Unit Name: Network Security and Resilience

Unit Code: TNE30009

Title: NSR/AS Lab 5 – Public Key Cryptography

Name: S M RAGIB REZWAN

ID: 103172423

Contents

Abstract:

This is a lab based upon TNE30009's week 10's lab4 task [1]. It begins with an "Introduction to Public Key Cryptography" section which contains brief background on what Public Key Cryptography is, its relation to RSA, alongside a brief explanation on the RSA algorithm. After this comes the "Breaking into the RSA algorithm" section where I explain how I broke the algorithm on Mathlab to obtain the private key (along with snippets of the codes that I used to break it). This links to the "Results" section where I use the values stored in the private key to run the decrypt function provided on the assignment page [1] (alongside the relevant parameters) to successfully decrypt the two messages. Lastly, it also has a "Conclusion" section where I summarize the main points of the report.

Introduction to Public Key Cryptography:

Public Key Cryptography is basically a way to encrypt messages passed between two known or unknown devices without having to exchange any confidential information before the communication **[2].** This ensures that the communication channels produced is both open and secure **[2].** That's because the details about the values used to form the keys are not released to any eves-dropper in setting up the channel (unlike in Diffie Hellman) and instead a pair of keys are used to decrypt and encrypt messages (where public key of user is revealed to everyone and used to encrypt messages sent to user, whilst the private key of user is kept fully hidden and used to decrypt those messages)

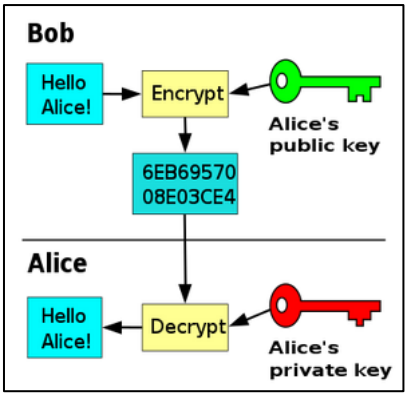

Figure 1: How encryption is performed using Public Key Cryptography when passing messages from user Bob to user Alice **[3]**

Although there are various ways to implement this technique, the most popular way to set it is by using RSA developed by Rivest, Shamir, Adleman in 1977 **[4]**:

1) Finding two large prime numbers and storing them into variables called "**p**" and "**q**". This will never be used to directly encrypt the messages and instead will be used to develop the keys that will be used to encrypt them. **[2].** Here, the primes numbers used must be very large to provide security against brute forcing and reverse engineering of the algorithm. Furthermore, by using prime numbers the n value that is produced will always be unique (ie not obtained by multiplying any other values)
2) Once this has been done, they are multiplied to get a value which will be stored in **"n".** This will be used as part of both public and private key and hence will be utilized in both encrypting and decrypting the message **[2].**
3) Then **"x"** will be computed by multiplying the numbers **"p-1"** and **"q-1"**. This will be used to choose e and calculate d later on **[2].**
4) Then an integer **"e"** will be chosen which needs to be relatively prime to **"n".** Relative primes are numbers who does not have any common factor between them, but are not necessary prime numbers

themselves. This will be used to make the **public key[n,e] [2].**

5) Then the modulo inverse of **"e"** with the base **"x"** will be calculated and stored in the variable **"d".** This will be used to create the **private key[n,d] [2].**
6) Now, that the keys have been made the messages send to the user will be encrypted using the public key and decrypted using the private key **[2].**

Since it is difficult to properly explain the calculations with words, a picture of the process **[2]** has been added below with sample values:

To create the public key select two large positive prime numbers p and q	p = 7, q=17
Compute n = p*q	n = 119
Compute x = (p-1)*(q-1)	x = 96
Choose an integer e which is relatively prime to x	e = 5
Public key is then [n, e]	[119, 5]
To create the private key compute d with (d*e) mod x = 1	d = 77
Private key is then [n, d]	[119, 77]
Data to encrypt is m	m=19
To encrypt m, compute c =(m^e)mod n	c = 66
To decrypt c, compute m=(c^d)mod n	m = 19

Figure 2: RSA calculation steps with sample values

Breaking the RSA algorithm:

From the previous process, we can see that in order to create the keys, certain linked calculations are performed in certain stages. Furthermore, we can also see that the root values used to perform all these calculations are the value **"p"** and **"q".** Thus, as long as we factorize the value **"n"** and obtain those values and have the value **"e"**, we can easily obtain the **"d"** value. Or in simple English, as long as one has the public key, it is possible to create it equivalent private key!

*[**Note:** Although it's possible to get the private, it's good to keep in mind being possible and being possible within time are two different aspects. Thus although this algorithm is breakable, if extremely large prime values of p and q are used, it will not be possible to get the private key within time using this method]*

Hence, I have written a simple function on Math lab that will perform the following calculations on my behalf:

1) It will basically take in **"n"** and **"e"** value (along with cipher **"c"** and store them in their respective variables)
2) Then it will factorize **"n"** to obtain **"p"** and **"q"** values and stores them.
3) Then it will find **"x"** value by multiplying "**p-1"** and "**q-1"** and store it.
4) Then it will run a simple loop for **"i"** value between **"1"** and **"n"** and store the modulus output of **"e*i"** and "**x"** in **"j".** Furthermore, it will keep on storing the value of **"i"** into **"d".** Once **"j"** value comes as **"1",** it means that the value of **"i"** is the modular inverse of **"e"** with respect to base **"x".** Hence the value passed to **"d"** will be the modulo inverse of **"e"** and hence will be the value that has been used to create **private key[n,d]**
5) Then it passes it to the string decryptor function (provided on canvas **[1]**) to decrypt the string using the private key in order to obtain the message

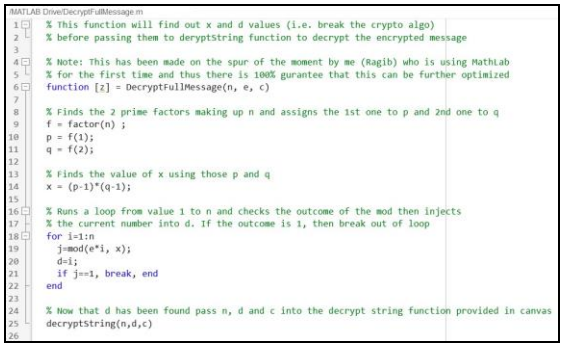

Figure 3: The algorithm used to break the RSA

Results:

When the function (i.e. the algorithm) was run using the values provided in the lab task **[1]**, it provided the following output.

Figure 4: Mathlab decryption output for both cipher cases

[Note: Here I have stored the following values in the respective variables:

1) **"n1"** and **"e1"** stores the values in the public key for first cipher,
2) **"c1"** stores the first cipher text,
3) **"n2"** and **"e2"** stores the values in the public key for first cipher,
4) **"c2"** stores the first cipher text]

```
c1= [2050 2296 640 479 640 2377 1274 479 640 2377
2395 194 476 2377 2395 602 2014 640 1205 2377 476
1888 2377 640 1142 1421 479 602 2014 2395 586 476
1142 749 2377 476 1142 640 2377 2395 2296 1274
2395 2377 194 586 1285 1285 2377 2014 479 640
1904 640 1142 2395 2377 602 476 540 479 2377 1205
586 1205 2395 640 479 2377 1888 479 476 2011 2377
479 640 1274 1741 586 1142 1019 2377 602 476 540
479 2377 1741 586 1274 479 602 2377];

n1=2407

e1=57

c2 = [2980 3647 1145 7023 4485 3647 7130 7023
6069 5363 2980 6069 7023 3911 2971 5943 5943 7023
1889 5561 7130 454 7130 3647 6069 7023 3243 4485
2957 5561 7023 5465 4485 454 7130 5561 3647 1883
7130 3647 6069 656 7023 6689 2206 5561 2957 4580
7130 7023 6238 4580 5363 3647 7130 2971 5561
1603];

n2=7663

e2=89
```

Figure 5: Values stored in the n1, e1, c1, n2, e2 and c2

Conclusion

Overall, the report I have created provides brief information regarding what a public key cryptography is and its link to RSA (alongside a brief introduction to the steps used to set up public and private keys using RSA). After that, it goes through explaining how I broke the algorithm and provides the function that I had developed to do so (which uses both information on RSA steps and also the function DecryptStrings provided on Canvas **[1]**). Then it shows the resulted output of the codes to ensure that the function can indeed break the RSA encryption.

Furthermore, through this report, it has been possible to understand that RSA is considered secure not because it's unbreakable, but because it is unable to be broken within a feasible amount of time (seen in cases when extremely large primes are used for p and q).

References

[1] A/Prof. P.Branch (2023). TNE30009/TNE80009 – Laboratory Week10 [Portable document format (pdf)]. Available: https://swinburne.instructure.com/courses/49751/pages/laboratory-week-10?module_item_id=3185496 . Accessed 9th May 2023.

[2] A/Prof. P.Branch (2023). Cryptography-Asymmetric Key - Lecture twenty-one [Portable document format (pdf)]. Available: https://swinburne.instructure.com/courses/49751/pages/laboratory-week-10?module_item_id=3185496 . Accessed 9th May 2023.

[3] Wikipedia Contributors. "Public-key cryptography." *Wikipedia*, Wikimedia Foundation, 19 July 2019, https://en.wikipedia.org/wiki/Public-key_cryptography . Accessed 9th May 2023.

[4] Wikipedia Contributors. "RSA (cryptosystem)." *Wikipedia*, Wikimedia Foundation, 19 July 2019, https://en.wikipedia.org/wiki/RSA_(cryptosystem) . Accessed 9th May 2023.