Substitution Ciphers

Brian Powell
(Dated: April 2013)

Monographic ciphers

A monographic substitution cipher works by replacing individual characters of plaintext with corresponding characters of ciphertext. It is perhaps the simplest encryption scheme ever devised: early monographic substitution ciphers were employed by Julius Caesar to secure private correspondence. These ciphers were low-tech, required virtually no mathematics, and encryption and decryption could be accomplished by counting on ones fingers. Perhaps surprisingly, then, the most secure modern cipher also happens to be a monographic substitution cipher, demonstrating the important cryptographic lesson that simplicity does not imply triviality. In this article, well study the monographic substitution cipher concept by examining several of its varied implementations, from the ancient to the state-of-the-art.

Shift cipher

Consider a simple cipher in which a single substitution rule is applied: each character of plaintext, p, is replaced by the ciphertext character, c, located m steps further ahead in the alphabet than p. For a plaintext letter located towards the end of the alphabet, if the number of steps m exceeds the number of remaining letters in the alphabet, the rule is to wrap around to the beginning. Mathematically, this operation is known as modular arithmetic, and we write $c = (p + m) \bmod N$, where here the values of c and p are not the characters themselves but numerical representations (for example, in the English alphabet we'd denote the letters `A-Z` by the numbers 0-25.), and where N is the number of letters in the alphabet. Expressions of the form $a = b \bmod n$ are called *congruences* and read "a is congruent to b modulo n." The mod operation gives the remainder, a, when b is divided by n. You can think of modular arithmetic as analogous to time on a clock, but with the n letters in place of the hours: once you count past the n^{th}, you start over at the 1^{st}. Thinking in this way, the cipher can be represented by two

FIG. 1: Representation of the shift cipher as two concentric rings: the outer is the plaintext and the inner, the corresponding ciphertext. For an alphabet with N characters, there are $N - 1$ implementations of this cipher.

concentric circles of characters: the outer one representing the plaintext and the inner the corresponding ciphertext. For this reason, the shift imparted by the cipher is sometimes called a *rotation*. Hence the name *shift* (or rotation) cipher. It's also called a Caesar cipher, after Julius Caesar who used the cipher with a shift of $m = 3$ to conceal military secrets. Consider the following plaintext:

```
We hold these truths to be self-evident, that all men are created equal, that
they are endowed by their Creator with certain unalienable Rights, that among
these are Life, Liberty, and the Pursuit of Happiness.
```

Application of the shift cipher with $m = 5$ gives

```
bjmtq iymjx jywzy mxytg jxjqk janij syymf yfqqr jfsfw jhwjf yjijv zfqym fyymj
dfwjj sitbj igdym jnwhw jfytw bnymh jwyfn szsfq njsfg qjwnl myxym fyfrt slymj
xjfwj qnkjq ngjwy dfsiy mjuzw xznyt kmfuu nsjxx.
```

where the punctuation has been dropped and only the letters have been encrypted (but we'll soon enlarge the alphabet to include punctuation.) You can see that I've grouped the ciphertext into "words" of five letters each; this is arbitrary but it's done to homogenize the sentence structure. The English language is characterized by certain regularities – its syntax and grammar – that give it an identifiable statistical structure. For example, the letter `S` occurs more frequently than the letter `Z`; the combination, `TH`, occurs more often than `RB`. The letter `E` occurs at the end of words more than any other letter, while `T` holds this distinction for the beginning words. The word `THE` is the most common three-letter word, whereas the word `RUG` occurs much less frequently. The grouping into five-letter "words" effectively erases any regularities based on words – we don't know where words begin or end, or how long they are, but it does nothing to hide the relative frequencies of the letters. And this is bad.

If we count the prevalence of letters in our ciphertext, we find that `J` occurs most frequently (22 times), followed by the letter `Y` (18 times). According to a frequency analysis of letters in English, the most common letters are `E, T, A, O, I,` and `N`, whereas the least common are `J, X, Q,` and `Z`. As an initial guess, then, a cryptanalyst intent on breaking this cipher might suppose that `J` codes for `E`, and `Y` for `T` (and he'd be right – even though our selection is only one sentence (and a very famous one at that), its two most common letters agree with the statistics of English at large.) Continuing in this manner, depending on the length of the ciphertext, he will succeed in cracking at least a few letters. He doesn't need to break the whole ciphertext this way – once he's convinced himself that a he's gotten even one letter correct, he can recover the shift amount, m. This number is the *key* of the shift cipher, and must be kept secret – once the cryptanalyst has it, he can recover the entire plaintext.

So we've seen how the inherent regularities and patterns of the English language can be leveraged by a cryptanalyst to break the code, and a good cryptosystem must do its best to obscure them. As advocated by Claude Shannon as early as 1949, the security of a cryptosystem is based on the properties of *confusion* and *diffusion* [1]: confusion means that the ciphertext depends in a very complex way on the plaintext and key; diffusion means that the statistics of the plaintext is spread out and diluted in the statistics of the ciphertext[1]. The shift cipher does nothing to diffuse the statistics of the English plaintext, and so it's easy to break.

It's worth noting that the shift cipher works with any alphabet: English letters, binary, whatever. The binary alphabet $\{0, 1\}$ is interesting: if m is divisible by 2, then the cipher acts like the identity on the plaintext and nothing gets encrypted. If m is chosen to be odd, something different happens. Suppose we have an English language plaintext, and we wish to encrypt it using the shift cipher based on

---

[1] A good encryption scheme generates a ciphertext that is highly randomized: if only a single bit of plaintext is changed, the ciphertext should change in a significant and unpredictable way (by, for example, flipping half of the output bits.)

the binary alphabet. We first translate the English text to binary according to the ASCII[2] conversion. Application of the shift cipher, with m odd, flips each bit – it effectively XOR's the plaintext binary with a string of 1's – a simple modification[3]. But when we then convert back to ASCII, we find that our plaintext has been scrambled in a seemingly random way:

```
NOT ON THE RUG, MAN.
```

translate to binary:

```
1001110 1001111 1010100 1001111 1001110 1010100 1001000
1000101 1010010 1010101 1000111 1001101 1000001 1001110
```

rotate by 1 bit:

```
0110001 0110000 0101011 0110000 0110001 0101011 0110111
0111010 0101101 0101010 0111000 0110010 0111110 0110001
```

translate back to ASCII:

```
10+ 01  +7: -*8 2>1
```

where, for clarity, I've ignored the punctuation and not encrypted the spaces. In contrast to the case above, each unique letter is shifted by a different amount. This makes things more complicated, but the primary weakness of the standard shift cipher remains: the statistics of the plaintext are preserved. I do have to work a little harder, though, because the key can't be recovered by decrypting only a few characters. Since each unique letter is shifted by a different amount, there are as many keys as there are letters in the alphabet – in this case, $2^7 = 128$ for the 7-bit ASCII alphabet that we're using. But it's the same game – use character frequencies to recover the plaintext letters, and once you've gotten the first few, the rest can probably be gotten from context.

Affine cipher

Instead of adding the key to the plaintext letters, there is a family of ciphers with a substitution rule based on multiplication by the key. These are known as *affine ciphers* and invoke the transformation

$$c = (dp + m) \bmod N, \tag{1}$$

where d and m are integers. The shift cipher is an affine cipher with $d = 1$. To focus on the effect of the multiplication, let's take $m = 0$ and see what happens for $d = 4$. If we label the letters of the English alphabet by the numbers 0-25, we get the following mapping shown in Table 1.

---

[2] ASCII is the American Standard Code for Information Interchange. It is a character-encoding scheme that converts 7-bit binary numbers into the many different printable characters used in written communication.

[3] XOR is short for *exclusive OR*, a logical operation that returns 0 unless the operands are different and 1 otherwise, *i.e.* $1 + 1 = 0$, $0 + 0 = 0$, $1 + 0 = 0 + 1 = 1$.

															These don't get encoded											
plaintext	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ciphertext	0	4	8	12	16	20	24	2	6	10	14	18	22	0	4	8	12	16	20	24	2	6	10	14	18	22
															and these are duplicates											

TABLE I: Affine cipher with $d = 4$ and $m = 0$. The ciphertext repeats after only 13 characters, and so not all plaintext characters are uniquely encoded. Notice how the ciphertext alphabet takes hops through the plaintext alphabet $d = 4$ times, repeating halfway through.

Half-way through the plaintext alphabet the ciphertext repeats! As a result, half of the plaintext letters (those beyond M, letter 12) have no unique ciphertext equivalent and so effectively cannot be coded. We evidently have an incomplete and, because of the duplicates, a non-invertible cipher. How did this happen? Let's look a bit more closely at the congruence Eq. (1) for the case of $m = 0$. The congruence $c = dp \bmod N$ can be written

$$c = dp - kN, \tag{2}$$

where k counts the number of times we wind around the mod$N$ "clock" (it takes on non-negative integer values.) So, for example, when $dp > N$, we take $k = 1$; when $dp > 2N$, we take $k = 2$, and so on. This expression is the equivalent of taking the remainder of dp upon division by N. If d and N have a greatest common divisor, $x > 1$ (written $\gcd(d, N) > 1$), then we can write $d = fx$ and $N = gx$ for integers $f < d$ and $g < N$. Then the congruence Eq. (2) becomes

$$c = x(fp - kg) = x(fp \bmod g).$$

This reveals that regardless of the value of p, the corresponding ciphertext, c, will always be a multiple of x. Since our ciphertext alphabet is equivalent to the set of numbers $[0, 25]$ – not all of which are multiples of the same integer x – it is evident that the affine cipher with $d = 4$ and $m = 0$ cannot generate a complete ciphertext alphabet. In the specific case of $d = 4$ and $m = 0$, $x = 2$ with the result that all the ciphertext characters are even. Since there are more plaintext than ciphertext characters, the cipher is also not unique[1] and so cannot be inverted. Only when $\gcd(d, N) = 1$ – when d and N are *relatively prime* – is the affine cipher complete and invertible[4].

This means that the use of the affine cipher is constrained by the existence of pairs (d, N) that are relatively prime. There are only 12 numbers for which $\gcd(26, d) = 1$, and with 26 possibilities for m, there are $26 \times 12 = 312$ different affine cipher prescriptions. Even if the affine cipher were difficult to break via frequency analysis, with only 312 versions it is almost more economical to recover the plaintext through brute force. But it turns out that the general affine cipher is just as susceptible to frequency analysis as the shift cipher. This family of ciphers preserves frequency statistics because each unique plaintext character is replaced by a unique ciphertext character – it is said that the ciphertext is drawn from a *single alphabet*. Even when the ciphertext alphabet appears totally random (as in the

[4] What we're actually discovering here about the congruence Eq. (2) when $\gcd(d, N) > 1$ is that it is not *onto* or *one-to-one*. A function, f, is onto if, for every value in the range, y_i, there exists an x_i in the domain such that $y_i = f(x_i)$. A function is one-to-one if $f(x_i) = f(x_j)$ only if $x_i = x_j$. Eq. (2) with $d = 4$, $m = 0$, and $N = 26$ is not onto because there are ciphertext characters that do not encode any plaintext characters (all the odd c_i); it is not one-to-one because each c_i encodes two plaintext characters, $p_i \neq p_j$. When the domain and range of a function are equivalent discrete sets (in this case $c = p = [0, 25]$), f is onto if and only if it is one-to-one.

example of the shift cipher mod 2) – even if the alphabet *is* totally random – there is still a one-to-one mapping between the p's and the c's and a careful analysis that leverages the grammar and syntax of the plaintext language will break the cipher. These techniques are wide-ranging and might include character frequencies, the expected locations of vowels in short ciphertext words, common word-beginning letters (like `T` and `A`) and word-ending letters (like `E` and `S`), words with tell-tale patterns (like `COMMITTEE`), even perhaps a frequency analysis of two- or three-letter groupings (so-called *digrams* and *trigrams*: `TH` and `AND` are quite common)[3]. Ciphers that employ a single ciphertext alphabet are known as *monoalphabetic*, so to get our bearings, the affine and shift ciphers are examples of monoalphabetic, monographic ciphers. They are instructive, but as some of the earliest known encryption schemes, they are simple and weak. As I said at the beginning – studying broken ciphers yields insights into strong cipher design. What's the lesson from studying affine ciphers? We need more alphabets.

From Vigenére to Vernam: the quest for more alphabets

Adding more alphabets is easy. Consider the shift cipher: instead of using a single key, m, to encrypt the entire plaintext, why not employ several? This is the idea behind the Vigenére cipher[5] – there is a different key, m_i, depending on position in the plaintext. For example, suppose we have three keys: $m_1 = 2$, $m_2 = 7$, and $m_3 = 4$, where the subscript denotes the position, mod i, of the plaintext character to be encrypted, *i.e.* we shift the first letter by 2, the next by 7, the next by 4, and the next by 2, and so on. All of the m_i can be enumerated in a table called the *tabula recta* (imaginatively, *ruled*

	p	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
	2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
	3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
m	4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
	5	F	G	H	K	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
	6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H

FIG. 2: First nine lines of the Vigenére *tabula recta*: plaintext alphabet, p, across the top with corresponding ciphertext alphabets, m, indexed by shift number. The table was useful for encryption and decryption back in the olden days; now it's mostly helpful for visualizing the different alphabets.

table). To create a Vigenére key we simply select a sequence of the m_i, say, $k = m_1m_2m_3 = 274$ as above. Or, better yet, we select a secret keyword, like `MARMOT`. Encryption proceeds as follows:

```
THE DUDE IS NOT INX
        +
MAR MOTM AR MOT MAR
-------------------
GIWQJ XRJKA DNVNP
```

We apply the key, `MARMOT`, in six-letter chunks across the whole plaintext[6]. Also, notice that we've *padded* the plaintext with `X`'s (here only one) so that the ciphertext can be made into groups of five. The salient

[5] The cipher is named for Blaise de Vigenére, a 16th century French diplomat who had nothing to do with it.

[6] Because the Vigenére cipher operates on characters in key-length blocks, it is sometimes described as a *block* cipher, or as a *polygraphic* cipher. Neither of these designations is correct. The Vigenére cipher still operates on single characters, with a position-dependent single character subkey (the m_i). Polygraphic or block substitution ciphers use multiple-character strings as the basis of their of substitutions, so for example, in the case of a six-character cipher, the word `RANSOM` would always map to the same ciphertext string. This is generally not true of the Vigenére cipher.

feature of this cipher is that like-plaintext characters do not generally encrypt to like-ciphertext. In our example, the first and third `D`'s in `DUDE` encrypt to `M` and `T`, respectively. Furthermore, the second `E` also encrypts to `M`. The cipher is not one-to-one – it's both many-to-one and one-to-many. In general, for a key with n unique characters, each individual plaintext character could represent any of n different ciphertext characters, depending on its position – the cipher is said to use n alphabets. What this means is that the frequencies of the plaintext letters – like `D` above – are distributed among several distinct ciphertext letters. By employing multiple alphabets, the Vigenére cipher achieves what the monoalphabetic ciphers failed to: it nicely hides the patterns of language and character frequencies of the plaintext by spreading them out among the ciphertext[7]. It offers up a fair helping of confusion to our fellow cryptanalyst. What can he do about it?

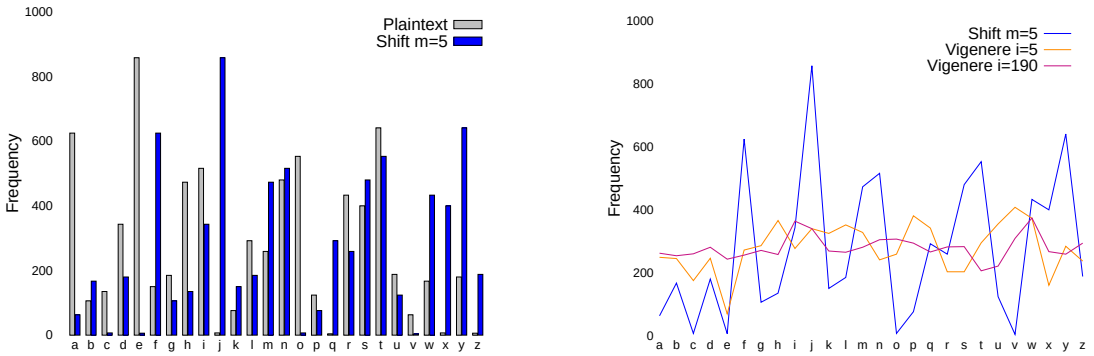

FIG. 3: (a) Frequency distributions of letters from the 1st chapter of the novel *Great Expectations* by Charles Dickens: plaintext (gray) and ciphertext (blue) generated from a simple shift cipher with $m = 5$. The distributions have identical statistics. (b) Frequency distributions of a few different ciphertexts: the same shift cipher from (a) (using a line graph instead of bars for clarity), a Vigenére cipher with a 5-character key (orange), and a 190-character key (purple).

For one, he can study the frequency distribution of the ciphertext to determine whether the cipher is a simple shift or Vigenére, and if it's the latter, how many alphabets were used. This involves bringing some statistical muscle to bear on a quantity called the *measure of roughness* that characterizes the choppiness of the frequency distribution [3],

$$\rho = \sum_{i}^{N} \left(P_i - \frac{1}{N} \right)^2 . \tag{3}$$

This measure gives the average squared deviation[8] of the observed character frequencies from a uniform distribution. P_i is the probability that the i^{th} character of the N-character alphabet appears in the message: if all characters are equiprobable, ρ is exactly zero. Notice that ρ has the form of the chi-square statistic, $\rho = nN\chi^2$, where n is the total number of characters in the message. This gives it some

[7] By "spread out" I mean that a frequency analysis of ciphertext characters reveals few features – that most characters appear with similar frequency (in the limit that the number of alphabets approaches 26, on average all ciphertext characters appear with equal frequency). I do not mean to imply that the Vigenére cipher *mixes* the statistics of the plaintext throughout the ciphertext, in the spirit of Shannon's diffusion criterion. Small, localized changes to plaintext still result in small, localized changes to the ciphertext.

[8] An average *squared* deviation is necessary because the average deviation is zero.

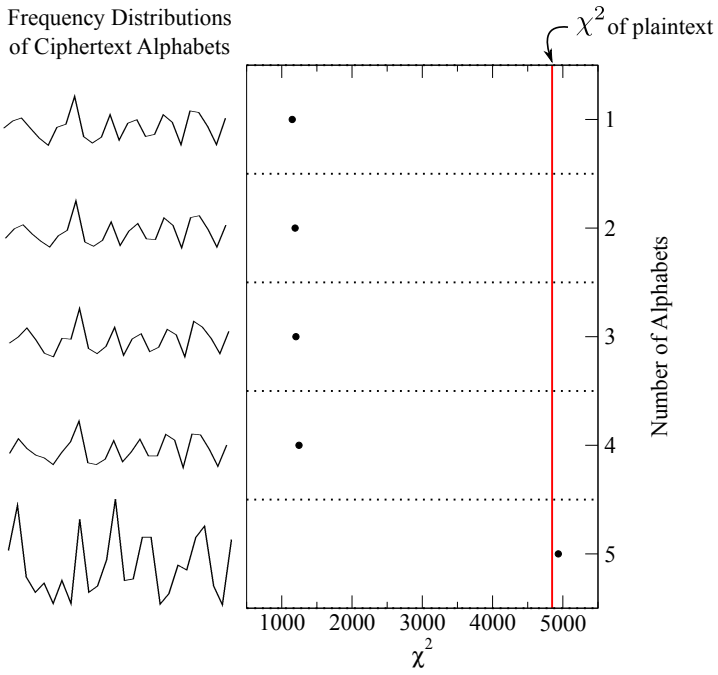

FIG. 4: χ^2 values of ciphertext character frequency distributions when the ciphertext is divided into 1, 2, 3, 4, and 5 monoalphabets. For multiple monoalphabets, there are multiple distributions; only one is shown. For multiple monoalphabets the average χ^2 is computed.

statistical street cred: the p-value[9] associated with a given value of ρ can be computed to determine how significantly, in a statistical sense, the ciphertext deviates from a uniform distribution. Because monoalphabetic ciphers retain the frequency characteristics of the plaintext, they have more roughness on average than polyalphabetic ciphers; the measure of roughness continues to decrease as alphabets are added (the distribution approaches uniformity on average when the number of alphabets is equal to the number of characters, N, in the alphabet.) In Figure 3 a frequency analysis of letters from the first chapter of the novel *Great Expectations* is presented, along with the frequency distributions of a couple different ciphertexts. As we've discussed, the shift cipher has no effect on the shape of the distribution, while the Vigenére cipher smooths out the choppiness.

If the key has a length of i, then every i^{th} plaintext character is shifted by the same key character, *e.g.* characters $\{p_1, p_{1+i}, p_{1+2i}, \cdots\} = \{p_{1 \bmod i}\}$ are shifted by the same amount, characters $\{p_2, p_{2+i}, p_{2+2i}, \cdots\} = \{p_{2 \bmod i}\}$ are shifted by the same amount, and so on up to $\{p_{(i-1) \bmod i}\}$. The full plaintext is effectively broken up into i smaller plaintext pieces, $\{p_{j \bmod i}\}$ for $0 \leq j < i$, each separately encrypted by a different shift cipher. The corresponding ciphertexts, $\{c_{j \bmod i}\}$, since they result from the $\{p_{j \bmod i}\}$ by simple shifts, share frequency statistics with the $\{p_{j \bmod i}\}$.

We can use this fact to discover the length of the key. Suppose the cipher is Vigenére with a 5-character key. First, we examine the frequency distribution of characters in the full ciphertext; if the cipher is anything other than a monoalphabetic substitution, this won't yield anything promising. Next, we test for a two-character Vigenére key by computing the measures of roughness of two groups of ciphertext, $\{c_{0 \bmod 2}\}$ and $\{c_{1 \bmod 2}\}$, then we do this for three groups, $\{c_{0 \bmod 3}\}$, $\{c_{1 \bmod 3}\}$, and

[9] The p-value is the integral over the chi-square distribution function $\int_{\chi^2}^{\infty} P_\nu(y)\, dy$ of $\nu = N - 1$ degrees of freedom. The p-value gives the probability that the observed deviations from uniformity arise from chance alone.

$\{c_{2 \bmod 3}\}$, and so on. This process is illustrated in Figure 4. Notice that the frequency of ciphertext characters is roughly uniform until we try five alphabets – the length of th key – at which point the χ^2 of the frequency distribution increases dramatically to match that of the plaintext.[10] Once the length of the key has been determined in this way, the problem has been effectively broken up into a more manageable collection of simple shift ciphers.

As the length of the key increases, this method of analysis gets progressively less reliable. For a key of length i and a plaintext message of length n, there are n/i characters in each of the i plaintext and ciphertext pieces, $\{p_{j \bmod i}\}$ and $\{c_{j \bmod i}\}$. If the number of characters in each ciphertext piece is too small, the frequency distributions of the $\{c_{j \bmod i}\}$ will not resemble a representative sample of the plaintext language and our ability to detect deviations from uniformity will be challenged. Indeed, as $i \to n$, we lose *all* ability as the plaintext pieces $\{p_{j \bmod i}\}$ reduce to individual characters. This result argues for a key with the same length as the message with as little repetition as possible (so that different instances of the same plaintext character encrypt to any ciphertext character with equal preference.) Repetition is minimized in practice by drawing each character in the key randomly from the alphabet, a system known as the *Vernam cipher*. When the key is random, so too is the ciphertext. The result is the most formidable of ciphertexts—a cryptanalytic nightmare that is unbreakable even in principle if implemented properly.

In 1949, Claude Shannon proved [1] that if the key is random, as long as the message, and never reused, then the Vernam cipher is theoretically secure, *i.e.* it cannot be broken even in principle. This system is also known as the *one-time pad*, named for the practice of writing the random key on notepads secretly shared between sender and receiver[11]. As keys were used, pages were torn out of the pad to ensure one-time use. Because the cipher employs as many alphabets as characters of plaintext, the key can be *anything*—a given ciphertext retains none of the characteristics of the message and might decrypt to any possible plaintext. Intercepting the ciphertext is therefore of no help to the cryptanalyst, since all plaintexts are equally likely: this condition is called *perfect secrecy*. One-time-pads were popular among Soviet spies during World War II, though they weren't always diligent with the "one-time" part of the concept, enabling U.S. and British intelligence services to break portions of Soviet encrypted communications[12]. Key reuse was undoubtedly due to the logistical difficulty of deploying and securing long, random keys. But what if we're willing to skimp a bit on the whole randomness thing?

One way to employ a key as long as the message is to somehow use the message itself as the key. The *autokey* is a substitution cipher that that uses the message plaintext, shifted by a certain amount relative to the message, as the key. A shared keyword prepended to the message imparts the shift and enables decryption. For example,

```
Weholdthesetruthstobeselfevidentthatallmenarecreatedequal...
+
JEFFERSONWeholdthesetruthstobeselfevidentthatallmenarecre...
```

---

[10] These are *huge* values of χ^2 per the 25 degrees of freedom, meaning that even the "uniform" ciphertexts are strongly statistically distinct from true uniform distributions. This is expected for such a short key, in which case we cannot use p-values to "detect" the key length, but must rely on the relative change in χ^2 (or ρ). See the end note[2] for more discussion.

[11] For long messages, one-time pads are difficult to implement because key exchange is impractical; instead, keys must be pre-deployed among a fixed set of parties and kept secure for long periods of time.

[12] This decades-long cryptanalysis effort was part of the famous U.S/U.K. VENONA project targeting the encrypted communications of Soviet intelligence agencies. Over 3000 messages were at least partially decrypted revealing important information, including Soviet espionage targeting the Manhattan Project.

`gjnuqvmwspjbggxbayhgykzfnxpxfjgyfnfpjpqayhisyddqnysewvxsq...`

The autokey is Vigenére cipher with a message-length key in which plaintext character p_i is shifted by p_{i+k}, for a k-letter keyword. Because the shifts are imparted by the plaintext itself, they are not random; rather, for long messages, their statistics conform to those of the English language (*i.e.* shifts by 5, corresponding to the letter `e`, will occur most often on average, followed by shifts of 1, corresponding to the letter `a`, and so on). But is it obvious that shifting non-random plaintext characters by amounts drawn from a non-random key won't result in a random ciphertext? Is shifting by, say, five characters more likely to result in a uniform ciphertext than shifting by ten? In general, how do the statistics of the ciphertext depend on the amount of shift between plaintext and key?

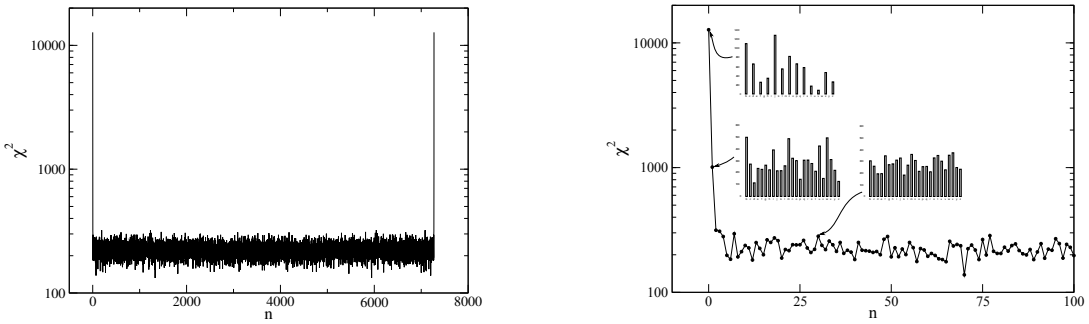

FIG. 5: (a) χ^2 values of ciphertexts generated by shifting the plaintext by amounts $n = 0, 1, 2, \ldots$. (b) Zoom in on the region $n \in [0, 100]$ with ciphertext character frequencies shown for $n = 0, 1$ and 30.

Since the keyword is typically short, it does not contribute significantly to the ciphertext statistics and so to isolate the effect of the shift amount, we'll do away with the keyword and just wrap the plaintext key back on itself[13]. First, consider a shift of zero: ignoring the fact that decryption would be impossible, we do very little to obscure the plaintext statistics. Because each character is added to itself, each ciphertext character, c_j, inherits the frequency statistics of the two plaintext characters p_i and p_l for which: $j = 2i = 2l \mod 26$. For example, `a` and `s` both encrypt to `b`. The result is that high-frequency plaintext characters transform to high-frequency ciphertext, and the lack of any ciphertext characters c_i for odd i is notable and further inhibits mixing. Alas, as Figure 5(a) shows, the χ^2 value of the ciphertext for a shift of $n = 0$ is near astronomical, $\chi^2 \approx 10,000$.

A shift of $n = 1$ is a sharp improvement, but the statistics are still far from uniform. This is interesting, since now we have much more mixing: each ciphertext character, c_j, can arise from any two plaintext characters p_i and p_l with $j = i + l \mod 26$. So what's going on here? Since each character of ciphertext results from combining consecutive characters of plaintext, we only expect the ciphertext to be uniform if each letter pair, $p_i p_l$, is equiprobable. But in English, we know this isn't the case: the letters `th` appear way more often than the letters `cz`; the combination `er` is more frequent than the pair `dl`. The ciphertext character distribution resulting from a shift of $n = 1$ therefore embodies the digraph statstics of English at large.

Interestingly, once we try $n = 2$ the χ^2 value more or less settles down. Evidently, in English, no pair of characters is more likely to be found separated by a single letter than as found separated

[13] For example, if the message is "`Get your shoes on Lebowski`" we execute a shift of 3 by making the key "`ski Get your shoes on Lebow`" rather than prepending a 3-letter keyword.

by two, three, or any number of letters. Notice, though, that the distribution settles down to around $\chi^2 \approx 250$ for $n \in [2, L-2]$ (where L is the length of the message), quite far from a uniform distribution (see Figure 6: strings generated by a typical pseudo-random number generator have $\chi^2 \approx 30$; not truly random, but much more uniform than the autokey.) This means that there must be some preference for some character pairs over others; I suspect though I've not verified that any preference is due to single character frequencies, *i.e.* that `e`'s are more likely to pair with other `e`'s solely because it is the most common character. In comparison with Figure 6, the autokey for $n \in [2, L-2]$ leads to a ciphertext with frequency distribution similar to a Vigenére cipher with a 30-character pseudo-random key.

---

[1] Shannon, C. E., Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28 (1949).

[2] Rosen, K. H., *Elementary Number Theory and its Applications*. Addison Wesley (2000).

[3] Sinkov, A., *Elementary Cryptanalysis: A Mathematical Approach*. Mathematical Association of America Textbooks, (1966).

[4] Gaines, H. F., *Cryptanalysis: A Study of Ciphers and Their Solution*. Dover Publications, (1989).

Notes

[1]For a general congruence, $a = b \bmod n$, a begins to repeat only when $b \geq n$. Here, though, multiplication by $d = 4$ has resulted in (at least) two distinct plaintext characters: p_1 and p_2 both < 26, that map to the same ciphertext character, *i.e.*,

$$\begin{aligned} c &= 4p_1 \bmod 26 = 4p_2 \bmod 26 \\ &= 4p_1 - 26k_1 = 4p_2 - 26k_2. \end{aligned} \tag{4}$$

If we solve for p_1 in terms of p_2, we get

$$\begin{aligned} p_1 &= p_2 + \frac{26}{4}(k_1 - k_2) \\ &= p_2 \bmod \frac{26}{4}(k_2 - k_1), \end{aligned} \tag{5}$$

as the condition that both p_1 and p_2 give the same c. If $p_1 = p_2 \bmod 26$, then p_1 and p_2 only give the same c if $p_1 = p_2$. If this were the case, then we'd get a well-behaved cipher and there would be no repeats. Now, from Eq. (5), we see that there is a dependence on $k_2 - k_1$. If we take $k_1 = 0$, then from Eq. (4) and the discussion following Eq. (2), we have $4p_1 < 26$, so that p_1 is from the first quarter of the alphabet. If we then take $k_2 = 1$, we've got p_2 coming from the second quarter of the alphabet. So the relation Eq. (5) with $k_2 - k_1 = 1 - 0 = 1$ tells us whether a p from the first quarter of the alphabet gives the same c as a p from the 2nd quarter. In this case Eq (5) becomes $p_1 = p_2 + 26/4$. Because both p_1 and p_2 must be integers, this equation has no solutions and so no two such p's exist. What about finding a match between a p in the first and a p in the 3rd quarter? For $k_2 - k_1 = 2$ we get $p_1 = p_2 \bmod 13$! This means that after making it only half-way through the plaintext alphabet, we stumble upon a character, p_2, that gives the same c as p_1. This is just what we see in Table 1. If the ideas of congruence and divisibility are new to you, see [2] for a nice introduction.

[2]As the size of a random key increases, we expect the ciphertext to become more uniform. This is reflected in a decreasing chi-squared per degrees of freedom. For the first chapter of *Great Expectations* examined in this article, the Vigenére cipher becomes more and more uniform with increasing key size until the key reaches a length of around 1000 characters, after which the degree of uniformity remains approximately constant (see Figure 7.) The ciphertext is still not statistically uniform: the average χ^2 of ciphertexts with key lengths ≥ 1000 is around 26, whereas $\chi^2 \leq 14$ for a 0.05 significance level (gray line in Figure 7). Is this lack of significant uniformity a feature of the type of cipher or the key? The average χ^2 of a sample of 1000-character-keys (blue line in Figure 7) agrees well with the average χ^2 of the ciphertexts encrypted with these keys. The Vigenére cipher therefore appears to achieve the same degree of uniformity as the key when the key length reaches 1000 characters, or around a length of $40N$ for an N-character alphabet. Presumably a more uniform key would increase the uniformity of the Vigenére ciphertext.

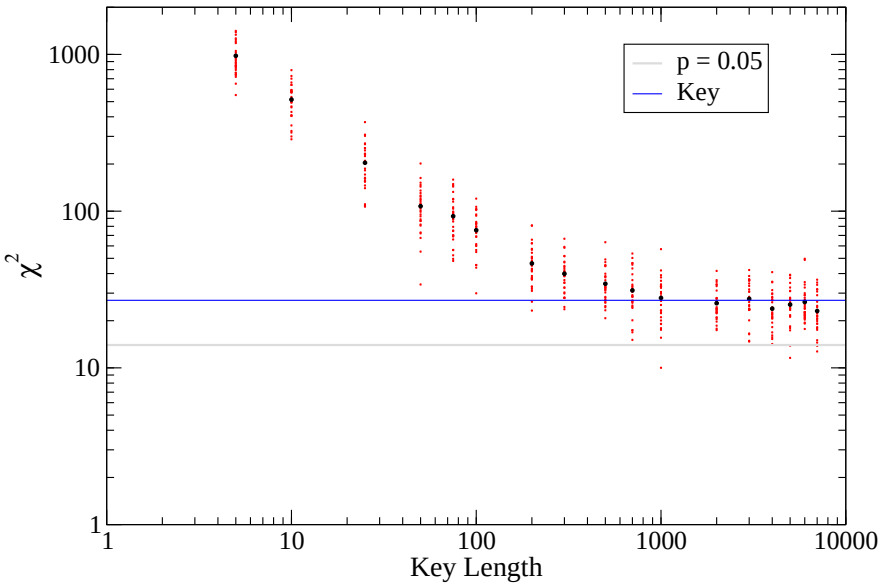

FIG. 6: χ^2 values of Vigenére ciphertexts encrypted with keys of increasing length. For each key length, the first chapter of *Great Expectations* was encrypted with 30 randomly generated keys (red points); the average χ^2 for each key length is also shown (black points). The gray line marks the χ^2 marking the 0.05 significance level. The blue line is the average χ^2 calculated from the sample of random 1000-character keys.