Cryptographic Engineering: Assignment 5

Submit a c code file + Quiz 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.

For this assignment, points in elliptic curves are defined using a struct (basically groups things together) of two 256-bit integers x and y. Using pointers is required in this assignment. The only thing you have to worry about is that when the point is defined as `point *P`, then you access the coordinates using `P->x` and `P->y`. When a point is defined as `point P`, you access the coordinates using `P.x` and `P.y` and you use `&P` to send it as a pointer to a function that requires a pointer.

All modular arithmetic operations from Assignment 3 have been provided for you. Use them to complete this assignment. Also, since the testbank takes a lot of time to complete in this assignment, 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.

1. Implement a secure **point addition function**. To be filled in function `point_add`.
2. Implement a secure **point doubling function**. To be filled in function `point_double`.
3. Implement a secure **point select function**. To be filled in function `select_point`.
4. Implement a secure **point multiplication function** which return the x-coordinate of point multiplication. To be filled in function `select_point`.
5. Implement a secure **recover_y** function which recovers the y-coordinate from a point's x coordinate and the elliptic curve. To be filled in function `recover_y`.
6. Implement a secure ECDH **key generation** function which generates secret key and public key. To be filled in function `keyGen`.

7. Implement a secure ECDH **shared secret generation** function which generates the shared secret from a secret key and a public key. To be filled in function `sharedSecret`.

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

Assignment B

1. **ECDH:** 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 = 70a4b2148fdd56d85cbaf80a012f312787bd9b13bcd69d0be2927f6d843bf57c
   ```

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

   (d) Alice and Bob are attempting to generate a shared secret using this ECDH implementation through a communication line. Eve gained access to the communication line between Alice and Bob and can read any traffic Alice and Bob is trying to send and can modify it before it reaches the other party. Alice and Bob exchanged their public keys through this communication line and generated their own shared secret. Now Alice and Bob want to use the shared secret in AES to exchange messages. Can Alice and Bob detect that the messages have not been read or modified by a 3rd party (Eve)? If yes, show how. If not, show why.

   (e) Same scenario as in (d) but Alice and Bob are able to detect that the public keys are tampered with. Eve can still read the traffic Alice and Bob are sending and can modify it except for the public keys because she will be detected. Can Alice and Bob detect that the messages have not been read or modified by a 3rd party (Eve)? If yes, show how. If not, show why.

   (f) Name two advantages for using ECDH over regular DH.

   (g) (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).

2. **ECDH verification in SageMath.** Parameters are provided below

   (a) Write a SageMath code that computes the public key using the secret key generated in 1(a).

   (b) Write a SageMath code that computes the shared secret using the public key provided in 1(b).

   ```
   Parameters:
   p = 2^255-19
   a  = 0x2aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa984914a144
   b  = 0x7b425ed097b425ed097b425ed097b425ed097b425ed097b4260b5e9c7710c864
   Gx = 0x2aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad245a
   Gy = 0x5f51e65e475f794b1fe122d388b72eb36dc2b28192839e4dd6163a5d81312c14
   ```

Remark: Use the SageMath functions provided in the prelab. **DO NOT** re-implement the functions in Assignment A.

Grading

	Exercise	Grade	Note:
Assignment A	1	10 pts	
	2	10 pts	
	3	10 pts	
	4	10 pts	
	5	20 pts	
	6	10 pts	
	7	10 pts	
Assignment B	1 (a)	10 pts	
	1 (b)	10 pts	
	1 (c)	10 pts	
	1 (d)	9 pts	3 pts for yes/no. 6 pts for explanation.
	1 (e)	9 pts	3 pts for yes/no. 6 pts for explanation.
	1 (f)	2 pts	
	1 (g)		+80 pts for correct answer and -10 pts for wrong answer. This is a bonus question and will increase your final grade. Incorrect answer will decrease the grade for this assignment. Leave blank to not lose points!
	2 (a)	10 pts	
	2 (b)	10 pts	
	Formatting		-5 pts. To note 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	150 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.