Cryptographic Engineering: Assignment 4

Submit a c code file + PDF through Canvas.

Assignment A

For this assignment, you are going to code some functions in C. A template (.c file) is created for you. Download it and complete the exercises below by filling the functions in the template. Compile the code and run it. At the end of the file, write **ALL** the output as a comment. Once completed, submit the C file through Canvas.

Remarks (These only apply to C code not Sagemath Code):

- Do **NOT** remove or add any function in the template.
- Do **NOT** modify the function name or parameters in the template.
- Do **NOT** remove, add, or modify any function in the template other than the ones stated below.
- Do **NOT** add any library to the C code.
- All variables you need have been created and initialized for you, you do NOT need any additional variable.
- Do **NOT** use any while loop or if statements. Only simple **for loops** are allowed `for(i=...;i<...;i++)` or `for(i=...;i>...;i-)`.
- Do **NOT** use / or % operators.

A test bank (contains tests and answers) has been created for you. This test bank will make sure that you coded the functions correctly. The only thing you have to do is copy `assign#a_bank` file from Canvas to the same folder as where you run the compiled executable. It will show whether each question passed or failed. You can disable testing for a specific exercise by switch #define EXn 1 to #define EXn 0. Make sure to re-enable all of them once completed.

For this assignment, modular multiplication and modular squaring from your previous assignments have been provided for you.

1. Implement a secure **select function**. To be filled in function `select_bigint256`.
2. Implement a secure **modular exponentiation** function $r = a^e \mod p$ using the provided functions assuming bit 254 (starting from 0) is the MSB bit of e and is always 1. To be filled in function `mod_exp`. Answer must be in radix-2^{16}.
3. Implement a secure Diffie-Hellman **key generation** function which generates the secret key and public key. To be filled in function `keyGen`. Answer must be in radix-2^{16}.
4. Implement a secure Diffie-Hellman **shared secret generation** function which generates the shared secret from a secret key and a public key. To be filled in function `sharedSecret`. Answer must be in radix-2^{16}.

DO NOT forget to copy/paste the output as a comment in the bottom of the assignment.

Assignment B

1. **RSA:** RSA is primarily used for encrypting and decrypting messages. RSA works both ways. If you use the private key to encrypt a message, then the public key will decrypt a message. If you use the public key to encrypt the message, then the private key will decrypt the message. The later is used primarily for signature verification (aka verifying that you are indeed the sender) while the former is used to exchange encrypted messages without generating a shared secret. Make sure to review RSA from class lecture before attempting this question.

   (a) You generated the following RSA data (prime p, prime q, $n = p \cdot q$, public key e, private key d). I sent you an encrypted message c using your public key e. Write SageMath code to decrypt the message with your private key d and **decode** it. Print the decoded message. Hint: The message is written in English (not Gibberish). Show code and output. Code MUST BE TYPED (Screenshots are NOT allowed).

   ```
   p = 1269b2df3bc2d8ef3626ef98a9ea77743
   q = 1104c4aa7525602cfaf5206afdb533bf7
   e = 96f54d8a969cd130d116e9e28d4cf01ad5f806c36e33be5398c1ee4ddfc1ccf5
   d = 501835ad04cce18b61c3c663db5ae973fa24ffb25b0ab538a27eeca427274599
   n = 1395cacce059063e3afa215c480c313787abd46e7516ff4ca6bfc836ea7c982a5
   c = 1333afedf43135fd5387f3dac6d7746450426e2d6fd6fee3b1aaf7efcf7decfa
   ```

   (b) In Sagemath, generate 128-bit security level RSA keys. Then encrypt the message "Cryptographic Engineering" (without quotes) with your private key. Then print your encrypted message and all necessary information for me to decrypt the message and verify that you are the sender of the message. Do NOT print any information that will let me encrypt messages with your information. Show code and output. Code MUST BE TYPED (Screenshots are NOT allowed).

2. **Diffie-Hellman:** After completing assigment A, go to line `#define REAL_RANDOM 0` and change it to `#define REAL_RANDOM 1`.

   (a) Generate secret key/public key pair by passing **`keygen`** argument. Write both down.

   (b) I generated my own secret key/public key pair. The public key generated is shared below. Generate the shared secret by passing **`shared sk pk`** where `sk` is the secret key in hex (without leading 0x) and `pk` is the public key in hex (without leading 0x).

   ```
   pk = 388769a0326458d0fe3bdeac7f8f76ca610b2b364433d04fea4b41bfb5af5f32
   ```

   (c) What is the approximate security level of this implementation? Is it secure? Explain.

   (d) Bob wants to generate a shared secret using Diffie-Hellman with Alice. Alice is not always online but Bob wants to generate their shared secret at any time. A 3rd party is able to host and share data and is always online. Is it possible for Bob to generate a shared secret at any time without Alice being online? If yes, how? And if not, why?

   (e) (Bonus) Crack the secret key that I generated in (b).

   Remark: To pass an argument, simply write it after the executable name. For example, if your generated executable is `a.out` and the argument is `keygen`, then simply write in the command line "`./a.out keygen`" (without quotes).

Grading

	Exercise	Grade	Note:
Assignment A	1	10 pts	
	2	10 pts	
	3	10 pts	
	4	10 pts	
Assignment B	1 (a)	10 pts	
	1 (b)	10 pts	
	2 (a)	10 pts	
	2 (b)	10 pts	
	2 (c)	10 pts	
	2 (d)	10 pts	
	2 (e)		+10 pts for correct answer and -10 pts for wrong answer. This is a bonus question and will increase your final assignment grade. Incorrect answer will decrease the grade for this assignment. Leave blank to not lose points!
	Formatting		-5 pts. To not lose any points, readability and proper indentation are required. Follow the same formatting as the template. Indent 4 spaces when going into the body of a block. Leave an empty line when working on a new block or when you are working on a new part in your function.
	Late Submission		-5 pts per day for 2 days. Afterwards a grade of 0 is given.
	Total	100 pts	

Remark 1: Code that does **NOT** compile on Linux (gcc) will receive a grade of 0 for the whole assignment.

Remark 2: Students are not allowed to modify any function other than the ones stated in the exercises. Modifying the function name or parameters will receive a grade of 0 for that function. Modifying functions not stated in the exercises will recieve a grade of 0 for the whole assignment. Adding any library (such as math.h) will receive a grade of 0 for the whole assignment.

Remark 3: A few points might be deducted if the code is ambiguous. Provide comments whenever the code is not clear or too complex. A good rule of thumb to know when to comment is to ask these 2 questions: Will you be able to quickly understand and update this part of your code if you looked at it in 5 years or if someone else wrote the code, will you be able to understand it in 5 years.