# Classical Cryptography, Part I

First, why should we cover old cryptography in a class on computer security? The main reason is that in order to understand modern cryptography, it helps to understand the older stuff it is built on. It's also kind of fun, and sometimes people do still use it.

## Basics of cryptography

The idea of cryptography is to disguise a message so that only the intended recipient can understand it. Probably the simplest type of encryption is the *Caesar shift*. In it, we shift all the letters of the message down by a certain number of letters. For instance, let's use a shift of 1 letter on the word SECRET. This gives TFDSDU. Each letter of SECRET is replaced with the next letter of the alphabet.

To read the message, the recipient will shift each letter backwards by 1 letter. Anyone who intercepts the message will see TFDSDU, which looks like a bunch of gobbledegook. They wouldn't know what the original message is unless they happened to guess that the Caesar shift was used, in which case they could break the cipher by trying various shifts until they get a readable message.

Let's introduce some vocabulary: The original message is called the *plaintext*. That message is *encrypted* to produce the *ciphertext*. A *key* is used in the encryption process in a similar way to how a real-life key is used to lock and unlock doors. For the Caesar shift, the key is the number of letters to shift by. The recipient *decrypts* the message to recover the plaintext. A *man in the middle* can intercept the message and use a *brute force* search of all possible keys to try to decrypt the message.

The Caesar shift can be used with 25 possible keys, corresponding to how many letters to shift by. A shift by 13 is sometimes called ROT-13. Note that you rotate around the alphabet if shifting takes you past the letter Z. For instance, shifting the letter Y by 2 places wraps back around to A.

## Substitution ciphers

The main weakness of the Caesar shift is that there are only 25 possible shifts. You could easily try those 25 shifts by hand, and a computer could check all of them really quickly. An improvement on the Caesar shift is the more general *substitution cipher*.

In the substitution cipher, each letter of the alphabet is replaced with another letter. For example, in the key shown below, every A in a message will be replaced with N, every B in the message will be replaced with J, etc.

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | J | G | V | F | K | A | D | H | C | S | T | E | L | X | I | U | O | P | B | Z | R | M | W | Q | Y |

So the message SECRET would be encrypted into PFGOFB. Decryption works similarly, just doing things in reverse. For a very short message like this, the substitution cipher is surprisingly secure. In fact, the number of ways to rearrange the alphabet is 26! (26 factorial), which is 403,291,461,126,605,635,584,000,000. Only large nation states have enough computing power to do a brute force search of all possible keys.

But there is a fatal weakness to the substitution cipher if it is used with a long enough plaintext message. Certain English letters show up more often than others. For instance, in ordinary English text, about 11% of letters are E, while only about 0.1% are Q. Since every copy of the letter E is replaced by the same letter, if we see a certain letter that shows up around 11% of the time in the ciphertext, then there's a good chance it's what E was encrypted to. We can also look at pairs of letters since combos like TH or ST are much more common than things like TN or SJ. This process is called *frequency analysis*, and it's an important tool for breaking many classical ciphers. In fact, substitution ciphers are a common puzzle in newspapers and puzzle books, where they are called cryptograms.

Since frequency analysis relies on statistics of letter frequencies, we often need at least few sentences of material to get decent statistics. For instance, I encrypted the short sentence "A BUZZY FLY ZIPS AROUND A ZEBRA" with

a substitution cipher to get “G TQUUZ ENZ UPSK GVRQAH G UMTVGAND”. I fed that into an online substitution cipher cracker, which returned “T CURRE SPE RING THOULD T RACHT”. There’s just not enough information here for that solver to have a chance. It’s a good exercise to try to program a substitution cipher cracker. It’s very doable, but there are a few tricks to it.

The substitution cipher is very old, and people have come up with a number of interesting tricks to make it harder to crack with frequency analysis. One simple approach is to introduce deliberate misspellings. Another is to use multiple substitutions for each letter in such a way to match its frequency. For instance, the letter E may have a dozen different symbols it could be replaced with, while less common letters would have a smaller sets of replacements.

## The Vigenère cipher

The *Vigenère cipher* was the state of the art in cryptography from the 1500s until the 1800s. Many people thought it was unbreakable. Some people did know how to break it and kept their knowledge to themselves, but finally in the 1800s a solution was published. It’s probably easiest to demonstrate it with an example. Our plaintext message will be THISISASECRET. The Vigenère cipher’s key is usually a word or short phrase. We’ll use the word ACE. We then line up the plaintext with a repeated copies of the key, like below:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| plaintext | T | H | I | S | I | S | A | S | E | C | R | E | T |
| key | A | C | E | A | C | E | A | C | E | A | C | E | A |
| ciphertext | T | J | N | S | K | X | A | U | J | C | T | J | J |

The letters of the key tell us how much to shift by. An A is a shift of 0, a C is a shift of 2, and an E is a shift of 4. So for the first letter of the message, T, we have an A in the key below it, so we do a shift of 0, which leaves it at T. For the second letter of the message, H, the letter below it in the key row is C, so we shift H by 2 letters to get J. The Vigenère cipher is essentially several different Caesar shifts in a repeating pattern.

This cipher is easy to do by hand, especially with the help of tables or specially built cipher wheels. The trick to cracking it relies on making an educated guess as to the key length with the help of frequency analysis and then breaking the individual Caesar shifts, also using frequency analysis. Again, this is a really nice exercise to try to program, though it’s a little tricky.

## Running key cipher

The main weakness of the Vigenère cipher is that the key repeats. This was a necessary part of its design because the cipher was meant to be used by humans, and shorter keys are easier to remember. If, instead of using a repeating key, we use a key that is as long as the message itself, then we have something called a *running key cipher*. The key could be, for instance, part of the text of a famous document.

A running key cipher is considerably harder to break than the Vigenère cipher. However, if the key is actual English text, then there are ways to break it. Typically this is done by making a guess for what a certain part of the plaintext might actually be, working out what the corresponding part of the key must be, and if you get something readable for that part of the key, then your guess for the plaintext is likely right. You could then extend what you know about the key to maybe work out the whole key if it came from a well-known source. Frequency analysis can also be used.

## One-time pad

The *one-time pad* is like a running key cipher except that we are more careful about the key. In particular, the key needs to satisfy three properties:

1. It must be as long as the plaintext message itself.
2. It must be totally random.

3. It must never be used to encrypt more than one message.

The last property is where the "one time" in the name comes from. A different key must be used for each message. The one-time pad has one remarkable property that no other encryption technique has: *it is unbreakable*. Yes, it is the only unbreakable cipher. It's not hard to both prove that it is unbreakable and to prove that any other cipher can theoretically be broken.

Here is a quick example of the cipher in action. Suppose our message is SECRET. We then must generate a perfectly random key as long as the message itself. I generated this key: ZVNNPU. These correspond to shifts of 25, 21, 13, 13, 15, and 20. We then encrypt in a very similar way to Vigenère to get the following:

| | | |
|---|---|---|
| plaintext | S E C R E T | |
| key | Z V N N P U | (shifts 25, 21, 13, 13, 15, 20) |
| ciphertext | R Z P E T N | |

So the ciphertext is RZPETN. That is, S gets shifted by 25 into R (which is a shift of 1 letter backwards), E gets shifted by 21 into Z, etc.

**Problems with the one-time pad** You might be wondering if there is a provably unbreakable system like this, why it isn't used more often. The problem is that it is hard to use well. First, a key as long as the message itself can take up a lot of space. If you are encrypting gigabytes of data, then your key needs to also be gigabytes long.

Second, that key needs to be perfectly random. Generating true random numbers is slow and time-consuming. The numbers generating by most programming languages are *pseudorandom*, which means they look random, but they are not truly random since they are generated by a predictable mathematical process. Generating true random numbers usually relies on using some physical phenomenon like atmospheric noise, and that can be slow.

Third, the key must never be reused. This is a difficult thing for humans to stick to in practice. The one-time pad was used for a time by Russian spies. They were given pads of random keys and they were supposed to use one sheet of paper from the pad for each message. But often they reused those sheets, and that made things breakable.

A fourth problem is *key exchange*. For two parties to use the one-time pad, they both have to have a copy of the key. This is fine for two people that can meet in person, but it's trickier to do remotely.

Despite these issues, the one-time pad is actually in use in applications that require the highest level of security. It was apparently in use to encrypt phone communications between the White House and the Kremlin.

**Why it's bad to reuse keys** Suppose we intercept two ciphertexts encrypted with the one-time pad. Suppose the first six characters of each are YLZUIN and VLDRRM. Suppose further that we know the first three characters of the second plaintext are THE. Below are tables showing what we have. Remember that the key was reused, so it's the same in both.

| | |
|---|---|
| plaintext 1 | ? ? ? ? ? ? |
| key | ? ? ? ? ? ? |
| ciphertext 1 | Y L Z U I N |

| | |
|---|---|
| plaintext 2 | T H E ? ? ? |
| key | ? ? ? ? ? ? |
| ciphertext 2 | V L D R R M |

Since we know the first three characters of both the plaintext and ciphertext for the second message, we can work out what the shifts must be. T to V is a shift of 2, H to L is a shift of 4, and E to D is a shift of 25. Therefore, the first three key values are 2, 4, 25 Since the first message was encrypted with the same key, we can use it to decrypt the first three characters of that message to get WHA.

The one fishy part of this is how we would know the first three characters of the second plaintext. There are two ways. First, a lot of messages begin or end with something predictable. This was one of the ways the Allies were able to break the German's Enigma machine codes in World War II. The Germans often sent out encrypted weather reports that had very predicable content.

A second way is something called *crib dragging*. This is where you make a guess as to what parts of one of the

plaintexts might say. Then work out the key as above and decrypt the other plaintext to see if the result looks like real English. The dragging part of it is where you take common words like THE, THERE, etc. and try them first at the start of the message, then starting at character 2, then starting at character 3, etc., until you get something readable in the other plaintext.

**Why it's unbreakable** Imagine we have intercepted the ciphertext DDU encrypted with a one-time pad. Note that the key $(1, 3, 1)$ decrypts this into CAT. Does that mean the original plaintext is CAT? Not necessarily. The key $(0, 11, 12)$ decrypts it into DOG, and the key $(14, 3, 1)$ decrypts it into RAT. Since the keys in a one-time pad are generated perfectly randomly, any key is as likely as any other key, so all of these decryptions are equally likely.

That is, we have absolutely no foothold by which to start breaking this code. However, if there is any bias at all in the random number generator, then we can exploit that using statistics to break the code. That's why the random numbers need to be perfectly random.