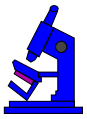



Chapter 5

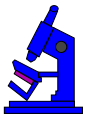
Lecture 5 - Information theory


Tutorial 3 Q1: Intuition...

Question: Longest time to transfer (i) 1000 random characters (ii) 1000 ’A’ characters followed by a ’B’ (iii) 1000000 ’A’ characters followed by a ’B’. (Look for size of information...)

Comment: The entropy for the (i) should be as high as possible, because the source is transmitting the MOST amount of information, the least predictable data. For 7-bit ASCII it will be 7 bits per symbol.

The entropy for the (ii) and (iii) should be as low as possible, as the information is almost completely predictable.



Tutorial 3 Q1: Comments

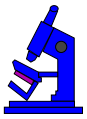
Comment: Since the source emits a fixed string, we can argue that the entropy of the source is zero - it is completely predictable.
But if the receiver is not aware of this, the first message does contain some (a small amount) of information.

Note also: The difference in information between
`1000 'A' characters followed by a 'B'`, and
`1000000 'A' characters followed by a 'B'` is very small, despite the difference in size of the data.

Tutorial 3 Q1: Maths...

Question: Prove it...

Comment: The entropy for the first source is 7 bits per symbol as discussed before, and so the 1000 characters will need 7000 bits = 875 bytes.
Can encode the other two sources much more compactly: **`1000 'A' characters followed by a 'B'`** is much shorter than 7000 bits...



Tutorial 3 Q2: Intuition...

Question: Estimate entropy..

Comment: The relative entropy for the (i) will be 1, whereas the other two will be close to 0, or 0 (depending on interpretation).

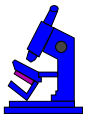
Tutorial 3 Q3: Maths...

Question: ... estimate the average size of a *best* encoded 128 character message. Relative entropy?

Comment: *The entropy for the system is*

$$\begin{aligned} H(X) &= \frac{1}{4}\log_2 4 + \frac{1}{8}\log_2 8 + \frac{6}{16}\log_2 16 + \frac{24}{4*24}\log_2 96 \\ &\approx \frac{1}{2} + \frac{3}{8} + \frac{3}{2} + 0.25 * 6.7 \\ &\approx 4.05 \quad \textit{bits/symbol} \end{aligned}$$

Fit 128 characters into 519 bits, or 65 bytes. Relative entropy is $\frac{4.05}{5} = 0.81$.



Tutorial 3 Q3: Oh ... and...

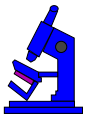
Comment: *The characters E T A O N S H R D L U are the most common letters in written English (in that order), and so English text does not have a high entropy (some symbols are more likely than others). As a result of this English text is easily compressible.*

Tutorial 3 Q4: Maths...

Question: the import of the "*necessary and sufficient condition for perfect secrecy*" in reading.

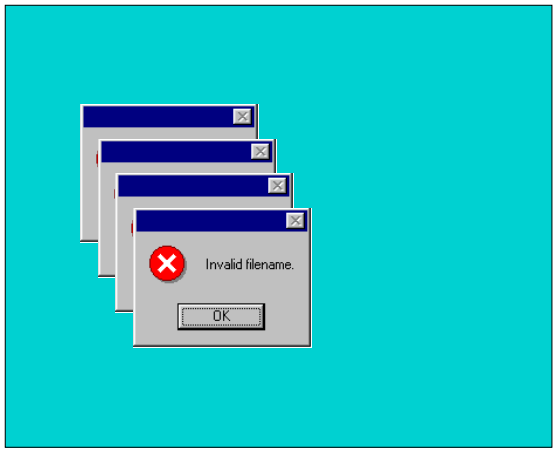
Comment: *Perfect secrecy implies that by observing the ciphered text, you can learn nothing about the original message. The import of the necessary and sufficient condition, is that for perfect secrecy, the key length $\#k$ must be the same number of bits as the message $\#m$. This sort of key is commonly called a one-time-pad.*

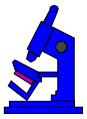
Modern cryptographic practice is not perfect secrecy - just computationally difficult...



An aside ...

The first worry...

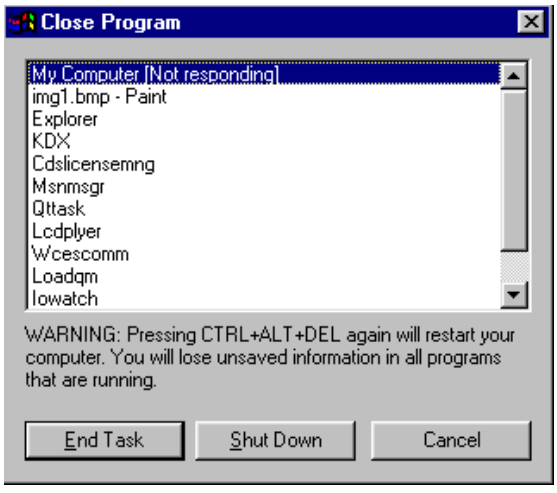


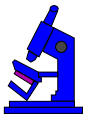


J'accuse! J'accuse!

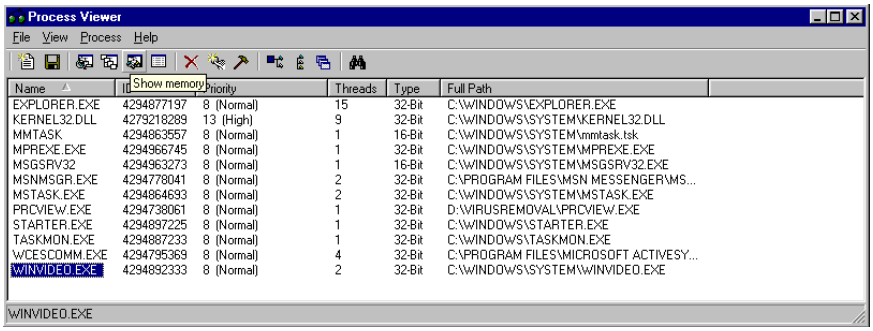
- ✔ Virusses, but also...
- ✔ Keyloggers, spy software in some products
  - ✔ Removal may lead to broken S/W (Kazaa)
  - ✔ CoolWebSearch: slowdowns in IE when typing into text boxes. The hijack installed a stylesheet that used a flaw in Internet Explorer and allowed a .css stylesheet file to execute Javascript code.

A lot of processes?



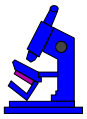


Look closely at processes...

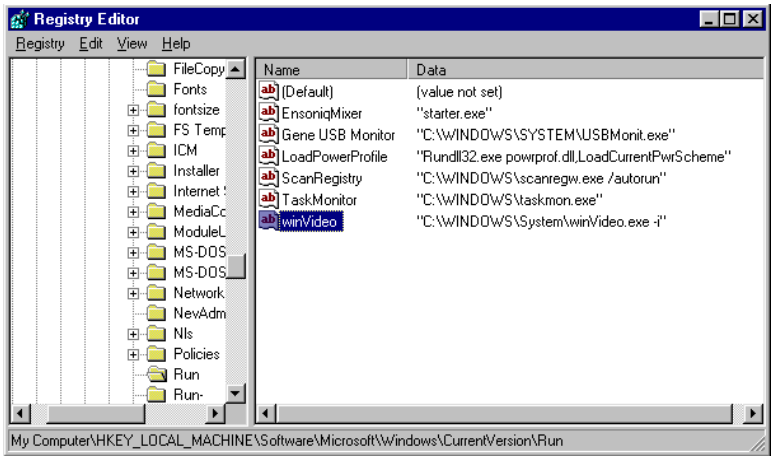


An exploit...DDoS Stinkbot

- This trojan is an IRC bot, controlled by a remote attacker to use infected systems to initiate a Distributed Denial of Service attack against others. When run, it copies its components to the SYSTEM directory with the following filenames:
  - SETUPINF.EXE (Server component)
  - WINVIDEO.EXE (Flooding component)



Using regedit...



Now - what about Ads?

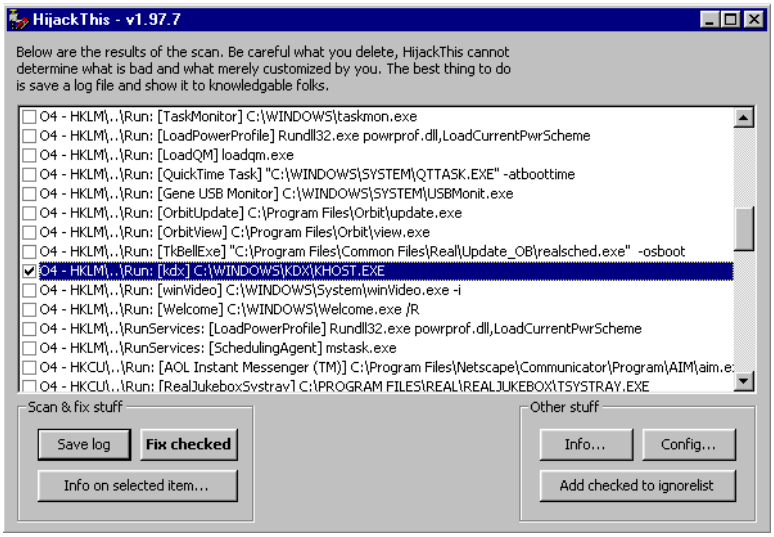
Hijack this: lists all installed browser add-on, buttons, startup items and removes them.

CWshredder: removes CoolWebSearch

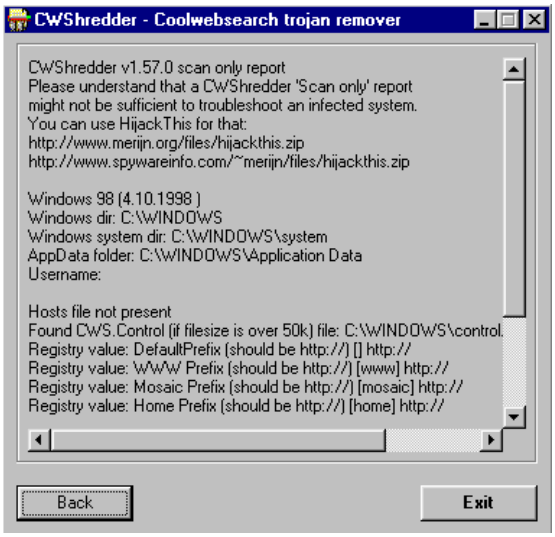
Spybot S&D: remove spyware

Ad-aware: Data-mining, aggressive advertising, Parasites, Scumware, selected traditional Trojans, Dialers, Malware, Browser hijackers, and tracking components...

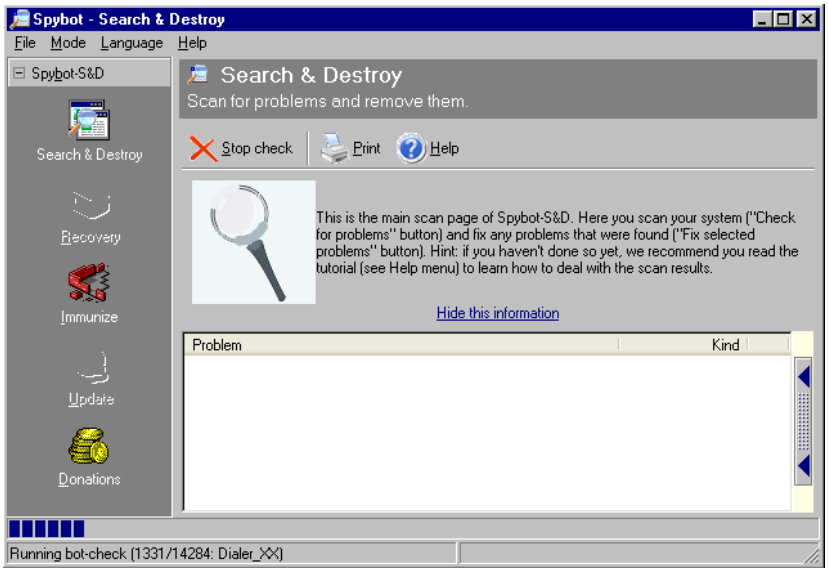
Hijack this



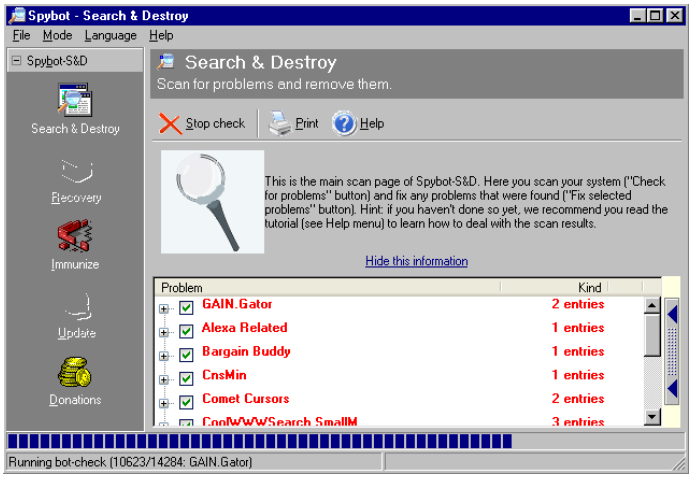
CWshredder



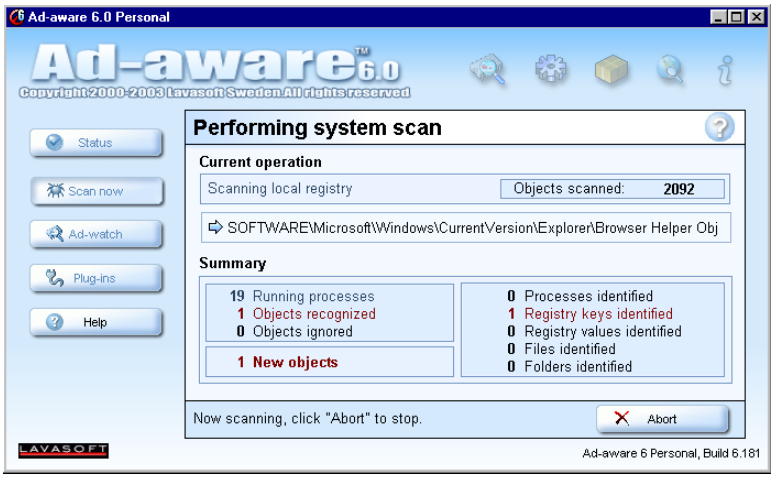
Spybot - Search and Destroy



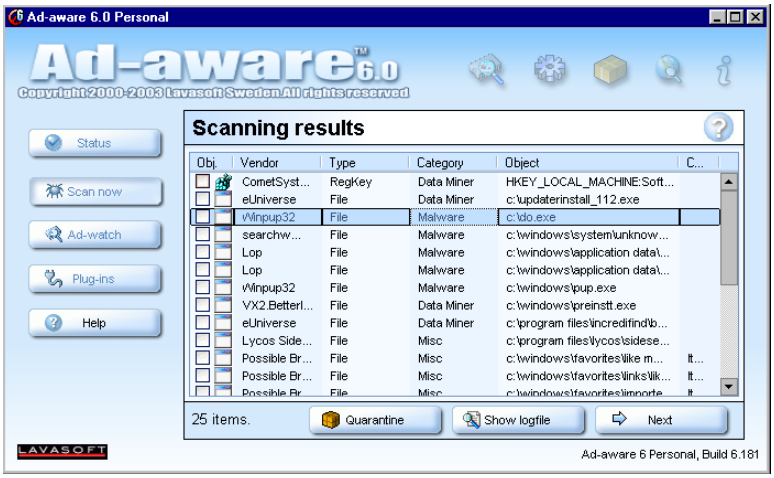
Search and Destroy

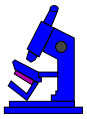


AD-aware...



Ad-aware...



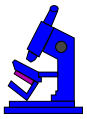


And the end result?

Hey - the computer is a whole lot faster!

but ... 1 month later ...

Back to slide 194

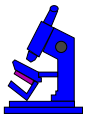


Algorithms

1. Pseudo-random number generation
2. Chinese remainder theorem
3. Extended Euclidean algorithm
4. Testing for primes

Pseudo-random number generation

- ✔ Random numbers or sequences are useful.
- ✔ Generate a long key sequence for an **`xor`**-encrypted message.
- ✔ Generate large primes at random



Linear congruential method

Generate the next number r_{n+1}, from r_n by calculating

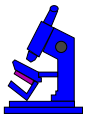
$$r_{n+1} = (ar_n + c) \bmod m$$

where a, m and c are carefully chosen to generate a long random sequence.

Linear congruential method

If we chose $c = 0$, $a = 16807$ and $m = 2147483647 = 2^{31} - 1$, then starting with $r_0 = 1$, we get the sequence

$$\begin{aligned} r_1 = 16807 r_0 \bmod 2147483647 &= 16807 \\ r_2 = 16807 r_1 \bmod 2147483647 &= 282475249 \\ r_3 = 16807 r_2 \bmod 2147483647 &= 1622650073 \\ r_4 = 16807 r_3 \bmod 2147483647 &= 984943658 \end{aligned}$$



Linear congruential weakness

Record three successive values of the pseudo-random number generator[6], and solve the simultaneous equations:

$$\begin{aligned} r_{n+1} &= (ar_n + c) \bmod m \\ r_{n+2} &= (ar_{n+1} + c) \bmod m \\ r_{n+3} &= (ar_{n+2} + c) \bmod m \end{aligned}$$

Should use technique not susceptible to this form of analysis.

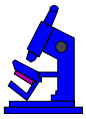
[6] Perhaps by knowing part of the plaintext...

Chinese remainder theorem

The following problem was posed by Sun Tse 2000 years ago:

> *There are certain things whose number is unknown. Repeatedly divided by 3, the remainder is 2; by 5 the remainder is 3; and by 7 the remainder is 2. What will be the number?*

This is now known as an example of the Chinese Remainder Theorem...



Chinese remainder theorem

Theorem 4: Two simultaneous congruences $n = n_1 \bmod m_1$ and $n = n_2 \bmod m_2$ are only solvable when $n_1 = n_2 \bmod (\gcd(m_1, m_2))$. The solution is unique modulo $\mathrm{lcm}(m_1, m_2)$.

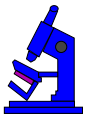
Knowing the remainder of n when it's divided by 3 and the remainder when it's divided by 5 allows you to determine the remainder of n when it's divided by $\mathrm{lcm}(5, 3) = 15$.

Chinese remainder theorem

The original problem:

$$\begin{aligned} x &= 2 \bmod 3 \quad (1) \\ x &= 3 \bmod 5 \quad (2) \\ x &= 2 \bmod 7 \quad (3) \end{aligned}$$

From (1), we know that $x = 3n + 2$ for some n. Substituting this in (2) gives $3n = 1 \bmod 5$. This reduces to a simpler equation $n = 2 \bmod 5$ which is equivalent to $n = 5m + 2$ for some m. Substituting this back into $x = 3n + 2$ gives us $x = 15m+8$. Substituting this in (3) gives $15m+8 = 2 \bmod 7$, or $m = 1 \bmod 7$, i.e. $m = 7o + 1$. From this we can see that $x = 105o+23$. Note that $105 = \mathrm{lcm}(3, 5, 7)$ and the solutions are $23, 128, \ldots$.



Import of CRT

Demonstrates to us that a number less than the product of two primes can be uniquely identified by its residue modulo those primes.

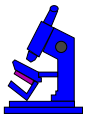
This is useful in the RSA cryptosystem to be investigated later.

Extended Euclidean algorithm

Some theory to compute the multiplicative inverses.

To find the **gcd** of two numbers, using the prime factorization of the numbers. For the **gcd** of 2394 and 154:

$$\begin{aligned} 2394 &= 2 * 3 * 3 * 7 * 19 \\ 154 &= 2 * 7 * 11 \\ \therefore \gcd(2394, 154) &= 2 * 7 = 14 \end{aligned}$$



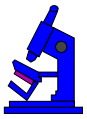
Euclidean algorithm

Unfortunately, it is not easy to find the prime factors of integers. The **gcd** of two integers can however be found by repeated application of division, using the Euclidean algorithm.

- ✔ You repeatedly divide the divisor by the remainder until the remainder is 0.
- ✔ The **gcd** is the last non-zero remainder in this algorithm.

Euclidean algorithm example

$$
\begin{aligned}
\text{dividend} &= \text{quotient} * \text{divisor} + \text{remainder} \\
2394 &= 15 * 154 + 84 \\
154 &= 1 * 84 + 70 \\
84 &= 1 * 70 + 14 \\
70 &= 5 * 14 \\
\therefore \gcd(2394, 154) &= 14
\end{aligned}
$$

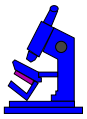


Extended Euclidean algorithm

- ✔ If the $\gcd(a, b) = r$ then there exist integers m and n so that $ma + nb = r$.
- ✔ This algorithm finds these integers m and n and give us a mechanism to calculate the multiplicative inverse of an element x modulo p.

Extended Euclidean algorithm

1. We begin by dividing p by x, and as we carry out each step i of the Euclidean algorithm discovering the quotient q_i, we also calculate an extra number, x_i. For the first two steps $x_0 = 0$ and $x_1 = 1$.
2. For the following steps, we calculate $x_i = x_{i-2} - x_{i-1}q_{i-2} \bmod p$. The calculation is continued for one step after the Euclidean algorithm finishes.
3. If the last non-zero remainder occurs at step k, then if this remainder is 1, x has an inverse and it is x_{k+2}.



Extended Euclidean algorithm

The inverse of 15 modulo 26, showing the method:

$$
\begin{array}{lrcl}
0: & 26 & = & 1 * 15 + 11 \\
1: & 15 & = & 1 * 11 + 4 \\
2: & 11 & = & 2 * 4 + 3 \\
3: & 4 & = & 1 * 3 + 1 \\
4: & 3 & = & 3 * 1 + 0
\end{array}
\qquad
\begin{array}{lcl}
x_0 = 0 & & \\
x_1 = 1 & & \\
x_2 = 0 - 1 * 1 \bmod 26 & = & 25 \\
x_3 = 1 - 25 * 1 \bmod 26 & = & 2 \\
x_4 = 25 - 2 * 2 \bmod 26 & = & 21 \\
x_5 = 2 - 21 * 1 \bmod 26 & = & 7
\end{array}
$$

This indicates that the inverse of 15 is 7. We can verify this by multiplying: $15 * 7 \bmod 26 = 1$.

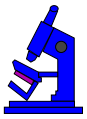
Primality testing

Commonly use statistical testing methods to determine primality.

Quick Quiz! Is $162, 259, 276, 829, 213, 363, 391, 578, 010, 288, 127$ prime[7]?

After choosing a large random (odd) number p, we can quickly see if p is divisible by 2, 3 and so on (say all primes up to 1000). If our number p passes this, then we can perform a statistical primality test.

[7] Note that this is only a 33 digit number, and we typically use prime numbers with hundreds of digits.



Lehmann test

1. Choose a random number w(for witness) less than p
2. If $w^{(p-1)/2} \not\equiv \pm 1 \bmod p$, p is not prime
3. If $w^{(p-1)/2} \equiv \pm 1 \bmod p$, the likelihood is less than 0.5 thatp is not prime

Repeat the test over and over, say n times. The likelihood of a false positive will be less than $\frac{1}{2^n}$. Other tests, such as the Rabin-Miller test may converge more quickly.

Primes is in P!

✔ A group at the Indian Institute of Technology have discovered the unexpected result that testing a number for primality can be done in polynomial time, rather than using probabilistic tests as just shown.

✔ This is unlikely to affect the effectiveness of public key systems.

✔ The paper is only 7 pages long and is beautifully written.