Cryptography: How Mathematics Keeps the World Safe

Jamie Cassar

April 18, 2018

Abstract

Cryptography is the practice and learning of techniques that allow secure communication in the presence of third parties. Modern cryptography is heavily based on mathematical theory, particularly the field of number theory. I will investigate the importance of mathematics behind the RSA algorithm and research how these methods are applied to keep us safe.

Lay Summary

Cryptography is the study of methods which enables us to protect sensitive information while it is in a position of vulnerability (i.e. an insecure channel). When any information is insecure it is under threat of an eavesdropper intercepting it, allowing them to find sensitive information.

With the advancement of technology there became a greater demand for more enhanced security, especially within the field of online computing. Society needed a cryptosystem that would work quickly and efficiently but also provide reliable results, meaning that the cryptosystem has a low chance of being deciphered by a third party.

To implement these cryptosystems which are required to keep people safe online, we require knowledge of fundamental number theory. This plays an underlying part in most modern types of cryptography. In this project the main types of cryptology will be explained, before focusing on the most used public cryptosystem to date, the RSA algorithm.

RSA is a cryptosystem that was created by three men Rivest, Shamir and Adleman which was based on a proposal from Diffie-Hellman. It relies on the notion that there is no quick way of factorising very large numbers, especially when the factorisation is a product of a pair of very large primes. The RSA therefore takes a remarkable amount of time and computer processing to finally decipher the system. By the time this can happen the secure connection has already taken place and the authentication of a message, signature or connection has been approved, dramatically reducing the chance of cybercrime.

The RSA algorithm is a 'trapdoor function'. This means that the function is easy to compute going in one direction, however it is near impossible to reverse the function. This project will show the mathematics behind the algorithm.

The RSA is designed to withstand different forms of attack due to its mathematical rigorousness. However there are still many attacks on the RSA, these usually spur from a lack of care by the users themselves rather than a direct attack on the algorithm (even though people do try to directly break RSA). Either the recipient or sender could lose a part of the important information (such as a key) to the third party eavesdropping, either by carelessness or by being hacked electronically, in which the hacker could take the private information. In the event of this occurring it would open possibilities to crack the cryptosystem in a short time, allowing the eavesdropper to withdraw the sensitive information that they are trying to steal. The reader will discover some of these methods and understand what information is needed to initiate an attack on the RSA algorithm.

Contents

1 Introduction **5**

2 Types of Cryptography **7**

2.1 Hash Functions 7

2.2 Private-Key Encryption 8

2.3 Public-Key Encryption 8

3 History of The RSA Algorithm **10**

3.1 Diffie, Hellman and Public-Key Encryption 10

3.2 Rivest, Shamir, Adleman and The RSA Algorithm 10

3.3 Applications of RSA 11

4 RSA Algorithm **13**

4.1 RSA Introduction 13

4.2 The RSA Algorithm 14

4.3 RSA Algorithm at Work - A Basic Example 15

4.4 RSA Algorithm at Work - A more Realistic Example 17

4.5 Why does the RSA Algorithm work? 19

5 Attacks on RSA **22**

5.1 Factoring the Modulus 22

5.2 Finding Decryption Key Without Factoring the Modulus or Computing $\phi(n)$ . . . 23

6 Conclusion **25**

7 Acknowledgments **26**

8 References **27**

9 Symbols Used **28**

10 Glossary **29**

11 Appendix **30**

11.1 Appendix A 30

11.2 Appendix B 30

1 Introduction

For thousands of years people have strived to send messages so that only the intended recipient can interpret them. Cryptography derived from the Greek word 'kryptos'[1]; is the way in which we achieve this secrecy amongst messaging. Early cryptography used some very basic methods such as denoting symbols for letters or shifting an entire alphabet by a certain number of spaces (i.e. a 3-shift would result in 'A' becoming 'D' and 'Z' becoming 'C'). As technology advanced the field of cryptography also had to advance. This resulted in cryptography relying more heavily on mathematical ideas, such as theorems and formulae, to improve the structure from which cryptic algorithms are built.

The idea of cryptography is that the sender will have a message which they wish to send via a path that may be insecure. The original message will be in plaintext which is readable by anyone who can read the language the message is written in, so in this form the message is by no means safe. The first step to securing the message is to encrypt it using some cryptic function. This enciphered message is now in a form of a string of numbers (Cipher text) which is not readable without knowing how to decipher it. The only way in which this message can be interpreted, is by inversing the function that was originally applied to the plaintext. The intended recipient should be the only person that has knowledge of secret information used in this inverse function, applied to the cipher text, which means that they are the only person that can access the text.

The exact date of the origins of cryptography are unknown, however there are many historic examples of it being applied. Tattersall (1999) acknowledges that early cryptography was present in Babylonian times 1500 years before Christ; he states 'a Babylonian cuneiform tablet, dating from about 1500 BC, contains an encrypted recipe for making pottery glaze.' Also throughout the Greek era much evidence has been discovered to show that Spartan military forces relied on encrypted conversations and orders to keep organisation amongst their regime. Cryptology helped the allied forces to triumph in the Second World War, it 'significantly shortened the war by more than two years, saving over 14 millions of lives.' this is an estimation present in The Imitation Game, Tyldum (2014). This was due to the persistence and determination of English mathematician Alan Turing who discovered that Enigma (a German encrypting device) could be decrypted by a mathematical algorithm; meaning that the allied forces were able to interpret German cipher text.

As technology advanced there became greater demands for enhanced security measures especially in the field of computing. Diffie et al (1976:644) state in their well established article, 'The

[1]Meaning secret or hidden

development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals.' The world needed an efficient way to remove these 'major barrier[s] to the transfer of business' ibid (1976:544) , a way in which processes such as providing online signatures, securing identity and communications as well as online certificates can be done via online connection, without the need of meeting before-hand.

The three main types of encryption that will be observed in this project are: hash functions, private-key and public-key. Both of the encryptions via key[2] are based on the same idea with some slight variations which will be explained in more detail later in the project. The methods have varying benefits and disadvantages so the method that should be employed depends solely on the individual task at hand. The cryptosystem that will be focused on throughout the project will be the RSA algorithm.

The RSA algorithm was released in 1978 it was one of the first public-key cryptosystems, which allows two users that have never previously met to produce an online authentication. RSA is the most widely used modern method of cryptography that is based upon the idea of public-key encryption and is one of the most important advances to date in the field of cryptography. This project will look into the origins of the RSA algorithm and also explain the algorithm itself, such as looking at the steps needed to create a successful encryption via RSA.

To date there is no way to prove that an encryption scheme is secure, the only scrutiny these schemes can come under is that of human application, trying to break them. The inventors of the RSA algorithm claim that the RSA algorithm is safe due to many great mathematicians over time trying to break the mathematical problem of factoring large numbers in a short time period, 'no one has yet found an algorithm which can factor a 200-digit number in a reasonable amount of time.' Rivest et al (1978:120). Even though it may appear that the RSA is completely secure there are ways in which human error can lead to faults in the secure scheme. This would normally consist of either one of the parties losing a sensitive piece of information or the sender or receiver being hacked which would allow a potential eavesdropper to steal this sensitive information and crack the system at hand. Some of these cases will be considered later in this project.

[2]An important piece of digital information that is required for both encryption and decryption.

2 Types of Cryptography

In this section we observe the most commonly used encryption methods by explaining the way in which they are used, the advantages and disadvantages of each method. Depending on the speed of encryption and the level of security the user requires, different methods of encryption should be employed.

2.1 Hash Functions

A hash function algorithm generates a hash value for a string of plaintext message. This value is comprised of bits, the possibility of receiving two identical hashes from varying messages is practically impossible, this will be shown shortly in the table below. It is also almost certain that a hash function cannot be inversed if only the hash value is known. Once the hash has been generated it is then encrypted using the sender's private-key, before being sent along with the ciphered message. The recipient of the message will decrypt the message which also contains the hash by using the sender's public-key. As the receiver knows the exact hashing function used, he can now reapply it to the message. If the hash received is equivalent to the original hash, then the identity of the sender is verified. The receiver also knows that no one has tampered with the original message sent to him.

Original Plaintext Message	**Hash Value**
The football team play in a red kit	EFD4B6DC3B62A184C352A77EDA6C169BF85DBC22
The football team play in a blue kit	D72C1163EBE872FD6CE13AE69946F28A8807AA48
The football team play in a yellow kit	98CE3D74A37D42DAC5ACC37573212F702958C586

Table 1: Original plaintext messages of similar content converted to hash values.

The table shows how a varying hash function can clearly highlight any tampering of an original message. The examples show a very basic change in the details within the plaintext however this results in very different hash values which can be observed in the right hand side of the table 1.

Hash functions are relatively quick and efficient; they also have the ability to reduce very large texts to hash values and this is why they are widely used. Computer users often opt for using a more secure encryption method such as public or private key encryption as these are known to be more rigorous than hash functions.

2.2 Private-Key Encryption

Private-Key encryption requires both parties to use the same key. The sender originally encrypts the plaintext using a private-key which both the sender and recipient know. The message is then sent via an insecure public channel to the recipient. To decipher the cipher text the inverse function must be applied using exactly the same private key as was used originally to encrypt the code. Figure 1 below shows a visualisation of how private key encryption works.

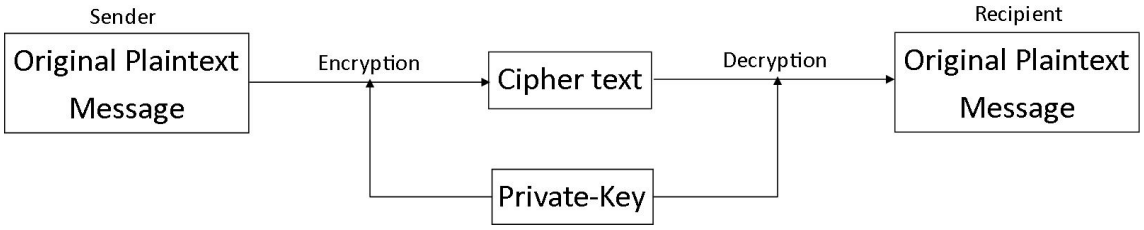

Figure 1: A Visualisation of Private-Key Encryption

Private key encryption is quicker than public key encryption and can be more efficient however there are some major drawbacks to private key encryption, the main reason being that both parties have to originally agree on a key via a secure method meaning that the parties have to meet in advance to achieve this. This is inefficient and in some cases not possible; let's say one party lives in Australia and the other in America; it would be very difficult for these parties to agree on a key. This could be the case for any connections where parties do not live relatively close. Moreover there is more chance in the key being lost or stolen if two parties have to keep the key. This is a contrast to public-key encryption as both parties use different keys.

2.3 Public-Key Encryption

Unlike private key encryption public-key does not require the users to rely on using identical keys; this means that the parties do not have to have met prior to creating a secure connection. Rather, the encryption algorithm relies on some fundamental mathematics to relate the keys, which will be used in the encryption. Many public key encryption systems are developed from prime numbers. To date there are no quick or efficient ways to factor large numbers and this is the underlying reason such cryptosystems are secure. Public key encryption is also known as asymmetric encryption due to varying keys being employed.

The process of public-key encryption is similar to private-key with the subtle difference in using different keys. Firstly the sender encrypts the plaintext message using an encryption function that

uses the public-key of the recipient. The message is then sent via an insecure public channel. When the message is received the recipient will be able to decrypt it using their own private-key and the inverse function. This is visualised below in figure 2. Between the original public-key and private-key that are used, there is a mathematical link, this link varies depending on the mathematical algorithm that is being used. We will look into a method of encryption that uses such a link later in the project.

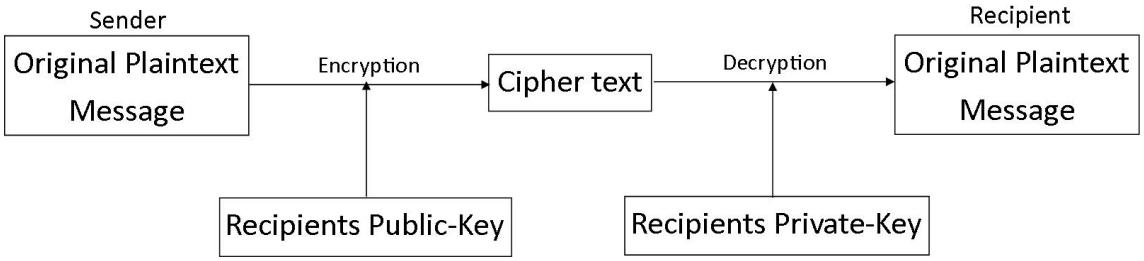

Figure 2: A Visualisation of Public key Encryption

Public key encryption has been very influential within modern day technology however in cases where large data needs to be encrypted the encryption process can take a long time. This may result in parties deciding to use other encryption methods, even if these methods do not provide as strong security.

3 History of The RSA Algorithm

3.1 Diffie, Hellman and Public-Key Encryption

Diffie-Hellman key exchange was first conceptualised by Ralph Merkle, who at the time was an undergraduate at the University of California, his paper which was received in August 1975 but not published until April 1978, states '[t]his is demonstrated by exhibiting a solution which allows two communicants to select a key publicly, but in such a fashion that no one else can easily determine it' Merkle (1978:295). This type of method allows a secure exchange of cryptographic key via an insecure channel.

Named after two electrical engineering students from Stanford University; Whitfield Diffie and Martin Hellman published their key exchange in their 1976 paper, 'New Directions in Cryptography'. Diffie et al (1976:644) states '[w]e propose some techniques for developing public key cryptosystems, but the problem is still largely open'.

The idea present in the Diffie-Hellman key exchange is that two users will no longer have to originally rely on a physically secure channel, such as secure post, to transfer a key amongst each other. This would erase what Diffie and Hellman claim to be 'a major barrier to the transfer of business' ibid (1976:44). The cryptosystem's enciphering stage uses arithmetic operations, which requires a straightforward process. Public key exchange is a massive revolution within cryptology due to the level of security the inverse process provides. In most cases the deciphering is computationally infeasible, due to the complexity of the algorithm and the time required to carry out the deciphering.

Diffie and Hellman are also credited for the invention of digital signatures. The idea of digital signatures was also released in the ground-breaking 'New Directions in Cryptography' paper, see section IV where Diffie et al (1976:649) discuss the problems faced at time of publication and why such a development needed to be worked on. Digital signatures allow the authentication of an electronically communicated document as well as verifying the senders' identity for the recipient.

3.2 Rivest, Shamir, Adleman and The RSA Algorithm

In 1978 Ron Rivest, Adi Shamir and Leonard Adleman from Massachusetts Institute of Technology released their ground-breaking article that would revolutionise electronic security. The cryptosys-

tem they developed, commonly known as RSA[3], was developed from Diffie and Hellman's principal set out in Diffie et al (1976) paper 'New Directions in Cryptography'. RSA was one of the first public-key cryptosystems and still to date, the most widely used cryptosystem.

The RSA cryptosystem revolves around the properties of prime numbers and the security relies on the difficulty of factoring a large composite integer which is a result of the product of two large primes. These primes are needed to decipher the cipher text. RSA 'is [widely used] today to ensure the security of every conceivable form of electronic communication.' Watkins (2014).

Kraft (2014) mentions the influence of RSA 'showed that number theory, long assumed to be purely theoretical branch of mathematics, had important practical applications.' This newly discovered importance of number theory attracted many researchers to this area of mathematics. Many mathematicians opted to research within number theory basing their research on topics such as factorisation algorithms.

3.3 Applications of RSA

RSA is widely used due to its ability to: protect confidentiality; secure data integrity; authenticate both source and recipient as well as preventing any possibility of repudiation. 'Computer, Network, Bluetooth technologies and other communication equipments such as mobile phone technologies' Caliskan (2011:1) all rely on strong and efficient cryptosystems. To date RSA encryption is one of the most efficient cryptosystems which explains why many technological companies apply RSA algorithm encryption to their software and devices.

Mail tracking is used to ensure that a message arrives at the intended recipient. It also helps to record the time and date that a message is received as well as the recipients IP address. Permadi (2018:1) states in his article, '[RSA is used] to provide security of a message or data in the process of delivery.' This use of RSA creates a much safer online environment for dispersing of electronic information.

E-commerce is built upon the security of RSA encryption. 'To ensure the realization of e-commerce information security, the international credit card organization, namely Visa International, Master-Card International cooperate with IBM, Microsoft, Netscape, GTE VeriSign and develop Secure Electronic Transaction ... to be used in all kinds of information security in e-commerce.' Liu (2010:88). This agreement amongst the companies mentioned previously uses application of the

[3]Derived from surnames of the three inventors. Rivest, Shamir and Adleman.

RSA encryption algorithm.

4 RSA Algorithm

4.1 RSA Introduction

Before jumping into the algorithm that makes the RSA so secure, the reader must understand some mathematical propositions that appear in the RSA algorithm. The theorems required are 'Euler's Totient Theorem for Primes' as well as an extension of this and 'Euler's Theorem'.

Euler's Totient Theorem for Primes

$$n \text{ is prime} \iff \phi(n) = n - 1 \text{ where } n \text{ is a natural number.}$$

Extension of Euler's Totient Theorem for Primes

$$If \ p \text{ and } q \text{ are two different primes and } N = pq \text{ we have :}$$

$$\phi(N) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$$

Euler's Theorem

$$If \ a \text{ and } n \text{ are relatively prime that is } gcd(a, n) = 1 \text{ then } a^{\phi(n)} \equiv 1 (mod\ n) \text{ where } a \text{ and } n \text{ are natural numbers.}$$

Understanding of the variables that will be used in the algorithm is also important. These are listed below:

- p, q - randomly selected distinct large primes
- n - the modulus value apart from when finding decryption key where $\phi(n)$ is used
- e - encryption key
- d - decryption key
- ρ - public key of the form (n,e)
- D - private key of the form (n,d)
- m - numeric form of plaintext message
- m' - split numeric form of plaintext message
- C - numeric encrypted message (Ciphertext)

4.2 The RSA Algorithm

The first stage of the RSA algorithm is to generate both the public and private keys. This is carried out by the intended recipient, before any messages can be encrypted. The following points are the steps carried out to setup both public and private keys, only the recipient is involved throughout this process.

1. Recipient selects two large distinct primes (p and q).
2. Recipient calculates n = pq and $\phi(n)$ = (p-1)(q-1) using the two primes selected and the 'Extension of Euler's Totient Theorem for Primes'.
3. Recipient chooses an e so that it satisfies both: gcd(e,$\phi(n)$)=1 and $1 < e < \phi(n)$.
4. The recipient calculates d using the property that ed $\equiv$1(mod $\phi(n)$). Hence d$\equiv e^{-1}(mod\phi(n))$
5. The recipient now has access to both ρ and D they are (n,e) and (n,d) respectively, where both ρ and D consist of two pieces of information each. ρ is made public and d, p and q are kept secret as the private key.

After the initial setup stage is complete, public information ρ and private information p, q and d is known to whom it concerns. The encryption process, which only involves the sender of the message can now begin. In this stage a plaintext message is converted to numerical form. To do this a letter coding conversion has to be introduced. The conversion shown in figure 3 below and also displayed in Appendix A is applied to the plaintext message.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Figure 3: Letter Coding Conversion Table

1. The encryption begins by the sender finding the recipients public key (ρ) which was made public in the initial setup.
2. The sender converts his message (m) to numerical form using the conversion and then splits the numerical string into blocks smaller than the size of the modulus by using m(mod(n)).
3. Sender encrypts the numerical message into numerical ciphertext partially by C$\equiv (m')^e$mod(n) using the components of the recipients public key (ρ = (n,e)). Where (m') is the split numeric form of the plaintext message.

4. The ciphertext is sent to the recipient this is where the encryption stage ends.

Now that the intended recipient has received the ciphered message, it can now be deciphered. Note that only the intended recipient can decipher the message as only they acquire the secret private key from the initial setup.

1. Using C the ciphertext that the recipient received; d the decryption key and the modulus n, the recipient computes $m \equiv C^d (\text{mod}(n))$ which results in the receiver being delivered the numerical plaintext message.

2. Now the message has been decrypted however it is still in numerical form. The reverse conversion has to be applied by using figure 3. A similar process to what took place in encryption but reversing the process.

4.3 RSA Algorithm at Work - A Basic Example

Now the steps of the RSA algorithm have been outlined, a working example will clarify how this method of encryption can help to enable secure transmission of a message. The following application would not be used in real world situations as the components are relatively small; it is applied here for illustration purposes. Later in the investigation a more complicated example will be carried out via the aid of a computer program, which has the capability to compute operations involving large primes. Let's say Jack, a secret agent, is being held captive. Sue, Jack's sidekick, is not aware of the capture of Jack. Jack has access to an Internet connection but cannot risk his message ('sos') being intercepted.

Sue has to first set up a RSA algorithm which will allow her to safely receive a message from Jack.

1. Sue selects primes to be p = 3 and q = 11.

2. Sue calculates n and $\phi(n)$.

$$n = p.q = 3.11 = 33$$

$$\phi(n) = (p-1)(q-1) = (3-1)(11-1) = 20$$

3. Sue must now select an e that satisfies: gcd(e , $\phi(n)$)=1 and $1 < e < \phi(n)$. We can select e=3 as gcd(3 , 20)=1 and it also satisfies $1 < 3 < 20$.

4. Now that Sue has selected her e to be 3 she can calculate d using ed≡1(mod$\phi(n)$) or similarly d≡ $e^{-1}(mod\phi(n))$.

$$3d \equiv 1(mod 20) \text{ or } d \equiv 3^{-1}(mod\phi(20)$$

If d = 7 the congruence above is satisfied, hence d = 7. As,

$$21 \equiv 1(mod 20)$$

Sue now has both public and private keys they are (33 , 3) and (33 , 7) respectively. That is ρ = (33 , 3) and D = (33 , 7). Note that ρ is the public key and D is the private key.

5. Sue now sends her public key (ρ) to Jack and keeps her private key (D) secret.

It is now Jack's job to encrypt his message using the public key information that Sue has sent through to him.

1. Jack begins by converting his plaintext message into numerical form using figure 3, which has also been included below. The message Jack wishes to send is 'SOS', by converting this using the table below the numerical result is m='191519'.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

2. Jack has found ρ sent by Sue, so knows that the value of the modulus he will work with is n=33. Jack splits the numerical value into blocks smaller than 33 and results at the following split m='19-15-19'.
3. Jack applies the encryption function C=m^e(mod(n)) to turn his numerical message (m='19-15-19') into ciphertext. To carry out this process, the split version of the numerical message will be considered.

$$\mathbf{19}^3(mod(33)) \equiv 6859(mod(33)) \equiv \mathbf{28}(mod(33))$$

and

$$\mathbf{15}^3(mod(33)) \equiv 3375(mod(33)) \equiv \mathbf{09}(mod(33))$$

Hence the encrypted split value of message m will be C=28-09-28.

4. Jack now sends the ciphertext to Sue. Sue will decrypt the ciphertext using her private key.

The decryption process can now begin, Sue uses the ciphertext that Jack has just sent and uses a decryption process to result at the original plaintext message.

1. Using the ciphertext (C=28-09-28) in parts as it is currently, and her own private key, Sue calculates $C^d(mod(n))$ that is:

$$28^7(mod(33)) \equiv 893871739(mod(33)) \equiv 19(mod(33))$$

and

$$09^7(mod(33)) \equiv 4782969(mod(33)) \equiv 15(mod(33))$$

This results in receiving the following numerical output: '19-15-19'

Now the ciphertext has been decrypted, Sue has her message but it is still in numerical form (19-05-19), using the standard letting coding displayed in 'Figure 3' Sue transfers her numerical message into text and receives 'SOS'. The exact message that Jack originally sent.

4.4 RSA Algorithm at Work - A more Realistic Example

To create a quick initial secure connection smaller primes, encryption key and decryption key are used. This would not be the case with connections that involve transactions such as banking as these use a very large modulus, which even super computers would not be able to factorise in relatively short time. The RSA Algorithm example below could be a possible real life application of the RSA, which could be used when creating a quick secure connection. This could be present when connecting to a website securely or even to secure messages within modern day message applications such as Messenger or WhatsApp depending on the encryption scheme the individual application uses.

1) *Firstly select primes* $p = 1481$, $q = 7487$ *and calculate the modulus.* $n = p.q = 1481.7487 = 11088247$

The integers selected have to be prime and so we use python program 'B.1 Prime Checker' which is displayed in Appendix B. This allows us to check that both p and q satisfy this condition, see the python output below.

```
Enter a prime number (p): 1481
(1481, 'is a Prime number!')

Enter a different prime number (q): 7487
(7487, 'is a Prime number!')
```

2) $\phi(n) is\ also\ required : \phi(n) = (p-1).(q-1) = 1480.7486 = 11079280$

3) *Now an encryption key selected must be relatively prime to the value of* $\phi(n)$. *I.e.* $gcd(e, 11079280) = 1$.

As the integer 11079280 has a total of 4064256 relatively prime numbers, that is $\phi(n)$=4064256, calculated using 'B.2 Relatively Prime Total Program' in Appendix B, a large encryption key will definitely be available for selection.

$$e\ is\ selected\ too :\ 55559$$

The relatively primeness is checked using python program 'B.3 Relatively Prime Check' also located in Appendix B, please see the python output below confirming e and $\phi(n)$ are indeed relatively prime.

```
e=55559

phi(n)=11079280
Relatively prime
```

The decryption key is now calculated by applying the property: ed≡1mod$\phi(n)$ to calculate this we use the 'B.4 Inverse Mod Program' located in Appendix B.

4) *From the python output below we can observe that the decryption key* (d) *is* 9117239

```
Input the encryption key "e": 55559

Input the value of phi "phi": 11079280
The Greatest Common Divisor of 55559 and 11079280 is 1
The Modular inverses of 55559 modulo 11079280 is: 9117239
Therefore the decryption key is: 9117239
```

Now that both encryption and decryption keys are known ρ=(11088247,55559) and D=(11088247,9117239) respectively, the encryption process can now begin. As with the previous steps a python program was created to aid with the huge calculations that are required in both the encryption and decryption processes.

The message that needs to be sent is ('RSA'), which needs to be numerically coded using the figure first introduced on page 14 that can also be found in Appendix A. Using the numerical coding we receive '181901' as R-18, S-19 and A-01. We know m which in this case is also m' as 181901 is less than the modulus, hence we can not split this numeric form into smaller chunks. So now the

encryption can take place. The 'B.5 Encryption Process Program' displayed in Appendix B created via python will carry out the tedious mathematics required due to the size of the calculations taking place. The output is displayed below:

```
Input the value of the modulus:11088247

Input numerical form of message:181901

Input the encryption key e:55559
The ciphertext is: 3857292
```

As the output above shows, the value of the ciphertext is: 3857292. As the message is now in ciphertext form, a eavesdropper who intercepts it will not be able to read the message.

5) *The numeric encrypted ciphertext,* $C = 3857292$.

As the encryption stage is fully complete, the decryption process can now begin. As with the encryption process the python program is very similar for decryption, with some slight differences as the computation is the reverse. Please see 'B.6 Decryption Process Program' code located in Appendix B . The output of the program is displayed below:

```
Input the value of the modulus:11088247

Input ciphertext form of message:3857292

Input the decryption key d:9117239
The numerical form of the plaintext message is: 181901
```

As can be observed in the output of the decryption python program the numerical form of the plaintext message is '181901', to receive the worded message, the conversion table must be used to convert 18-R, 19-S and 01-A and this delivers the initial message RSA, without any hacker having the chance to read, tamper or destroy the message.

4.5 Why does the RSA Algorithm work?

To understand the underlying mathematics behind the RSA algorithm we require Fermat's Little Theorem as well as Euler's Totient Theorem which was stated on page 13. Fermat's Little Theorem is stated below:

Fermat's Little Theorem

If p is prime and a is an integer then $a^{p-1} \equiv 1(mod\ p)$ *and* $p \nmid a$.

For the integer message M that is relatively prime,

$$M^{\phi(n)} \equiv 1(mod\ n) \tag{1}$$

Using Euler's Totient Theorem:

$$\begin{aligned} \phi(n) &= \phi(p)\phi(q) \\ &= (p-1)(q-1) \\ &= n - (p+q) + 1 \end{aligned} \tag{2}$$

As d is relatively prime to $\phi(n)$, it has multiplicative inverse e^{-1} modulo($\phi(n)$) that is:

$$ed \equiv 1(mod\phi(n)) \tag{3}$$

If e and d are chosen to satisfy the condition above we can now prove the following equations ((4)(5)) hold they are:

$$D(E(M)) = M \tag{4}$$

and

$$E(D(M)) = M \tag{5}$$

Equation (4) is the deciphering of the enciphered form of a message M. Equation (5) is when firstly the deciphering takes place before the enciphering. Both equations result in M.

We have:

$$D(E(M)) \equiv (E(M))^d \equiv (M^e)^d \equiv M^{ed}(mod\ n)$$

$$E(D(M)) \equiv (D(M))^e \equiv (M^d)^e \equiv M^{ed}(mod\ n)$$

and

$$M^{ed} \equiv M^{k.\phi(n)+1}(mod\ n)$$

This application can be made as from (3): ed≡ k$\phi(n)$+1

From (1) we derive that for all M such that p ∤ M combining this equation with Euler's Totient Theorem for Primes.

$$M^{p-1} \equiv 1(mod\ p)$$

$$M^{k.\phi(n)+1} \equiv 1xM \equiv M(mod\ p)$$

so since p-1 divides $\phi(n)$:

$$M^{k.\phi(n)+1} \equiv M(mod\ p) \tag{6}$$

Similarly for for (mod q) we get:

$$M^{k.\phi(n)+1} \equiv M(mod\ q) \tag{7}$$

Combining the two previous equations (6),(7) implies for all M:

$$M^{ed} \equiv M^{k.\phi(n)+1} \equiv M(mod\ n)$$

By applying the following rule: a ≡b(mod n) & a≡b(mod m) $\implies$ a≡ (mod mn)

Hence (4) and (5) are satisfied for all 0 < M < n. Therefore E and D are inverse operations and hence the reason the RSA Algorithm functions so well.

5 Attacks on RSA

The RSA algorithm is used to create secure connections and certificates amongst two online users, however it does not protect the user from the potential liability of a third party stealing a decryption key. Resulting that users must be careful in the way which they store important information; just as people would be with a pin number. In this section, obvious attacks that could be carried out on the RSA algorithm will be investigated. In Rivest et al (1978:125) it states 'approaches for breaking our system are at least as difficult as factoring n'. A selection of these methods will be researched. Looking into the reason why these methods are not successful enough to complete a full crack of the RSA Algorithm.

5.1 Factoring the Modulus

Knowing both primes p and q would allow an eavesdropper to crack the RSA. Knowledge of these primes would allow the eavesdropper to compute $\phi(n)$, from this they could root out d the decryption key allowing them to easily decode any ciphered text. This is why the complexity of factoring a large composite number created by the multiplication of two large primes is such a fundamental part of the algorithm. At the time the RSA Algorithm was developed and released the fastest method for factorising numbers was created by Richard Schroeppel an American mathematician from Illinois. The figure below a table published in 'A Method for Obtaining Digital Signatures and Public-Key Cryptosystems' shows the number of iterations and time taken to factorise a number with Schroeppel's method.

Digits	*Number of operations*	*Time*
50	1.4×10^{10}	3.9 hours
75	9.0×10^{12}	104 days
100	2.3×10^{15}	74 years
200	1.2×10^{23}	3.8×10^{9} years
300	1.5×10^{29}	4.9×10^{15} years
500	1.3×10^{39}	4.2×10^{25} years

Figure 4: A table showing number of iterations and time to factor a number with a certain number of digits via Schroeppel's method.

New methods for factorising large numbers such as general number field sieve and the Pollard-Strassen method have enhanced the ability of factoring numbers since the release of the RSA Algorithm in 1978, but to date there is still not a single factorising algorithm that provides accurate results within a short time period.

5.2 Finding Decryption Key Without Factoring the Modulus or Computing $\phi(n)$

Carrying out an attack on RSA Algorithm via the decryption key is also viable, however the set of possible numbers that d could be is very large and so it is also impossible to directly compute in respectively short time as is with the modulus. The table below shows output values a python program has produced (Python program B.2 is located in Appendix B). It shows how many relatively prime integers a certain n has. The value for $\phi(n)$ has been selected to be 97 rather than 99 or 100, as 97 does not have a clear small integer divisor such as 2, (2|100) or 3, (3|99).

n	97	997	9997	99997	999997	9999997
$\varphi(n)$	96	996	9216	94392	997920	8571420

Figure 5: Python $\phi(n)$ totals output.

Figure 5 shows a general pattern that as the value of $\phi(n)$ increases, the number of relatively prime numbers generally increases.

The larger the value of $\phi(n)$ the larger the encryption key can be and this results in a greater possibility of having a larger decryption key (d) which we are looking for.

If a eavesdropper manages to get hold of the decryption key he will be able to compute (e.d)-1 which will provide a multiple of $\phi(n)$. Miller (1975:234-239) showed that n can be factored via the application of a multiple of $\phi(n)$. This attack once again highlights the importance of n being a large composite number composed from two primes, that preserves the functioning capability of the RSA Algorithm.

This will now be demonstrated using the basic example from earlier in the investigation located in section 4.3.

The following information was used $p = 3,\ q = 11,\ n = 33\ \phi(n) = 20\ e = 3$ *and the decryption key* $d = 7$.

By calculating e.d-1 for this example we get e.d-1=20. This will be a a multiple of $\phi(n)$. We can now use the following to find the factors of n:

$$x^2 - (n + 1 - \phi(n))x + n$$

We substitute the value of $\phi(n)$ and n into the above and then factorise to find p and q as required:

$$x^2 - (33 + 1 - 20)x + 33 = 0$$

By solving this equation we result at (x-3)(x-11)=0 $\implies$ Values of p and q are 3 and 11 respectively. Hence the modulus has been factorised.

In some cases a user will purposely choose a small decryption key to make the RSA Algorithm work quickly. This makes the particular encryption vulnerable to an attack, if a hacker was attempting an attack via trialling low decryption keys he may find the key relatively quickly and this will enable the hacker to decrypt the enciphered message.

6 Conclusion

The RSA Algorithm is still one of the most rigorous methods of encryption available to companies as well as anyone who wishes to make a secure connection via online communications. If the methodology of the user implementing the algorithm is of a good standard meaning they pick their primes (p,q) to be large; have a large enough public and private key (ρ,D), and most importantly keep vital information safe then the RSA will be both convenient and suitable for users to set up an initial secure communication.

Gardener (1977:123) states 'Rivest and his associates have no proof that at some future time no one will discover a fast algorithm for factoring composites'. Many great mathematicians have worked within the topic of factorisation and to date there is still no system that allows for quick factorisation of very large composite numbers, created by multiplying to large primes. Continuous development within the field of quantum computing may give some, belief that the search for new large primes and the ability to factorise large numbers with ease is not far away. But for now the RSA algorithm provides users with a security level required to achieve a secure online connection and only an astonishing breakthrough within the factorisation of numbers will change this.

7 Acknowledgments

I would like to express my appreciation toward my investigation supervisor, Kuldeep Singh, for the guidance provided when writing this report as well as the continued support and commitment that he has given throughout my studies at university. His passion for mathematics and lecturing of 'Number Theory' fuelled my interest of the ways in which mathematics keeps the world safe and this is what encouraged me to research within the field of cryptography. During this period I learnt the necessary foundations that allowed me to base my investigation on this topic.

I would also like to thank all of the PAM lecturers that have dedicated their time over the past three years, for helping me gain knowledge and succeed throughout my time at university.

8 References

Caliskan, D. (2011). 'An Application of RSA in data transfer', pp. 1.

Diffie, W and Hellman, M. (1976) 'New Directions in Cryptography', IEEE Transactions on Information Theory, vol. 22, no. 2.

Gardner, M. (1977). 'A new kind of cipher that would take millions of years to break', MATHEMATICAL GAMES.

Kraft, J.S and Washington, L.C. (2014). 'An Introduction to Number Theory with Cryptography', CRC Press, Taylor & Francis Group, pp. 169.

Liu,D, Chen, Y and Huai-ping, Z. (2010). 'Secure Applications of RSA System in the Electronic Commerce', 2010 International Conference on Future Information Technology and Management Engineering, pp. 88.

Merkle, R. (1978). 'Secure communications over insecure channels', ACM, vol. 21, no. 2, pp. 295.

Miller, G,L. (1976). 'Riemann's hypothesis and tests for primality', Journal of Computer and System Sciences, vol. 13, Issue. 3, pp. 234-239.

Permadi, S,P, Kusworo, A and Gernowo, R. (2018). 'Application Mail Tracking Using RSA Algorithm As Security Data and HOT-Fit a Model for Evaluation System', E3S Web of Conferences 31, pp. 1.

Rivest, R , Shamir, A and Adleman, L. (1978). 'A method for obtaining digital signatures and public-key cryptosystems', ACM, vol. 21, no. 2.

Tattersall, J.J. (1999) Elementary Number Theory in Nine Chapters, Cambridge University Press, Cambridge.

Tyldum. (2014). [Film]. USA.

Watkins, J.J. (2014). 'Number Theory A Historical Approach', Princeton University Press, Princeton and Oxford, pp. 369.

9 Symbols Used

Symbol	Definition
gcd(a,b)	Greatest common divisor of a and b
$\iff$	If and only if
$\phi(n)$	Euler's Totient function - Total number of relatively prime numbers to n
$\equiv$	Equivalent too
p	First prime in RSA Algorithm
q	Second prime in RSA Algorithm
n	Value of the modulus p.q
e	Encryption key
d	Decryption key
ρ	Public key of the form (n,e)
D	Private key of the form (n,d)
m	Numeric form of plaintext message
m'	Split numeric form of plaintext message
C	Numeric encrypted message (Ciphertext)

10 Glossary

Algorithm	A set of calculations following steps to obtain a final result.
Asymmetric encryption	Encryption that does not rely on the same key, i.e. public and private key differ.
Ciphertext	The scrambled version of the plaintext, which cannot be understood.
Coprime	Two numbers are coprime if they have no other factors in common other than 1.
Cryptography	The skill of sending a disguised message to an intended person.
Cryptosystem	A system which uses an algorithm to encrypt data.
Decryption	A process to undo the encryption, that results at the plaintext message.
Eavesdropper	A person or organisation who has is trying to intercept important information.
Encryption	The process of turning plaintext into scrambled ciphertext.
Key	An piece of digital data information that is required for both encryption and decryption.
Plaintext	The original form of a message in letter form.
Relatively prime	The same as coprime, relatively prime numbers have no common factors other than 1.
Symmetric encryption	An encryption in which the same keys are used hence the name symmetric.
Trapdoor function	A function easy to compute one way, however nearly impossible to calculate the reverse.

11 Appendix

11.1 Appendix A

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

11.2 Appendix B

```
A program to check that p and q selected are prime.
"""
def checkForPrime(n):

    result = False

    if n > 1:
        for i in range(2, n):
            if (n % i) == 0:
                print("i is ", i)
                print(n, "is not prime!")
                print(i, "x", n//i, "is", n)
                result = False

            else:
                print(n, "is a Prime number!")
                result = True

            return result

p = int(input("Enter a prime number (p): "))
while checkForPrime(p) == False:
        p = int(input("Enter a prime number (p): "))


q = int(input("Enter a different prime number (q): "))
while checkForPrime(q) == False:
```

Figure 6: B.1 A Prime Checker Python Program

```
"""
A program that counts the number of relatively prime integers a number n has.
"""
from fractions import gcd


n= int(input("n="))
list=[]
count = 0
for i in range (1,n):
    if gcd(i,n)==1:
        count+=1
        list.append(i)
        #print("Relatively prime to",i)

#print('The following are relatively prime to,')
#print b
#print list

print ('The number of relatively prime numbers to,')
print n
print 'Is:'
print len(list)
```

Figure 7: B.2 Relatively Prime Total Program

```
"""
A program to check that the encryption key (e),
is relatively prime to the value of phi(n)
"""


from fractions import gcd

e=int(input("e="))
phi= int(input("phi(n)="))

if gcd(e,phi)==1:
     print("Relatively prime")
else:
     print("Not relatively prime")
```

Figure 8: B.3 Relatively Prime Check Program

```
"""
A python program to work out both the gcd and inverse modulo of a
congruency.
"""

def inverseMod(e, m):
    for i in range(1,m):
        if ( m*i + 1) % e == 0:
            return ( m*i + 1) // e
    return None

def gcd(e, phi):
    """Calculate the Greatest Common Divisor of e and phi.
    Unless phi==0, the result will have the same sign as phi (so that when
    phi is divided by it, the result comes out positive).
    """
    while phi:
        e, phi = phi, e%phi
    return e


e = input("Input the encryption key \"e\": ")
phi = input("Input the value of phi \"phi\": ")


print("The Greatest Common Divisor of %s and %s is %s" % (e, phi, gcd(e,phi)))
print("The Modular inverses of %s modulo %s is: %s" % (e, phi, inverseMod(e,phi)))
print("Therefore the decryption key is: %s") %(inverseMod(e,phi))
```

Figure 9: B.4 A program to find the inverse mod

```
"""
A program to carry out the encryption process of the RSA Algorithm.
Where n is the modulus,  m is the message in numerical form, e is the
encryption key and C is the resulting ciphertext.
"""

# Request information needed to carry out the encryption process
n = input('Input the value of the modulus:')
m = input('Input numerical form of message:')
e = input('Input the encryption key e:')

# Calculate the ciphertext (C) using the encryption formula
C = (m**e)%n

# Command to  print the resulting ciphertext
print 'The ciphertext is:', C
```

Figure 10: B.5 Encryption Process Program

```
"""
A program to carry out the decryption process of the RSA Algorithm.
Where n is the modulus, C is the ciphertext, d the decryption key and pm
the plaintext numerical form of the message
"""

# Request information needed to carry out the decryption process
n = input('Input the value of the modulus:')
C = input('Input ciphertext form of message:')
d = input('Input the decryption key d:')

# Formula to carry out the decryption
pm = (C**d)%n

# Command to print the plaintext message result
print 'The numerical form of the plaintext message is: ',pm
```

Figure 11: B.6 Decryption Process Program