Alin Tomescu, CSE408
Tuesday, February 1st, Lecture #1

What is security?

Let us introduce *Alice* and *Bob*, two people who want to communicate "securely". Also, let us introduce *Eve*, as their evil enemy who will always eavesdrop on Alice and Bob's conversations.

Since Eve is always listening, sending a message between Alice and Bob is risky business. Encryption will allow Alice and Bob to send messages between each other securely.

Alice will never send the message m to Bob, because Eve can completely understand it. Alice will send the encrypted version of m, called $E(m)$, which only Bob can decrypt, getting back the actual message m.

From this perspective, we can think of "good security" in the following way:

If we give Eve the encrypted message $E(m)$, then she gains no information whatsoever about m.

The following figure illustrates this communication scenario:

- Alice and Bob share the same secret key together
- Alice and Bob both know how to use an encryption algorithm
- Alice and Bob are communicating over an unsecure channel, that Eve can listen to
- Eve sits in the middle attempting to decipher the encrypted messages
    - With good security, Eve is unable to do so.

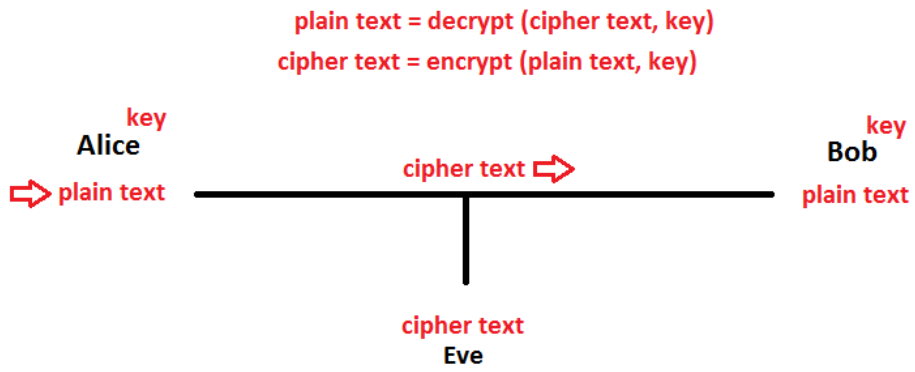

Figure 1: Alice and Bob communicating while Eve is listening

One-time pad analysis

The **one-time pad** (OTP) is an encryption algorithm proven to be impossible to crack if used correctly.

How does it work?

With the **one-time-pad**, each bit (or character) from the **message** is XOR'ed with the next bit (or character) from the **secret key**, obtaining the **ciphertext.**

As long as the following conditions are met, the ciphertext will be impossible to decrypt without the **secret key**.

- the key is truly random
- the key is as large or greater than the message
- the key is never reused (fully or partially)
- the key is never divulged (obvious)

Alin Tomescu, CSE408
Tuesday, February 1st, Lecture #1

One time pad analysis (simple case)

We will analyze the one-time pad algorithm for the most basic case: encrypting a single bit.

- Our message m can be either 0 or 1
- Our key k can be either 0 or 1
- Therefore, the ciphertext $c = m \oplus k$ can be either 0 or 1

Our goal is the following: *If Eve has the ciphertext c, then she gains no information whatsoever about the original message m.*

Mathematically, this is equivalent to proving that: $\Pr[m = 0|c = 0] = \Pr[m = 0]$ (there are 3 other possible combinations for m and c, but the proof for them remains the same)

Proof

We assume that $\Pr[k = 0] = \Pr[k = 1] = \frac{1}{2}$, as it should be if the key is picked randomly. Also note that $\Pr[m = 0] = \frac{1}{2}$ and that $\Pr[c = 0] = \frac{1}{2}$ since both the message and the ciphertext can be either 0 or 1 with equal probability.

$$\Pr[m = 0|c = 0] = \frac{\Pr[c = 0|m = 0] \times \Pr[\mathrm{m} = 0]}{\Pr[c = 0]} = \frac{\Pr[k = 0] \times \Pr[m = 0]}{\Pr[c = 0]} = \frac{\frac{1}{2}}{\frac{1}{2}} \times \Pr[m = 0] = \Pr[m = 0]$$

Problems with the one-time pad

One problem with the one-time pad is that the **key has to be as large as the message** in order to achieve perfect secrecy. Therefore, the problem of transmitting the message securely is now reduced to transmitting the secret key securely, which is no good. Also, real-world applications demand smaller, more portable keys.

Key reuse

Suppose you have two messages m_1 and m_2 encrypted under the same key k as $c_1 = m_1 \oplus k$ and $c_2 = m_2 \oplus k$.

If Eve intercepts c_1 and c_2, and XORs them together she will get $c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$ (thanks to XOR's associativity and commutativity).

Now, once Eve has $m_1 \oplus m_2$ she effectively got the key k out of the equation and she can again use XOR's properties to decipher m_1 and m_2 out of $m_1 \oplus m_2$.

Note that if Eve knows part of m_1 (let's say she knows $m_1[i \dots j]$), she can obtain part of m_2 (specifically the same part of m_2 overlapping the known part of m_1, that would be $m_2[i \dots j]$) by XORing $m_1 \oplus m_2$ with the known part of m_1.

Example

Suppose m_1 is "alan has apples and oranges".

Suppose m_2 is "brian likes network security".

If Eve knows that m_1 contains the sentence "has apples" she can take this known sentence w and XOR it with $m_1 \oplus m_2$. An important detail is that Eve needs to try XORing w in at different offsets (assuming she does not know where w actually starts within m_1) until she determines w's offset within m_1. Eve will figure out the right offset by looking at the results she gets from $w \oplus (m_1 \oplus m_2)$. Some of these will be complete junk, but some of them will reveal letters, or even better, full words from m_2.

Eve can repeat this process by guessing words (or doing a dictionary attack) on either one of m_1 or m_2. If the guess is correct, it will reveal more words or partial words from the other message, which Eve can now continue to work with. This will go on until the entire message is revealed.