Lab 8:
Encrypting and Decrypting with RSA

In this lab, you will play out several encryption scenarios using simple 16-bit RSA.

The scenarios you will play out:

- Forging a message by manipulating a non-cryptographic hash
- Cracking encryption using factoring

Before you can play out these scenarios, you will need the following:

- Code to create and use a public & private key
- Code to hash data (with a non-cryptographic hash)

Once your code is written, assign the roles of Alice, Bob, and Trudy to each person.

Procedure

1. ***Download*** rsa.py
2. ***Put*** your names at the top of the file.
3. Create a ***design*** for the methods `create_keys`, `compute_checksum`, and `apply_key` in rsa.py. See the documentation for these methods in the rsa.py template.
4. ***Fill*** out the design for the methods in part 3.
5. **Bob:** ***Run*** the program with the `compute_checksum` option to create an encrypted checksum for the message "Bob owes Trudy $100.99". ***Save*** the public & private keys, as well as the encrypted checksum for your records. ***Provide*** Alice and Trudy with the public key. ***Provide*** Trudy with the message and encrypted checksum. (Suppose that Trudy is an unscrupulous online store…)
6. **Trudy**: ***Create*** a message that results in the same checksum as Bob's message, but implies that Bob owes a larger amount of money. Hint: If you rearrange the characters in the string, how does that change the checksum? ***Supply*** Alice with the forged message and the encrypted checksum that Bob gave you.
7. **Alice**: ***Check*** Trudy's message using the `verify_checksum` option of the program. Does it check out OK? If not, Trudy should keep trying. If so, how could Trudy be prevented from performing this trick in a real application? (Suppose Alice is the bank responsible for transferring the money from Bob to Trudy…)

8. As a team, create a ***design*** for the method `break_key`.
9. As a team, ***implement*** `break_key`.
10. **Bob**: ***Run*** the program and create a public key. Deliver this key to Alice. (You can reuse the key from Step 5 if you like.)
11. **Alice**: ***Create*** a secret message. ***Encrypt*** it with Bob's private key using the `encrypt_message` option of the program. ***Supply*** Bob and Trudy with the message. (You may need to email the hexadecimal characters to Bob and Trudy – or share them on IM.)
12. **Bob**: Run the program with the `decrypt_message` option to read Alice's secret message.
13. **Trudy**: ***Run*** the program with the `break_key` option to read Alice's secret message.
14. **Whole team**: In the comments at the end of the lab, ***answer*** the questions and ***comment*** about what you learned in the lab. Your comments should include:
    a. Answers to the questions included in the comments at the end of the template.

    b. A description of the functionality you implemented and the results of your testing.

    c. Comments on your experience in completing the lab, including any problems you encountered. Briefly explain what you learned.

    d. Questions and suggestions.