ASYMMETRICAL CRYPTOGRAPHY

Qcamp 2018, Experimental Session 01/06/2018

1 Introduction

In Mission 1, you construct a basic communication channel which uses infrared light as the communication medium. However, as we have learned over the weeks, this communication channel is not secure. Not even the channel that you usually use to access the internet!

In Mission 2 and 3, we will explore a quantum key distribution (QKD) method to secure the channel. However, QKD is not the current way to keep internet communication secure. The security of our internet connection at this moment relies upon an implementation of asymmetrical cryptography.

This exercise will explore a basic pedagogical implementation of kid-RSA [1], which hopefully will get you to appreciate the big picture of the internet cryptography business. If you are interested to go deeper and geeky into this subject, this article [2] is a good place to start.

1.1 Main Idea

Asymmetrical cryptography (also known as public-key cryptography) works "somewhat" like this:

1. From some "random" numbers, Alice generates a pair of keys, known as the public key and the private key. Alice publishes the public key so everybody can get it. The private key stays with Alice and is kept secret by her.
2. Bob takes Alice's key and "locks/encrypts" his message with the public key, rendering it unreadable to anybody else than the private key owner (Alice). Bob then sends the encrypted message to Alice.
3. Alice can then "open/decrypt" the message with the private key.

The key idea in this scheme is that, the key to "lock" and "open" the message are two different keys, hence the name asymmetrical cryptography. The main assumption is that from the public key, it is "extremely hard" to obtain the private key.

One can quantify the "(extreme) hardness" of obtaining the private key from the public key by estimating the number of years it takes to crack the message. The record for the longest RSA key length broken is 768 bits, and it took them an equivalent of almost 2000 years of computing on a single-core 2.2 GHz AMD Opteron-based computer [3]. The internet nowadays uses 1024 or even 2048 RSA-bits, and the time it takes to crack them with today technology is probably much more than the current age of the universe. So far no one has successfully crack beyond 768 bits, but if you want to try your luck, you can try [4].

[1] Neal Koblitz., Cryptography As A Teaching Tool, https://sites.math.washington.edu/ koblitz/crlogia.html

[2] How RSA Works: TLS Foundations, https://fly.io/articles/how-rsa-works-tls-foundations/

[3] In case you are wondering, they used a lot of computers in parallel. https://eprint.iacr.org/2010/006.pdf

[4] RSA numbers, https://en.wikipedia.org/wiki/RSA_numbers

1.2 Kid-RSA

Kid-RSA is very similar to RSA, as it has most of the properties of RSA encryption algorithm. However, it is not secure as it can be broken by mathematicians who have studied number theory [5].

Follow these steps to implement kid-RSA:

1. Choose four "random" numbers a, b, a', and b'.
2. Evaluate the following numbers:
   - $M = a \times b - 1$
   - $e = a' \times M + a$
   - $d = b' \times M + b$
   - $n = (e \times d - 1)/M$
3. From these numbers, e and n are the public keys, and d is the private key.
4. To encrypt the message P, use the operation $C = e \times P \pmod{n}$.
   Note that the message P is an integer and can only have values between 0 and $n - 1$.
5. To decrypt the ciphertext C, use the operation $P' = d \times C \pmod{n}$.

For those of you who are unfamiliar with modular arithmetic $y \pmod{z}$, it is the remainder of the division y and z. For example, by dividing 13 by 4:

$$13 = \mathbf{3} \times 4 + \mathbf{1}$$

The division result is 3 and the remainder is 1. Thus, we can write $13 \pmod{4} = 1$.

Now, for the kid-RSA to work nicely, we require that n is an integer (see Assignment below), and that $P' = P$, by performing decryption on the encrypted message, you will obtain the original message:

$$\begin{aligned}
&P' = d \times C \pmod{n} \;= d \times e \times P \pmod{n}\\
&P' = (n \times M + 1) \times P \pmod{n}\\
&P' = (n \times M \times P) \pmod{n} \;+ P \pmod{n}\\
&P' = 0 + P \pmod{n}\\
&P' = P
\end{aligned}$$

Note that $n \times y \pmod{n} = y$ and P can only have values between 0 and $n - 1$.

1.3 Representation

To perform the kid-RSA operation above with characters or words, we need to find some way to represent those words into numbers. In the experimental session, we are using ASCII representation, but for the purpose of this exercise, we will use a simpler 26 alphabet-number representation.

[5] If you are interested, you can read about Extended Euclidean Algorithm. There is even an online calculator version in https://planetcalc.com/3298/

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Table 1: Conversion table from alphabet to number.

For example, if you want to write the word "WORD" in this representation, it would be:

$$(\text{WORD})_{26} = \mathbf{23} \times 26^3 + \mathbf{15} \times 26^2 + \mathbf{18} \times 26^1 + \mathbf{4} \times 26^0$$
$$(\text{WORD})_{26} = 404248 + 10140 + 468 + 4 = 414860$$

2 Assignment [10 pts]

Task 1 [2 pts] Verify that n is always an integer, i.e. $e \times d - 1$ is divisible by M.

Task 2 [2 pts] Find what word does the number 2490 represents in the 26 alphabet-number representation. Show your working steps.

Task 3 [2 pts] Try to encrypt the message $(\mathrm{QC})_{26} = 17 \times 26 + 3 = 445$. Show your working steps.

Task 4 [2 pts] Try to decrypt the encrypted message obtained in Task 3, and verify that you obtained $445 = (\mathrm{QC})_{26}$. Show your working steps.

Task 5 [2 pts] You overheard an encrypted message 78025 with the public keys $(e, n) = (12413, 323279)$. Try to crack the message, and express your answers in the form of $(???)_{26}$.
Hint: You can find the private key d from public keys (e, n) from the equation:

$$d \times e \pmod{n} = 1$$
$$d \times e + k \times n = 1$$

Also, you might want to consult some mathematicians to do this task.