Math 418/818
Term Project: Comments and hints

Note: Answer all parts of each question. (Some parts are more important than others.)

• Number your parts (a) - (g) and keep them separate. (They are marked separately.)
• Remember to always justify your assertions by using references to results from class and/or the text.

1(a)
• Be sure to identify the public and private keys properly. Note that the key size depends on security considerations, not on the size of the message. (A long message can always be subdivided into smaller message units.)
• Remember that \( g \in \mathbb{F}_p \), does not need to be a generator of \( \mathbb{F}_p^\times \). Thus, your attacks usually involve properties of \( g \) (and not necessarily those of \( \mathbb{F}_p^\times \)).
• Explain how the Diffie-Hellman assumption is relevant here.

1(b)
• Explain clearly the consequences for the design. This is not the same as explaining when you can apply the attack.
• When discussing consequences, give precise (numerical) values for concepts like “small” and “large” (making realistic assumptions). Moreover, explain how you arrived at these values. (Give a reference or a justification for them.)

1(c)
• “Arbitrarily large bit-size” means that the bitsize is an input parameter for the key generation. (The output should have bitsize \( \geq \) to the given one.) Thus, you cannot choose your primes from a predetermined list.
• “Step-by-step” means that you list one step after the other (number your steps). Each step should be detailed enough so that it is clear how you would implement this step on a computer. Do not include Maple code here, but write out the main steps in words. If you plan to use any built-in functions in MAPLE, state the ones that you are using. (You can use \texttt{isprime} and \texttt{nextprime} and \texttt{safeprime}.)

• Do not use any algorithms (built-in or otherwise) which are not polynomial-time algorithms, for otherwise you won’t be able to do part (g). Thus, you cannot use programs like \texttt{ifactors} or \texttt{order} or \texttt{primroot} in your program design.
• Do not explain the ElGamal encryption/decryption procedure again. (This was done in part (a).)
• The security of the system depends mainly on the choice of \( \mathbb{F}_p \), so the prime \( p \) has to chosen/constructed carefully. Explain clearly how your program design guards against your two attacks of part (b). You have to be sure that your program is safe against these attacks. Being “probably sure” is not enough.
• Your method of picking \( g \) should depend on how you picked \( p \). This should be analyzed when you discuss the pros and cons of your two methods in part (d). (A probability analysis might be useful here.) Discuss the pros and cons of the two methods relative to each other.
• Do not restrict the secret key \( b \) to a small range, otherwise a (partial) log table attack could be used.
1(d) • Both algorithms for key generation should be of the same caliber. In particular, their run-times should be similar and both should work for bitsize 300.
  • The second method should not be a refinement of the first (or vice versa), but should be based on a different idea. In particular, the primes should be chosen by a different method.
  • Repeat the steps of part (c) for the second method. Do not include Maple code in your description of the second procedure.

1(e) • For each of the two methods, put your 100 choices of \([p, g]\) into a list, but do not display the list. However, display (print out) the first and last element of each list.

1(f) • Do not use \texttt{mlog} in your decoding program. (Your key should have been chosen in such a way that \texttt{mlog} would not work in this situation.)
  • Read Maple’s help pages (or the homework solutions) on how to time Maple commands.

1(e)–(g) • You will be marked on the clarity and completeness of your MAPLE documentation. The documentation/description for each program should come before the program, not afterwards. On the other hand, the discussion of the results of a Maple computation should come afterwards.
  • The documentation should explain the input and output of each program in terms of the variables that are passed to the program.