

Introduction to FinTech

Assignment for Bitcoin/Blockchain

★ Welcome to the Exciting World of Cryptocurrency! In this assignment, you'll embark on a journey to explore the fascinating technical foundation behind Bitcoin and blockchain. Get ready to dive deep into the world of cryptography and understand how these groundbreaking technologies work at their core.

Using the elliptic curve **secp256k1**, the same one employed by Bitcoin and Ethereum, you'll learn to work with fundamental concepts like elliptic curve arithmetic and the base point **G** defined in the standard. This is your chance to connect theory with real-world applications in the cryptocurrency ecosystem.

Don't worry if it feels complex at first; think of this as a puzzle waiting to be solved! Tackle the challenges step by step, experiment, and let your curiosity guide you. By the end, you'll not only deepen your understanding but also gain confidence in handling advanced cryptographic tools. Have fun, and enjoy it!

The assignment package contains the following files:

1. **Assignment4.pdf**: This Assignment specification.
2. **main.py**: The testing framework for your reference only.
3. **mySubmission.py**: The ONLY file you need to submit to the gradescope for grading.
You are not allowed to import any additional package, or you will receive "0" grade for this assignment.

Assignment Details:

Please read this paragraph carefully. It can save you tremendous time on this Assignment.

1. **main.py** is the main framework TA used for reference only. You don't need to submit this file to gradescope.
2. You can browse how we grade your submission on each Problem through **main.py**. Then, you can start to complete each corresponding function in **mySubmission.py**.
Ex. `python3 main.py your_private_key < testdata.txt`
This assignment is supposed to be completed by simply completing the predefined function body without designing any additional functions.
3. You are encouraged to utilize TA hour to help you initialize your assignment. The TA, Wen Hsiao, who has office hours at 9am on Thursday is responsible for this assignment.
4. TA has the responsibility to help you clarify the understanding of the assignment, and coach you to learn all necessary concepts on solving this assignment. TA has NO responsibility to assist you on the debugging of your assignment.

The Problem Set (105%):

Problem 0. (2%, realtime grading) Find the correct base point used in Bitcoin's public-key cryptography. Hint: See reference hints in code

Problem 1. (2%, realtime grading) Evaluate $4G$. Hint: Lecture slide p14

Problem 2. (2%, realtime grading) Evaluate $5G$. Hint: Lecture slide p14

Problem 3. (9%, realtime grading) Evaluate $Q = dG$.

Problem 4. (20%, post-grading with hidden data) Write a standard Double-and-Add algorithm mentioned during the class for scalar multiplications, and return the scalar multiplication result, and the number of doubles used and the number of additions used respectively when evaluating Q ? Hint: Lecture slide p16

Sample Input: 4 31

Sample Output:

6a245bf6dc698504c89a20cfdded60853152b695336c28063b61c65cbd269e6b4 e022cf42c2bd4a708b3f5126f16a24ad8b33ba48d0423b6efd5e6348100d8a82 4 4

Problem 5. (40%, post-grading with hidden data) Note that it is effortless to find $-P$ from any P on a curve. If the addition of an inverse point is allowed, try your best to evaluate dG as fast as possible. Then, use the optimized Double-and-Add algorithm to compute arbitrary scalar multiplications. Hint: $31P = 2(2(2(2(2P)))) - P$.

Sample Input: 5 31

Sample Output:

6a245bf6dc698504c89a20cfdded60853152b695336c28063b61c65cbd269e6b4 e022cf42c2bd4a708b3f5126f16a24ad8b33ba48d0423b6efd5e6348100d8a82 5 1

Problem 6. (15%, post-grading with hidden data) Sign given Bitcoin transactions with a random number k and your private key. Hint: Lecture slide p31

Sample Input: 6 4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b

Sample Output: Depends on the private key you used

Problem 7. (15%, post-grading with hidden data) Verify a given digital signature with your public key. Hint: Lecture slide p32

Sample Input: 7 signature_from_problem_6

Sample Output: True

Frequently Asked Questions (FAQ):

Q1: Should I import other packages?

A1: No. This assignment was designed without the need of non-build-in packages.

Q2: Should the initial configuration in Problem 4 be calculated as one addition?

A2: No. Please follow the convention in lecture slides.

Q3: How can I test my program on my desktop?

A3: `echo [problem#] [test_input] [expected_output] | python main.py [private_key]`

For more information, please refer to main.py.

Q4: Discussions on Problem 5.

A4: Practically, the double operation is slightly faster than point addition, so the optimized number of ((double + addition) should be smaller or equal to the non-optimized one. If the equality holds, the algorithm using less point addition is better.

Q5: What is the value range of d ?

A5: d is a value under field F_p , so it should be ranged at $[0, p)$.

Q6: What $0 * G$ is not $(0, 0)$?

A6: $(0, 0)$ is not a point on the Koblitz curve (secp256k1).

$0 * G$ point to the origin of F_p , denoted O . We call it zero point or point at infinity (*ellipticcurve.INFINITY*). For more information, please refer to lecture slide page 15.

Q7: Which version of ecdsa package is used for grading?

A7: We use ecdsa v0.19.0. However, only *ellipticcurve.PointJacobi* and *ellipticcurve.INFINITY* in this library are used. The version might not be an issue to this problem.

Q8: What are the input and return types of `sign_transaction()` in Problem 6?

A8: `(int, int) = sign_transaction(int, str, callback function, callback function, callback method)`

Be noted that hashID is a hashed transaction ID. For more detailed information, please refer to the comments in *main.py*, including how your implemented function will be invoked.

Q9: What is the meaning of PASS on Gradescope?

A9: PASS means your result for this test data is correct.

Q10: Is there any time constraint on this homework?

A10: A 10-minute bond was set on the grading server. However, all problems in this assignment are supposed to be completed in 1 second. Time constraints should not be a concern in this assignment.

Q11: Will the *callback_randint* be fixed to *random.randint*?

A11: You can view it as fixed to *random.randint*. Actually, we might change it with another overridden function to know which random nonce you are using but keeping the same prototype of the callback function. Therefore, you can directly assume that it is just the *random.randint*.

Q12: In Problem 5, when adding two points, is one of the points mandatory to be P or $-P$?

A12: No, we don't have this limitation. Problem 5 requests you to implement an optimal algorithm to calculate $n * P$. Our optimized solution was considered the optimal. If you believe your solution is optimal(better) than the one grader uses but failed by the judge system, feel free to submit your code for regrading after the deadline. We will manually review your code and give you a bonus if the algorithm is proven correct. We hope this mechanism makes you feel at ease without worrying about the algorithm grader used. Just focus on the optimal solution as best as you can. :)

Q13: Can you give me a sample and its corresponding optimization result?

A13: $31P$ takes 4 doubles and 4 additions in the traditional *double_and_add* algorithm.

However, the optimized algorithm takes only 5 doubles and 1 addition.

As to the definition of "optimal," please refer to Q4 and Q12.

===== END OF ASSIGNMENT =====