Total points: 40. Total time: 75 minutes. 6 problems over 6 pages. No book, notes, or calculator

1. [14 points]
Are n=187 and e=9 valid numbers for RSA. Explain. If you answer yes, obtain the corresponding d.

Solution

There are two requirements:

- n must be a product of two primes
- e must be relatively prime to $\phi(n)$ (so that d, which equals e^{-1} mod-n, exists)

First requirement **[2 points]**
n = 187 = 11·17. 11 and 17 are primes. So this holds.

Second requirement **[6 points]**
Recall that if n =p·q where p and q are distinct primes, then $\phi(p \cdot q) = (p-1)\cdot(q-1)$
So $\phi(187) = (11-1)\cdot(17-1) = 160$.
e, which equals 9, is relatively prime to 160 (because $9=3^2$ and $160=10\cdot16=2\cdot5\cdot2^4=2^5\cdot5$)
So this requirement holds.

Obtaining d **[6 points]**
So we need to find 9^{-1} mod 160,
i.e., need to find a and b such that $1 = a\cdot9 + b\cdot160$ (then $a = 9^{-1}$ mod 160).
We can use Euclid's algorithm to obtain a and b
[Below, rows n = −2 and n = −1 are initialization.
$r_n \leftarrow$ remainder (r_{n-2}/r_{n-1});
$q_n \leftarrow$ quotient (r_{n-2}/r_{n-1});
$u_n \leftarrow u_{n-2} - q_n \cdot u_{n-1}$;
$v_n \leftarrow v_{n-2} - q_n \cdot v_{n-1}$;
]

n	q_n	r_n	u_n	v_n
−2		160	1	0
−1		9	0	1
0	17	7	1	−17
1	1	2	−1	18
2	3	1	4	−71
3	2	0		

From row n=2, we have
$r_n = \gcd(9,160) = 1$ (which we already knew), and
$1 = (4)\cdot(160) + (-71)\cdot 9 = 1$
So d = −71 mod 160 = 160−71 mod 160 = 89

---

2. [6 points]
Consider a sensor X that periodically sends a 64-octet measurement to a receiver Y. One day the administrator decides that X should encrypt the measurement data using DES in CBC mode. How many octets does X now send for each measurement? Explain your answer.

Solution

DES takes a 8-octet (64-bit) plaintext block and yields a 8-octet cipherblock. **[2 points]**

CBC requires a 8-octet initialization vector (IV) to be sent along with the cipherblocks. **[2 points]**

So X now sends 64 octets of cipherblocks **[1 point]**
plus 8 octets of IV, for a total of 72 octets. **[1 point]**

[3 points if you don't say anything wrong and you say that CBC sends cipherblocks + IV.]

---

3. [8 points]
Lish, Pish, and Kish are three languages like English, except that each of them has an alphabet of 4 characters, namely, "A", "B", "C", and "D". The frequency (as percentage) of letter usage in these languages is as follows:

	"A"	"B"	"C"	"D"
Lish	35	15	35	15
Pish	40	30	20	10
Kish	20	20	40	20

Let P be plaintext that can be in either Lish, Pish, or Kish. You are given ciphertext Q obtained from P using a permutation cipher (e.g., "A, B, C, D" → "D, C, B, A"). Q has 1300 A's, 3700 B's, 1700 C's, 3300 D's. Which language is P most likely to be in. Justify your answer.

Solution

Because Q is obtained from P by a permutation cipher, the likely language of P would be the one whose letter frequencies are the closest to the letter frequencies of Q **[2 points]**
after accounting for possible permuations (e.g., by ordering the two frequency vectors). **[3 points]**

The measure of closeness, as in project 1, is the correlation $\sum_{i=0,1,2,3} a_i \cdot b_i$

Listing the frequencies in decreasing order for each language and for Q (and scaling them out of 10 for convenience), we have

Q	3.7	3.3	1.7	1.3
Lish	3.5	3.5	1.5	1.5
Pish	4.0	3.0	2.0	1.0
Kish	4.0	2.0	2.0	2.0

Correlation(Q, Lish) = (3.7)·(3.5) + (3.3)·(3.5) + (1.7)·(1.5) + (1.3)·(1.5) **[1 point]**
= (3.5)·(3.7 + 3.3) + (1.5)·(1.7 + 1.3)
= (3.5)·(7.0) + (1.5)·(3.0)
= 24.5 + 4.5 = 29.0

Correlation(Q, Pish) = (3.7)·(4.0) + (3.3)·(3.0) + (1.7)·(2.0) + (1.3)·(1.0) **[1 point]**
= 14.8 + 9.9 + 3.4 + 1.3
= 29.4

Correlation(Q, Kish) = (3.7)·(4.0) + (3.3)·(2.0) + (1.7)·(2.0) + (1.3)·(2.0) **[1 point]**
= 14.8 + (2.0)·(3.3 + 1.7 + 1.3)
= 14.8 + (2.0)·(6.3)
= 14.8 + 12.6
= 27.4

So P is most likely in Pish.

[5 points if your answer was correct except for not accounting for the possible permuations.]
[5 points if your answer was correct did not have a precise and reasonable "closeness" metric.]

---

4. [4 points]
In the authentication protocol below, pw is A's password and J is a key derived from pw. Can an attacker that can eavesdrop messages (but not intercept or spoof messages) obtain pw by off-line password guessing. If you answer no, explain briefly. If you answer yes, describe the attack.

A (has pw)	**B** (has J)
send [conn] to B	
	generate random challenge R send [R]
compute J from pw compute X ← encrypt(R) with key J send [X] to B	
	compute Y ← decrypt(X) with key J if Y = R then A is authenticated

---

Solution

Yes, an eavesdropping attacker can do off-line password guessing. **[1 point]**

The attacker has R and X, and does the following: **[3 points]**

```
repeat {
  choose candidate password cpw;
  compute cJ from cpw;
  compute cX ← encrypt(R) with key J
}
until cX = X;
 // cpw = pw
```

[Lose 1 point for saying: obtain J from R and X, and then use offline password guessing to get pw from J.]

---

5. [4 points]
In the authentication protocol below, pw is A's password, J is a key derived from pw, and L is a high-quality key (which A gets from B as shown below). Can an attacker that can eavesdrop messages (but not intercept or spoof messages) obtain pw by off-line password guessing. If you answer no, explain briefly. If you answer yes, describe the attack.

A (has pw)	**B** (has J, L)
send [conn] to B	
	X ← encrypt(L) with key J generate random challenge R send [X, R]
compute J from pw L′ ← decrypt(X) with key J Y′ ← encrypt(R) with key L′ send [Y′] to B	
	compute Y ← encrypt(R) with key L if Y′ = Y then A is authenticated

.

Solution

Yes, an eavesdropping attacker can do off-line password guessing.

The attacker has X, R, and Y′, and does the following: **[4 points]**

```
repeat {
  choose candidate password cpw;
  compute cJ from cpw;
  compute cL ← decrypt(X) with key cJ
  compute cY ← encrypt(R) with key cL
}
until cY = Y′;
// cpw = pw
```

[0 points if you say: obtain J from R, X, and Y′, and then use offline password guessing to get pw from J.]

---

6. [4 points]

The chart below shows an authentication protocol, followed by data exchange, followed by disconnection.Only an initial part of the authentication protocol is shown; here, pw is A's password, J is a key derived from pw, and L is a high-quality key. Assume an attacker that can (1) eavesdrop messages and (2) intercept and spoof messages sent by A (but not those sent by B). Complete the authentication protocol (i.e., supply the part indicated by the "• • …. • •") so that inspite of this attacker

- B authenticates A,
- this authentication is not vulnerable to off-line password guessing, and
- A and B establish a session key S (for encrypting data) such that after A and B disconnect and forget S, even if the attacker learns pw, the attacker cannot decrypt the data exchanged.

A (has pw)	**B** (has J, L)
send [conn] to B	
	X ← encrypt(L) with key J send [X]
compute J from pw L′ ← decrypt(X) with key J	
• • • • • • • • •	
<------------------ A and B exchange data ---------------->	
<------------------ A and B disconnect -------------------->	

Solution to 6

There are several solutions based on some form of authenticated Diffie-Hellman (DH). Two are outlined below.

Solution 1 (detailed handshake at end):
After obtaining L′, principal A initiates DH to establish a session key S, where the DH messages are encrypted by L′. Encrypting the DH messages by L′ ensures that the attacker cannot hijack the data exchange phase (otherwise the attacker can spoof A in the DH and thus have the session key established between itself and B).

Solution 2:
After obtaining L′, principal A initiates DH (with unencrypted messages) to establish a session key S. Then B sends a challenge, say R, encrypted with S, to which A responds with a message M applying S and L to R, for example:

- [encrypt(encrypt(R) with L) with S]
- [encrypt(hash(R | L) with S]
- [encrypt(R+1) with S]

Grading:
2 points for Diffie-Hellman.

2 points for secure authentication:

- −1 point for using a challenge-response (as in problem 5) before doing DH; this is vulnerable to offline password guessing.
- −1 point if you have B authenticate A based receiving a static message generated by A, such as [encrypt(J) with L]. This is vulnerable to replay attack (i.e., attacker eavesdroping response from an earlier authentication and then replaying it here).
- −1 point if you have B authenticate A based receiving a message generated by A with no input from B, such as [encrypt(R) with L, R], where R is a random number generated by A (not B). This is vulnerable to replay attack.

Solution 1 handshake details

	A (has pw)	**B** (has J, L)
	send [conn] to B	
		X ← encrypt(L) with key J send [X]
	compute J from pw L′ ← decrypt(X) with key J	
•	<- Solution Start ->	
•	choose Diffie-Helman g, p **[2 points]**	
	generate random number S_A	
•	compute $T_A \leftarrow g^{S_A} \bmod p$	
	compute $Y_A \leftarrow$ encrypt(T_A) with L′	
•	send [Y_A , g, p] to B	
•		
		generate random number S_B **[2 points]**
•		compute $T_B \leftarrow g^{S_B} \bmod p$
		compute $Y_B \leftarrow$ encrypt(T_B) with L
•		send [Y_B] to B
		session key $S \leftarrow T_A^{S_B} \bmod p$
•	session key $S' \leftarrow T_B^{S_A} \bmod p$	
	A and B now have session key S (= S′)	
	<- Solution End ->	
	<------------------ A and B exchange data ---------------->	
	<------------------ A and B disconnect --------------------->	