NSR/AS Lab 5 – Public Key Cryptography – Dylan Rodwell – 105341089

Dylan Rodwell

(105341089)

105341089@student.swin.edu.au

# **Abstract**

This lab session used MATLAB to break a cypher text (c) encrypted with a simple RSA public key system. By exploiting the key’s small modulus size (n), it demonstrates how the algorithm can be compromised by factoring n into it’s secret prime components (p, q). These components can then be used to calculate Euler’s totient (x = φ(n)), and use it to brute force the private exponent (d), such that (d\*e) mod x == 1. MATLAB was used to create a script that automates this process and decrypts the message. The routine *decryptString.m* was used to decrypt the message. It’s “decryptString” function takes n, d, and c in as parameters and outputs the decrypted cypher (c) in plain text.

*-Key words: RSA, cryptography, public key*

# **Introduction to Public Key Cryptography**

Public key cryptography uses two virtual keys to encrypt and decrypt messages. The public key is used to encrypt a message, and only the related private key can be used to decrypt the message.

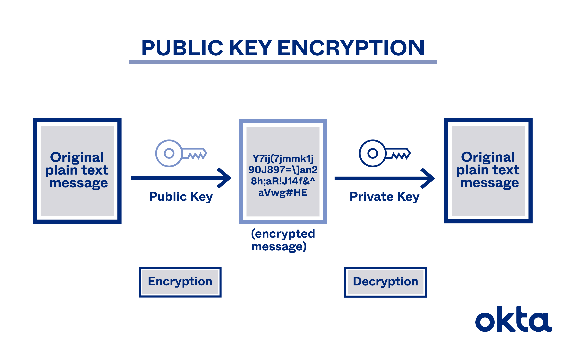


Figure . [1] Public Key Encryption Diagram.

To generate the keys, the RSA algorithm is used. In RSA, two large positive prime numbers are chosen and multiplied to create the modulus. The public key consists of this modulus and an encrypted exponent. The private key also uses this modulus and a decrypted exponent. The security of the keys depends on the difficulty of factoring the modulus. If the prime numbers selected are small or poorly chosen, as demonstrated in this lab, it is easy to factor out the primes from the modulus and calculate the private key.

A couple of people sitting at a desk

AI-generated content may be incorrect.

Figure . [2] RSA Key Exchange diagram.

# **Breaking the RSA Algorithm**

The first step to break the weak RSA key is to factor the prime numbers from the modulus “n”, giving us “p” and “q”.

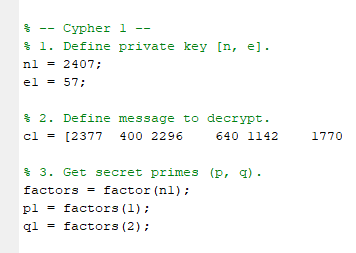


Figure . Break 1.

The next step is to compute Euler’s totient “x = (p-1) \* (q-1)”.

A white background with black and green text

AI-generated content may be incorrect.

Figure . Break 2.

After finding “x”, we can use a loop to brute force finding the decrypted exponent “d”. We know that “d” is the correct value by computing the modular inverse “mod(d\*e, x) == 1”.

A white background with text

AI-generated content may be incorrect.

Figure . Break 3.

Now we use the “decryptString” routine to decrypt the message “c” into plain text. The routine takes in the modulus “n”, the decrypted exponent “d”, and the message vector “c”.

A close up of words

AI-generated content may be incorrect.

Figure . Break 4.

# **Results**

**Message 1:**

A screenshot of a computer

AI-generated content may be incorrect.

Figure . Cypher 1 code.

A screenshot of a computer

AI-generated content may be incorrect.

Figure . Cypher 1 variables.

**Public Key 1=** [2407, 57]

**p1=** 29

**q1=** 83

**x1= (p1 - 1)\*(q1 - 1)=** 2296

**d1=** 1289

**Message1=**

“When public interest in cryptography was just emerging in the late seventies and early eighties, the National Security Agency made several attempts to quash it.

**Message 2:**

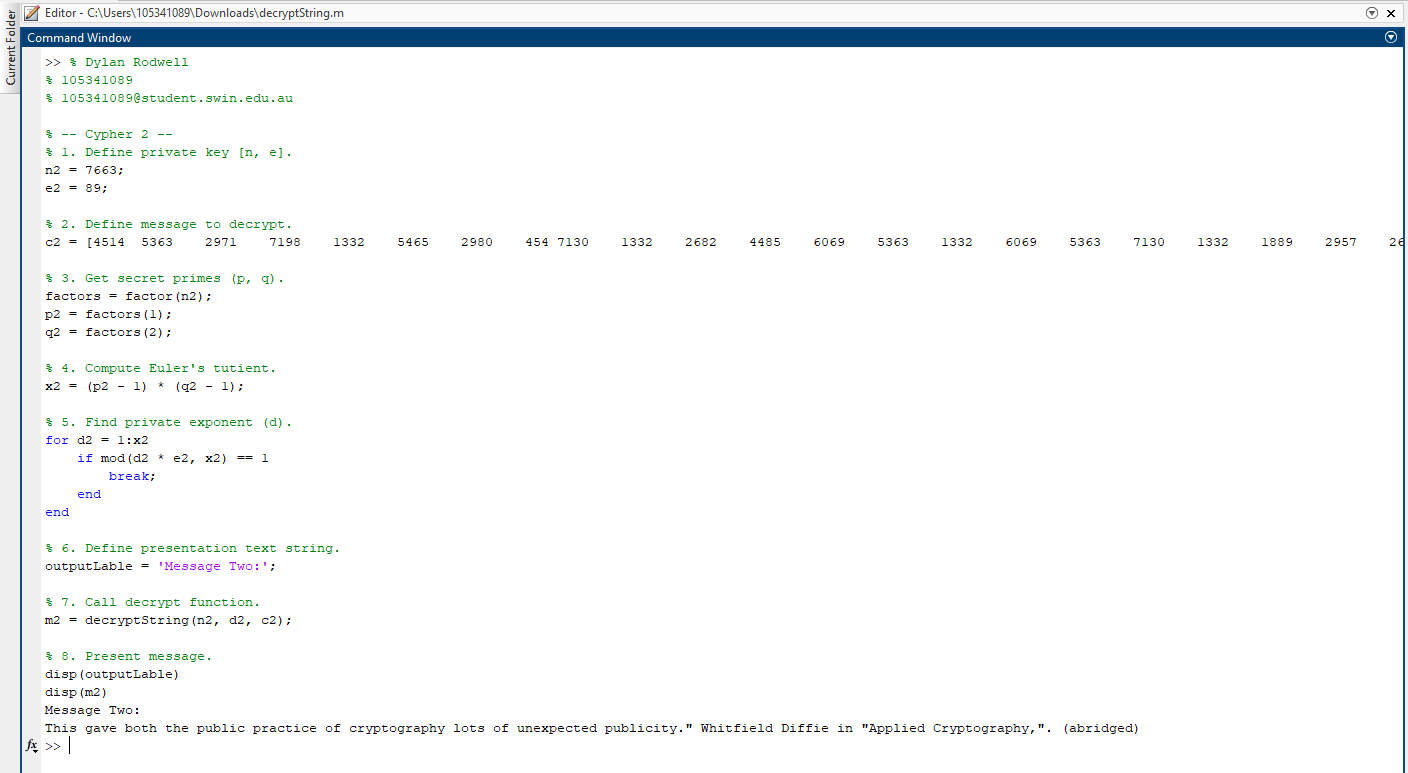


Figure . Cypher 2 code.

A screenshot of a computer

AI-generated content may be incorrect.

Figure . Cypher 2 variables.

**Public Key 2=** [7663, 89]

**p2=** 79

**q2=** 97

**x2= (p2 - 1)\*(q2 - 1)=** 7488

**d2=** 3113

**Message2=**

This gave both the public practice of cryptography lots of unexpected publicity." Whitfield Diffie in "Applied Cryptography,". (abridged)

# **Conclusion**

This lab demonstrates how easy it is to break RSA encryption when the key size is small. This is done by factoring the modulus and computing the modular inverse to find the decrypted exponent.

The decrypted message confirms that our attempts of cracking the cypher was successful and shows that for more security, longer keys should be used. This can be accomplished by simply picking much larger prime numbers to calculate the modulus.

# **References**

[1] “Public Key Encryption: What Is Public Cryptography? - Okta AU & NZ,” *www.okta.com*. <https://www.okta.com/au/identity-101/public-key-encryption/>.

[2] M. Cobb, “RSA algorithm (Rivest-Shamir-Adleman),” *TechTarget*, Nov. 2021. <https://www.techtarget.com/searchsecurity/definition/RSA>.