the reader.

We have performed these calculations, and have determined that the method which maximizes lazy-lokis is hacking into baby monitors. Though this method generates fewer bits of entropy per bit of information than some of the other methods such as passcode stealing or using toasters and fridges, the sheer number of bits per time and the ease of access puts it over the top by several orders of magnitude.

Perhaps a phone passcode contains more entropy, but to steal phone passcodes I have the tremendous activation energy of driving to a local public shop or gathering point, potentially purchasing a beverage at said location to appear inconspicuous, and making sure to hit a different location each time so that I am not sampling the same set of passcodes each time. Similarly, the time delay in waiting for individuals to use their toasters and fridges limits random generation to specific time intervals and collecting larger collections of random numbers relies on hacking into more devices and waiting for each of those. Thus for the purest form of distilled random numbers, toaster and fridge micro-input information, the cost of acquiring these numbers is also maximal.

The baby monitor approach, however, allows me to get a continuous stream of data from other people's homes throughout the world from the comfort of my own home. The activation energy cost approaches 0 as I have around 2 years worth of data that I can get from any individual baby without the baby's data becoming stale or predictable. There is, of course, the issue of having a male baby vs a female baby as there is no way to tell what kind of baby is in the household. This gets tedious to find out, but it's a lot more comfortable than going to the coffee shop and buying something to fit in with everyone else that is there.

5 Experimentation

Our non-existent lawyers have told us that to say that we DEFINITELY have NOT, do NOT intend to, and do NOT encourage experimentation with these methods; however, if one was to hypothetically attempt to test one of these methods there are various things that they could try.

The following case study discusses many ways that have worked in the past to remotely access any baby monitor's audio feed [7]. Using the national vulnerability database, it is trivial to find more examples on which the same methods are effective. We have been advised to say that we have NOT tried any of them and do NOT advocate taking a look at various methods for getting remote access to baby monitors and trying them for yourself.

We are not lawyers and carrying out any of these tests, (which we did NOT), would be a legal gray area because, according to the definitions of what constitutes a crime in the Computer Fraud and Abuse Act (United States Code 18 Section 1030), taking entropy from someone without them knowing via one of their smart devices does not appear to directly violate this act [8]. Seriously. Take a look. Then consult a lawyer just to be safe. And let us know what they say.

The closest thing to violating this act that we could find can be found in the statement: "knowingly and with intent to defraud, accesses a protected computer without authorization,

or exceeds authorized access, and by means of such conduct furthers the intended fraud and obtains anything of value, unless the object of the fraud and the thing obtained consists only of the use of the computer and the value of such use is not more than $5,000 in any 1-year period" [8]. Considering that the thing obtained consists only of the use of the computer and the entropy we are gathering probably isn't worth more than $5,000 then you are PROBABLY safe, but then again, we aren't lawyers.

6 Conclusion

Thanks to this careful research we have reached a number of practical conclusions about random number generation that we expect to be taught in every information theory class as early as next semester. Zerothly, the higher quality the random numbers are the rarer and more difficult they are to acquire. Firstly, ease of access and random number quality are inversely proportional which implies that the lazy-loki metric as a function of entropy is continuously decreasing on the open interval (0,1). Secondly, male babies exhibit higher variance in entropy generation when compared to their female counterparts at all ages which lends evidence to the greater male variability hypothesis.

It can be trivially seen that in general using the cries of babies is the best option for collecting random numbers from other people and that the highest quality of random numbers are quite rare and come from smart toasters. It is left as an exercise to the reader to continue testing this idea.

References

[1] A Cursory Google Search

[2] Avinery, Ram *Universal and accessible entropy estimation using a compression algorithm* The Raymond and Beverly Sackler School of Physics and Astronomy, Tel Aviv University, 2019. Print.

[3] Brown, Robert G. *dieharder* Duke University Physics Department, 19 June 2017. Web.

[4] Knuth, Donald *The Art of Computer Programming Volume 2: Seminumerical Algorithms* Addison-Wesley, 1968. Print.

[5] Miessler, Daniel *SecLists: The Pentester's Companion* GitHub Repository, 2012. Data.

[6] Shannon, C. E. *Prediction and Entropy of Printed English* Manuscript, 1050. Print.

[7] Stanislav, Mark *HACKING IoT: A Case Study on Baby Monitor Exposures and Vulnerabilities* Rapid7, 2015. Print.

[8] United States Legislative Branch *Computer Fraud and Abuse Act (United States Code 18 Section 1030)* Comprehensive Crime Control Act of 1984. Print.

[9] Veres, Gabor *Donate a Cry Corpus* GitHub Repository, 2015. Data.

[10] Wurm, Jacob *Security Analysis on Consumer and Industrial IoT Devices* Department of Electrical and Computer Engineering, University of Central Florida, 2016. Print.