CS 198 Codebreaking
Fall 2022 Ryan Cottone, Imran Khaliq-Baporia

Note 2

XOR

The XOR operation, sometimes called the exclusive-OR, is a very powerful tool in cryptography. It's special properties lie in the fact that XOR-ing two binary strings does not lose any *information*.

For example "1010" $\oplus$ "1111":

Plaintext:	1010
Key:	1111
Ciphertext:	0101

This can be reversed by XOR-ing the ciphertext with the key again:

Cipher:	0101
Key:	1111
Plaintext:	1010

We can rewrite the equations as follows:

$$\begin{aligned} m \oplus k &= c \\ m \oplus k \oplus k &= c \oplus k \\ m &= c \oplus k \end{aligned}$$

As you can see, the plaintext is recovered in all cases. This property of XOR can be formalized as such:

$$\begin{aligned} &A := \{0,1\} \\ &f : (A,A) \to A \\ &f(X,Y) := X \oplus Y \\ &(\forall x,y \in A)(f(f(x,y),y) = x) \end{aligned}$$

That is to say, XOR-ing a string against a key, then XOR-ing that result against the key again will return the original string. Astute readers might notice this is equivalent to addition $\pmod 2$. Thus, a XOR cipher is equivalent to a substitution cipher with the alphabet defined as 0 and 1. The concepts from Note 1 thus apply, and we can be assured the message is recoverable in all cases.

Critically, however, the inverse only exists *if you know the key*. One could chain an arbitrary amount of XOR with different keys at different steps together, and losing the key to *any one* of those steps would result in the entire message being lost.

Information Theory

However, the phrase "lost message" begs the question, how do we quantify a message being lost? The answer is: a message is considered to be lost when the ciphertext's entropy is equal to the entropy of a random bitstring.

Definition 1: Entropy

Unlike in thermodynamics, the **entropy** of a given information source quantifies the amount of uncertainly in a random variable. While precise definitions are outside the scope of this class, the Shannon entropy (in bits) of a random variable x is given by:

$$H(x) = -\sum_i x_i \log_2(x_i)$$

where x_i represents the probability of event i occuring.

It follows from the definition of entropy that a random variable x with equal probability to be 0 or 1 (think coin flip) has 1 bit of entropy. A weighted coin would have less entropy, since we can be more certain of the outcome than of a truly random coin. This is why the entropy is higher – there is more uncertainty in the outcome.

Returning to the original question, one can see that a random bit is exactly like a coin flip, and thus has 2 equal outcomes. For a n-bit random message, there is n bits of entropy – and 2^n possible messages. Notably, the messages we analyzed in the previous note *do not* have n bits of entropy, as their key is repeated. The repeated key provides us with valuable information and allows for frequency analysis attacks. However, if the key was just as long as the message (and completely random), it is **mathematically impossible** to break the ciphertext! This holds for base 2 and other bases (one can simply convert back and forth between base 2 for proof). This type of encryption is called a one-time pad.

One-Time Pad

Definition 2: One-time Pad

A **one-time pad** is a substitution cipher utitlizing a perfectly random key at least as long as the message to be encrypted. Such a cipher is information-theoretically secure, also known as *perfect secrecy*.

Definition 3: Perfect Secrecy

A cipher that is **perfectly secret** is one in which the ciphertext provides zero information about the plaintext, such that the entropy of the ciphertext is equivalent to the entropy of a perfectly random string. An adversary with unlimited computational resources and time would not be able to crack it.

Let's consider what makes a one-time pad perfectly secret. Consider the very simple example of a single character long ciphertext – "1". Given this, could you determine whether the original bit was 0 or 1 (without knowing the key)? The nature of modular arithmetic makes it equally likely to have been 0 or 1. That is to say, the entropy of the ciphertext is equal to the entropy of a random bit – 1.

Extending this to an arbitrary number of bits does not change the security – since the key does not repeat and is truly random, no bit gains any information from the others. If you revealed 99% of the key, you would still have no more information about the last bit.

What is even more interesting about perfect secrecy and one-time pads is that a perfectly secure ciphertext can be "decrypted" into *anything* the user wants it to be. Consider the example ciphertext, which is the integer 40 converted to binary:

$$00101000$$

Say the intended message was 87. In that case, one can find the key by XOR-ing the ciphertext and the plaintext to receive: 127. We can then verify that this key is correct: $87 \oplus 127 = 40$.

Now, say the intended message was 12. We do the same thing and get a key of 36. Verify: $12 \oplus 36 = 40$.

Consider a more concrete example – a encrypted message from a general to his lieutenants was intercepted by the enemy. Using their powerful computers, they find two promising examples – "Attack at noon!" and "Attack at dawn!", with no way of discerning between the two (or the billions of other options).

This way, we can't even analyze the possible outcomes to look for "correct" sounding outcomes – because it could be *any combination of letters* with length n. This is why it is **impossible**, not just hard, for an attacker to recover the message. It's because the intended message has the same probability of being "correct" as every other possible message.

This is not exclusive to XOR – it would work with a basic substitution cipher like Vignere covered in Note 1 - assuming the key was equally as long as the message and perfectly random. XOR is simply popular for use in computers due it its speed while preserving critical properties of modular arithmetic.

Drawbacks and Pitfalls

You might be wondering, why is the field of cryptography such a big deal when this perfect cipher exists? Unfortunately, there are some severe drawbacks to using one-time pads. Notably:

- Setup – the sender and receiver must have met in person to trade the key between themselves beforehand.
- True Randomness – generating a perfectly random key is rather difficult (this will be expanded upon in a later note).
- Convenience – you need a **lot** of these random bits to convey messages of appreciable length.
- Reusability – the key can never be re-used, or else the one-time pad loses perfect secrecy.
- Lack of Message Authentication – an attacker can replace the ciphertext with any encrypted message they desire if they know the ciphertext and plaintext.

Due to this, there is almost no use of one-time pad in consumer contexts. Despite these drawbacks, one-time pads have been in use throughout modern history by spy agencies and anything requiring utmost secrecy.

Contributors:

- Ryan Cottone