

Homework #1: Encryption and Security: Answers (total 100 pts)

Task 1: Pencil and Paper: (40 pts)

Question 1: If $p = 7$ and $q = 13$, what are the five smallest possible numbers for e ?

e must be relatively prime to $(p - 1)(q - 1) = 72$. The first five numbers that fit that bill are 5, 7, 11, 13, 17

Question 2: If $e = 7$, what is d (its multiplicative inverse)?

The multiplicative inverse is the number d such that $d * 7 \pmod{72} = 1$. The unique inverse is 31.

Question 3: Use d to decrypt the following message, encrypted with the private key e : 3 50 14 84 60 1 6 70 12 1 6 9 50 33 25 50 70 4 47 3 50 4 47 4 6 57 47 13 47 33 33 1 84 47

CONGRATULATIOYOUDECODEDTHEMESSAGE

(sorry for the misspelling.)

Part B - keyspaces.

Key size	Time to find it	Keyspace	Average number of keys/second
40	3.5 hours	2^{40}	8.7×10^7
48	313 hours	2^{48}	2.5×10^8
56	265 days	2^{56}	3.14×10^9
64	1757 days	2^{64}	1.21×10^{11}

Estimated time to crack:

- : 72 bit key: 1468 years, or 535820 days, or 1.28×10^8 hours.
- : 80 bit key: 375831 years, or 1.37×10^8 days, or 3.29×10^9 hours.
- : 88 bit key: 9.62×10^7 years, or 3.5×10^{10} days.

This is definitely exponential. A linear increase in keysize leads to an exponential increase in the time needed to find the key.

Task 2: Practical encryption. (20 pts)

The solution was to send me back a message encrypted with my public key (so that only I could read it) and sign it with your private key (proving that it came from you).

Task 3: Cipher-block chaining. (40 pts)

Here's some lisp code that does CBC - I'll put up some C code in the next day or so.

```
(defun xor-bits (b1 b2)
  (if (equal b1 b2) 0 1))

(defun xor (list1 list2)
  (mapcar #'(lambda (x y) (xor-bits x y)) list1 list2))
```

```

;;; for simplicity, I assume the key and plaintext are stored as lists.
(defun cbc (key prevblock plaintext)
  (cond
    ((null plaintext) nil)
    (t
     (let
        ((cblock
          (xor prevblock
               (xor key (subseq plaintext 0 (length key))))))
       (append cblock
                (cbc key cblock (nthcdr (length key) plaintext)))))))

```

Part 4: Steganography/Digital Watermarking using jsteg (20 pts)

The secret message: Hey! Don't try to bootleg this image. There's a watermark in here.

Some ways of defeating this watermark are: converting the image to a different format, scaling it, altering the colors, adding noise to or degrading the image.

Some workarounds to prevent this: embed the watermark in many places, don't use an LSB technique.