Example 36. (terrible jokes, parental guidance advised)

There are I0 types of people... those who understand binary, and those who don't.

Ok, ok, of course you knew that. How about:

There are II types of people... those who understand Roman numerals, and those who don't.

It's not getting better:

There are I0 types of people... those who understand hexadecimal, and F the rest...

Example 37. (yet another joke) Why do mathematicians confuse Halloween and Christmas?

Because 31 Oct = 25 Dec.

Get it? $(31)_8 = 1 + 3 \cdot 8 = 25$ equals $(25)_{10} = 25$.

Fun borrowed from: https://en.wikipedia.org/wiki/Mathematical_joke

Modern ciphers

Example 38. For modern ciphers, we will change the alphabet from $A, B, ..., Z$ to $0, 1$. One of the most common ways of encoding text is **ASCII**.

In ASCII (American Standard Code for Information Interchange), each letter is represented using 8 bits (1 byte). Among the $2^8 = 256$ many characters are the usual letters, as well as common symbols.

For instance: $\text{space} = (20)_{16}$, "0"$=(30)_{16}$, $A = (41)_{16} = (0100, 0001)_2 = 65$, $a = (61)_{16} = (0110, 0001)_2 = 97$

See, for instance, http://www.asciitable.com for the full table.

One-time pad

Definition 39. The "exclusive or" (XOR), often written $\oplus$, is defined bitwise:

	0	0	1	1
$\oplus$	0	1	0	1
=	0	1	1	0

Note. On the level of individual bits, this is just addition modulo 2.

By the way. Best thing about a boolean: even if you are wrong, you are only off by a bit.

Example 40. $1011 \oplus 1111 = 0100$

Example 41. Observe that $a \oplus b \oplus b = a$.

One way to see that is think bitwise in terms of addition modulo 2. Then, $a + b + b = a + 2b \equiv a \pmod{2}$.

A **one-time pad** works as follows. We use a key k of the same length as the message m. Then the ciphertext is

$$c = E_k(m) = m \oplus k.$$

To decipher, we use $m = D_k(c) = c \oplus k$.

As the name indicates, we must never use this key again!

Note. Observe that encryption and decryption are the same routine.

Comment. If that is helpful, a one-time pad is doing exactly the same as the Vigenere cipher if we use a key of the same length as the message (also, we use $0, 1$ as our letters instead of the classical $A, B, ..., Z$).

Armin Straub
straub@southalabama.edu

Example 42. Using a one-time pad with key $k = 1100, 0011$, what is the message $m = 1010, 1010$ encrypted to?

Solution. $c = m \oplus k = 0110, 1001$

If a one-time pad (with perfectly random key) is used exactly once to encrypt a message, then **perfect confidentiality** is achieved (eavesdropping is hopeless).

Meaning that Eve intercepting the ciphertext can draw absolutely no conclusions about the plaintext (because every text of the right length is actually possible, and equally likely without information on the key), see next example.

Example 43. A ciphertext only attack on the one-time pad is entirely hopeless. Explain why!

Solution. The attacker only knows $c = m \oplus k$. The attacker is unable to get any information on m, because every other message m' (of the right length) could have resulted in the same ciphertext c.

Indeed, if the key was $k' = k \oplus m' \oplus m$, then the encryption of m' is $m' \oplus k' = m' \oplus (k \oplus m' \oplus m) = k \oplus m = c$ as well! [Moreover, every plaintext m' is equally likely because it corresponds to a unique key.]

The next example highlights the importance of only using the key once.

Example 44. (attack on the two-time pad) Alice made a mistake and encrypted the two plaintexts m_1, m_2 using the same key k. How can Eve exploit that?

Solution. Eve knows the two ciphertexts $c_1 = m_1 \oplus k$ and $c_2 = m_2 \oplus k$.

Hence, she can compute $c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$.

This means that Eve knows $m_1 \oplus m_2$, which is information about the original plaintexts (no key involved!). That's a cryptographic disaster: Eve should never be able to learn *anything* about the plaintexts.

In fact. If the plaintexts are, say, English text encoded using ASCII then Eve very possibly can (almost) reconstruct both m_1 and m_2 from $m_1 \oplus m_2$. The reason for that is that the messages are expressed in ASCII, which means 8 bits per character of text. However, the **entropy** (a measure for the amount of information in a message) of (longer) typical English text is frequently below 2 bits per character.

Some details and beautiful graphical illustration are given at:

http://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse

Using the one-time pad presents several challenges, including:

- keys must not be reused (see Example 44)
- while perfectly protecting against eavesdropping (if done correctly), the one-time pad is not secure against tampering (example coming soon)
- key distribution and management

  Alice and Bob have to somehow exchange huge amounts of keys, so that, at a later time, they are able to communicate securely.
- for perfect confidentiality, the key must be perfectly random

  But how can we produce huge amounts of random bits?

  Especially, how to teach a deterministic machine like a computer to do that? Think about it! This is much more challenging that it may seem at first...

These issues make one-time pads difficult to use in practice.

Historic comment. During the Cold War, the "hot line" between Washington and Moscow apparently used one-time pads for secure communication.