Spring 2015
Chris Christensen
Cryptology Notes

Extending Keys

The Vigenère cipher is an example of a periodic cipher. For the Vigenère cipher, the period comes from repetition of the keyword; the cipher rotates among a small number of Caesar ciphers – the rotation is described by the letters of the keyword and the period is the length of the keyword.

The Vigenère cipher, by rotating among several alphabets, destroys the value of single-letter frequency for cryptanalysis, but patterns still remain because the cipher is periodic.

We have considered several periodic ciphers in addition to the Vigenère; e.g., the Gronsfeld cipher, the Beaufort cipher, and the variant Beaufort. We also, at least in exercises, considered Vigenère ciphers having a square that consisted of alphabets other than the 26 Caesar cipher alphabets. For the remainder of this discussion of some problems with periodic ciphers, we will focus on the Vigenère cipher (with the usual square).

How can we improve its security?

The first step in cryptanalysis of a periodic cipher is to discover its period. To do this, we have used several attacks: we used the Kasiski attack, we aligned ciphertext messages in such a way as to produce many "doubles," and we used brute force to separate alphabets until we have alphabets that are monoalphabetic. Each of these attacks provides information about the length of the key – the length of the period. Once that has been discovered, breaking a Vigenère cipher becomes fairly easy – provided each of the separated alphabets contains enough ciphertext letters.

Let's say we have a plaintext message of 600 characters. If our keyword has length 5, to encrypt we will rotate among five Caesar ciphers. 600/5 = 120 characters will be encrypted with each alphabet. If someone attacks our cipher and is able to determine the length of the keyword, they can strip the ciphertext into five alphabets of 120 characters each. Because each cipher is a Caesar cipher, they can probably spot the shift from the frequencies of the 120 letters. So, the longer the length of the keyword, the more secure the cipher. A keyword of length 10 would give 10 alphabets of length 60; the shift could probably be determined. A keyword (or phrase) of length 20 would result in 20 alphabets of length 30; the shift would be harder to spot. A keyword or phrase of length 60 would result in 60 alphabets of length 10. … A keyword or phrase of length 600 (the length of the plaintext message) would result in 600 alphabets of length 1; that would clearly be secure. So, one way to strengthen a Vigenère cipher is to use a longer keyword. A keyword equal in length to the message would be ideal.

A problem, of course, would be remembering the keyphrase. We could write it down, but then we have the possibility of its being stolen. To create a more secure cipher, we would create problems of key security.

Sometimes people specify long key phrases by portions of a page of a common book. A person who wanted to steal the key might not recognize the book as the key (especially if it were in a collection of books), but it might be possible to attack this cipher by using the frequencies in the key – the frequencies of plaintext English.

The cipher described in the paragraph above is called a running key cipher. We will explore an attack on running key ciphers in a later section.

In this section we will discuss two ways to extend memorable keys. In the next section we will consider "infinitely long" keys.

Re-encryption of Vigenère Cipher

One way to extend memorable keys is to re-encrypt. For example, enciphering a plaintext message with a Vigenère cipher using the keyword `notices` (length is 7) and then re-encrypting it using a Vigenère cipher with the keyword `dream` (length is 5) is equivalent to enciphering it using a Vigenère cipher with the 35-letter keystring shown on the last line below:

```
noticesnoticesnoticesnoticesnotices
dreamdreamdreamdreamdreamdreamdream
rgyjpikspgmujtaslndrwftuvgwxobxahff
```

Notice that the length of the equivalent keystring is the least common multiple of 7 and 5: LCM[7, 5] = 35. After 35 letters the `n` in `notices` and the `d` in `dreams` are – for the first time -- again aligned and the sequence repeats.

Using for keys `word` and `cipher` would result in a keystring of length LCM[4, 6] = 12.

```
wordwordword
ciphercipher
```

Using for keys `notices`, `dream`, and then `plaintext` result in a keystring of length LCM[7, 5, 9] = 315.

Composing ciphers could be used to effectively generate a long key.

Autokey Ciphers

Another way to extend memorable keys is called autokey.

Here are two such schemes. One uses a keyword and extends the keyword by plaintext, and the other uses a keyword and extends the keyword by ciphertext.

Extending by Plaintext

The keyword is `norse`.

Key	`norsethemostfamousrotormachinewasenigma`
Plaintext	`themostfamousrotormachinewasengima`
Ciphertext	`GVVMESLAJMAGNXRAHIJDOVVZZEYAAREGEE`

Extending by Ciphertext

The keyword is `norse`.

Key	`norsegvvmeslajmagnxrahijdovvzzeyaaregee`
Plaintext	`themostfamousrotormachinewasengima`
Ciphertext	`GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM`

Cryptanalysis of Autokey

Extending by ciphertext is easier to break than extending by plaintext.

We know that the ciphertext enciphers the message after the keyword. So, we just keep sliding the ciphertext along the ciphertext message and deciphering until we get plaintext.

```
Presumed key    GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Ciphertext      GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Plaintext       aaaaa

Presumed key     GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Ciphertext      GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Plaintext        pajog

Presumed key      GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Ciphertext      GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Plaintext         pjn

Presumed key       GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Ciphertext      GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Plaintext          mdt

Presumed key        GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Ciphertext      GVVESYOAEEMISVSFWJIUHDRIWDDJMJLJVM
Plaintext           stfam
```

The last has some hope (and we know that it is the correct choice). We might have to do some guessing for the beginning of the message.

Extending by plaintext is a bit harder to break.

Let’s say that we know the length of the keyword. Say, it’s 5. Look at encryption:

```
Key  n      t      s      o      t      c      w
p/t  t      s      o      t      c      w      i
C/T  GVVME  SLAJM  AGNXR  AHIJD  OVVZZ  EYAAR  EGEE
```

Just reverse that process.

```
C/T  GVVME  SLAJM  AGNXR  AHIJD  OVVZZ  EYAAR  EGEE
Key  n      t      s      o      t      c      w
p/t  t      s      o      t      c      w      i
```

We decipher every fifth letter in the string by using as the key the previously deciphered letter. How did we know that the first letter of the key was an `n`? We didn’t; we just tried all 26 possible letters until we found one that “worked.” How do we know it “worked?” One possibility is to check the collection of presumed plaintext to see whether they (probably) come from a single alphabet; i.e., check the index of coincidence. How did we know that the length of the keyword was 5? We just tried all possibilities until we got one that worked – 1, 2, 3, 4, 5, … .

A lot of trial and error, but it works.

History of Autokey Ciphers

Two men are noted a having developed early methods of autokey encryption.

> The inventor of the first and imperfect system was Girolamo Cardano (1501 – 1576), Milanese physician and mathematician who is known today chiefly as one of the first popularizers of science and as author of the world's first text on the theory of probability.
>
> Cardano employed plaintext as a key to encipher itself; starting the key over from the beginning with each plaintext word. (*The Codebreakers,* David Kahn)

Here is an example using Cardano's scheme and our usual Vigenère square and method. Remember that the key repeats beginning with each start of each plaintext word.

Key	`thethemthemosthemothemostthethemos`
Plaintext	`themostfamousrotormachinewasengima`
Ciphertext	`MOIFVWFYHQAIKKVXAFFHGUWFXPHWXUKUAS`

> But, while the autokey was a brilliant idea, Cardano formulated it defectively … the decipherer is in exactly the same position as the cryptanalyst in trying to figure out the first plaintext word. This, once obtained, unlocks the rest of the message. (*The Codebreakers,* David Kahn)

The two autokey methods that we gave above – extending by plaintext and extending by ciphertext are due to Vigenère (1523 – 1596). Although Vigenère used only one keyletter to begin each process.

> [Vigenère] perfected Cardano's [autokey cipher] in two ways. First, [he] provided a priming key. This consisted of a single letter, known to both encipherer and decipherer, with which the decipherer could decipher the first cryptogram letter and so get a start on his work. … Secondly, Vigenère, unlike Cardano, did not recommence his key with each plaintext word, which is a weakness, but kept it running continuously. (*The Codebreakers,* David Kahn)

Exercise

Cryptanalyze the following ciphetext that was encrypted using autokey extended by ciphertext.

```
VVHTD  VTMTL  MVUIS  QMQIK  TQLMV  HFPPW  FHWPN
QLOLU  ULHDN  IYL
```