CPSC 435/535 Algorithms, Project 1 - RSA & Digital signature, Spring 2025

Despite the threats posed by quantum computing, RSA and Elliptic Curve Cryptography (ECC) remain the most widely used public key cryptography algorithms. Your first programming assignment is to implement RSA Encryption and apply it to digital signature (http://en.wikipedia.org/wiki/RSA_(algorithm)). To facilitate the grading, you are to use **Python** to complete the project (see the Jupyter notebook template for the specifications). C++ doesn't have a built-in BigInteger library. If you are to do the project in C++, make sure you include the bigInteger package you use and detailed instruction on how to run your code in your submission.

Part I: RSA key generation.

- Implement Fermat test;
- Use Fermat's test to generate two large prime numbers (p, q), each should have a size between 128 and 256 bits (>=128 and <256);
- Save p and q in a file named p_q.txt, one integer per line and making sure no white space saved;
- Use the extended Euclidean algorithm to generate two pairs of keys: (e,n), (d,n), where n=p*q;
- Save the two pairs of keys in two separate files: e_n.txt and d_n.txt, one integer per line and no white space;

Part II: Generate and verify digital signatures using a SHA-256 hash.

- Unless you want to implement the hash function yourself, make sure you import Python hashlib.

```
1 import hashlib
2 print(hashlib.algorithms_guaranteed)
```

```
{'blake2b', 'shake_256', 'sha3_384', 'sha256', 'sha224', 'md5', 'sha3_224', 'sh
a3_512', 'shake_128', 'sha1', 'sha512', 'sha3_256', 'sha384', 'blake2s'}
```

- You will use sha256:

```
1 h = hashlib.sha256(b'computer science at UA is the best')
2 m = h.hexdigest()
3 m
```

```
'e697db5d9a543ecf04f2cc76fe105eb9d65edb08e06e2ef53788157bb05dc7e6'
```

- Sign a given file
  - Generate a SHA-256 hash of the content of the file to be signed (e.g., "file.txt");
  - Sign/"decrypt" this hash value using the private key stored in d_n.txt;
  - Combine the original content and the signature into one document filename.signed (e.g. "file.txt.signed"). Append the 64-byte signature (512/8=64) at the end of the original content. (Note: n has <512 bits)

- Verify the signed file
  - Separate the signature from the content of the file in the signed document (e.g. "file.txt.signed");
  - Generate a SHA-256 hash of the content of the file you have signed.
  - Check if the signature (old hashcode/m) = new SHA-256 hashcode/m.

What to submit.

1. **Submit your program.** Make sure to test your code on more than one set of data. DO NOT submit programs that are not *reasonably correct*! To be considered *reasonably correct*, a program must be completely documented and work correctly for sample data provided with the assignment.
2. This is a classroom project. If you use Python, I see one file is enough. Use **the template** to get it started.

Grading. Your code will be graded **on correctness**, efficiency, clarity, and elegance.