Fall 2005
Chris Christensen
Cryptology Notes

# One-time Pad (OTP)

The security of the Vigenère cipher can be enhanced by using a long key. A key that is as long as the plaintext message would result in having no period. But, there still might be some pattern in the key that a cryptanalyst might find helpful.

Our goal in this section is to create randomness in the key.

## Cryptography

One way to impose randomness on a plaintext message is by overlaying it with "noise." Of course, the receiver must be able to remove the "noise."

The noise can be done by adding stings of random numbers to plaintext. For example, let us convert the message

```
Arlington Hall analysts had first to strip off
the layer of additive.
```

to a string of numbers using the usual a = 01, b = 02, …, z = 26. We obtain

```
01181  00914  07201   51408   01101   00114   01102   51920
19080  10406  09181   92020   15192   01809   16150   60620
08051  00125  05181   50601   04040   92009   2205
```

Now we "find" a string of random number of the same length, say

```
14159  26535   89793  23846   26433   83279   50288   41971
69399  37510   58209  74944   59230   78164   06286  20899
86280  34825   34211  70679   82148   08651   3282
```

We overlay the randomness on the message by "adding;" but not in the usual way. We add vertical pairs of digits, and we add “without carrying;” i.e., we

add the vertical pairs modulo 10. For example, here are the results from adding three pairs of digits:

The result of adding the first block of the plaintext and the first block of the random keystring is:

| | |
|---|---|
| plaintext | 01180 |
| keystring | 14159 |
| ciphertext | 15230 |

Here is the ciphertext "merge" of the two strings:

```
15230  26449  86994   74244   27534   83383   51380   92891
78379  47916  57380   66964   64322   79963   12336   80419
84231  34940  39392   20270   86188   90650   5487
```

Random is applied to patterned, and random wins. This ciphertext message is a random string, and, hence, unbreakable. There is no pattern for the cryptanalyst to discover.

It is easy to see this intuitively. For example, let us assume that a three-letter word was enciphered to `MRV`. What was the word? `MRV` could be the ciphertext of any trigraph; there is no way to determine which three-letter word is the correct plaintext because there is no way to determine what random keystring was used. And, if we happened to know the three-letter word to which `MRV` corresponds, the remainder of the message would be unpredictable because it is enciphered with a string that has no relationship to the string used to encipher the three-letter word into `MRV`.

## Deciphering

To decipher our example, we need to subtract the keystring from the ciphertext string by subtracting (modulo 10) the vertical pairs of digits. We added without carrying; so, we should borrow as needed while subtracting;

Here is a comparison of addition (on the left) and subtraction:

| | |
|---|---|
| 01181 | 15230 |
| 14159 | 14159 |
| 15230 | 01181 |

Wouldn't it be nice if we didn't have to subtract? Wouldn't it be nice if we could use the same process to both encipher and decipher? That's possible if our addition is modulo 2. Here is some history.

## Vernam

From *The Codebreakers*:

> The foundation for such a cipher was laid by an American Telephone and Telegraph (AT&T) engineer Gilbert Vernam, who, in 1917, was working in the telegraph section of the company and concentrating on the newest development in telegraphy – the teletype(writer). A few months after World War One broke out, the telegraph section members began investigating the security of the teletype.
>
> The teletype used a code similar to Morse code to transmit messages. The code was named the Baudot code after its French inventor J.M.E. Baudot. The code consisted of 32 combinations of marks and spaces – 26 for the letters of the alphabet and 6 for keyboard commands (like *space*, *carriage return*, and *shift*). The letters of the alphabet were enciphered as follows (1 = mark and 0 = space):

```
A = 11000 B = 10011 C = 01110 D = 10010 E = 10000
F = 10110 G = 01011 H = 00101 I = 01100 J = 11010
K = 11110 L = 01001 M = 00111 N = 00110 O = 00011
P = 01101 Q = 11101 R = 01010 S = 10100 T = 00001
U = 11100 V = 01111 W = 11001 X = 10111 Y = 10101
Z = 10001
```

> Through an electrical arrangement involving rotating commutators, the proper sequence of pulses is sent out when a character's key is struck on the keyboard. … At the receiving end, the incoming pulses energize electromagnets that, in combination, select the proper character and print it. In the punched paper tape which is frequently used to run teletypewriters, marks are represented by holes and spaces

by leaving the tape intact. To read the tape, metal fingers push through the holes to make contact and thereby send pulses [this is a 1]; where there is a space, the paper keeps the fingers from completing the circuit [this is a 0].

Vernam suggested punching a tape of key characters and electromechanically adding its pulses to those of the plaintext, the "sum" to constitute the ciphertext. The addition would have to be reversible so that the receiver could subtract the key pulses from the cipher pulses and get the plaintext. Kahn, David, *The Codebreakers: The comprehensive history of secret communication from ancient times to the internet*, Scribner, 1996.

Here are the four possible cases and Vernam's rules for addition:

| plaintext | | key | | ciphertext |
|---|---|---|---|---|
| mark | + | mark | = | space |
| mark | + | space | = | mark |
| space | + | mark | = | mark |
| space | + | space | = | space |

Addition modulo 2 is exactly what is needed. Notice that with 1 = mark and 0 = space, Vernam's rules exactly correspond to addition modulo 2.

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

Vernam's rules also exactly corresponds to the operation XOR – exclusive OR. A XOR B means that either statement A is true or statement B is true but not both.

| | B is false. | B is true. |
|---|---|---|
| A is false. | A XOR B is false. | A XOR B is true. |
| A is true. | A XOR B is true. | A XOR B is false. |

Let us take the word `the` in Baudot code (with 1 and 0 rather than mark and space) 000010010110000 and add to it the random key string 100010000101100

$$\begin{array}{r} 000010010110000 \\ 100010000101100 \\ \hline 100000010011100 \end{array}$$

Notice that (because in addition modulo 2, $0 + 0 = 0$ and $1 + 1 = 0$) adding the random key string 100010000101100 to the ciphertext 100000010011100 returns plaintext

$$\begin{array}{r} 100000010011100 \\ 100010000101100 \\ \hline 000010010110000 \end{array}$$

In addition modulo 2, the additive inverse of 0 is 0 and the additive inverse of 1 is 1. So, the effective of adding the key twice (once to encrypt and once to decrypt) is to do nothing.

$$\begin{array}{r} 100010000101100 \\ 100010000101100 \\ \hline 000000000000000 \end{array}$$

Therefore, the same random keystring can be used for encrypting and decrypting.

> To combine the pulses electrically Vernam devised an arrangement of magnets, relays, and bus-bars. Since encipherment and decipherment were reciprocal, the same arrangement served for both. He fed the pulses into this device from two tape readers – one for the keytape, the other for plaintext tape. The mechanism closed a circuit, resulting in a mark, when the two incoming pulses were different, and opened a circuit, resulting in a space when they were the same. The output of marks and spaces could be transmitted just like an ordinary teletypewriter message to the received. Here the Vernam apparatus

subtracted out the key pulses, which were supplied by an identical keytape, and recreated the original plaintext pulses. …

Plaintext went in and plaintext came out, while anyone intercepting the message between the two endpoints would pick up nothing but a meaningless sequence of marks and spaces. … The advantage [to Vernam's method] was not the mechanical encipherment and printing of the message. That had been accomplished as far back as the early 1870s by two Frenchmen … . Rather it was the assimilation of encipherment into the overall communication process. Vernam created what came to be called "on-line encipherment"… . He freed a fundamental process in cryptography from the shackles of time and error. He eliminated the need for a human being – the cipher clerk – from the chain of communication.

…

In the first days of development, the Vernam keys took the form of loops of tape perforated with characters drawn from a hat, giving a random keytext. The engineers … soon spotted the flaw with this. The Vernam system is polyalphabetic. A $32 \times 32$ tableau may be set up with the 32 characters of the Baudot alphabet across the top as plaintext and down the side as keys. Because the Baudot alphabet is public information, the composition of the 32 cipher alphabets filling the body of the tableau would be known. Secrecy in the Vernam system thus resides entirely in its keys. Looped keytapes would pass through the Vernam mechanism at regular intervals, permitting a simple Kasiski solution, even though the key recovered would be incoherent. The engineers made the keytapes extremely long to increase the difficulty of such a solution. But then the keytapes became too hard to handle.

… Morehouse surmounted these difficulties by combining two short keytapes of different lengths in a Vernam device as if one were enciphering the other … . If one loop were 1,000 characters long and the other 999, the one-character difference would produced 999,000 combinations before the sequence would repeat. Kahn, David, *The Codebreakers: The comprehensive history of secret communication from ancient times to the internet*, Scribner, 1996.

## Mauborgne

[Joseph O.] Mauborgne [who was later to become the Army's Chief Signal Officer] recognized that even this system was not immune to cryptanalysis. [From studying the AT&T system, Mauborgne noted that any] repetition of any kind in the keys of cryptograms under analysis imperils them and perhaps dooms them to solution. It does not matter whether the repetitions lie within a single message or among several … . Repetitions in the key could not be permitted. … He therefore welded together the randomness of the key, created, perhaps almost accidentally by Vernam, and the nonrepetition of the key, discovered by the Army Signal School cryptologists, into what is now called the "one-time [pad]." It consists of a random key used once and only once. It provides a new and unpredictable key character for each plaintext character in the whole ensemble of messages ever to be sent by a group of correspondents.

And it is an unbreakable system. Some systems are unbreakable in practice only, because the cryptanalyst can conceive of ways of solving them if he had enough text and time. The one-time [pad] is unbreakable both in theory and practice. No matter how much text a cryptanalyst had available in it, or how much time he had to work on it, he could never solve it. … Kahn, David, *The Codebreakers: The comprehensive history of secret communication from ancient times to the internet*, Scribner, 1996.

## One-time Pads

Cryptography experts in different countries were conducting research similar to Mauborgne's, and between 1918 and the early 1920s other independently developed one-time techniques appeared. In Germany, the German Foreign Office formalized a single-use system that became known as the "one-time pad." Its name was derived from the two sheets of paper typed with a sequence of random numbers that became the key. A series of these pages were placed in two identical groups, or pads, one for the sender and one for the receiver. The numbers on the pads' pages were intended to be used just once and then discarded.

When an encryptor used a pad for encipherment, he included a prearranged means of identification called an "indicator group." The indicator changed with each transmission and identified the new pad sheet to be used. The numbers on each page provided a random key that was added to a second group of digits formed from the plaintext words of the message.

[Various schemes were used to convert plaintext to a string of numbers.]

During World War II, U.S. agents of the Office of Strategic Services (OSS) and their allies in Britain's Special Operations Executive (SOE) used one-time methods that combined alphanumeric grids and numerical key sheets. The grid form was often made of easily-disposed-of silk and contained 26 alphabet letters with columns of sequential numerals and cyclical alphabets aligned beneath them. The key sheets of five-digit groups were printed on paper or a very flammable synthetic material. By 1944 technicians had developed pages made of film that could be read with a hand-held device.

By the early 1960s, pads had become sheets the size of postage stamps or scrolls the size of large pencil erasers. Some were printed on paper that was photographed and sent as microfilm for extra concealment. Pads have also been made of foil-like material and have had pages of extremely combustible cellulose nitrate for rapid destruction in case of emergency. Wrixon, Fred B., *Codes, Ciphers & other Cryptic & Clandestine Communication: Making and breaking secret messages from hieroglyphs to the internet*, Black Dog & Leventhal Publishers.


This is a picture of one of the teletypewriters used in the US/Soviet Union Hotline. The key is a one-time pad that contained on the paper spool on the left. The machine is on display at NSA's National Cryptological Museum.

## Unbreakable

Yes, one-time pads are unbreakable – *if used properly*. Because the random keystring contains no predictable aspects, standard techniques of cryptology are defeated.

And, a brute force attack is not generally feasible. Consider the ciphertext message above 100000010011100 which is an enciphered version of the Baudot code for `the`. The string of numbers has length 15. For each of the 15 digits of the string, the random key string could have a 0 or a 1. There are $2^{15} = 32768$ strings of 0s and 1s of length 15. All of these could be added one at a time to the ciphertext string. But, this brute force attack would produce all possible trigraphs! How would we know which was the correct decipherment? Is it `the, and, she, her, him, etc`?

Even worse. Consider the enciphering of the message about Arlington Hall

```
15230  26449  86994   74244   27534   83383   51380   92891
78379  47916  57380   66964   64322   79963   12336   80419
84231  34940  39392   20270   86188   90650   5487
```

The string of digits corresponding to the plaintext of the message was added to a random key string of 114 digits – each digit could be 1, 2, 3, 4, 5, 6, 7, 8, 9, or 0. That means there would be $10^{114}$ possible key strings to check.

Furthermore, even if we were successful in decrypting part of the message, the decryption would provide no information about the remainder of the message because the remainder of the key has no relationship to the portion of the key that was discovered.

*… if used properly*

This is a big *if*. Using a one-time pad involves all the usual problems of a cipher. It requires generation of keys, and it requires that those keys be distributed and kept secure. There also needs to be some agreement between sender and receiver that indicates where the key string for the message begins.

But, there are some problems unique to using a one-time pad.

In particular, the keys must be long, random strings. The sender and receiver must possess random key strings that are long enough to encipher all messages between them without repeating any portion of the random key string in a message *or in subsequent messages*. So, it is necessary to have a method to generate random strings. But, *method … random* is often an oxymoron. Methods are often not random.

How might we generate a string of random numbers? We could just write each of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 on a slip of paper, toss the slips in a box, and draw out slips one at a time while replacing the slip and shaking up the box after each draw. If we needed a string of 0s and 1s, we could just write 0 and 1 on equal numbers of slips. This would not be exciting work.

Or, we could try to program a computer to generate a string of random numbers, but doing that is exactly the opposite of what computers are designed to do – computers are designed to do things in a very structured way – not in a random way. Random number generators on computers usually generate pseudo-random strings. Sometimes linear feedback shift registers are used to generate a key string. In either case, the string is not truly random, and the lack of randomness might be exploitable by a cryptanalyst.

Or, we might monitor naturally occurring chaotic processes (e.g., radioactive decay and radio static in the universe) and use their output as random strings.

Or, we might use a known random string of digits. The string of digits of the decimal expansion of the number pi is, for example, known to be a random string.

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717766914730359825349042875546873115956286388235378759375195778185778053217122680661300192787661119590921642019
```

The first 1000 digits of Pi, according to *Mathematica*.

To use a OTP, there must be some way to generate long keys, and, for complete security, the keys must random strings and they must not be reused.

It is sometimes tempting to reuse a key – the assumption is that no one will notice. First, we give an example of how reusing a key leads to depth that can be exploited by a cryptanalyst, and, then, we briefly look at the VENONA project which is a case when someone did notice.

## Depth

Vigenère ciphers were broken by locating repetitions of alphabets within the cipertext. It is just as bad to allow the repetitions to occur among several ciphertext messages. This is called depth. Here is an example of how depth can be used to cryptanalyze ciphertext messages.

Here are the first fifteen characters of one hundred ciphertext messages each of which has been enciphered with a Vigenère cipher using the same random string of letters.

```
hxtcp wndcw tciov
hxtod hcola ugwyh
ojitd kkycv zulny
gybbg wneva zjxda
dugtv hndbm yjqys

gybug smvsz kqpjj
hxxeo lcoiz ejwlv
kyitd foryv kyxqb
wmpez sosho yvtyl
wjxed ekyma ocpna

cdtaa lcolm gtswz
wjlmn aipuk zbhnc
hxtdz lcogi zuiay
rkgui yoryg kbvbp
beqay qrkmx gsxrj

vyhqs shshi zjswv
hxtdz xjbya sjxqy
hxtmm ljpwz eqxxn
wjxen ldvfu usilb
hxtnj gfmil kjwxw

vugqh smumb nbxco
belwi grxua ziith
crkuj mnvsq lbvnw
hxxea sxdca ziikh
velqq wmdbm rfxcl

muifc wpxcw tfwcz
kxtzv vzmcx nfvvl
iduam lpxub kmcov
kyitd fvxbw assop
kxpfx skduq tdyrn

cdtaa vzfco koial
kxxxz aidbi zdmcf
tykqd kvfie kmmcw
dqgfg qorca sfxqv
pkiuo aomuv xfemp
```

aegqj nzbnp knewb
qechz jnofg ziiop
obdzb ujela kpjru
vubmt zjvxa uumpo
wdiij gadbm rprpl

aehfv jhsya uoxql
qyetz jnklm ziybz
ryexj evmsp ktmch
pqrai anwnp kulnv
qhnbo gbbux njgjs

qecej fvxna mssdw
tegwz jxuin ltemc
hxteo wkdbi zgsus
gebqo ahomi fjkih
jugkz xaowb owidz

teaxj odxab nfwny
reczz dgiww aoxnk
qugfv aixie zieco
otdbo aiqnp oteba
owday vzkfk uvpmi

kxntv voryk uojnk
gymbz jnyha cfvna
muitd koolz ousap
ifiao zdcnq nfrap
uykqi lcknb nfwlp

dhdrz knylv kxfxs
thdyd lcomw uoiga
fuhqv jxrcv umhnu
cdakm wgknq bfphp
aehfg qoryt kuxny

kyaxo zzimp uxtny
qejzo aiqzz unxql
puvui fdxai zulnm
ofgay absic yoyvi
qyetz jnrul hfiwy

muiic agolc ytmju
zuifz jngcb nulnb
beedd kjxyz tpvju
wmpez sosho yvtyl
aoemm liolv uxywk

dqgfj xorya acwcp
oiagx cryot jieel
kqaed fbruu yisdz
pkiai dtpiz gtlxy
peitn winyz gohal

```
idauf whbmo gmpdw
tqred edvya ugxql
hhpzn eddnm xtiay
sltzo zzqlm gujal
pedfc kkbiw ltxql

ofgaa wnccw tbpco
owpui fdovc nsljk
sltzo zzxuu ktxqy
pkiua gioua yvqvl
wditz hvslw zfbcz

obafc jjeap ziinp
hxtmg dzqiz odeuj
keard kvvnp kgebo
dhpkt wodbi zzsdt
sdvxd kcmcx nfvru

sfxeo dzyhb nfrds
hhxfc whsoa lppuv
vucdd zvngi tbknk
ryskj mcouz ciecq
wjxed ekyma ocpna
```

The first alphabet is the set that consists of the first letter of each message. This should correspond to a Caesar cipher.

Here is the first alphabet. Remember that each letter of this alphabet corresponds to an initial letter of a word; so, the frequencies are shifted toward letters that are more frequent initial letters. Recall that when we are looking for a Caesar cipher we usually look for frequency peaks at `aeinorst.`

`a _ _ _ e _ _ _ i _ _ _ _ n o _ _ r s t _ _ _ _ _ _`

But `enr` are not typically used as initial letters; so, we should be looking for frequency peaks corresponding to `aiost`.

`a _ _ _ _ _ _ _ i _ _ _ _ _ o _ _ _ s t _ _ _ _ _ _`

```
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 = *
```

What is the shift?

For the second alphabet and subsequent alphabets, we should be looking for the more usual frequency peaks at

```
a _ _ _ e _ _ _ i _ _ _ _ n o _ _ r s t _ _ _ _ _ _
```

Here is the second alphabet.

```
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 =
```

What is the shift?

The third alphabet:

```
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 =
```

What is the shift?

The fourth alphabet:

```
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 = *******
```

What is the shift?

The fifth alphabet:

```
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 = ***************
```

What is the shift?

The sixth alphabet:

```
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 = ******
```

What is the shift?

The seventh alphabet:

```
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 = *********
```

What is the shift?

The eight alphabet:

```
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 = *********
```

What is the shift?

The ninth alphabet:

```
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 = *
```

What is the shift?

The tenth alphabet:

```
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 = **********
```

What is the shift?

The eleventh alphabet:

```
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 = ****************
```

What is the shift?

The twelfth alphabet:

```
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 = *
```

What is the shift?

The thirteenth alphabet:

```
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 =
```

What is the shift?

The fourteenth alphabet:

```
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 =
```

What is the shift?

The fifteenth alphabet:

```
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 = ******
```

What is the shift?

The key for the first 15 letters of each message was: `OQPMV SVKUI GBEJH`.

The key string can only be used once! If it is reused for subsequent messages, there is the possibility of cryptanalysis.

## VENONA

Although the fact was known only to a few, a small group of codebreakers had in fact been working on Russian code problems during [World War II]. In 1943, American intelligence began to worry about a possible alliance between Nazi Germany and Russia as part of a comprehensive peace deal. Such a merger would have been a nightmare for the Allies. As a result, a few Army cryptanalysts were pulled away from work on German systems

and assigned to a highly secret new unit with the goal of attempting to solve the enormously complex Soviet codes and ciphers.

Since 1939, thousands of encrypted Soviet messages, sent between Moscow and Washington had been acquired from Western Union and other commercial telegraph companies. A major break occurred when it was discovered that identical code groups turned up in seven pairs of messages. To find even a single pair was a billion-to-one shot. Army codebreakers had discovered a "bust," an error or anomaly that opens a crack into the cipher system. Such a bust might be caused, for example by a malfunction in a random-number generator. This bust, however, was caused by the Soviets reusing pages from one-time pads – the violation of a cardinal cryptographic rule. One-time pads had become two-time pads. Cecil Phillips, a former senior NSA official, played a key role in the early Soviet-watching program. "For a few months in early 1942," he said, "a time of great strain on the Soviet regime, the KGB's cryptographic center in the Soviet Union for some unknown reason printed duplicate copies of the 'key' on more than 35,000 pages … . Thus, two sets of the ostensibly unique one-time pad page sets were manufactured."

The decision by the Soviet codemakers to duplicate the pages was likely the result of a sudden shortage of one-time pads, a result of Hitler's invasion of Russia in June 1941. To quickly fill the enormous demand for the pads, Russian cryptographers likely chose the easiest course: carbon paper. Suddenly production was doubled while, it was reasoned, security was diminished only slightly.

Phillips estimated that between 1942 and 1948, when the last one-time pad was used, more that 1.5 million messages were transmitted to Soviet trade and diplomatic posts around the world. Of those, American codebreakers obtained about a million, 30,000 of which had been enciphered with the duplicate pages. But despite the bust, days and weeks of frustrating work were required to squeeze out a clear-text message from a cipher text. Even then, usually the most they would have was a long, out-of-date message concerning such things as shipping schedules of the Soviet Purchasing Commission.

For more than thirty years the codebreakers worked on those messages. By the time the file drawer was closed for the last time, in 1980, they had managed to read portions of more than 2,900 Soviet diplomatic telegrams

sent between 1940 and 1948. Codenamed Venona, the program was one of the most successful in NSA's history. It played a major role in breaking up key Soviet espionage networks in the United States during the postwar period, including networks aimed at the secrets of the atomic bomb. Bamford, James, *The Puzzle Palace: Inside the National Security Agency, America'a Most Secret Intelligence Organization*, Penguin Books, 1983 (< Houghton Mifflin Company, 1982).

[From NSA's *Introductory History of VENONA and Guide to the Translations*] These messages disclose some of the clandestine activities of Julius and Ethel Rosenberg, Harry Gold, Klaus Fuchs, David and Ruth Greenglass, and others such as the spy known by the covername MLAD or the equally important, but still unidentified, PERS. The role played by the person covernamed VEKSEL remains uncertain but troubling. A number of other covernames of persons associated with atomic bomb espionage remain unidentified to this day.

## Spy Stations

Perhaps, one way the random key strings are distributed to spies is by radio. Several shortwave radio stations feature people reading long strings of numbers at regular times. One suspicion is that these stations are transmitting messages and random key strings to spies in the field. The stations are called "Spy Number Stations." Lists of the stations are available on the internet.

Exercises

1. First, convert the letters of the word `Friedman` to numbers using the scheme

```
a    b    c    d    . . .    z
01   02   03   04             26
```

Use the first 16 digits of the third row of the list of digits in the expansion of pi as additives to encipher the plaintext string of numbers.

2. The string of numbers

```
66351051497943
```

was obtained by converting letters to numbers using the scheme in exercise 1 and then using the random key string

```
58209749445923
```

as additives. Decipher the message.

3. Use the Baudot code to convert the word `Venona` to a binary string. Then use the random key string

```
11001101000011100011111100001
```

to encipher the message.

4. The binary string

```
11000101101011000110000011001O
```

was obtained by converting a word to a binary string using the Baudot code and then using the binary key string

```
10111001101110000000110011010 1
```

as additives. Decipher the message.

5. Assume that two messages are converted to Baudot code and the same random key string is used to encipher each of them. What is the result of adding together the two ciphertext messages? Could a cryptanalyst exploit this?