Total points: 60. Total time: 75 minutes. 6 problems over 7 pages. No book, notes, or calculator

1. [14 points]
Are n=323 and e=5 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 = 323 = 17·19. 17 and 19 are primes. So this holds.

Second requirement **[4 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 $\phi(323) = (17-1)\cdot(19-1) = 288$.
e, which equals 5, is relatively prime to 288 (because 5 is prime and does not divide 288 exactly)
So this requirement holds.

So $d = 5^{-1}$ mod 288 **[2 points]**

Obtaining d **[6 points]**
Use Euclid's algorithm to get a and b such that $1 = a\cdot 5 + b\cdot 288$ (then $a = 5^{-1}$ mod 288).
[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		288	1	0
−1		5	0	1
0	57	3	1	−57
1	1	2	−1	58
2	1	1	2	−115
3	2	0		

From row n=2, we have
r_n = gcd(5, 288) = 1 (which we already knew), and
$1 = (2)\cdot(288) + (-115)\cdot 5$
So d = −115 mod 288 = 288−115 = 173

---

2. [5 points]
Recall that a **DES encryption operation** takes a 64-bit plaintext block and a 56-bit key and produces a 64-bit ciphertext block. Recall also that each DES encryption operation itself consists of a number of iterations, which we shall refer to as **basic iterations**.

For the DES encryption in CBC mode of a plaintext message of N 64-bit blocks, obtain the following (in terms of N):
a. Total number of DES encryption operations.
b. Size of the output. Explain briefly.
c. Total number of basic iterations. Explain briefly.

Solution

Let the plaintext message be $[m_1, m_2, \ldots, m_N]$.
Its CBC encryption is given by C_j = DES_Encrypt(C_{j-1} XOR m_j) for j = 1, …, N, where C_0 = IV.

a. The DES-CBC encryption involves N DES encryption operations. **[1 point]**

b. The output is $[IV, C_1, \ldots, C_N]$, which is (N+1) 64-bit blocks. **[2 points]**
Only 1 point if incorrect answer but IV is mentioned.

c. Each DES encryption operation has **[2 points]**

- 16 iterations to transform the plaintext block into the ciphertext block. .
  So there are 16N of these iterations.
- 16 iterations to produce the 16 48-bit keys from the 56-bit key.
  But these 16 iterations need be done only once for the entire message.

So the answer is 16N + 16 iterations. 16N and 32N are also acceptable.
1 point if you don't give an answer in terms of N but say there are 16 iterations per DES encryption operation.

---

3. [6 points]

Is there an integer K in the range 1, …, 47 such that K^{48} mod-105 is not equal to 1?
If you answer yes, produce such a K and the value of K^{48} mod-105 (as an integer in the range 1, …, 47).
If you answer no, explain.

Solution

105 = 5·21=3·5·7. So 105 is a product of distinct primes. **[1 point]**
$\phi(105) = (3-1)\cdot(5-1)\cdot(7-1) = 2\cdot 4\cdot 6 = 48$

By Euler's theorem: K^{48} mod-105 = 1 for all K relatively prime to 105 **[1 point]**
But this does not account for all K in 1, …, 47.
[Also, by generalization of Euler's theorem, K^{48+1} mod-105 = K for all K in 1,…, 47,
but K^{48+1} mod-105 = K does not imply K^{48} mod-105 = 1.]

So need to look for a counter-example K that is not relatively prime to 105. **[1 point]**

Calculating K^{48} mod-105 **[3 points]**

Example calculations

K=3
3 is not relatively prime to 105, so we try that (all lines below are mod-105):
$3^3 = 27$
$3^6 = 27\cdot 27 = 729 = -6$
$3^{12} = (-6)\cdot(-6) = 36$
$3^{24} = (36)\cdot(36) = 1296 = 36$
$3^{48} = (36)\cdot(36) = 36$

K=5
5 is also not relatively prime to 105.
$5^2 = 25$
$5^3 = 125 = 20$
$5^6 = 20\cdot 20 = 400 = -20$
$5^{12} = (-20)\cdot(-20) = 400 = -20$
$5^{24} = (-20)\cdot(-20) = -20$
$5^{48} = (-20)\cdot(-20) = -20 = 85$

Note K=2 would not work because 2 is relatively prime to 105.

---

4. [10 points]
Consider a public key infrastructure with principals $A_1, A_2, \ldots, A_{20}$ and $B_1, B_2, \ldots, B_{20}$. There are three certification authorities, namely, X, Y, and Z. Each principal (i.e., A_i and B_i) has X's public key. X issues certificates for Y and Z. Y issues certificates for $A_1, A_2, \ldots, A_{20}$. Z issues certificates for $B_1, B_2, \ldots, B_{20}$.

Suppose A_1 wants the public key of B_2. What are the documents (e.g., certificates) that A_1 looks for. For each document, describe its fields and any constraints that must hold.

Solution

A_1 looks for

- Certificate issued by X for Z that **[1 point]**
  has not yet expired **[1 point]**
- CRL issued by X that **[1 point]**
  is recent enough **[1 point]**
  and does not include the serial number of the above certificate for Z **[1 point]**

- Certificate issued by Z for B_2 that **[1 point]**
  has not yet expired **[1 point]**
- CRL issued by Z that **[1 point]**
  is recent enough **[1 point]**
  and does not include the serial number of the above certificate for B_2 **[1 point]**

0 points for giving a KDC-based approach.
(Note that X, Y, Z need not be online, and A_1 does not talk to them.)

---

5. [10 points]
The chart below shows a skeleton of an authentication protocol. Initially, principals A and B share a secret key K and public Diffie-Hellman parameters g and p. Assume an attacker that can eavesdrop, intercept messages, and send messages with another's sender id. Supply an authentication protocol (i.e., the part indicated by the "● ● …. ● ●") such that:

- A initiates the protocol.
- A and B authenticate each other (i.e., the attacker cannot impersonate one to the other).
- 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 K, the attacker cannot decrypt the data exchanged.
- The authentication involves *at most* 4 messages (it can be fewer). (Only one cell can be used in each row.)

	A (has K, g, p)	**B** (has K, g, p)
●		
●		
●		
●		
	<------------------- A and B exchange data -----------------> <------------------- A and B disconnect --------------------->	

Solution to 5

The solution is to do an authenticated Diffie-Hellman (DH) using the shared key K.

Solution 1.
The easiest solution is to do DH using K to encrypt the DH messages:

	A (has K, g, p)	**B** (has K, g, p)
1	generate random S_A $T_A \leftarrow g^{S_A}$ mod p $U_A \leftarrow$ encrypt(T_A) with K send [A, B, U_A]	
2		receive [A, B, U_A] extract T_A from U_A using K generate random S_B $T_B \leftarrow g^{S_B}$ mod p $U_B \leftarrow$ encrypt(T_B) with K send [B, A, U_B] session key $S_B \leftarrow T_A^{S_B}$ mod p
3	receive [B, A, U_B] session key $S_A \leftarrow T_B^{S_A}$ mod p	
	<--------------- session key $S = S_A = S_B$ -------------------------> <------ A and B exchange data using S, then disconnect -------->	

Note that at the end of step 3, it is possible A and B are both talking via a "man-in-the-middle" attacker; however, the attacker will not have the session key S, and so cannot impersonate A to B or B to A any further in the session.
Even this can be avoided by using nonces, as described in solution 2 below.

Grading

6 points for the Diffie Helman operations
- generate random S_A, $T_A \leftarrow g^{S_A}$ mod p, etc, corresponding operations for B **[3 points]**
- session key $S \leftarrow T_B^{S_A}$ mod p, etc, corresponding operations for B **[3 points]**

4 points for authenticating the DH exchange using K
- One way is to encrypt the DH exchange using K (as shown above).
- Another way is to do unencrypted DH and then use the DH session key to encrypt a challenge-response involving K.

At most 2 out of 4 points if K is not involved in the DH session key construction or subsequent verification.
- One example is if the DH handshake is not encrypted with K.
- Another example is if K alone is used to encrypt a challenge-response.

In such cases, a man-in-the-middle attack is possible where the attacker hijacks session after the authentication handshake (as shown in solution attempt 3 below).

At most 2 out of 10 points if session key obtained from other than DH (which would allow the attacker to decrypt data if it learns K later).

Solution 2 (detects authentication attack earlier)

	A (has K, g, p)	**B** (has K, g, p)
1	generate random N_A generate random S_A $T_A \leftarrow g^{S_A}$ mod p $U_A \leftarrow$ encrypt(T_A, N_A) with K send [A, B, U_A]	
2		receive [A, B, U_A] extract T_A and N_A from U_A using K $M_A \leftarrow N_A + 1$ generate random N_B generate random S_B $T_B \leftarrow g^{S_B}$ mod p $U_B \leftarrow$ encrypt(T_B, N_B, M_A) with K send [B, A, U_B] session key $S_B \leftarrow T_A^{S_B}$ mod p
3	receive [B, A, U_B] extract T_B, N_B, M_A if $M_A = N_A + 1$ then B authenitcated $M_B \leftarrow N_B + 1$ session key $S_A \leftarrow T_B^{S_A}$ mod p send [A, B, K{M_B}]	
		receive [A, B, K{M_B}] extract M_B from message using K if $M_B = N_B + 1$ then A authenticated
	<--- A and B exchange data with session key $S = S_A = S_B$ -----> <------------------------- A and B disconnect ----------------------->	

Solution attempt 3 (does not use K and DH in conjunction, hence does not work)

	A (has K, g, p)	**B** (has K, g, p)
1	generate random N_A and S_A $T_A \leftarrow g^{S_A}$ mod p send [A, B, K{N_A}, T_A]	
2		receive [A, B, K{N_A}, T_A] $M_A \leftarrow$ decrypt K{N_A} using K generate random N_B and S_B $T_B \leftarrow g^{S_B}$ mod p send [B, A, M_A, K{N_B}, T_B] session key S $\leftarrow {T_A}^{S_B}$ mod p
3	receive [B, A, M_A, K{N_B}, T_B] if $M_A = N_A$ then B authenticated else abort $M_B \leftarrow$ decrypt K{N_B} using K session key S $\leftarrow {T_B}^{S_A}$ mod p send [A, B, M_B]	
		receive [A, B, M_B] if $M_B = N_B$ then A authenticated else abort
	<--- A and B use session key $S=S_A=S_B$ for data and closing---->	

Here is a man-in-the-middle attack on solution attempt 3

	A (has K, g, p)		**Attacker C**		**B** (has K, g, p)
1	generate random N_A and S_A $T_A \leftarrow g^{S_A}$ mod p send [A, B, K{N_A}, T_A] //msg 1	→	intercept msg 1 generate random S_C $T_C \leftarrow g^{S_C}$ mod p session key $S_{AC} = T_A^{S_C}$ mod p forward msg 1 with $T_A \rightarrow T_C$	→	
2		←	intercept msg 2 session key $S_{BC} = T_B^{S_C}$ mod p forward msg 2 with $T_B \rightarrow T_C$	←	receive [A, B, K{N_A}, T_C] $M_A \leftarrow$ decrypt K{N_A} using K generate random N_B and S_B $T_B \leftarrow g^{S_B}$ mod p send [B, A, M_A, K{N_B}, T_B] //msg 2 session key S $\leftarrow T_A^{S_B}$ mod p
3	receive [B, A, M_A, K{N_B}, T_C] $M_A = N_A$ so B is authenticated $M_B \leftarrow$ decrypt K{N_B} using K session key $S_A \leftarrow T_C^{S_A}$ mod p send [A, B, M_B] // msg 3	→	no need to modify msg 3	→	
					receive [A, B, M_B] $M_B = N_B$ so A authenticated
	<---- A shares session key S_A with C --> A thinks it shares it with B				<---- B shares session key S_B with C --> B thinks it shares it with A
			C does following for every msg that A sends to B (including the disconnection handshake messages): • intercept the message, • decrypt encrypted fields with S_{AC} and re-encrypt with S_{BC}, • forward modified msg to B C does the same for every msg that B sends to A (with the roles of S_{AC} and S_{BC} interchanged).		

6. 15 points]
In the authentication protocol below, pw is A's password and J is a key derived from pw.

A (has pw)	**B** (has J)
send [A, B, conn] // msg 1	
	receive [A, B, conn] generate random challenge R_B $S_B \leftarrow$ encrypt(R_B) with key J send [B, A, S_B] // msg 2
receive [B, A, S_B] compute J from pw $T_B \leftarrow$ decrypt(S_B) with key J $U_B \leftarrow$ encrypt(T_B+1) with key J generate random challenge R_A $S_A \leftarrow$ encrypt(R_A) with key J send [A, B, U_B, S_A] // msg 3	
	receive [A, B, U_B, S_A] $V_B \leftarrow$ decrypt(U_B) with key J if V_B= R_B+1 then A is authenticated else abort $T_A \leftarrow$ decrypt(S_A) with key J $U_A \leftarrow$ encrypt(T_A+1) with key J send [B, A, U_A] // msg 4
receive [B, A, U_A] $V_A \leftarrow$ decrypt(U_A) with key J if V_A= R_A+1 then B is authenticated else abort	

a. Consider an attacker that can **only eavesdrop** (i.e., can hear messages in transit but cannot intercept messages or send messages with somebody else's sender id). Can this attacker obtain pw by off-line password guessing. If you answer no, explain briefly. If you answer yes, describe the attack.

b. Consider an attacker that can **only spoof A** (i.e., send messages with sender id A and receive messages with destination id A, but not eavesdrop or intercept messages). Can this attacker obtain pw by off-line password guessing. If you answer no, explain briefly. If you answer yes, describe the attack.

c. Consider an attacker that can **only spoof B** (i.e., send messages with sender id B and receive messages with destination id B, but not eavesdrop or intercept messages). Can this attacker obtain pw by off-line password guessing. If you answer no, explain briefly. If you answer yes, describe the attack.

Solution to 6

Part a. [5 points]

Yes, off-line password guessing is possible. **[5 points]**

Attacker does following with S_B and U_B (note $S_B = J\{R_B\}$ and $U_B = J\{R_B + 1\}$):

```
for each cpw in dictionary do {
        cJ ← key constructed from cpw;
        cR ←decrypt S_B using cJ;
        cRplus1 ←decrypt U_B using cJ;
        if cRplus1 = cR + 1 then done; //  pw = cpw;  J = cJ,
}
```

Same attack possible with S_A and U_A.

0 points for saying "yes" without any explanation or with a completely wrong explanation.

Part b. [5 points]

No, off-line password guessing is not possible. **[5 points]**

The attacker can get S_B (= $J\{R_B\}$) by sending [A, B, conn], but it cannot get anything more.
Because it does not have J, it cannot compute U_B (= $J\{R_B+1\}$).
So whatever msg 3 the attacker sends will not elicit a responding msg 4 from B.

0 points for saying "no" without any explanation or with a completely wrong explanation.
At most 2 points if you do not explain why the attacker cannot get B to send msg 4.

Part c. [5 points]

Yes, off-line password guessing is possible. **[5 points]**

Attacker waits until A requests a connection, upon which it sends msg 2 with random S_B.
A responds with msg 3 in which U_B = encrypt(decrypt S_B using J) using J.
Then do the following with S_B and U_B (exactly as in part a):

```
for each cpw in dictionary do {
        cJ ← key constructed from cpw;
        cR ←decrypt S_B using cJ;
        cRplus1 ←decrypt U_B using cJ;
        if cRplus1 = cR + 1 then done; //  pw = cpw;  J = cJ,
}
```

0 points for just saying yes without any explanation or with a completely wrong explanation.
