10 secret codes

Secret codes as a way to communicate privately has been an important activity for centuries. Modern commerce relies on the integrity of large data bases, and a common method of ensuring integrity is encryption (encoding the data so that it is meaningful only to someone familiar with the coding technique). Accountants have as much interest in maintaining data integrity as anyone. Further, given the skills acquired in the analysis of accounting problems so far, we are well positioned to study cryptography, the science of secret codes.

A fundamental result for encryption was worked out over 300 years ago by Pierre de Fermat. Another central result is even older, over 1,000 years from Euclid.[1] But the application to encryption had to await the development of powerful, high-speed computers. Cryptography is an intriguing mixture of old and new science.

10.1 Fermat's theorem

Pierre de Fermat was an amateur mathematician who explored the mysteries of numbers for the pure joy of discovery. It is ironic that some of his results should turn out to be of such great practical application. The practical use of Fermat's theorem is due to the fact that the relationship is true for all numbers, no matter how big they are. Computer encryption uses extremely large numbers.

Theorem 10.1 *Fermat's theorem*

[1]A discussion of Fermat's theorem and Euclid's algorithm is in Ore, 1948.

$$a^{p-1} \equiv 1 \pmod{p}$$

where p is any prime number, and a is any number not a multiple of p.

Example 10.1 *For an easy example, 3 is a prime number. Then checking Fermat*

$$2^{3-1} = 2^2 = 4 \equiv 1 \pmod{3}$$

Example 10.2 *17 is a prime number. Calculate* 12^{16}*. By Fermat when* 12^{16} *is divided by 17, the remainder is* 1.

12^{16} is already a fairly large number, but fortunately we can verify the example without actually calculating 12^{16}. We can go in small stages. $12^2 = 144$, and when 17 is divided into 144, the answer is 8 with a remainder of 8. The remainder is what we want.

$$12^2 \equiv 8 \pmod{17}$$

The next step is to calculate 12^4.

$$12^4 = \left(12^2\right)^2 \equiv 8^2 \pmod{17} = 64 \equiv 13 \pmod{17}$$

Similarly,

$$12^8 = \left(12^4\right)^2 \equiv 13^2 \pmod{17} \equiv (-4)^2 = 16 \equiv -1 \pmod{17}$$

Notice when it came to squaring 13, it was easier to use $13 \equiv -4 \pmod{17}$ and square -4 instead. One more step gets us to 12^{16}.

$$12^{16} = \left(12^8\right)^2 \equiv (-1)^2 = 1 \pmod{17}$$

The last line verifies Fermat for the example.

As Fermat is true for all prime numbers, there are an infinite number of examples, and it is a good idea to complete several of them. For now, though, it would be nice to demonstrate Fermat with a little more generality. That can be done, since the theorem follows directly from the two properties of multiplication using a prime modulus: there are no zeros and no repeats. For example, consider modulo 5. Multiply all the numbers less than 5 by 2, and then multiply the answers together.

$$\begin{aligned} & (2 \times 1)(2 \times 2)(2 \times 3)(2 \times 4) \\ \equiv\ & 2 \times 4 \times 1 \times 3 \pmod{5} \end{aligned}$$

We know because of no zeros and no repeats, the numbers $(1, 2, 3, 4)$ will return after multiplying by 2, or indeed any non-zero number in the multiplication table. Only the order will change.

We can rewrite the result using the factorial (!) operator.

$$2^4\,(4!) \equiv (4!)\,(\bmod\,5)$$

We can effectively divide out the factorial term.[2]

$$2^4 \equiv 1\,(\bmod\,5)$$

The above algebra verified Fermat for $p = 5$ and $a = 2$. The logic for any p and a is similar. Each step is allowed because of the two properties. The product never disappears because there are no zeros, and the two expressions are congruent because they are composed of numbers in the same set without repeats (although probably in a different order).

10.2 an encryption technique

Fermat's theorem suggests a secret coding technique. Express the message in numbers. Encode the message by raising to a high power. That, then, is the cyphertext sent to the receiver. Decoding is accomplished by raising the cyphertext to another power until we reach 1, as Fermat guarantees we will. Then multiply one more time and the message will return.

Denote the message by a, and let p be the prime modulus.

$$\begin{aligned} a^{p-1} &\equiv 1\,(\bmod\,p) \qquad \text{(Fermat's theorem)} \\ a^{(p-1)n} &\equiv 1\,(\bmod\,p) \\ a^{(p-1)n+1} &\equiv a\,(\bmod\,p) \end{aligned}$$

Encryption involves accomplishing the above in two steps. Let e be an encryption parameter, so the cyphertext is a^e. What is required is a decryption parameter, d, such that

$$a^{e^d} = a^{ed} = a^{(p-1)n+1} \equiv a\,(\bmod\,p)$$

The designer of the code has the freedom to pick any p and e. But d must satisfy

$$ed = (p-1)\,n + 1$$

Or, in congruence notation, the problem is, given p and e, solve the following congruence for d.

$$ed \equiv 1\,(\bmod\,p-1)$$

[2]Actually, we can't strictly divide. However, we know (because of no zeros and no repeats) that there is some number, when multiplied by 4!, the product is congruent to 1 modulo 5.

The problem is actually not a difficult one to solve; a solution is achieved using Euclid's algorithm, one of the oldest (perhaps the oldest) algorithm in mathematics. Euclid's algorithm is the subject of the next section, so we will temporarily skip that part. For now, here's a small numerical example.

Example 10.3 *Let $p = 67$, $e = 35$, and $d = 17$. It is easy to verify that e and d will work as the encryption and decryption parameters.*

$$35 \times 17 = 595 = 9 \times 66 + 1 \equiv 1 \pmod{66}$$

With this small code any number less than 67 can be encrypted. The following tabulates the cyphertext form of messages from 2 to 29.

a	a^e	a	a^e
2	63	16	55
3	58	17	21
4	16	18	11
5	42	19	26
6	36	20	2
7	18	21	39
8	3	22	15
9	14	23	60
10	33	24	40
11	13	25	22
12	57	26	6
13	32	27	8
14	62	28	20
15	24	29	37

It is not too hard to verify individual entries, and they should be verified. For example, the calculations for $a = 2$ and $e = 35$ follow. (Notice the use of negative numbers when they are easier to raise to a power.)

$$\begin{aligned} 2^8 &= 256 \equiv 55 \equiv -12 \pmod{67} \\ 2^{16} &\equiv (-12)^2 = 144 \equiv 10 \pmod{67} \\ 2^{32} &\equiv (10)^2 = 100 \equiv 33 \pmod{67} \\ 2^{35} &= \left(2^{32}\right)\left(2^3\right) \equiv 33 \times 8 = 264 \equiv 63 \pmod{67} \end{aligned}$$

Decoding is accomplished with $d = 17$.

$$\begin{aligned}
63 &\equiv -4 \pmod{67} \\
63^4 &\equiv (-4)^4 = 256 \equiv -12 \pmod{67} \\
63^8 &\equiv 144 \equiv 10 \pmod{67} \\
63^{16} &\equiv 100 \equiv 33 \pmod{67} \\
63^{17} &\equiv 33 \times (-4) = -132 \equiv 2 \pmod{67}
\end{aligned}$$

And the original message, $a = 2$, is returned.

The cyphertext, a^e, bears no systematic relationship to the message a. Because of this, it is difficult (probably impossible) to "break" the code, that is, invert from a^e to a. Another way to state the relationship is that there is no information in a^e about a. Indeed, many random number generators work by raising a large number to a large power and then reporting the remainder. In this sense a^e will look to the bad guys like random numbers, and will prove no use to them.

It will be easier to visualize the randomness in a^e when we get to examples with large numbers. For now, we have some unfinished business: solving for the decryptor, d.

10.3 Euclid's algorithm

The problem in the last section was to find a decryptor for the prime modulus, $p = 67$, and encryptor $e = 35$. It was easy to verify $d = 17$ does the trick. It satisfied the congruence

$$ed \equiv 1 \pmod{p-1}$$

Or, in this case,

$$\begin{aligned}
35d &\equiv 1 \pmod{66} \\
&\text{or} \\
35d &= 66n + 1
\end{aligned}$$

Euclid's algorithm provides a method to discover the answer in the first place. The method of Euclid starts with 66 and 35; then divides the smaller into the larger to get a remainder (here it is 31). Then the division is repeated with the smaller numbers 35 and 31. And repeated still further until the remainder is as small as it can get. (For there to be a solution to the congruence, the final remainder must be 1.) The equations are as follows.

$$\begin{aligned}
66 &= 1 \times 35 + 31 \\
35 &= 1 \times 31 + 4 \\
31 &= 7 \times 4 + 3 \\
4 &= 1 \times 3 + 1
\end{aligned}$$

The last equation is in the form we want; we can substitute into that equation for $3, 4$, and 31 to get everything in terms of 66 and 35. The substitutions to use are restatements of the first 3 equations above.

$$\begin{aligned} 31 &= 66 - 35 \\ 4 &= 35 - 31 \\ 3 &= 31 - 7 \times 4 \end{aligned}$$

Now substitute into the last equation for $3, 4$, and 31 sequentially.

$$\begin{aligned} 4 &= 3 + 1 \\ 4 &= (31 - 7 \times 4) + 1 \\ 35 - 31 &= (31 - 7 \times (35 - 31)) + 1 \\ 35 - (66 - 35) &= ((66 - 35) - 7 \times (35 - (66 - 35))) + 1 \end{aligned}$$

Combining terms to get the coefficients of 66 and 35:

$$\begin{aligned} 2\,(35) - 66 &= 8\,(66) - 15\,(35) + 1 \\ 17\,(35) &= 9\,(66) + 1 \end{aligned}$$

The last expression is in the form we want; we can see that $d = 17$ and, also, we get $n = 9$ for free.

The algorithm works, but parts of it, especially the substituting part, can get tedious. It would be cool if there was a parsimonious way to conduct the algorithm, and, indeed, there is. It amounts to keeping three columns of numbers. The two left hand columns accomplish taking successively smaller remainders (going down), and the right hand column does the substitution (coming back up). Start the procedure with the two numbers of interest.

$$\begin{array}{c} 66 \\ 35 \end{array}$$

Divide the smaller number into the larger. The middle column has the number of time 35 goes into 66, and the remainder is placed in the first column.

$$\begin{array}{cc} 66 & \\ 35 & 1 \\ 31 & \end{array}$$

Repeat the procedure until the remainder is 1.

$$\begin{array}{cc} 66 & \\ 35 & 1 \\ 31 & 1 \\ 4 & 7 \\ 3 & 1 \\ 1 & \end{array}$$

Start the third column at the bottom by placing 1there.

66		
35	1	
31	1	
4	7	
3	1	
1		1

Now work up the third column. Each entry is calculated by multiplying the entry in the second column times the third column entry one row lower. Then add the element in the third column two rows lower. For the first calculation there is only one element in the third column, so the calculation is $1 \times 1 + 0 = 1$.

66		
35	1	
31	1	
4	7	
3	1	1
1		1

Repeat. The next calculation is $7 \times 1 + 1 = 8$.

66		
35	1	
31	1	
4	7	8
3	1	1
1		1

The next calculations are $1 \times 8 + 1 = 9$, and $1 \times 9 + 8 = 17$.

66		
35	1	17
31	1	9
4	7	8
3	1	1
1		1

This scheme goes pretty fast with a little practice as in another example or two.

Example 10.4 *Let the prime modulus p=103 and encryptor e=91. Find d.*

The first two columns yield successively smaller remainders.

102		
91	1	
11	8	
3	3	
2	1	
1		1

The elements in the third column are calculated by multiplying the associated second column element times the element in the third column down one row, then add the third column element down two rows.

102			102			102			102		
91	1		91	1		91	1		91	1	37
11	8		11	8		11	8	33	11	8	33
3	3		3	3	4	3	3	4	3	3	4
2	1	1	2	1	1	2	1	1	2	1	1
1		1	1		1	1		1	1		1

The verification is $37 \times 91 = 3367 = 33 \times 102 + 1 \equiv 1 \pmod{102}$. The example can be further verified by encrypting and decrypting to see if the original message returns.

Example 10.5 *Let* $p = 103$, $e = 91$, *and* $d = 37$. *Encrypt and decrypt* $a = 2$.

Encrypt:

$$\begin{aligned} 2^9 &= 512 \equiv 100 \equiv -3 \pmod{103} \\ 2^{54} &= \left(2^9\right)^6 \equiv (-3)^6 = 729 \equiv 8 \pmod{103} \\ 2^{36} &= \left(2^9\right)^4 \equiv 81 \pmod{103} \\ 2^{91} &= \left(2^{54}\right)\left(2^{36}\right)(2) \equiv (8)(81)(2) = 1296 \equiv 60 \pmod{103} \end{aligned}$$

Decrypt:

$$\begin{aligned} 60^2 &= 3600 \equiv -5 \pmod{103} \\ 60^8 &= \left(60^2\right)^4 \equiv 625 \equiv 7 \pmod{103} \\ 60^{24} &= \left(60^8\right)^3 \equiv 7^3 = 343 \equiv 34 \pmod{103} \\ 60^5 &= \left(60^4\right)(60) \equiv (25)(60) = 1500 \equiv 58 \pmod{103} \\ 60^{37} &= \left(60^{24}\right)\left(60^8\right)\left(60^5\right) \equiv (34)(7)(58) = 13{,}804 \equiv 2 \pmod{103} \end{aligned}$$

One more example illustrates a potential problem that might arise and its solution.

Example 10.6 *Let* $p = 103$ *and* $e = 29$. *The regular calculation for* d*:*

102		
29	3	7
15	1	2
14	1	1
1		1

In this case things don't quite work: $7\,(29) = 203 = 2\,(102) - 1 \equiv -1 \pmod{102}$, but they are easily fixed. Simply change the sign:

$$-7\,(29) = -203 = -2\,(102) + 1 \equiv 1 \pmod{102}$$

d, then, is -7; a negative d is a little bit uncomfortable, so recall $-7 \equiv 95 \pmod{102}$, and $d = 95$ works fine.

$$95\,(29) = 2755 = 27\,(102) + 1 \equiv 1 \pmod{102}$$

10.4 a real example

Sending a meaningful message requires a large prime modulus, which, in turn, implies use of a computer. A large modulus, encryptor, and decryptor increase the security of the code, as well. In the example to follow the programming language is Mathematica, and none of the commands requires more than a second or two to execute, at least on the machine I am using.

Start with a sample message: "Transfer $1,000,000 to Nish's account." And we require the message in numerical form. It is not hard to write a crude routine which, in this case, changes each symbol into a two digit number.

a_1=5828112429161528378265757474747574747437302537521929189329371113132531243038

"T" is transformed to '58', "r" to '28', "a" to '11', and so forth. Simple substitution cyphers such as this one are popular in literature: see, for example, "The Gold Bug" by Edgar Allan Poe and "The Adventure of the Dancing Men" by Arthur Conan Doyle. They are also common on the puzzle pages of daily newspapers. In any event, they are not very secure. To access Fermat's theorem we require some large numbers. The Mathematica command "Random[Prime,{1,10^100}]" returns a prime number between 1 and 1 followed by 100 zeros.

p=57656350467772590842968926675632437856669578908091633629232
74645552392498986648935712483504631040643

e=15709770121388900982473516951863122516186766705982763868988
10281172240891679239034775140086266134343

Mathematica's command for Euclid's algorithm is ExtendedGCD[p-1,e], and a decryptor is easily obtained.

d=15658322322815747764012219315548941836749339823780334074029
13173160713796989428801131649044416494831

A patient reader can type the numbers into a computer and verify that $ed \equiv 1 \pmod{p-1}$. It's more instructive, and much more fun, however, to use p, e, and d to encrypt and decrypt some messages. The Mathematica command to generate a cyphertext is PowerMod[a_1, e, p]; in other words, raise a_1 to the power e, and take the remainder using the modulus p. Denote the cyphertext as c_1.

c_1=2537971453639642635622750110104879189317450917631082084271
781641502958590430167649770860315527660657

To return the message a_1, it is simply a matter of running PowerMod once more, this time with d and c_1: PowerMod[c_1, d, p]. While it is certainly possible to verify the example with the numbers given, it would be better to make up different examples.

The cyphertext, c_1, is essentially a random number, and does, indeed, look that way to an eavesdropper. A way to emphasize its random character is to make a trivial change in the original message: "Transfer $1,000,000 to Oish's account." The numeric version is hardly distinguishable from the original; indeed, it's hard to find the single digit at variance.

5828112429161528378265757474747574747437302537531929189329371
113132531243038

Using p and e to generate cyphertext c_2, however, results in a sequence with no apparent relationship to c_1.

c_2=4117321761273706571604525053743054740759147055954957638538
186298827008076055619317468196994629382843

Even if the bad guys had the original message in both encrypted and unencrypted forms, it would be of no help in deciphering cyphertext for a very similar message. This is a powerful property of Fermat encryption, and one which strengthens the security of the codes.

There is, however, one area where Fermat encryption is vulnerable. The vulnerability is known as the private key problem. In order to establish a code, it is necessary to communicate the prime modulus and the encryption number so that the sender and the receiver can coordinate the transmissions. But this original communication can not be sent over the derived secure Fermat channel, as the secure channel did not come into existence until after the communication of p and e (or d). The next section, and the next chapter, are different ways of confronting the private key problem.

10.5 public key encryption

One way of avoiding the private key problem is to publicly announce the modulus and the encryptor, but do so in a way that is of no use to people wishing to intercept coded messages. This is called public key encryption and relies on two things: a theorem from Euler, and the fact that some problems are hard to solve, even by high speed computers. First Euler's theorem.

Fermat's theorem works only for prime numbers, the logic requiring that there be no repeats or zeros in the multiplication table. Euler generalizes Fermat's result to non-prime numbers, as well, by restricting consideration to rows of the multiplication table which satisfy the no zeros and no repeats conditions. Recall the multiplication table modulo 10 from chapter 9 reproduced here.

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	0	2	4	6	8
3	3	6	9	2	5	8	1	4	7
4	4	8	2	6	0	4	8	2	6
5	5	0	5	0	5	0	5	0	5
6	6	2	8	4	0	6	2	8	4
7	7	4	1	8	5	2	9	6	3
8	8	6	4	2	0	8	6	4	2
9	9	8	7	6	5	4	3	2	1

The rows (and columns) associated with $1, 3, 7$, and 9 satisfy the necessary properties; the other rows do not. The difference is that 1,3, 7, and 9 are relatively prime to the modulus 10.

Definition 10.1 *Two numbers are relatively prime if, when comparing their respective (unique) factorizations into prime numbers, there are no common factors.*

The factorization of 10 is 2×5. All the numbers less than 10 except for 1, 3, 7, and 9 have either a 2 or 5 in their factorization into prime numbers. Euler denoted the size of the set of numbers relative prime numbers to m and less than m as $\phi(m)$, called "phi" or the totient.

Theorem 10.2 *Euler's theorem: for any modulus m, and any number a which is relatively prime to m,*

$$a^{\phi(m)} \equiv 1 \pmod{m}$$

Verifying the theorem for $m = 10$ we have $\phi(10) = 4$ and

$$\begin{aligned} 3^4 &= 81 \equiv 1 \pmod{10} \\ 7^4 &= 2401 \equiv 1 \pmod{10} \\ 9^4 &= 6561 \equiv 1 \pmod{10} \end{aligned}$$

Euler's generalization allows for a way around the public key problem. When the modulus is prime, and the modulus and encoder are known, the bad guys can solve for the decoder using the congruence $ed \equiv 1 \pmod{p-1}$. Because of Euclid this is an easy congruence to solve. On the other hand, when the modulus is not prime the congruence to solve for the decoder is $ed \equiv 1 \pmod{\phi(m)}$.[3] This is also an easy congruence to solve but only when $\phi(m)$ is known. This is the part that trips up the bad guys: finding $\phi(m)$ requires knowing the prime factors of m.

[3]Note that for a prime number $\phi(p) = p - 1$.

Example 10.7 *Suppose $m = r \times s$, where r and s are both prime numbers. Then any multiple of r less than m is not relatively prime to m, since they have common prime factor r. There are $s - 1$ multiples of r less than m. Similarly there are $r - 1$ multiples of s less than m. The total of relatively prime numbers $\phi(m)$ is*

$$\begin{aligned} \phi(m) &= (m-1) - (r-1) - (s-1) \\ &= (rs-1) - r + 1 - s + 1 \\ &= (r-1)(s-1) \end{aligned}$$

Finding the prime factors of a large number, say 100 digits or more, is computationally hard and beyond the capacity of present day computers to find in a reasonable time. Anyone who wishes to receive secret messages can disclose publicly a modulus and an encoder, and be (sort of) confident that the decoder can not be inferred. The receiver can easily derive their own decoder by constructing the modulus as the product of large prime numbers. This is the essence of public key encryption.

Example 10.8 *Let $r = 911$ and $s = 1009$. also, let the encryptor, e, be 601. The code modulus, m, then, is*

$$m = rs = 911(1009) = 919,199$$

And the number of relatively prime numbers less than m is

$$\begin{aligned} \phi(m) &= (r-1)(s-1) = (910)(1008) \\ &= 917,280 \end{aligned}$$

Finding $\phi(m)$, of course, is the difficult part for an outsider unaware of the prime factors of m. (It is even a little bit difficult for this small example.) But once $\phi(m)$ is known, it is relatively simple to find the decryptor, d, that solves

$$ed \equiv 1(\operatorname{mod} \phi(m))$$

The calculations using Euclid's algorithm are tabulated.

917,280		
601	1,526	244,201
154	3	160
139	1	41
15	9	37
4	3	4
3	1	1
1		1

d is computed to be 244,201 and is verified by the calculations.

$$\begin{aligned} 244,201(601) &= 146,764,801 \\ 160(917,280) &= 146,764,800 \end{aligned}$$

With some computer assistance, the code can be tested. For example,

$$\begin{aligned} 121^e &\equiv 405,061 \pmod{m} \\ 405,061^d &\equiv 121 \pmod{m} \end{aligned}$$

Technology advances, however, and factoring large numbers might not always be difficult enough to support public key encryption.

> How many computational steps are needed to find the prime factors of a 300-digit number? The best classical algorithm known would take about 5×10^{24} steps, or about $150,000$ years at terahertz speed. By taking advantage of innumerable quantum states, a quantum factoring algorithm would take only 5×10^{10} steps, or less than a second at terahertz speed. (M. Nielsen, *Scientific American*, May 31, 2003)

The ability to harness quantum processes has already begun. The effect of quantum capabilities on encryption is the subject of the next chapter.

10.6 infinitude of primes

Encryption techniques use as raw material large prime numbers. Further, the existence of lots of prime numbers is necessary both for designing a secret code and ensuring the code does remain, in fact, secret. This section has a couple of theorems verifying the existence of a sufficient number of large primes; in fact, there are infinitely many. Besides being relevant to our coding activities, there is something satisfying about peering into the domain of very large numbers and identifying the regularities, and even beauty in that domain.

As primes become sparse as the numbers get large (there is, on average, more and more distance between two primes), the concern might be that we'll run out of prime numbers. Fortunately, for coding and a variety of other applications, that concern is unfounded. The number of primes is infinite, and the proof thereof, from Euclid, is elegant.[4]

Theorem 10.3 *There is an infinity of prime numbers.*

First assume there exists a largest prime number, and then derive a contradiction, thereby showing the original assumption to be false. Suppose the largest prime number is P. Construct the product of all the primes up to and including P, and to the product add 1.

$$Q = (2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdots P) + 1$$

Q is not divisible by any of the primes used in its construction: the remainder is always 1. Then Q must itself be prime, or divisible by primes larger than P.

[4]See G. H. Hardy's discussion of the proof in *A Mathematician's Apology*.

Either way we have a contradiction to the assumption of a largest prime, and that assumption must then be false.

Another important theorem concerning the behavior of primes is known as the prime number theorem and was first stated by Gauss. The actual number of primes less than a particular number, n, is conventionally denoted as $PrimePi(n)$. For example, $PrimePi(10) = 4$,and the four prime numbers are 2, 3, 5, and 7. Gauss' insight was that $PrimePi(n)$ behaves a lot like $n/\ln n$.

Theorem 10.4 *Prime number theorem:*

$$\underset{t\to\infty}{Limit}\frac{PrimePi(t)}{t/\ln t} = 1$$

The prime number theorem gives another verification of the infinitude of primes, as the ration $t/\ln t$ is always increasing.

$$\begin{aligned}\frac{d}{dt}\frac{t}{\ln t} &= \frac{\ln t - 1}{(\ln t)^2}\\ &= \frac{1}{\ln t} - \frac{1}{(\ln t)^2} > 0\end{aligned}$$

Furthermore, we can see there are plenty of large primes. Just as 90% of the numbers below 10^t exceed 10^{t-1}, the same is approximately true for primes. 90% of the primes less than 10^{100}, say, exceed 10^{99}. Doing approximate computations

$$\begin{aligned}\frac{\frac{10^{100}}{100\ln 10} - \frac{10^{99}}{99\ln 10}}{\frac{10^{100}}{100\ln 10}} &\approx \frac{10^{100} - 10^{99}}{10^{100}}\\ &= 1 - .1 = .9\end{aligned}$$

The computation presumes $99\ln 10$ is "close enough" to $100\ln 10$; being a little more careful yields a result of 89.9%.

10.7 cyphertext entropy

A necessary condition of a good secret code is that the cyphertext (that is, the message that is sent after encryption takes place) should not be useful to the bad guys. One way to evaluate this characteristic of the cyphertext is to compute its entropy.

To investigate the entropy of messages in words, consider the name "kyle kerner." The name is composed of three e's, two k's and two r's, as well as one apiece of y, l, n, and a space, for a total of 11 symbols. The relative frequency of e, for example, is 3/11. Using relative frequencies instead of probabilities, applying the entropy operator yields

$$\begin{aligned} & Entropy[\text{"kyle kerner"}] \\ = & -\left(\frac{3}{11}\ln\frac{3}{11}+\frac{4}{11}\ln\frac{2}{11}+\frac{4}{11}\ln\frac{1}{11}\right) \\ \simeq & \ 1.8462 \end{aligned}$$

Here is a quotation which appeared as an encrypted puzzle in the daily newspaper, presented for simplicity without upper case and punctuation.

> "the details vanish in the birdseye view but so does the birdseye view vanish in the details william james"

Using the same relative frequency technique as before, the computed entropy of the quotation is 2.649. Some perspective is added by comparing the actual entropy to the most the entropy could be . In this case the maximum possible entropy is ln 27, as there are 27 possible characters (26 letters and a space). And the ratio of actual to maximum is approximately 80%.

For the puzzle in the newspaper the cyphertext appears as follows.

> "axl hladgcj tdvgjx gv axl pgyhjlul tgle pka ji hilj axl pgyhjlul tgle tdvgjx gv axl hladgcj egccgdf odflj"

"t" in plaintext becomes "a" in cyphertext, "h" becomes, "x", and so forth. The entropy ratio of the cyphertext is the same 80%, of course. It is instructive to compute the ratio for a random sequence of symbols of the same length: for this example there are 103 characters.

A computer simulation experiment generating the ratio for 27 possible numbers generated randomly in lengths of 103 routinely returns an entropy ratio of about 95%, greater than the 80% entropy ratio in the puzzle. So the codebreaker (or newspaper puzzler) has a significant advantage just with the relative frequency of individual symbols. There are, of course, other useful patterns besides individual letters, including word patterns, letter patterns within words, and so forth.

The next part of the experiment involves a computer, as well. The idea is to check the entropy ratio for cyphertexts using the encryption systems under consideration in this chapter. Here we generate cyphertext using a prime modulus and an encryptor of 100 digits. The entropy ratio is computed as the entropy of the cyphertext divided by ln 10, as there are 10 possible digits. For the cyphertext the ration is routinely about 98%. For a list of 100 random integers, again the ration is about 98%. Indeed, many computerized random numbers are generated by raising a seed number to a very high power, divide by another large number, and report the remainder: effectively this reproduces the transformation of plaintext to cyphertext.

So the relative frequency wedge available to the codebreaker with substitution cyphers does not exist for the "Fermat encryption" secret codes in this chapter. Of course, as mentioned earlier, there are a variety of other possible patterns. But they can be checked in a similar fashion, and the interested reader with access to some computer capability is encouraged so to do.

10.8 summary

The process of sending secret messages, or maintaining the secrecy and integrity of databases, has changed as technology has changed. Centuries old mathematical results from Fermat, Euler, and Euclid have become central to secret codes as computational technology has improved. Technology continues to improve; encryption follows along, and different mathematical results become important. In the next chapter quantum processes and their effects on encryption are discussed.

10.9 references

Doyle, Arthur Conan, "The Adventure of the Dancing Men," in The Return of Sherlock Holmes, 1903.

Hardy, G. H., *A Mathematician's Apology*.

M. Nielsen, *Scientific American*, May 31, 2003

Ore, Oystein, *Number Theory and Its History*. McGraw-Hill Book Company, 1948.

Poe, Edgar Allan, "The Gold Bug," 1843.

10.10 exercises

Exercise 10.1 *Consider a private key secret code with prime modulus* $p = 2633$ *and encoder* $e = 43$. *What is the decoder, d? Suppose the received cyphertext is 2477. What is the message? (Use an Excel spreadsheet and the function "mod.")*

Exercise 10.2 *Using Fermat's theorem, verify that 7387 is not a prime number. Consider a multiplication table with modulus equal to 7387. Find some entries in the multiplication table which are zero. (An Excel spreadsheet will be useful.) Using what you learned from the position of the zero entries, estimate a value for Euler's totient (phi) by eliminating the affected numbers. Use Euler's theorem to verify (or disprove) your estimate of phi.*

Exercise 10.3 *For* $p = 97$ *and* $e = 37$, *find the decoder,* d. *Verify that the code works by encoding and decoding a message, say* $a = 2$. *Excel might be useful for the verification.*

Exercise 10.4 *Repeat the previous exercise for* $p = 1009$ *and* $e = 97$.

Exercise 10.5 *Consider modulus* $m = 33,277 = (107)(311)$. *The code encryptor is* $e = 2003$. *What is the decryptor, d?*

Exercise 10.6 *The code parameters are* $p = 79$ *and* $e = 41$. *Compute* d. *Compute the cyphertext for plaintext* $a = 2$.

Exercise 10.7 *From the proof of the infinitude of primes, when the first seven primes are multiplied together, and* 1 *is added to the product, the result is either prime or divisible by some prime number greater than the seventh prime. Which is it?*

How hard would it be to repeat the exercise for the first ten primes?

Exercise 10.8 *Consider a secret code modulus* $m = 221 = (13)(17)$. *Let the code encryptor be* $e = 97$. *Find the decryptor* d, *and encrypt, and then decrypt, a few messages. Do you see any weakness(es) with this code? Does the phrase "fixed point" have any relevance?*

Exercise 10.9 *Using the prime number theorem, approximately how many prime numbers are between* 10^{200} *and* 10^{201}*?*

Exercise 10.10 *What is the relative frequency entropy of* "*sandy koufax*"*?* "*leonhard euler*"*?*