One-time pads

So, we need to eliminate patterns in the key as well as in the message. We could do this by making the key a random string of letters or, in the case of the Gronsfeld cipher, a random string of numbers.

But, how do we create a random string? We could place the letters of the alphabet on slips of paper and draw them from a bag with replacement to create a random string of letters, or do that same thing with the digits 0, 1, …, 9 to create a random string of digits. We could monitor a random process; e.g., radioactive decay or signals from space.

For a random string of numbers, we could use the digits of π; this string of digits is known to be random. Here are the first 1000 digits of π according to the computer algebra system *Mathematica*.

```
3141592653589793238462643383279502884197169399375105820974944592307816406286
2089986280348253421170679821480865132823066470938446095505822317253594081284
8111745028410270193852110555964462294895493038196442881097566593344612847564
8233786783165271201909145648566923460348610454326648213393607260249141273724
5870066063155881748815209209628292540917153643678925903600113305305488204665
2138414695194151160943305727036575959195309218611738193261179310511854807446
2379962749567351885752724891227938183011949129833673362440656643086021394946
3952247371907021798609437027705392171762931767523846748184676694051320005681
2714526356082778577134275778960917363717872146844090122495343014654958537105
0792279689258923542019956112129021960864034418159813629774771309960518707211
3499999983729780499510597317328160963185950244594553469083026425223082533446
8503526193118817101000313783875288658753320838142061717766914730359825349042
8755468711595628638823537875937519577818577805321712268066130019278766111959
09216420199
```

If sufficiently long string of these digits was used to encrypt a message using a Gronsfeld cipher, then there would be no depth within the message and no pattern in the key.

But, say, that we had 100 messages to encrypt with a Gronsfeld cipher. If we used the same keystring for each message, we would have created depth – not within a single ciphertext message – but among the collection of ciphertext messages. Stripping the first letter from each ciphertext message would result in 100 letters from the same alphabet; the frequencies of these letters could be used to determine the first shift. Stripping the second letter from each ciphertext message would result in 100 letters from the same alphabet; the frequencies of these letters could be used to determine the second shift. Etc. The keystring cannot be re-used or depth would be created. The keystring must be random, and it can only be used once.

Such an encryption scheme is called a one-time pad (OTP). Claude Shannon (1916 – 2001), the Bell Labs engineer and mathematician who founded the field of information theory, proved that the one-time pad is unbreakable (1949). It is the only provably secure cryptosystem.

So, why doesn't everyone use a one-time pad? The practical problems associated with a one-time pad make it a little-used encryption scheme. The practical problems are:

Generating long random keystrings.

Exchanging the keystrings between sender and receiver.

There are other cryptosystems that, while not provably secure, are "secure enough" and are more convenient.

Stream ciphers

Block ciphers, like Playfair and Hill ciphers, encrypt plaintext of a fixed length – digraphs for the Playfair cipher and *n*-graphs for *n*-dimensional Hill ciphers. If the length of the plaintext message is not an integral multiple of the length of a block, the plaintext message must be padded.

Stream ciphers can encrypt plaintext messages of variable length. The one-time pad can be thought of as an example – each message uses a portion of the key with length equal to the length of the plaintext message. (Then that portion of the key is never re-eused.)

The ideas that resulted in modern stream ciphers originated with another AT&T Bell Labs engineer, Gilbert Vernam (1890 – 1960). In 1917, Vernam developed a scheme to encrypt teletype transmissions. Unlike Morse code, which uses symbols of different lengths to substitute for letters of the alphabet, teletype transmission used what we would today call a 5-bit code for the letters of the alphabet and certain keyboard commands. Its use was similar to the way that the 8-bit ASCII code is used today in computing.

A = XX•••	B = X••XX	C = •XXX•	D = X••X•	E = X••••
F = X•XX•	G = •X•XX	H = ••X•X	I = •XX••	J = XX•X•
K = XXXX•	L = •X••X	M = ••XXX	N = ••XX•	O = •••XX
P = •XX•X	Q = XXX•X	R = •X•X•	S = X•X••	T = ••••X
U = XXX••	V = •XXXX	W = XX••X	X = X•XXX	Y = X•X•X
Z = X•••X				

Like today’s keyboards, two functions were associated to each key – a “letter” function and a “figures” function.

letters	figures
-	A
B	?
C	:
D	who are you ?
E	3
F	%
G	&
H	local currency
I	8
J	ring bell
K	(
L	)
M	.
N	,
O	9
0	P
Q	1
R	4
S	‘
T	5
7	U
V	=
W	2
X	/
Y	6
+	Z

The 26 letters of the alphabet and the corresponding “figures” used 26 of the 32 possible 5-bit strings.

••••• was not used.

The remaining five 5-bit strings were used for commands.

••X••	space
•••X•	carriage return
•X•••	line feed
XX•XX	shift to figures
XXXXX	shift to letters

This code is called the International Telegraph Alphabet number 2 (ITA2) and is still in use in TDDs and some amateur radio applications.   It was based upon a 5-bit teletype code developed by the French telegraph engineer Émile Baudot (1845 – 1903) around 1874. (The rate of symbol transmission called the baud is named after Baudot.) Around 1901, Baudot’s code (ITA1) was modified by Donald Murray (1865 – 1945), a New Zealand sheepfarmer. A later modification resulted in ITA2.

The idea of using 5-character strings to represent the letters of the alphabet seems to be due to Sir Francis Bacon (1561 – 1626). Bacon used the first two letters of his name for the characters in the strings.

a	AAAAA	i/j	ABAAA	r	BAAAA
b	AAAAB	k	ABAAB	s	BAAAB
c	AAABA	l	ABABA	t	BAABA
d	AAABB	m	ABABB	u/v	BAABB
e	AABAA	n	ABBAA	w	BABAA
f	AABAB	o	ABBAB	x	BABAB
g	AABBA	p	ABBBA	y	BABBA
h	AABBB	q	ABBBB	z	BABBB

When Vernam developed his encryption scheme during the early Twentieth Century, he was not thinking about 5-bit strings of 1's and 0's; he was thinking about cross X or dot •, mark or no mark, low voltage or high voltage, etc., but today it is easiest to think of 1's or 0's.

Thinking this way, Vernam's encryption scheme can be easily described. The key was a random string of 0's and 1's. The plaintext message was converted to a string of 0's and 1's using ITA2. The two strings were XORed to generate the ciphertext. Pattern (plaintext) plus ($\oplus$) random (*key*) yields random (CIPHERTEXT).

Exercise 3.2. Encrypt the message nku using a Vernam cipher with keystring 011111110010010.

The operation XOR is equivalent to bitwise addition modulo 2.

$$\begin{array}{c|cc} \oplus & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

0 is the identity for XOR; i.e., XORing a bit with 0 does not change the bit. $0 \oplus 0 = 0$ and $1 \oplus 0 = 0 \oplus 1 = 0$. Notice that, for XOR, each bit is its own inverse; i.e., a bit XORed with itself yields 0. $0 \oplus 0 = 0$ and $1 \oplus 1 = 0$.

The last property makes decryption easy. Any string XORed with itself is a string of 0's.

$$\begin{array}{r} 011111110010010 \\ \oplus\ 011111110010010 \\ \hline 000000000000000 \end{array}$$

Therefore, ciphertext message can be decrypted by XORing (again) with the key.

$$\begin{aligned}&\text{CIPHERTEXT} \oplus key\\ &= (\text{plaintext} \oplus key) \oplus key\\ &= \text{plaintext} \oplus (key \oplus key)\\ &= \text{plaintext} \oplus 00\ldots 0\\ &= \text{plaintext}\end{aligned}$$

Exercise 3.3. Decrypt the message that was encrypted in exercise 3.2.

Exercise 3.4. Decrypt the following message that was encrypted using a Vernam cipher with keystring 110000100011110111111111110101.

101111100010100110010011110010

Vernam's cipher was not a one-time pad. It was implemented using relays, and the keystring was punched on a (long) paper tape. Even though the tape was long, it would eventually repeat and create depth. An AT&T engineer Lyman Morehouse developed an idea that created long keystrings by combining two shorter strings of different lengths. The idea can be exhibited using two small strings. Let the first string be of length 6

101110

and the second of length 7

1111011

Create the keystring by XORing these two strings.

101110101110101110101110101110101110101110**101110**
111101111110111111011111101111110111111011**1111011**
010011010000010001110001000001011001010101**010011**

Because of the difference in lengths and because the lengths are relatively prime, combining these strings, which have periods 6 and 7, results in a string having period 42. A string of 999 bits combined with a string of 1000 bits would result in a keystring of 999000 bits. But, it would still not be a one-time pad.

The need for the key to be both "endless and senseless" was recognized by Joseph Mauborgne (1881 – 1971) an officer in the United States Army Signal Corps.

The United States – Soviet Union hotline that was constructed during the 1960's used a one-time pad with the keystring on a paper tape. The voice encryption device SIGSALY that was developed by Bell Labs for the United States Army during World War II used a one-time pad with the key consisting of random noise recorded on a record. Concern about the poor encryption systems that they used during World War I caused the Soviets to implement a one-time pad, hand cipher for their spies during World War II and after. Unfortunately for them, the need for many keys caused them to re-use keys. This was noticed by American cryptanalysts who were able to break many messages. The breaking of these messages was called the VENONA project and led to the arrests of the Soviet atomic spies.

From the 1920's until the 1970's encryption machines dominated cryptography. The purpose of these machines was to implement ciphers with long keys and, therefore, protect against attacks using the period.

Known plaintext attack on depth of two

A known plaintext attack can succeed against a stream cipher with a depth of two. If two messages are encrypted with the same keystream, XORing the two ciphertexts will remove the keys and result in the XOR of the plaintexts.

$$\begin{aligned}
&\text{CIPHERTEXT1} \oplus \text{CIPHERTEXT2} \\
&= \left(\text{plaintext1} \oplus key\right) \oplus \left(\text{plaintext2} \oplus key\right) \\
&= \left(\text{plaintext1} \oplus \text{plaintext2}\right) \oplus \left(key \oplus key\right) \\
&= \left(\text{plaintext1} \oplus \text{plaintext2}\right) \oplus 00\ldots0 \\
&= \text{plaintext1} \oplus \text{plaintext2}
\end{aligned}$$

If a crib, a portion of the plaintext, is known, an attack can be made.

Exercise 3.5. Here are two ciphertext messages that were encrypted using a Vernam cipher and the same keystring.

CIPHERTEXT1

```
11111111101111001000001110001111010000110101001000
11000010010110010010010000001100111101110100101011
100000100110101011110011010011
```

CIPHERTEXT2

```
11000110000010010011110100011010101111010110100110
11011100000010010110001101011100111110110001110100
10010
```

XOR the two ciphertexts to obtain the XOR of the plaintexts.

It is known that the word `jayhawk` appears in one of the plaintext messages. Drag the crib through the XOR of the plaintexts. For example, assume that the crib begins one of the plaintext messages

```
Crib                 11010110001010100101110001100111110
XOR of plaintexts    00111001101101011011111010010101111
                     -----------------------------------
                     11101111100111111110001011110010001
                     q    k    v    k    h    u    z
```

If the crib is in the correct location, plaintext should result.

$$\begin{aligned} &\text{plaintext1} \oplus (\text{plaintext1} \oplus \text{plaintext2}) \\ &= (\text{plaintext1} \oplus \text{plaintext1}) \oplus \text{plaintext2} \\ &= \text{plaintext2} \end{aligned}$$

This crib is not in the correct location; shift it one place to the right and try again. Etc.

Once the crib is in place extend each plaintext message forward and backward until both plaintexts have been obtained. Also determine the keystring.

Modern stream ciphers

Modern stream ciphers operate much the same as Vernam's original cipher. They are not one-time pads; their keystrings are pseudorandom. Because the XOR operation and the methods used to generate keystrings are not complex operations, stream ciphers are typically faster than block ciphers. Stream ciphers are often used in situations (for example, wireless communications) in which the length of the plaintext message is not known beforehand.