Cryptography

1. Consider the following message: ‘meet me midnight’
   a) Encrypt the message using a transpositional cipher with key 3, and show ciphertext

   M T E I I T
   E _ _ D G *
   E M M N H *

   Ciphertext: MTEIITE__DG*EMMNH*

   b) Encrypt the message using a Vignere cipher with key ‘bed’, and space as the 27th letter of the alphabet.

   Plaintext: m e e t _ m e _ m i d n i g h t
   Add key: 2 5 4 2 5 4 2 5 4 2 5 4 2 5 4 2
   Ciphertext: O J I V E Q G E Q K I R K L L V

2. Break the following simple transpositional cipher: FIOROSDYOGFODOOU

   Try k = 2 columns: rows of 16/2 = 8
   F I OROS DY
   OG FODOOU

   Can’t be k = 3 columns: 16/3 is not an integer and there are no filler characters

   Try k = 4 columns:
   F I O R
   O S D Y
   O G F O
   D O O U

   Plaintext: Foodisgoodforyou

3. Frequency analysis and looking for common short words are the best cryptoanalysis techniques I think – or brute force given that it has a simple key!

   It’s a Caesar cipher with *key* 5

Plain-alphabet:
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Crypto-alphabet:
F G H I J K L M N O P Q R S T U V W X Y Z A B C D E

Plaintext:
here is a suggestion for the tutorial: read up on known attacks against the aes, rijndael, cipher; which class of attacks, that is an attack on the implementation rather than the cipher itself, is the most practical one?

4. The main known attacks against AES, are *side-channel* attacks: they measure the time that the implementation takes or other observable parameters to check a potential key; in particular, implementations that use lookup tables internally for parts of the steps in the AES code, are susceptible to cache-timing attacks, because a second usage of a value in the lookup will have different lookup times, due to the presence of a cache; from that information it may be possible to detect the secret key. See the section on side-channel

attacks of the AES Wikipedia page (and see the referenced paper by Osvik, Shamir, Tromer for details).

5. Use a Vernam cipher with key [1,2,3,4,5,6,7,8,9,10,11,12] to encode the plain text "some message". To decrypt the message, the receiver needs the entire key, which is of the same length as the plain text message itself. How can the exchange of such large keys be avoided?

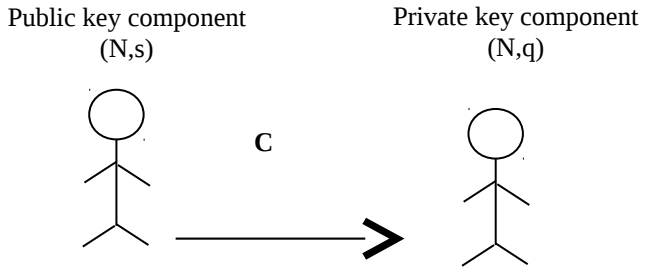
   Shift each letter in the plain text: "some message"
   by the amount prescribed by the key: [1,2,3,4,5,6,7,8,9,10,11,12]
   which gives the ciphertext: "tqpi slabkrq"
   You can test this by trying:
   *Caesar> vernam [1..] "some message"

   The exchange of large keys can be avoided, if the same pseudo-random-number generator (PRG) is used by both sender and receiver. In this case, the only secret is the seed for the PRG, and because the PRG's behaviour is deterministic, the same key sequence will be produced on sender and receiver side. If the key sequence is a random number sequence, the encryption is as strong as a genuine Vernam cipher.

6. Exercise RSA key generation and encryption by doing the following:

The message 4 is encrypted by computing: 4^5 `mod` 91 = 2^10 `mod` 91 = 1024 `mod` 91 = 23. The ciphertext is 23.
In order to crack the private key, the n component of the public key-pair (5,91) needs to be factored: factors 91 = [7,13]. Now, it is easy to compute phi = (7-1)*(13-1) = 72, and the private key is d = 5^-1 mod 72 = mod_inv 72 5 = 29
You can now crack the ciphertext: 23^29 `mod` 91 = 4
Trying to use (6, 91) as the public key fails, because 6 is not relatively prime to 72, and therefore mod_inv 72 6 (which is needed to compute the private key) fails.

11.



SENDER

$P(M) = C = M^{p} \bmod N$

P = public key
M = plain message
C = encrypted message

RECEIVER

$S(P(M)) = M = C^{s} \bmod N$

P = public key
S = private key
M = plain message
C = encrypted message

N, s and q are prime numbers – typically very large, e.g. 100 digits.

RSA algorithm was worked out by three guys, who worked out the algorithm to encode/decode by converting the message to a number and put through a mathematical formula shown left.