UC Computer Science and Software Engineering

COSC362 Data and Network Security
Semester Spring, 2021

Lab 7: CrypTool Part 3

Exercises from Lectures 12 and 13 (and a little bit Lecture 11).

Background

This exercise sheet has been written for use with CrypTool v2.1.

You can download the open-source CrypTool program at `https://www.cryptool.org/en/ct2-downloads` from the Windows Virtual Machine. Please download the stable version of CrypTool v2.1. You can get more information about the CrypTool program at `https://www.cryptool.org/en/`.

Exercise 1: RSA Encryption

We have a look at RSA encryption.

Cryptography → Modern → Asymmetric → RSA Encryption

Before doing anything, examine the selected values in the RSA Key Generator box (top left).

QUESTION 1

(a) Approximately how many decimal digits long are each of the primes?

(b) Approximately how many decimal digits long will the RSA public modulus n be?

(c) Approximately how many bits long will the RSA public modulus be?

(d) Is this RSA key large enough to be used in a modern application such as TLS (Transport Layer Security)?

Memorise the public key exponent e, then change it to 8. Play.

QUESTION 2

What just happened, and why?

Press Stop and then change the public key exponent e back to what it was. Play.

The ciphertext should now appear as a string of hex characters in the top right box.

Press the Data icon in the RSA Key Generator box (third icon from the left). Toggle between the public exponent e and the private key d.

cont/...

QUESTION 3

(a) These values are very different! Can you explain why?

(b) Mathematically, we could swap these values around. Why would this be a very bad idea from a security perspective?

Press the Settings icon in the RSA Key Generator box (left icon) and Stop.

We'll now investigate what happens when you choose larger (more realistic) key sizes.

In the RSA Key Generator box, choose:

- *Source → Generate random primes*
- *Number of bits*
- *Range → 512*

Play.

QUESTION 4

(a) How long did the encryption process take? Check the upper green bar.

(b) Press Stop and then change the range to 1024. Play.
How long did the encryption process take?

(c) Press Stop and then change the range to 2048. Play.
How long did the encryption process take?

(d) Do you think the delay was key generation, or encryption, or both?

(e) What practical lessons can we extract from examining these timings?

Exercise 2: RSA Security

This exercise will give you an indication of RSA security by examining the hardness of factorization. We will start by opening two fresh CrypTool templates from the Start Center.

Cryptography → Mathematics → Factorization with Quadratic Sieve (QS).

Cryptography → Modern → Asymmetric → RSA Encryption.

We will refer to these two templates as QS and RSA. Start by examining the default input number to be factorized in QS.

Note how big it is, then press Play.

cont/...

QUESTION 5

What happens next, and how long will this process take to complete?

Press Stop in QS and then open RSA to revisit our first RSA encryption example. Press Play in RSA. Press the Data icon in the RSA Key Generator box (third icon from the left) and copy the value in the Public Modulus box. Paste this number into the Input box of QS. Press Play in QS.

QUESTION 6

How long did factorization take?

Check that the results are correct by revisiting the values in RSA. Press Stop in QS and RSA. In the RSA Key Generator box choose:

- *Source → Generate random primes*
- *Number of bits*
- *Range → 150*

Press Play in RSA. Press the Data icon in the RSA Key Generator box (third icon from the left) and copy the value in the Public Modulus box. Paste this number into the Input box of QS. Press Play in QS.

QUESTION 7

How long is factorization going to take this time? Be patient, there may be a delay before you get this information.

Repeat the above steps for an RSA modulus of 160 bits, and again for 170 bits.

QUESTION 8

What is this telling you about the security of RSA?

Exercise 3: Using a Hash Function to Protect Passwords

This first part of this exercise just checks that you recognise one important feature of a hash function.

Hash functions → SHA-1.

cont/...

The box on the left should contain some text that represents the input to the hash function (the data we want to hash). Feel free to use the default data, or replace it with anything you like.

Play.

The output of the hash function should now appear in the right hand box.

QUESTION 9

How many bits long is the output of the hash function?

Make a very small change to the input to the hash function by editing the box on the left. Then observe the resulting changes to the hash function output in the box on the right.

QUESTION 10

What do you notice? Is this what you would expect to happen?

We'll now see why choosing basic passwords and simple hashing of passwords is insecure.

Now place the word "alligator" into the Input box. Play. Copy the hash output. Go to Start Center.

Hash function → Dictionary attack on a password hash value.

Change to SHA-1 in the SHA box. Paste the hash output into the Test Password box. Play.

QUESTION 11

(a) How many English dictionary words were tested before the password was recovered?

(b) How could this search have been made more difficult to conduct?

We will now look at a different technique for protecting passwords.

Hash function → PBKDF-1.

QUESTION 12

(a) What does PBKDF stand for? Look online.

(b) How long is the PBKDF-1 output? Look online.

(c) How has PBKDF-1 made it harder to conduct a dictionary attack? See slide 19 from Lecture 11.

cont/...

(d) We have been thinking about the use of PBKDF-1 for protecting passwords, but what else could you use PBKDF-1 for? Look online. (Hint: check in the full name of PBKDF.)

Exercise 4: Other Public Key Encryption Algorithms

We have been focusing on RSA, since this is the most well established, and arguably the simplest, public key encryption algorithm. There is one other important family of public key encryption algorithms in use today. These are based on elliptic curves.

If you wish to get an idea of how elliptic curve based public key encryption works, then it is suggested you search for details about the Elgamal public key encryption algorithm, which is a simpler mathematical algorithm to understand and is the one on which elliptic curve algorithms are based (see Wikipedia article "Elgamal encryption"[1] and Lecture 13 (from slide 13)).

QUESTION 13

What is the advantage of elliptic curve based public key encryption over RSA? Check slide 24 of Lecture 13.

A major frontier in cryptography is to seek new public key encryption algorithms fit for the future.

Visit `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`[2] *to read about a major initiative to design new public key encryption algorithms.*

QUESTION 14

(a) Why is NIST running this competition?

(b) What stage has the NIST competition got to, and when is it expected to conclude?

[1] `https://en.wikipedia.org/wiki/ElGamal_encryption`

[2] and also `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization`