Sketch of Lecture 6

Mon, 1/22/2024

Representations of integers in different bases

We are commonly using the **decimal system** of writing numbers. For instance:

$$1234 = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0.$$

10 is called the base, and $1, 2, 3, 4$ are the digits in base 10. To emphasize that we are using base 10, we will write $1234 = (1234)_{10}$. Likewise, we write

$$(1234)_b = 1 \cdot b^3 + 2 \cdot b^2 + 3 \cdot b^1 + 4 \cdot b^0.$$

In this example, $b > 4$, because, if b is the base, then the digits have to be in $\{0, 1, \ldots, b - 1\}$.

Comment. In the above examples, it is somewhat ambiguous to say whether 1 or 4 is the first or last digit. To avoid confusion, one refers to 4 as the **least significant digit** and 1 as the **most significant digit**.

Example 35. $25 = 16 + 8 + 1 = \boxed{1} \cdot 2^4 + \boxed{1} \cdot 2^3 + \boxed{0} \cdot 2^2 + \boxed{0} \cdot 2^1 + \boxed{1} \cdot 2^0$.
Accordingly, $25 = (11001)_2$.

While the approach of the previous example works well for small examples when working by hand (if we are comfortable with powers of 2), the next example illustrates a more algorithmic approach.

Example 36. Express 49 in base 2.

Solution.

- $49 = 24 \cdot 2 + \boxed{1}$. Hence, $49 = (\ldots 1)_2$ where $\ldots$ are the digits for 24.
- $24 = 12 \cdot 2 + \boxed{0}$. Hence, $49 = (\ldots 01)_2$ where $\ldots$ are the digits for 12.
- $12 = 6 \cdot 2 + \boxed{0}$. Hence, $49 = (\ldots 001)_2$ where $\ldots$ are the digits for 6.
- $6 = 3 \cdot 2 + \boxed{0}$. Hence, $49 = (\ldots 0001)_2$ where $\ldots$ are the digits for 3.
- $3 = 1 \cdot 2 + \boxed{1}$. Hence, $49 = (\ldots 10001)_2$ where $\ldots$ are the digits for 1.
- $1 = 0 \cdot 2 + \boxed{1}$. Hence, $49 = (110001)_2$.

Other bases.
What is 49 in base 3? $49 = 16 \cdot 3 + \boxed{1}$, $16 = 5 \cdot 3 + \boxed{1}$, $5 = 1 \cdot 3 + \boxed{2}$, $1 = 0 \cdot 3 + \boxed{1}$. Hence, $49 = (1211)_3$.
What is 49 in base 5? $49 = (144)_5$.
What is 49 in base 7? $49 = (100)_7$.

Example 37. Bases 2, 8 and 16 (binary, octal and hexadecimal) are commonly used in computer applications.

For instance, in JavaScript or Python, $0\mathrm{b}\ldots$ means $(\ldots)_2$, $0\mathrm{o}\ldots$ means $(\ldots)_8$, and $0\mathrm{x}\ldots$ means $(\ldots)_{16}$.
The digits $0, 1, \ldots, 15$ in hexadecimal are typically written as $0, 1, \ldots, 9, A, B, C, D, E, F$.
Example. FACE value in decimal? $(FACE)_{16} = 15 \cdot 16^3 + 10 \cdot 16^2 + 12 \cdot 16 + 14 = 64206$
Practical example. `chmod 664 file.tex` (change file permission)

664 are octal digits, consisting of three bits: $1 = (001)_2$ execute (x), $2 = (010)_2$ write (w), $4 = (100)_2$ read (r)
Hence, 664 means rw,rw,r. What is rwx,rx,-? 750
By the way, a fourth (leading) digit can be specified (setting the flags: setuid, setgid, and sticky).

Armin Straub
straub@southalabama.edu

Example 38. (substitution cipher) In a substitution cipher, the key k is some permutation of the letters $A, B, \ldots, Z$. For instance, $k = FRA\ldots$. Then we encrypt $A \to F$, $B \to R$, $C \to A$ and so on. How large is the key space?

Solution. Key space has size $26! \approx 10^{26.6} \approx 2^{88.4}$, so a key can be stored using 89 bits. That's actually a fairly large key space (for instance, DES has a key size of 56 bits only). Too large to go through by brute force.

However, still easy to break. Since each letter is always replaced with the same letter, this cipher is susceptible to a **frequency attack**, exploiting that certain letters (and, more generally, letter combinations!) occur much more frequently in, say, English text than others. For instance, Lewand's book on Cryptology lists the following frequencies:

E: 12.7%, T: 9.1%, A: 8.2%, O: 7.5%, I: 7%, N: 6.7%, S: 6.3%, H: 6.1%, R: 6%, D: 4.3%, L: 4%, C: 2.8%, $\ldots$

The rarest letters are Q and Z with a frequency of about 0.1% only. (The exact frequencies and precise ordering various between different sources and the body of text that the frequencies were obtained from.)

The most common letter pairs (digrams) are TH HE AN RE ER IN ON AT ND ST ES EN OF TE ED OR TI HI AS TO.

More information at: https://en.wikipedia.org/wiki/Letter_frequency

Comment. Note that the frequencies and even the ranking depend considerably on the source of text. For instance, using government telegrams, a military resource lists EN followed by RE, ER as the most frequent digrams. That same manual suggests SENORITA as a mnemonic to remember the most frequent letters.

http://www.umich.edu/~umich/fm-34-40-2/ (Field Manual 34-40-2, Department of the Army, 1990)

Example 39. It seems convenient to add the space as a 27th letter in the historic encryption schemes. Can you think of a reason against doing that?

Solution. In most texts, the space occurs more frequently and more regularly than any other letter. Adding it to the encryption schemes would make them even more susceptible to attacks.

Example 40. (bonus challenge!) You intercept the following message from Alice:

WHCUHFWXOWHUQXOMOMQVSQWAMWHCUHFXOLNWXQMQVSQWAWMQLN

Your experience tells you that Alice is using a substitution cipher. You also know that this message contains the word "secret". Can you crack it?

Note. In modern practice, it is not uncommon to know (or suspect) what a certain part of the message should be. For instance, PDF files start with "%PDF" (0x25504446).

See https://en.wikipedia.org/wiki/Magic_number_(programming) for more such instances.

(To collect a bonus point, send me an email within the next week with the plaintext and how you found it.)

Modern ciphers

Example 41. For modern ciphers, we will change the alphabet from $A, B, \ldots, Z$ 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: $\text{space} = (20)_{16}$, $\text{“0”} = (30)_{16}$, $A = (41)_{16} = (0100, 0001)_2 = 65$, $a = (61)_{16} = (0110, 0001)_2 = 97$

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

Example 42. The new (8/2018) insignia of **FinCEN** features binary digits. What do they mean?

01000110 01101001 01101110 01000011 01000101 01001110 https://www.fincen.gov

By the way. If you ever have more than $\$10,000$ in foreign accounts, you must file a report to FinCEN.

One-time pad

Definition 43. 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 44. $1011 \oplus 1111 = 0100$

Example 45. Observe that $a \oplus b \oplus b = a$.

One way to see that is to 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 $A, B, \ldots, Z$).

Example 46. 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, without information on the key, every text of the right length is actually possible and equally likely), see next example.

Example 47. 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, the key $k' = m' \oplus c$ encrypts m' to c as well (because $m' \oplus k' = m' \oplus (m' \oplus c) = c$). 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 48. (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 illustrations are given at:

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

We saw in Example 47 that ciphertext only attacks on the one-time pad are entirely hopeless. What about other attacks?

Attacks like known plaintext or chosen plaintext don't apply if the key is only to be used once.

Yet, the one-time pad by itself provides **little protection of integrity**. The next example shows how tampering is possible without knowledge about the key.

Example 49. Alice sends an email to Bob using a one-time pad. Eve knows that and concludes that, per email standard, the plaintext must begin with `To: Bob`. Eve wants to tamper with the message and change it to `To: Boo`, for a light scare.

- Eve wants to change the 7th letter of the plain text m from b to o.
- Since b is $0x62$ and o is $0x6F$, we have $b \oplus o = 0x0D$. Hence, $b \oplus 0x0D = o$.
- Therefore, if $e = 0x\underbrace{000000000000}_{6\text{ characters}}0D00\ldots$, then $\underbrace{\text{“TO: Bob...”}}_{m} \oplus e = \underbrace{\text{“TO: Boo...”}}_{m'}$.
- Alice sends $c = m \oplus k$. If Eve changes the ciphertext c to $c' = c \oplus e$, then Bob receives c' and decrypts it to $c' \oplus k = \overbrace{\underbrace{m \oplus k}_{=c} \oplus e}^{c'} \oplus k = m \oplus e = m'$, which is what Eve intended.

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

- keys must not be reused (see Example 48)
- while perfectly protecting against eavesdropping (if done correctly), the one-time pad is not secure against tampering (see Example 49)
- 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.