MAT 302: LECTURE SUMMARY

We began class by discussing why the algorithms of cryptography are typically described in binary notation. Here were a few of the ideas we came up with:

(1) Computers store information in binary.
(2) Binary provides a standard easy way to write statements down, using a very small number of symbols.
(3) 1 and 0 correspond to True and False, which is convenient.
(4) Arithmetic is easier in binary.

These are all good arguments for using binary in our future discussions. On the other hand, there's an excellent reason not to use binary – we're not used to it! This is true for arithmetic (quick, what's 3^3 base 10? base 2?) but even more so when dealing with text. For example, how does one write the letter A in binary? It turns out there's a standardized system for translating the English alphabet (and other symbols) into binary, with each symbol corresponding to an 8-digit binary number. (The system is called ASCII; each binary digit is called a 'bit'; 8 bits form a 'byte'.) This is convenient, but unhelpful when it comes to developing intuition for an algorithm. For this reason, we will go back and forth between using binary or not throughout the course; most algorithms can be freely translated between different bases.

Take the One Time Pad, for example. Last time the idea was explained in terms of a keystream consisting of 0's and 1's. Personally, I find it easier to think about the OTP as a variation of the Shift cipher, where rather than shifting each letter in the plaintext by the same amount, Alice shifts each letter by a random amount. Bob then decrypts the ciphertext by simply undoing all the shifts.

The One Time Pad is information-theoretically secure – it cannot be broken, even with unlimited resources. Why don't we use it for all of our encryption, then? We came up with several reasons:

- The length of the key is as long as the length of the message.
- It's hard for Alice to communicate the key to Bob.
- If Alice and Bob plan on having multiple communications, they cannot reuse the key.

The resolution of these problem is to generate not truly random numbers (e.g. results of flipping a coin or tossing a die), but to generate a sequence which is random enough that it's hard to crack. The usual approach is to agree on a *starting seed* s_0 and a function $f(x)$, and then generate a sequence s_i recursively by setting $s_{i+1} = f(s_i)$ for each i. For example, the Linear Congruential generator Aaron discussed in the last lecture is the case where one takes

$$f(x) = Ax + B \text{ (mod } m)$$

for some constants A, B, m. Let's see this in action.

Date: January 25th, 2011.

Example (Linear Congruential Generator)**.** *Let* $f(x) = 2x - 1 \pmod 7$*, and use a starting seed of 3. Applying* f *over and over, we generate the sequence* $3, 5, 2, 3, 5, 2, 3, 5, \ldots$

Note that the above sequence is periodic. This isn't a surprise, in hindsight, since as soon as we hit a value we've seen before, we know exactly what f will do to it (f is *deterministic*). And there are only finitely many values f can take $\pmod 7$, so it will have to take the same value twice eventually (after at most 6 iterations). Encrypting a message by adding a periodic key shift is called a polyalphabet cipher. They were long thought to be secure, but an efficient attack was demonstrated already by the mid-19th century; the attack relies on the periodicity of the key. This problem can be resolved by making m (the modulus we reduce by) much larger than the length of the plaintext.

What's the advantage of using Pseudo-RNG in place of a true RNG? The main advantage is that Alice no longer has to transmit the entire key stream to Bob, as in the OTP; she needs only communicate the starting seed s_0 and the transformation $f(x)$. Of course, this is simultaneously a disadvantage, since Oscar now needs much less information to crack the cipher.

We finished the lecture with a discussion of a famous open problem, the $3x + 1$ problem (also known as the Collatz conjecture). This problem, though not directly linked to cryptography, serves as a good demonstration of how little mathematicians currently understand about the behaviour of iterated functions.

The $3x + 1$ **Conjecture** (aka the Collatz conjecture)**.** *Define the function* $f : \mathbb{Z} \to \mathbb{Z}$ *by*

$$f(n) = \begin{cases} 3n + 1 & \textit{if } n \textit{ is odd} \\ \frac{n}{2} & \textit{if } n \textit{ is even.} \end{cases}$$

Then for any positive starting seed, a finite number of applications of f will reach 1. In other words, for each $a \in \mathbb{N}$ *there exists a positive integer* N_a *such that* $f^{N_a}(a) = 1$*, where* $f^1(a) = f(a)$ *and* $f^{i+1}(a) = f\Big(f^i(a)\Big)$*.*

For example, 3 reaches 1 after seven applications of f:

$$3 \xrightarrow{f} 10 \xrightarrow{f} 5 \xrightarrow{f} 16 \xrightarrow{f} 8 \xrightarrow{f} 4 \xrightarrow{f} 2 \xrightarrow{f} 1$$

or in other words, $f^7(3) = 1$. If you prove that this happens for any starting seed, you will become very, very famous![1] For a thorough reference on the subject, check out
`http://www.math.lsa.umich.edu/~lagarias/3x+1.html`

[1] In certain circles.