SPR200 Final Exam

Seneca Polytechnic
April 15, 2025

Student Submission

Course Number: SPR200
Section Number: NAA
Student Name: Zakariya Outbih
Seneca ID/Email: zoutbih@MySeneca.ca
Student Number: 100184233

Question	Full Mark	**Points**
1	2	
2	8	
3	10	
Total	20	

Instructions:

[label=0.]This examination contains 6 pages, including this page. You have **100 Minutes** to complete the exam. Make sure you submit before the time ends. You may submit unlimited number of times. List of websites you are allowed to visit during the exam: BlackBoard (learn@Seneca), GitHub.com, Overleaf.com. You are not allowed to visit any other websites. Submission Guidelines:

4.
   - Write your answers in the .tex documents, and compile `main.tex` to .pdf and upload only the .pdf file to the Blackboard.
   - Submit BOTH of the .tex and .pdf files to your GitHub.com repository in the folder of `Final-Exam` and make the tag name `Final-Exam`.
   - If you miss submitting to either Blackboard or GitHub in time, you will unfortuneately get a ZERO(0) score and you'll have to retake this course. If two versions of your submitted files are different, the one on GitHub.com will be used.
   - Please make sure you have enough time to submit your files, late submission will not be accepted and you will get ZERO(0) score.

5. You may leave the classroom at any time and not come back during the exam EXCEPT during the last 10 minutes. During the final 10 minutes, you must remain in the classroom until the exam ends.

6. Please sign the Honor Code statement by below filling in your name.

In recognition of and in the spirit of the Seneca Polytechnic, I certify that I will neither give nor receive unpermitted aid on this examination.

Signature: Zakariya Outbih

Part 1: Read the Source Code and Compile to PDF [2 Points]

This is basically free 2 marks for you, but you should still be careful to:

1. Read each line of the source files (Important: also the comments). If you don't, you lose 2 points.
2. Fill in what are required in the first page. If you don't get everything right (e.g., have your course section number wrong, or your name is not in the right format, you filled your student number in your Seneca ID field, etc.), you lose 2 points.
3. Complete all questions in `main.tex`.
4. Compile `main.tex`. to a .pdf file, file name should be [Firstname].[Lastname].[SenecaID].pdf (e.g., "John.Doe.jdoe2.pdf"). If you get any parts of the file name wrong, you lose 2 points.
5. Restrain yourself from asking about what to submit – it is all in the instructions. But if you really need to ask "Is it the right file/style/file name/.. to submit?" You will get a confirmation, but lose the marks of this question.

The .pdf file name I submit to BlackBoard is:

Zakariya.Outbih.zoutbih.pdf

If you have issues with .pdf file compilation and you cannot fix it, you can submit a .tex file in addition to the .pdf file bu tyou may lose some marks.

Part 2: Multiple Choice Questions [8 Points]

You may choose one of more options for each question.
You need to get all the options right to get marks.

Example Question and Answer (you don't need to answer this question):

When writing an initial email to your professors for discussing a formal academic matter, which **salutation** is considered to be impolite/annoying and might result in a bad impression?

A Hey Professor Huang
B Hi Teacher
C Hi Sir
D Hi Wei
E No salutation, just start with the body of the email.

Answer: B C E

Explanation (You don't have to provide explanations in your submission):

"Hi Teacher" (Option B) Using "Teacher" as a name substitute shows a lack of effort to address the professor respectfully by name and position, and is generally considered inappropriate or annoying in professional academic communications.

"Hi Sir" (Option C) is inappropriate because "Sir" shows you don't even bother to address the professor's actual name or title, potentially signaling a lack of attention or courtesy.

"No salutation" (Option E) Obviously, it is not polite to not address your email recipient when starting the email thread.

On the other hand:

"Hey Professor Huang" (Option A), while informal ("Hey"), still correctly acknowledges the professor's title and last name, and might be acceptable in a slightly casual context.

"Hi Wei" (Option D), addressing by first name, might be appropriate depending on the context and whether your professor is a casual person, for example, in this SPR200 course, your professor is happy to be addressed by just his first name. However, not everyone else is.

Question 1: We have talked about all the following algorithms in class. Which one(s) utilize(s) elliptic curve cryptography?

A AES
B EdDSA
C ECDSA
D RSA
Question 1 Answer: B C

Question 2: Which property(ies) is/are NOT desirable in a cryptographic hash function?
A Fast computation
B Collision resistance
C Preimage resistance
D Easy reversibility
Question 2 Answer: D

Question 3: When we are talking about Blockchain Technology in class, we mentioned that the blockchains commonly employs which cryptographic method(s)?
A RSA encryption
B AES encryption
C SHA-256 hashing
D ECDSA signatures
Question 3 Answer: C D

Question 4: RSA encryption's security primarily depends on the complexity of:
A Hash collision problems
B Prime number factorization
C Elliptic curve discrete logarithm
D Integer overflow issues
Question 4 Answer: B

Question 5: What is a fundamental objective of zero-knowledge proofs?
A Efficiently encrypt data
B Verify information without revealing extra knowledge
C Rapidly compute hash values
D Digitally sign documents
Question 5 Answer: B

Question 6: ECC provides advantages over RSA such as:
A Resistance against quantum computing
B Shorter cryptographic keys
C Guaranteed provable security
D Improved computational efficiency
Question 6 Answer: B D

Question 6: Post-quantum cryptography is primarily focused on:
A Increasing RSA encryption speed
B Making encryption quantum-based
C Securing data against quantum attacks
D Removing the need for hash functions
Question 6 Answer: C

Question 8: Which foundational mathematical problems are used in post-quantum cryptography? (hint: this question has more than one choice as the right answers)
A Integer factorization
B Elliptic Curve Discrete Logarithm Problem (ECDLP)

C Lattice-based computations
D Solving multivariate polynomial equations
Question 8 Answer: C D

Question 9: Bitcoin mining primarily requires solving:
A RSA encryption problems
B Elliptic curve discrete logarithms
C Cryptographic hash puzzles
D Integer factorization challenges
Question 9 Answer: C

Question 10: Which statement correctly describes asymmetric cryptographic operations?
A Secure key distribution beforehand is always needed
B Encryption is done using public keys, decryption using private keys
C Both encryption and decryption require the same key
D Digital signatures can involve private key encryption and public key decryption
Question 10 Answer: B D

Part 3: Practical Tasks [10 Points]

Your client provided you files of a pair of ECC cryptographic keys, but they did not tell you which ECC curve they are using. Now your client went camping in a place without cellphone signal or internet access ("I hope they suffer the life without internet," you think, "they are so clumsy and careless and I hope they don't burn their campsite or left food outside to get eaten by bears!").

Anyways, now you are here to complete the following simple tasks in a few minutes (hopefully) with the keys your client provided, and then you can get off work early and enjoy your weekend with your loved ones too. Here is a list of the files your client left for you:

- `private-key.pem`
- `public-key.pem`
- `client-message-to-sign.txt`
- `client-signature.txt`

Task 1: Find out what ECC curve is used when the client generated the key pair. [2 points]

(Hint-1: In the lab, you use `openssl` to generate an ECC public key from a private key. How does the `openssl` command know which ECC curve to use to generate public key from private key? It must know, right? But how?)

(Hint-2: All details of openssl commands can be found in the manual, e.g., `$man openssl ec`)

The ECC curve used by the client is:

secp521r1

How did you find out?

openssl ec -in private-key.pem -noout -text

Task 2: Print Out the Signature Clients Left. [2 points]

So the client left you a signature: `client-signature.txt`. Now you need to print it out and send it to someone else along with the message `client-message-to-sign.txt`, so they can verify that it is indeed the client's signature on the message.

However, the signature is not readable when you try to open it, so you need to print it out in hex format. You wrote a python/shell program to do this and the output of your program by executing the following command is the hex format content of the signature file:

```
$ python task2.py
```

or,

```
$ ./task2.sh
```

The client's signature in hex format is:

3081 8802 4201 4c6a 50a9 6a9d 19c1 e021 4c31 cec3 fc39 1a57 9fed 53cf dc40 c7ba ea4c 4ead 1a1f 9b32 4d6b 899c 96d3 9f11 0dd6 cd16 05f1 8982 f33d f5db e2a3 661e 3403 18df e364 3002 4201 9fa3 78a5 3ef5 cf3a 0242 9d16 5635 4bed cadb 8613 8101 f34c 4128 e47b e4f0 40da b896 a528 4b6f 37e3 c694 2e9e edce 5a59 3f0e b75a 86d5 3eb4 52b2 527c f090 41eb 8a

If you used a python program to solve it, please provide (task2.py) and submit it on GitHub in the same folder of your other files.

If you used a shell script to solve it, please provide (task2.sh) and submit it on GitHub in the same folder of your other files.

Task 3: Verify the Signature. [2 points]

After you sent the signature and message file to someone else, they immediately responded to you:

> Dear XXX,
> I received your signature and message file. However, I cannot verify the signature because although I have the public key of the client, I don't know which hash algorithm the client used to generate the digest of the message.
> Please let me know which hash algorithm the client used.
> Best regards,
> Someone Else

The client's message digest algorithm used was:

-sha3-512

In one or a few sentences, explain briefly how you found out. If you wrote a program to solve it, please include how you used the program in the explanation:

first I used this command: openssl dgst -list to list all possible digests, then I used the verify command for every digest replacing them each time until I got a Verified OK openssl dgst -sha3-512 -verify public-key.pem -signature client-signature.sig client-message-to-sign.txt. Essentially brute force

If you used a python program to solve it, please provide (task3.py) and submit it on GitHub in the same folder of your other files.

If you used a shell script to solve it, please provide (task3.sh) and submit it on GitHub in the same folder of your other files.

Task 4: Python Program to Verify the Signature. [2 points]

Someone else also has many client messages and signatures to verify. Please write a python program to verify the signatures of the client. The program should run like this:

When the signature can be verified successfully:

```
$ task4.py client-message-21.txt client-signature-21.sig public-key.pem

Signature verified successfully.
```

When the signature cannot be verified successfully:

```
$ task4.py client-message-136.txt client-signature-136.sig public-key.pem

Signature verification failed.
```

Please provide (task4.py) and submit it on GitHub in the same folder of your other files.

Copy and paste the 9th non-empty line of your (task4.py) source code here:

Task 5: Answer your client's question about post-quantum cryptography. [2 points]

Your client is a bit paranoid about the recent news about quantum computing and its potential impact on the security of the cryptographic systems they are using. They are concerned whether the signature scheme they used is still safe when quantum computers are powerful enough in the future. Please write an email to your client to explain:

1. What can quantum computers do?
2. What can quantum computers do to the security of the cryptographic systems?
3. Is the ECC signature scheme still safe when quantum computers are powerful enough?
4. If yes, why the current ECC signature scheme is still safe? If not, what can the client do if they want to improve the security of the signature scheme against future quantum attacks?

Write the email to your client in the following box:

Dear Valued Client,
Thank you for your inquiry regarding the security implications of quantum computing on our current cryptographic systems. I appreciate your proactive approach to understanding these important security considerations.

I would like to address your concerns with the following detailed explanation:
Quantum computers can do alote of things that classical computers cannot. They do not operate like classical computers that use regular bits (1 or 0) they use qubits (1 and 0), they are highly effective at specialised tasks

Quantum computers can affect the security of current cryptographic systems specifically RSA and those using eliptic curve cryptography. Algorithms like shor's algorithm reduce the security of current cryptographic systems to the point that they are no longer secure, for example it can turn an aes 2 to the power 128 key into 2 to the power 56 which is very breakable

No the ECC signature scheme is not safe when quantum computers are powerful enough as explained above

To better protect against quantum computer on cryptographic algorithms I highly reccomend using post quantum cryptographic algorithms like sphincs+

I trust this information will assist you in making informed decisions regarding your cryptographic security strategy. Please do not hesitate to contact me if you require any clarification or have additional questions.
Best regards,
XXX

Submission Checklist

You need to submit your files to **BOTH** BlackBoard and GitHub.

On BlackBoard, you need to submit the following files:

- Your compiled .pdf file

On GitHub, you need to submit the following files:

- Your .tex file
- Your compiled .pdf file
- All the source codes of your programs including:
  - task2.py or task2.sh
  - task3.py or task3.sh
  - task4.py