[Mandeep Singh, Albert Carlson. (May 31, 2024). An Approach to Watermarking Using Polymorphic Algorithms For Increased Data Security. Austin Community College/IEEE.]

An Approach to Watermarking Using Polymorphic Algorithms For Increased Data Security

Mandeep Singh[*], **Albert Carlson**[†].

[*]Department of Computer Science & Engineering, Santa Clara University, USA
[†]Chair for Entropy and Encryption, Quantum Security Alliance &
Computer Science Department, Austin Community College, Austin, TX, USA
Email: [*]msingh5@scu.edu, [†]albert.carlson@austincc.edu

***Abstract*—Watermarking is a method of identifying and controlling ownership of distributed media. It is also extended to documents with suggestions about how steganography can be applied to documents that are reproduced via xeroxing and electronic means. In this paper, we show how polymorphic random number generators can continuously morph watermark locations so that they are untraceable to the human eye or ear and cannot be replaced without being detected. As part of the paper, we introduce a new method for randomization based on wave action and common waveforms found in electrical engineering, applying that technique to the placement of watermarking data. These techniques are then applied to securing data and the ownership of documents.**

Keywords: Data Security, watermarking, steganography, polymorphic algorithms, wave randomization, polymorphic random number generators

I. Introduction

Watermarking is a form of steganography that provides both a method of positively identifying the origin, control, and ownership of a document as well as a way to insert and transport data. The method involves techniques to insert unique codings or patterns within the original file. This can be done by adding words to a document, images to a graphic, or even manipulation at the bit level. Such techniques can also be used in large, streaming files used for video and data delivery to prevent the piracy of intellectual property.

Watermarking, including steganography, is both related and complementary to encryption. It is used as a method of non-repudiation and identification of the owner of the data or intellectual property. A defining characteristic of watermarking is that the technique is invisible to normal users. However, static watermarks can be easily detected and replicated by a malicious entity. This paper explores different methods using polymorphic algorithms similar to how they are used in encryption [1] to implement deterministic yet unpredictable watermarks into files for increased data security.

The paper is organized into five sections. The first is the introduction to the paper, followed by a necessary background of random number generators. Section 3 discusses the wave method of randomization, with the requisite mathematics. Next, Section Four covers watermarking with polymorphic principles. Section Five gives the conclusions for the paper and also introduces some future directions of research.

II. Background

A. Randomization

Random number generators (RNGs) are functions that provide a number stream that is seen as random for a "short" sequence. In this case, short is a number ($m << \infty$). Each RNG has a characteristic sequence of size $|\lambda|$ and to be secure it is desired that the cycle length be greater than the number of accesses made to it by the algorithm/program. RNGs have the property that each number in the sequence (x_i) from an arbitrary starting point in the sequence (known as the "seed") is unique. Uniqueness comes from the RNG function which is

$$x_{i+1} = f(x_i) \tag{1}$$

where $f(x_i)$ is the RNG function. Since the function cycles, $\forall n > 0$

$$x_i = x_{i+n\lambda} \tag{2}$$

A "maximal cycle" occurs when there are at most two cycles and one cycle has at least $2^b - 1$ values, where there are b bits in the size of the random number.

Every RNG is a function and the information that is found in examining each new number in the sequence allows an observer to gain information about the RNG. This is similar to the concept of entropy [2] in information theory (IT) [3]. Similarly, the "unicity distance" for RNGs is given as the number ρ, describing the number of entries required to be able to determine the particular RNG and its location in the output cycle. At that point, an observer can accurately predict the next number in the sequence.

Random numbers are required for several algorithms in IT in addition to obscuring. Those applications include both compression and modes. Modes are highly dependent on RNGs and are widely used to add security to encryption.

III. Wave Randomization

As in most encryption or reordering functions, randomization functions are key to success. IT indicates that redundancy should be avoided if possible. Creating a function ($f(x)$) that avoids predictability requires one that is both asymmetrical in its period (λ) and sufficiently long to exceed the unicity distance (n) in the message [4], [2], or portion of the message, encrypted. Most common keys are selected and then used for the entire length of the message, if not for many messages. Modern cryptography teaches that the bigger the key the larger the key space and, consequently, the more secure the message. There are, however, flaws to this general line of analysis. First, not all keys result in unique encryptions. This may be due to limited characters used in the message [5]. The entropy of a message determines how much information is needed to break that message. Most decryption techniques cannot break an encryption at the theoretical minimum for an encryption. Normally it requires from 200 - 500 times that number (n) in order to have a reasonable chance at recovering the message [6]. One way to vary things is known as the One Time Pad (OTP) [2], [7]. OTPs were originally called "Vernam ciphers" [8] and were introduced in 1917. In this type of encryption, every character in the message is given its own key and/or cipher. The technique of varying the key is instructive, but it requires governments and large corporations to be able to afford the process and its overhead. If the key is changed on an irregular basis, then more characters are sent using the same key. If the key is changed on a calculable schedule, it is then possible to attack the encryption via the function that selects the keys [9]. Therefore, a randomizing function whose values change over the duration of the message is needed.

Cipher and random function designers strive for true randomness, but even proving randomness is difficult [10]. Some functions are known, and have been proven to be, random. Calculating those functions is not possible, so a function with a long periodicity, compared to the message, is required. One such way of doing this is to create a function that describes something similar to the waves in the ocean. In that case, the frequency of waves ($w(t)$) is due to a number of factors: regular natural phenomena (such as tides [ω_n]), irregular natural phenomena (such as wind, [ω_d]), extraordinary events (such as man-made events like boats sailing), and reflections off of obstructions (both man-made and natural) that dies off over distance and time according to the function $f_d(t, l)$. That is

$$\omega(t) = \sum_{i=1}^{n} \omega_n(t) + \sum_{j=1}^{m} \omega_d(t) + \sum_{q=1}^{p} \omega_m(t) + f_d(t, l) \sum_{v=1}^{x} \omega_r(t) \tag{3}$$

assuming that all perturbations are accounted for from the beginning of time. After a sufficient amount of time waves can be dropped from consideration, depending on the the rate at which they are dampened.

The main problem with using such an algorithm is that sine waves are well understood. Other similar periodic waves can be added to help change the regularity of the system and cause what appear to be discontinuities at various points in the wave flow. Adding different wave functions also allows for adding or subtracting offsets in the period. However, it is important to ensure that the periods of the different waves are NOT integer multiples of each

other. That is, for two waveforms ω_i and ω_j with the associated wavelengths λ_i and λ_j,

$$\exists!(n \in \mathbb{I}),\ (i \neq j)\ n\lambda_i \neq \lambda_j \tag{4}$$

where $\lambda_i < \lambda_j$.

Other candidates for inclusion in the randomization function are the unit step function, ramp function, and triangle wave. Of the three, the ramp and triangle wave functions are similar, so only one should be included. The ramp function is most like the unit step function, in that it monotonically changes whereas the triangle wave is more like the sign wave in that it changes in both the positive and negative directions. Choosing the ramp function is easier to deal with mathematically, so it was chosen for inclusion. The unit step function $u(t_i)$ is described by the equation

$$u(t_i) = \begin{cases} 0 & \text{if } t < t_i \\ 1 & \text{if } t \geq t_i \end{cases} \tag{5}$$

If added to a waveform there is a sudden jump to the total value of the function of $f(s)\ u(t_i)$ at time t_i, where $f(s)$ represents some scaling function associated with the unit step function. That is to say, the change in the function is not restricted to unity. Nor is it required that $f(s) > 0$. The effect may be a subtraction from the summed wave, as well as an addition. Similarly, an instantiation of a ramp function can also cause a gradual rise in the same way. The ramp function $r(w_i, t_i)$ is described by the equation

$$r(\omega_i, t_i) = \begin{cases} 0 & \text{if } t < t_i \\ \frac{t - t_i}{\omega_i - t_i} & \text{if } t_i < t \leq t_i + \omega_i \\ 0 & \text{if } t \geq t_i + \omega_i \end{cases} \tag{6}$$

Randomizing using these component functions requires choosing a value between $\pm m$ for cyclic functions that can be mapped as limits for a pseudo-random number generator (PRNG) used in standard software languages. Three approaches are possible. The first is to consider each application of a new wave pattern and normalize the effect to the range of the number of functions applied between time 0 and t_i. That is, let

$$f(s) = \frac{m}{|\omega_i|} \tag{7}$$

or give each wave the weighting of the number of waves that are active at the point in time ($|w_i|$) described by s.

The second approach is to replace $|w_i|$ by the total number of waveforms over time, $|w_t|$ so that the discontinuity is normalized over all time. This changes the formula to

$$f(s) = \frac{m}{|\omega_t|} \tag{8}$$

Finally, the third approach is to decide on some weighting function $K(x)$ that allows for normalizing each wave function separately and not necessarily equally. If this approach is taken as the general approach, then a general form of the combined randomization function is given by

$$\begin{aligned} p(t_i) = &\frac{K_s}{a} \sum_{i=1}^{a} \sqrt{-1}^{2m_i s_i} sin\omega_i(t) \\ &+ \frac{K_u}{b} \sum_{i=1}^{a} \sqrt{-1}^{2m_j u_j} u(t_j) \\ &+ \frac{K_r}{c} \sum_{1=1}^{c} \sqrt{-1}^{2m_k s_k} r(\omega_k, t_k) \end{aligned} \tag{9}$$

which has a periodicity of

$$\lambda = 2 \prod_{i=1}^{n} \omega_{max} m_{max} r_{max} s_{max} u_{max} \tag{10}$$

assuming that $\omega_{max} m_{max} r_{max} s_{max} u_{max} \in \mathbb{P}$. In this equation, the use of the terms of the form $\sqrt{-1}^{2m_j u_j}$ allows a waveform to be added and subtracted over time, creating further discontinuities that appear to be random. It also allows for additional degrees of freedom to be added into a calculable function for obscuring the nature of the waveforms and location of exact transitions. These functions are also well known and characterized, as well as each to calculate on the fly.

If the size of the message ($|M|$) is known, then the goal is to achieve a periodicity (λ) such that

$$\lambda \geq |M| \tag{11}$$

Although it is preferable that for each type of function, denoted by $f(x)$ that coefficient also have the property $\forall x_i \in f(x) \rightarrow x_i \in \mathbb{P}$, it is not required. Further, it is also desirable that if the coefficients are not all prime numbers that the coefficients not

all be integer multiples of the maximum coefficient. That is, $\forall x_i inf(x)$

$$x_{max} \; mod \; x_i \neq 0 \tag{12}$$

If the size of the message is not known at the beginning of the encryption process an arbitrarily large number can be assumed. A message larger than the assumed number can then be treated as if it were a series of smaller messages whose size ($|M - i|$) is smaller than λ. In that case, all that has to be done is to Diffie-Hellman a new set of parameters and then continue the encryption with the new parameters and reparameterize as needed. A list of prime numbers that can be used as parameters will speed reinitializing things up. Values chosen from that list should be randomly selected with a unique mapping employed. Seeds for the PRNG are then selected by calculating the function at time s. This does not guarantee unique values at any two times, s_i, s_j where $i \neq j$, but it does randomize the seed value, especially if there is fine granularity in the function. Randomization is used as the seeding value.

Consider the waves on a beach. The pattern of waves seems to change randomly. Waves are shaped by many factors such as wind, weather, seismic events, passing boats, currents, reflected waves, etc. that create a complex set of stimuli. These factors cause changes to the patterns of wave arrival that appear to be random, but are actually deterministic. A mathematical model of this effect can be expressed as

$$w(t) = \sum_{i=1}^{n} w_n(t) + \sum_{j=1}^{m} w_d(t) + \sum_{q=1}^{p} w_m(t) + f_d(t,l) \sum_{r=1}^{x} w_r(t) + ... \tag{13}$$

where individual instances of each type of wave component add together to make a composite wave train in the water. A similar approach with the wave components provides a model for simulating randomness [11]. Let each term represent a different type of wave function. Now let each wave function represent an RNG. By having a composition of RNGs to cycle through, it is possible to create the desired wave randomization. The more components used, the more random the result will look. The cyclic nature of these component functions provides a complex function and therefore, a more difficult analysis for an attacker. Complexity does not always equate with security, but most secure functions are also complex. With this RNG function, it is possible to better protect other algorithms, such as watermarking.

IV. Watermarking

Watermarks are an identifying message embedded in a frame, program, or file. Most users are familiar with video watermarks, a picture or logo added to a video program in one corner that identifies the company (or channel) that is transmitting the program for public consumption. However, a watermark does not have to be visible to be effective. Invisible watermarks are more effective if the watermark is to be used as a means of establishing identity and progression of possession. Invisible watermarks are typically disguised by placing them somewhere that the data comprising the watermark will not be noticed. In a video file, most formats record the color or color and intensity of each picture element, or "pixel." If the value for each pixel is encoded in binary and a matrix is made for the values of each weighted binary bit, it can be taken as a group and the values can be viewed as a whole. That is, let a pixel be represented by three base colors, red (R), green (G), and blue (B). Each color has an intensity represented from $0 \leq x \leq 2^{|b|}$, where $|b|$ is the number of bits in the intensity encoding. For current encodings, the number of bits is almost always at least 8 bits. In the future, this should continue to increase. This type of encoding is known as "bit layer encoding," or BLE [12].

Assume that current video formats use 8 bit RGB to describe a pixel. The human eye is not able to distinguish variations in colors for more than 6 bits in any of the color domains. Blue color sensitivity is even lower than that for R and G, only 5 bits. Therefore, the 2 least significant bits (LSBs) of each color are effectively indistinguishable. The same can be said of the 3^{rd} LSB of the B layer. Messages can be placed in the bits of those layers and will be invisible to the human eye. This provides the means to place a watermark in a picture. Any watermark that can be encoded and whose size is less than

$$|w| \leq 7 l_p h_p \tag{14}$$

where l_p is the width (in pixels) and h_p is the height (in pixels) of the frame. Placing the watermark in the frame and leaving it in the same place in the frame, a static watermark, can be found and recovered by applying a simple modified intersection function

$$l_w = \hat{\bigcap}_{i=1}^{n} F_i \tag{15}$$

where

$$\hat{\bigcap}_{i=1}^{n} = \begin{cases} b_i & \text{if } b_i = b_j \\ x & \text{otherwise} \end{cases} \tag{16}$$

On the average it will take

$$|F| \approx lg|l_w| \tag{17}$$

to find the watermark.

Unfortunately, a watermark that stays in the same place from frame to frame can be identified and attacked using this approach. Once identified, a simple filtering program can be written to randomly select a new bit value for each of those bits. This program is known as "scrubbing" the watermark. If the replacement uses a probability of $p(x = 1) = 0.5$ to make the replacement, the probability of the watermark retaining intact is

$$p(l_w) = \frac{1}{2^{|l_w|}} \tag{18}$$

Scrubbing does not affect the video as long as the bit replacement value replacing the data is random. Preventing scrubbing requires that multiple copies of the watermark be placed in the frame and that the locations of the watermarks be randomized and moved throughout the frame. Inserting the watermarks in the frame is equivalent to a permutation cipher (or any other equivalent function) using a random key and permuting the watermarks in their various forms into the combined RGB representation for the correct layers of each color. Let X_i be the i^{th} LSB for the color X. Further, let RGB' be the following bit layers concatenated together

$$RGB' = B_3||R_2||G_2||B_2||R_1||G_1||B_1 \tag{19}$$

A permutation key (K_p) is constructed for a block the size of $|RGB'|$. The first $|l_w|$ substitutions are then made using a mapping of

$$l_w \mapsto RGB' \tag{20}$$

This mapping allows for substitutions in any order and produces a possible number of mappings given by

$$|k| = \binom{|RGB'|}{|l_w|} \tag{21}$$

The key changes for every frame. Therefore, the total number of possible watermarks for $|F|$ frames is

$$|K_f| = |K|^F \tag{22}$$

If the watermark includes the identification of the stream and the frame number, then the watermark is easily recovered. This identification may be encrypted or plaintext. The number of watermarks can also be varied to further compound the problem of finding and scrubbing all watermarks from the video. This is applicable for any RGB combination now and in the future.

Watermarking is not limited to video files. One possible extension to watermarking is to interpret watermarks that do not correspond to the node of an audio file that is paired to a high pitched tone. In this case, a signal is added to the audio mix and a tone that is based on the embedded watermark is subtracted from the base frequency to create a new frequency. Let the base program be represented by

$$p_p = \sum_{i-1}^{n} Asin(\omega_i t) \tag{23}$$

where $s_i \in (!it)$ represents the sound coming from a single source i. The source may be an instrument, voice, or any other source contributing to the final sound. Further, let

$$\omega = sin(\omega_\lambda t) \tag{24}$$

where ν is a frequency selected due to its ability to annoy the listener. Therefore, with the watermark added, the audio program (p_a) is described by

$$p_a = p_p + B(\omega - \omega_d) \tag{25}$$

where ω_d is the waveform derived from the node pairing. The difference between the two waveforms creates a beat frequency oscillator (BFO) whose frequency is

$$\omega - \omega_d = Bsin(\omega_\lambda - \omega_d t) \tag{26}$$

If the frequency of the BFO (ω_{BFO}) is in the audible range

$$300Hz \leq \omega_{BFO} \leq 20kHz \tag{27}$$

then an audible sound is added to the program. The sound can vary or stay in the same range. In either case, it will be detectable and may degrade the listening experience. If a continuous tone is desired, then the watermark can be created to return a constant value if incorrect. That constant value (K) can be then used to produce the tone by inserting it into the modified BFO equation as

$$(\omega + K) - xK = Bsin(\omega_\lambda - xK)t \tag{28}$$

where x is defined as

$$x = \begin{cases} 1 & \text{if watermark is correct} \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

Files that display characters as part of their output can embed watermarks by varying the space between words or the height that a character is located with respect to the horizontal spacing from line to line. An offset of 4 indicates a bit with a value of "1" and no difference is a value of "0". There are some limitations to this method. If a vertical offset is added to the character, then this can only be read if the character is not a blank space. No such limitation is imposed for differences in horizontal spacing. The intent is to add offsets can are small enough to pass for the normal variation in printing and are virtually undetectable to the human eye. This method has the advantage of remaining detectable either in the electronic version or when reproduced. Care must be taken when justifying text to ensure that the justification retains the offset and conforms to the proper offset.

There exist quantized vertical spacings, $|s| + \delta$, $2|s| + \delta$, where $s \in \{s, 2s, s_{j,1}, ...s_{j,n}s\}$ that act as quantized distances for quick comparison that determine the bit value of the embedded bit at that character location. For a character spacing (a) a logic '0' occupies the area $ns - \Delta \leq a \leq ns$ while the logic '1' occupies the space $ns \leq a \leq ns + \Delta$. To add a bit of a buffer the offset for the bit can be adjusted to $(-1)^b \frac{\Delta}{2}$, where b is the value of the bit being encoded.

Applications of the polymorphic environment can be illustrated via watermarking videos. A video can be watermarked invisibly to a viewer. This watermark is used to identify the source of a video that has been pirated [13]. Human eyes cannot distinguish between the color difference of a single pixel value. Color is represented by three 8-bit numbers - one for red, one for green, and one for blue. Combining the three colors results in the final blended color seen by humans. This encoding for color data results in 16,777,216 possible colors. Now, consider the number of pixels in a picture. For a 2-D picture, like video, there are $l * w$ pixels. The current standard resolution is 1920 x 1080 pixels which results in 2,073,600 pixels per frame. The human eye can recognize patterns easily. If the same pattern were to be used in all frames, it could be detectable by a user. Subtle changes, however, can be made and remain undetected. Consider changing a set of random bits in the picture to embed a message or identification tag. There are 74,649,600 possible locations to hide the bits of the message with $b!$ possible orderings. That makes 74,649,600 $\times b!$ possible orderings and locations per frame. The standard video is approximately 30 frames per second and most movies are about 120 minutes long. So there are approximately $1.6 * 10^{13}$ possible locations for data in a movie. With these many locations, it is relatively easy to embed multiple copies of the message or tag. Now think of the frame as a rectangle except this is really just a 3-D cube with one layer for each color bit. All that is needed is to define the data, select locations using some function(s), and change the appropriate bits to be the contents of the embedded message or tag. Note that the lower three bits of all colors are indistinguishable to the human eye and the fourth lowest bit in the blue range is also indistinguishable. In practice, these bits are not static because using the same values in each byte results in a frame that looks "blocky" and unnatural. Therefore, these bits are typically randomized before transmission. Slipping in a bit does not seem unnatural and would be unremarkable to an analyst.

The next step is to add watermarks with a polymorphic RNG such as those mentioned for creating keys [13] using the previously mentioned signatures. The signatures will be used to fill in the required settings for the waver approach. Since the wave randomization functions change rapidly and at dif-

ferent frequencies, they help defeat the prediction of the next key or action. Many changes can be made in a text file that can be read. The program usually sets very specific formatting parameters. Small changes in the formatting can be read as embedded bits. For example, changes in margins, indentation, and spacing can be embedded. The changes in the negative direction can be seen as a '0' bit and changes in the positive direction can be seen as a '1' bit. Again a polymorphic RNG can be used and the number of possible changes depends on the file type and length. Tracing the flow of confidential documents can also be done via this type of watermarking. Watermarking gives new tools for assuring security and passing messages between users.

V. Conclusions and Future Work

Watermarking can benefit from polymorphic practices. A message or encryption placed in a file requiring security can benefit from the large number of positions in which data can be placed, requiring a binomial choice of possible locations in a file and a factorial treatment of the information to recover the correct order of that data. Patterns of placement can be changed in pages of data or pictures using those units (or portions of those units) in the same manner as shards in encryption. Since the method used for hiding data in any type of stenographic file using polymorphic mathematics requires "randomness," a polymorphic RNG should be used as the strongest option. Design for such an RNG, one based on morphing physically unclonable functions (PUFs) [14] and decoupled from the original RNG seed, needs to be developed.

The treatment of the presented material has been largely theoretical. Results from implementing the system will be compiled and presented in a later paper to prove the efficacy of the mathematics and algorithm. This work will show that both a continuous and discrete approach to the random placement of information is possible. Empirical data on the security of the placement of the data will then be compiled and reported.

A better information placement algorithm requires that sharding[15] and data positioning in the file be developed from local, rather than global, language statistics. Presently, global statistics are almost universal. A study of the local entropy and unicity distance resulting in a simple calculation algorithm is required. More information is also required to estimate the required size of a masking file for the size of a message that it can hide. Knowing the relationship between the two will help to properly select a file in which to hide the data. A survey of such a value has not shown work in this area to date. A study of the effectiveness of hiding plain text information in a watermarked file is also required. The object of the study is to decide if encryption of the message before insertion of the data is needed. This analysis should also include the effect of compression as a security measure for hiding data in a file.

References

[1] Mandeep Singh and Albert Carlson. Exploring polymorphic algorithms and their use in cryptography. *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0428–0434, 2024.

[2] Claude Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656 – 715, 1949.

[3] Thomas Cover and Joy Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc, New York, 2nd edition, 2005.

[4] Paul Garrett. *The Mathematics of Coding Theory*. Pearson/Prentice Hall, Upper Saddle River, 2004.

[5] Albert Carlson. *Set Theoretic Estimation Applied to the Information Content of Ciphers and Decryption*. PhD thesis, University of Idaho, 2012.

[6] Robert E. Hiromoto, Albert H. Carlson, and Richard B. Wells. An information based approach to cryptography. In *The 6th Computer Information Systems and Industrial Management Applications*, 2007.

[7] Uli Maurer. A universal test for random bit generators. *Journal of Cryptography*, 5(2):89–105, 1992.

[8] Gilbert Vernam. Secret signal system.

[9] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In *Fast Software Encryption, Fifth International Workshop Proceedings (March 1998)*, pages 168 – 188. Springer-Verlag.

[10] Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley, Massachussets, 1997.

[11] Albert Carlson, Steven B. Cohen, Lawrence duBoef, and H. Stan Johnson. Block management unifi system and methodcation, patent number us 9,436,815.

[12] Richard Wells. *Applied Coding and Information Theory*. Prentice Hall, Upper Saddle River, 1999.

[13] Albert Carlson, Steven B. Cohen, and H. Stan Johnson. Digital watermarking for secure transmission between a source component and a node device, patent number 8,924,730.

[14] Basal Halak. *Physically Unclonable Functions, From Basic Principles to Advanced Hardware Security Applications*. Springer, Cham, Switzerland.

[15] Claude Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 30:50 – 64, 1951.