Assignment 1, part 2 by Staffan Lundstrom, roll no 03005201

Generation of RSA key pair

Key size (bits):	*128*	*256*	*512*	*768*	*1024*
Time (millisec): 1	171	270	621	791	4927
2	220	190	771	1693	8993
3	210	180	781	1993	5758
4	150	181	821	3725	6239
5	130	271	681	1702	6019
Avg_time (ms):	176,2	218,4	697,8333	1980,8	6387,2

Analysis: Up to 512 bits it seems like the generation time is proportional to the key size. After that it takes off. The second result using key size 1024 might be abnormal. I think it's difficult to draw any conclusion with certainty. The complexity could be O(k^3) or in worst case O(x^4). It depends on how the curve look after 1024.

Encryption/decryption of messages using different key sizes

Message size:	1K				
Key size (bits):	*128*	*256*	*512*	*768*	*1024*
Encrypt time (ms):	1572	9183	67266	210162	556400
Decrypt time (ms):	1423	8983	67607	208980	616998

Analysis: The test results indicate strongly that the encryption and the decryption time is of O(k^3), if k is the key size. For example, doubling the key size implicates an eight times longer encryption time.

Encryption/decryption of messages using different message sizes

Key size used:	*128*			
Message size	1K	2K	3K	4K
Encrypt time (ms)	1612	2924	4246	5748
Decrypt time (ms)	1453	2744	4086	5608

Key size used:	*512*			
Message size	1K	2K	3K	4K
Encrypt time (ms)	65564	132200	199076	266493
Decrypt time (ms)	65885	132360	201199	262497

Analysis: Usually small or smart choices of the public exponent like 3, 17 and 65537 are used in order to reduce the operations. That makes encryption faster than decryption. This can't be read from these data and the natural reason is because it's not implemented in this implementation. The data suggests that the encryption and decryption time grow with linear complexity at these message sizes. But increasing the key size by 4 takes almost 4^3 times longer time, which supports the conclusion above that the time complexity is O(k^3), if key size is k.

Probability of p, q being prime

In the key generation above, the probability that each of p and q represents a prime number will exceed 1 - 1/(2^x), where x is 10.

Using different values on x gives the following data:

Key size used:	*256*		
x (see explanation above)	*100*	*50*	*5*
Time (ms): 1	291	221	180
2	200	250	170
3	290	260	161
4	201	290	281
5	361	260	150
Avg_time (ms):	268,6	256,2	188,4

Analysis: At least in the tested certainties for finding a prime, increasing the probability parameter does not cost so much time. Without verifying it empirically I think the cost will be significantly larger if you have extreme requirements on certainty