MAT 302: LECTURE SUMMARY

Last class we discussed two classical ciphers, both of which turned out to be rather insecure (as evidenced by your cracking them manually during lecture):

- The Scytale cipher
- The Caesar (aka shift) cipher

We began today's lecture by writing the latter down in mathematical notation. To this end, it is convenient to use the set $\mathcal{A} = \{0, 1, 2, \ldots, 25\}$ to represent the English alphabet, i.e. 0 represents A, 1 represents B, etc.

The Caesar Cipher. *The Caesar cipher (or shift cipher) consists of a key $k \in \mathbb{Z}$, an encryption function*

$$\begin{aligned} e_k : \mathcal{A} &\longrightarrow \mathcal{A} \\ x &\longmapsto x + k \pmod{26} \end{aligned}$$

and a decryption function

$$\begin{aligned} d_k : \mathcal{A} &\longrightarrow \mathcal{A} \\ x &\longmapsto x - k \pmod{26}. \end{aligned}$$

The Caesar cipher is insecure because the space of all possible keys is rather small (how many are there?), so it is a trivial matter for Oscar to check all possible keys. We next discussed a variant of this, whose key space is somewhat larger: the affine cipher.

The Affine Cipher. *The affine cipher consists of a key $(a, b) \in \mathbb{Z}^2$, an encryption function*

$$\begin{aligned} e_{(a,b)} : \mathcal{A} &\longrightarrow \mathcal{A} \\ x &\longmapsto ax + b \pmod{26} \end{aligned}$$

and a decryption function

$$\begin{aligned} d_{(a,b)} : \mathcal{A} &\longrightarrow \mathcal{A} \\ x &\longmapsto a^{-1}(x - b) \pmod{26} \end{aligned}$$

Actually, as stated this isn't well-defined: a^{-1} doesn't always exist $\pmod{26}$. Recall that a^{-1} is the element of $\mathcal{A}$ satisfying

$$a^{-1}a \equiv 1 \pmod{26}.$$

For example, 0 has no (multiplicative) inverse $\pmod{26}$. Less trivially, 2 has no multiplicative inverse. (To check this, one could test whether $2a \equiv 1 \pmod{26}$ for every $a \in \mathcal{A}$, but there are much faster ways to see that 2 has no inverse... can you come up with one?) By contrast, we figured out that $1^{-1} = 1$, and $3^{-1} = 9$.

Date: January 6th, 2011.

This brought us to the subject of group theory, which you hypothetically learned in a previous course. We will review what we need from group theory more intensely in the future; for now we contented ourselves with a couple of basic examples of groups. We started with the group $\mathbb{Z}_{26}$ (more commonly denoted $\mathbb{Z}/26\mathbb{Z}$), which is just the set $\mathcal{A}$ together with the binary operation of addition $\pmod{26}$. Although this binary operation is very similar to the ordinary addition we all know and love, it is better to think of it as a totally unrelated operation (since it happens in a different group). To emphasize this, I will temporarily denote this operation as $\oplus$. For example, $19 \oplus 10 = 3$ in $\mathbb{Z}_{26}$.

What makes $\mathbb{Z}_{26}$ a group? And why should we care? We will postpone the answer to the latter question until next week. In this lecture, we focused on a technical answer to the first question:

(1) **Closure**: $a \oplus b \in \mathbb{Z}_{26}$ for any $a, b \in \mathbb{Z}_{26}$
(2) **Associativity**: $a \oplus b \oplus c$ is unambiguously defined, for any $a, b, c \in \mathbb{Z}_{26}$
(3) **Identity**: The element $0 \in \mathbb{Z}_{26}$ is called the *identity* because it has the special property that $0 \oplus a = a \oplus 0 = a$ for all $a \in \mathbb{Z}_{26}$
(4) **Inverses**: Every $a \in \mathbb{Z}_{26}$ has an *inverse* in the group, traditionally denoted a^{-1}, such that $a \oplus a^{-1} = a^{-1} \oplus a = 0$.

Associativity is an important but somewhat subtle point. Why wouldn't $a \oplus b \oplus c$ be unambiguously defined? The issue is that $\oplus$ is a *binary* operation – it's a way of combining two elements, not three! This makes $a \oplus b \oplus c$ a questionable quantity, and it's somewhat miraculous that it's unambiguous. By contrast, the binary operation $\div$ is not associative: $4 \div 2 \div 2$ might be 4 or 1, depending on how you group the terms. For this reason, associativity is often presented as the following technical condition:

$$(a \oplus b) \oplus c = a \oplus (b \oplus c) \quad \text{holds for all} \quad a, b, c \in \mathbb{Z}_{26}.$$

Before moving on to our second example of a group, let's take a moment to discuss inverses. You might object to the notation a^{-1}. Wouldn't $-a$ be a more natural notation for the inverse of a? After all, $a + (-a) = (-a) + a = 0$. The merits of one notation over the other are debatable, but I prefer a^{-1} for the same reason I prefer $\oplus$ over $+$: it reinforces that we are working in a totally different universe from that of $\mathbb{Z}$ and ordinary addition. After all, -5 isn't even an element in $\mathbb{Z}_{26}$! Of course, there are advantages to thinking of -5 as just another name for the element 21, but I think this is secondary to becoming fully conscious that we're not in the comfortable universe of $\mathbb{Z}$.

We next considered the group $\mathbb{Z}_{26}^{\times}$, the multiplicative group $\pmod{26}$. This group is similar to $\mathbb{Z}_{26}$, except that rather than the additive operation $\oplus$ we use the multiplicative operation $\otimes$, multiplication $\pmod{26}$. For example, $5 \otimes 7 = 9$ in $\mathbb{Z}_{26}^{\times}$. However, $\mathbb{Z}_{26}^{\times}$ cannot contain all the elements of $\mathcal{A}$, since many do not contain multiplicative inverses (as discussed above). It turns out that

$$\mathbb{Z}_{26}^{\times} = \{a \in \mathcal{A} \,:\, (a, 26) = 1\}$$

(recall that (m, n) is the greatest common divisor of m and n). By counting directly, we figured out that there were 12 elements in $\mathbb{Z}_{26}^{\times}$. This can be written in shorthand in the following way:

$$\left|\mathbb{Z}_{26}^{\times}\right| = 12.$$

Let's return to the affine cipher for a moment. How big is the key space? Well, a must be an element of $\mathbb{Z}_{26}^{\times}$, whereas b can be anything in $\mathbb{Z}_{26}$. This leads to a total of 12×26 different possible keys (a, b). This seems like a lot, but in fact can be checked by a household PC in under a second. (Even by hand it wouldn't take particularly long to check every possibility.)

What if we're working with a language other than English? Or we add symbols to the English alphabet? Let's say we're working with an alphabet of n letters. Then again we have the additive group $\mathbb{Z}_n$, and the multiplicative group $\mathbb{Z}_n^{\times}$. It's clear that $\left|\mathbb{Z}_n\right| = n$; what's not at all clear is the size of $\mathbb{Z}_n^{\times}$. This is traditionally called Euler totient function, $\varphi(n)$, i.e.

$$\varphi(n) = \left|\mathbb{Z}_n^{\times}\right|$$

This function has all sorts of nice properties. It is clear, for example, that $\varphi(p) = p - 1$ for any prime p. In class you then came up with a proof that $\varphi(p^2) = p(p-1)$, by writing all the integers between 1 and p^2 in the following row:

$$\begin{array}{cccc} 1 & 2 & 3 & \cdots & p \\ p+1 & p+2 & p+3 & \cdots & 2p \\ 2p+1 & 2p+2 & 2p+3 & \cdots & 3p \\ \vdots & \vdots & \vdots & & \vdots \\ (p-1)p+1 & (p-1)p+2 & (p-1)p+3 & \cdots & p^2 \end{array}$$

In every row, the only number *not* relatively prime to p^2 is in the rightmost column; this implies that all the other numbers in the table are coprime to p^2, giving the formula above. The same proof also shows that $\varphi(p^n) = p^{n-1}(p-1)$ for any prime p and positive integer n.

I also mentioned (but didn't prove) that φ is a multiplicative function, i.e.

$$\varphi(mn) = \varphi(m)\varphi(n)$$

whenever $(m, n) = 1$. We shall explore this in the future, but for now, we apply it to calculate $\varphi(26)$:

$$\varphi(26) = \varphi(2 \times 13) = \varphi(2) \times \varphi(13) = 12.$$

This is much quicker than counting!

Both the Caesar and affine ciphers involved a permutation of the alphabet $\mathcal{A}$. A natural generalization of both of these is

The Substitution Cipher. *Suppose $\mathcal{A}$ is an alphabet, and let $S(\mathcal{A})$ denote the set of of all bijections from $\mathcal{A}$ to itself. The substitution cipher consists of a key $\sigma \in S(\mathcal{A})$, which serves as an encryption function*

$$\begin{aligned} \sigma : \mathcal{A} &\longrightarrow \mathcal{A} \\ x &\longmapsto \sigma(x). \end{aligned}$$

Accordingly, the decryption function is

$$\begin{aligned} \sigma^{-1} : \mathcal{A} &\longrightarrow \mathcal{A} \\ x &\longmapsto \sigma^{-1}(x) \end{aligned}$$

For example, if $\sigma(A) = T$, $\sigma(B) = L$, $\sigma(C) = A$, $\sigma(D) = K$, etc. then the message "A BAD CD" would be encrypted as "T LTK AK ".

How secure is the substitution cipher? The key space has size $\big|S(\mathcal{A})\big|$, which in the case of the English alphabet is 26!. This is quite huge, around $4 \times 10^{26} \approx 2^{88}$. (We approximated this using Stirling's formula, which asserts that

$$N! \sim \sqrt{2\pi N}\Big(\frac{N}{e}\Big)^N.$$

We will prove this formula next week.) How long would it take for a computer to run through all these possibilities? The fastest supercomputer in the world can run at around 2 petaflops, i.e. it can compute 2×10^{15} operations (e.g. addition, subtraction, or multiplication) per second. (FLOPS stands for floating-point operations per second. The prefix 'peta-' means 10^{15}.) However, huge computer clusters can achieve rates on the order of 100 petaflops. This means that it would take well upward of $26!/10^{17} \approx 4 \times 10^9$ seconds for the fastest computing systems to check all possible keys to a given substitution cipher – roughly a century! So, the substitution cipher is secure against the brute-force attack of simply testing every possible key in the key space.

However, there are other approaches to cracking the substitution cipher, rendering it highly insecure. One is to examine letter frequencies. For example, the letter E is the most frequently occurring letter in the English language, followed by T, A, O, I, N, etc. For a long message encrypted using the substitution cipher, one can simply look at the letter frequencies of the encoded message – this gives a hint of what the key might be. Moreover, one can look for short words, vowels, combinations of letters (qu-, th-, etc.), double letters, ... you name it. This turns out to be a robust attack, as you will explore on your first problem set.

In summary, the main weakness of the substitution cipher is that while it disguises the letters in the original message, it does not disguise their statistical properties.

We have now presented four ciphers, each of which are appealing, but none of which are secure. In the rest of the course, we will focus on ciphers which seem to be secure.