Understanding the Mathematics Behind Cryptography

Nicolas Kyriacos

May 15, 2021

Abstract

Today the internet is being used more than ever in order to store, transfer and access information. As a result, our private information is vulnerable to malicious parties who wish to use it for their own personal benefit. Cryptography uses mathematics in order to hide our private data so that it cannot be accessed by anybody other than ourselves. We begin this paper by exploring the most basic kinds of cryptosystems such as shift transformations, linear transformations and affine transformations as well as the how these cryptosystems are cracked using frequency analysis. We then take a look at how these concepts translate to enciphering matrices. Furthermore, we explain the differences between private and public key cryptography and investigate the RSA cryptosystem which is one of the most common types of public key cryptosystems in use today.

Contents

1 Background

In today's age, we are more dependent on the Internet than ever before. From education to business to social media, much of our daily lives take place over the realm of the Internet and as a natural consequence we use cryptography everyday without even realising it. Consider daily activities such as logging into a bank account, changing a password online, sending an email or even searching 'funny cat pictures' on Google. In each of these examples we are sending our private data (bank account passwords, credit card pins etc.) onto public networks and each time we do so there is the possibility that an adversary, such as a hacker, can intercept our private information an use it for their own benefit. This is where cryptography comes to the rescue.

Cryptography, which derives from the Greek word 'krypto' meaning hidden, is study of mathematical functions which are used to scramble information so that it can only be interpreted by the intended recipient. In this way, our private data will appear meaningless to anybody that wasn't meant to get their hands on it. In other words, cryptography can be appropriately thought of as the 'science of keeping secrets secret' [3] In order to do so, cryptography makes use of encryption and decryption algorithms which respectively scramble our private data into an unintelligible form, and unscramble the scrambled data back into its original form. These algorithms make use of specific mathematical functions as well as parametric keys which determine how the message is scrambled and unscrambled. Encryption keys "lock" our private information so that it is not accessible by others, and decryption keys "unlock" this private information, so that we can access it.

2 Cryptography and Basic Cryptosystems

Before dissecting some basic cryptosystems, it is a good idea to develop our understanding of what cryptography is all about. Cryptography, as we had mentioned in

the introduction, is the study of mathematical functions which are used to conceal the content of a message by scrambling its information. In this way, a message which is intercepted by an adversary cannot be understood thereby keeping its information secure all the way until it reaches its intended recipient. The original message, that is the message prior to encryption, is referred to as the *plaintext* while the scrambled message is referred to as the *ciphertext*. Both the plaintext and ciphertext are written in an alphabet consisting of all the possible characters, letters, punctuation marks, and symbols found in the text. In this paper, we will assume that N represents the number of characters in an alphabet. *Encryption* refers to the action of converting plaintext into ciphertext which can also be though of as the "scrambling" of the message. Decryption, on the other hand, refers to the action of converting ciphertext into plaintext, thereby retrieving the message in its original form.

The conversion of plaintext into ciphertext takes place by means of an *enciphering transformation*. In mathematical terms, an enciphering transformation is a function which maps the elements of our alphabet to other elements in our alphabet, thereby scrambling the alphabet. In view of the fact that we wish to decrypt messages, enciphering transformations must be invertible. Furthermore, we know that a function is invertible only if it is 1-1. This tells us that enciphering transformations must be 1-1 in order to produce both enciphering and deciphering transformations, where a *deciphering transformation* refers to the inverse function used to decrypt the message. The enciphering transformation, deciphering transformation and the elements of our alphabet altogether form a *cryptosystem*.

For cryptographic purposes, it is convenient for us to use functions whose domains and ranges are integers. For this reason cryptography uses a process called *labeling* to assign numeric values to the elements of our alphabet. The most basic example of this would by labeling the letters A-Z by the numbers 0-25. For example, the message 'fire' could be represented numerically as the numbers 5 8 17 4. This is not

the only way in which we can represent the elements of our alphabet as numbers. For example, we can pick our alphabet to be the set of all possible digraphs which are individual message units consisting of 2 characters. In this way our alphabet will have N^2 characters and each digraph will be an element of the set $\mathbb{Z}/N\mathbb{Z}$. We can they say that the numerical equivalent of each digraph is determined by the convention $xN+y$ where x, y are the indices of the individual characters x and y in our alphabet and N is the numbers of letters in the alphabet [4], page 59. As example, if we were working in the standard alphabet, then the digraph 'MN' would take on the numerical equivalent $12 \cdot 26+13 = 325$. We can use similar methods to label n-graphs consisting of n letters. Take note that this is just one way of labeling message units, while other cryptosystems can label message units as vectors (Enciphering Matrices)as well as positions on an elliptic curve.

Now that we know how plaintext message units are broken up and assigned numeric equivalents through the process of labeling, let's consider the Caesar's Cipher as an introduction into basic cryptosystems. This cryptosystem was named after its creator, Julius Caesar and was used by him order to send encrypted messages to his army. This cryptosystem makes use of a very simple substitution cipher over the standard 26 letter alphabet [1].

Example 1. Suppose we are working in the 26-letter alphabet where a message unit is a single letter. Let $P \in \{0, 1, 2, 3, \ldots, 25\}$ represent the numeric equivalents of all plaintext message units. Then we define the function f as follows:

$$f(P) = \begin{cases} \text{P+2}, & \text{if } x < 24 \\ \text{P-24}, & \text{if } x \geq 24 \end{cases}$$

In short, this function shifts each letter 2 letter down the alphabet. We can therefore express the function as :

$$f(P) = P + b \ mod \ 26.$$

Take note that throughout the rest of the paper we define $a \equiv b \ mod \ N$ to be equivalent to $a \equiv_N b$. Therefore we will represent the shift transformation above as

$$f(P) \equiv_{26} P + b$$

Let's consider the word "CAR" in an attempt to better understand this shift transformation. Since each letter is its own message unit, the word can be numerically expressed as the numbers 2 0 17 . By applying the enciphering transformation we see that

$$f(2) \equiv_{26} 2 + 2 \equiv 4,$$

thus "C" maps to "E", while

$$f(0) \equiv_{26} 0 + 2,$$

so "A" gets shifted down two letters "C" and

$$f(17) \equiv_{26} 17 + 2$$

which tells us that "R" becomes "T". Therefore the word "CAR" is scrambled into the word "ECT".

While this provides a specific example of a shift transformation, a general shift transformation is defined as follows:

$$f(P) \equiv_N P + b \tag{2.1}$$

such that $b \in \mathbb{Z}/N\mathbb{Z}$. As we can see from Example 1, without knowledge of the specific shift parameter b as well as the number of letters is in the alphabet given by N, we would not be able to decrypt this scrambled message. In other words, *code breaking* requires knowledge of the structure of the cryptosystem (i.e. that is the alphabet that

is used and the type of enciphering transformation) as well as the specific parameters which act as the enciphering/deciphering key. The science of breaking code without knowledge of both of the structure and parametric keys is known as *cryptanalysis* [4], page 56.

2.1 Frequency Analysis

Note that while Example 1 is interesting and provides a basic notion of what cryptography is about, it makes for a very poor cryptosystem. Such a cryptosystem could be easily broken by a cryptanalytic technique known as *frequency analysis*. This method takes advantage of the fact that certain letters are repeated more often than others. Knowing this, a person could look out for the repeated usage of a particular letter in a message. They could then attempt to substitute that letter by the letter which appears more or less as frequently in that particular language in an attempt to deduce the shift parameter [2]. In other words, we count the number of times that each ciphertext letter occurs in a message. We then take the most frequently occurring letter and deduce that it most likely belong to the most frequently occurring letter of the English alphabet which is "E". Similarly, we take the second most frequently occurring letter and deduce that it most likely corresponds to the second most frequently occurring letter in the English alphabet"" and so on. In this way a shift transformation becomes very easy to break because knowing the shift of a single letter allows us to decrypt the entire message. Note that while a frequency analysis can be used to break code, a computer doesn't even need to perform a frequency analysis for a cryptosystem that uses a shift transformation. It can simply iterate through every possible shift and find the parameter for which the message makes sense.

2.2 Affine Transformations

An *affine transformation* makes for a slightly better cryptosystem than a shift transformation. Specifically, an affine transformation is defined as:

$$C \equiv_N aP + b \tag{2.2}$$

where a and $b \in \mathbb{Z}/N\mathbb{Z}$, which together give us the enciphering key (a, b). Suppose we were working in the 26 letter alphabet and wanted to encipher the message "STRIKEFIRST" using parameters $a = 6$ and $b = 15$. We can perform the enciphering transformation on the letter "S" which has the numerical equivalent 18. Substituting this value into the enciphering transformation gives us that

$$6 \cdot 18 + 15 \equiv_{26} 19$$

and so 'S' maps to 'T'. By applying the mapping to the entire plaintext we get that 18 19 17 8 10 4 5 8 17 18 19 maps to 19 25 13 11 23 13 19 11 13 19 25 which is "TZNLXNTLNTZ".

Now let's explore how deciphering works with an affine transformation. By rearranging equation (2) we get that

$$C - b \equiv_N aP$$

Multiplying by a^{-1}(the inverse of a modulo N) on both sides gives us that

$$a^{-1}(C - b) \equiv_N a^{-1}aP$$

which implies that

$$P \equiv_N a^{-1} - a^{-1}b$$

We will represent a^{-1} as $a^{'}$ and $-a^{-1}b$ as $b^{'}$, thus producing the deciphering transformation

$$P \equiv_N a^{'}C + b^{'}.$$

We choose a such that $\gcd(a, N) = 1$ in order to ensure that a is invertible, which in turn ensures that the enciphering is invertible.

Let's take a look at what would happen if a were not chosen to be an invertible element in $\mathbb{Z}/N\mathbb{Z}$. Suppose that we are working with the affine transformation is defined by the function

$$C \equiv_{26} 2P + b$$

In this case it's clear that $gcd(a, N) = gcd(2, 26) = 2$. We also know that

$$C \equiv_{26} 2P + b \equiv_{26} 2P + b + 26$$

which can be rewritten as

$$C \equiv_{26} 2(P + 13) + b.$$

This implies that both plaintext message units P and $P + 13$ map to the same ciphertext message meaning that we have a non-unique deciphering. A non-unique deciphering means that we do not have a 1-1 mapping between plaintext and ciphertext message units, thus we do not have a cryptosystem.

Now lets look at a more advanced problem where we wish to decipher a ciphertext message which was enciphered using an affine transformation

Example 2. Suppose that we have intercepted a string of ciphertext which was enciphered using an affine transformation. We also know that this transformation takes place using the 28 letter alphabet consisting of the letters A-Z which take on

numerical equivalents 0-25 as well as the characters and their numerical equivalents blank = 26 and ? = 27. We then perform a frequency analysis on the ciphertext and find that the two most frequently occurring letters are "F" and "D" in that order. Furthermore, we perform a frequency analysis on a large body of English text that uses this alphabet and find that the two most frequently occurring letters are "X" and "F" in that order. We conclude that "F" most likely maps to "X" and "D" most likely maps to "F". Mathematically this tells us that

$$5a^{'} + b^{'} \equiv_{28} 23$$

and that

$$3a^{'} + b^{'} \equiv_{28} 5.$$

Subtracting these two congruences allow us to eliminate the variable $b^{'}$, giving us that

$$2a^{'} \equiv_{28} 18$$

In this way, $a^{'}$ can be either 9 or 23. This is because

$$2a^{'} \equiv_{28} 18 = 2a^{'} \equiv_{28} 18 + 28$$

since 28 *mod* 28 = 0. By substituting these possibilities into our congruence equations we get that $b^{'}$ is either 6 or 20. Therefore we have two possible deciphering transformations, either that

$$P \equiv_{28} 9C + 6$$

or that

$$P \equiv_{28} 23C + 20.$$

We could try both and see which one produces a plaintext message that makes sense, or we could continue on with our frequency analysis and obtain the third most frequently occurring letter in the ciphertext. Doing so will give us another congruence equation which can be used to determine which possibility is correct. After performing another frequency analysis on the ciphertext, we find that "C" is the third most frequently occurring letter. Therefore we conclude that it most likely represents the third most frequently occurring letter in the body of English text that uses this alphabet, which is "Y". This additional piece of information gives us that

$$2a' + b' \equiv_{28} 24$$

and by subtracting it from either one of the original equations and solving for the parameters a and b, we get that $(9C + 6)$ *mod* 28 must be the correct enciphering transformation.

2.3 Affine Transformations Using Digraphs

Now that we have explored affine transformations of single message units, let's explore what affine transformations look like for digraphs. Recall that a digraph is simply a message unit which is made up of two letters. For example the word "READ" can be split up into the digraphs "RE" and "AD". Note that it is possible that a message is made up of an odd number of characters. In such a case, we simply append an additional character onto the end of the message so that each character gets its own digraph. Next we need to label each digraph as a number which we refer to as the *numerical equivalent*. Let's suppose that we are working with an alphabet consisting of N letters. The simplest way to label each digraph would be to calculate $xN + y$

for each digraph, where x,y are the numerical equivalents of the first and second letters in the digraph respectively. For example, if we were working in the regular A-Z alphabet, "BZ" would become $(1)(26) + 25 = 51$. Labeling each digraph in this way allows us to form a 1-1 mapping between each possible digraph and its numerical equivalent in the set $\mathbb{Z}/N^2\mathbb{Z} = \{0, 1, 2, 3, \ldots, N^2 - 1\}$.

Now that we have labeled each digraph, we decide on an affine enciphering transformation which rearranges the numerical equivalents $\mathbb{Z}/N^2\mathbb{Z}$. In this case we define the encryption to be $C \equiv_{N^2} aP + b$, where P represents the numerical equivalents of plaintext message units and C represents the numerical equivalents of P after they have undergone enciphering, in other words, the ciphertext numerical equivalents. Just as before, we need to ensure that $gcd(a, N) = 1$, which implies that $gcd(a, N^2) = 1$ as well, in order to have a deciphering transformation: $(P \equiv_{N^2} a^{'}C + b^{'}$ such that $a^{'} = a^{-1}\ mod\ N^2$ and $b^{'} = -a^{-1}b\ mod\ N^2$

Example 3. Now let us consider an example from *A Course in Number Theory*. In this example our adversary is using an affine cryptosystem over the 27 letter alphabet consisting of the letters A-Z as well as the '_' (blank) character which takes on the value 26. Suppose that we intercept a large body of ciphertext, perform an extensive frequency analysis on it and find that the most frequently occurring digraphs happen to be "ZA", "IA", and "IW" respectively. We know that these digraphs most likely correspond to the three most frequently occurring digraphs in the English language are "E ", "S ", and " T" respectively. Our goal is to find the deciphering key and decrypt the message "NDXBHO".

Solution Since we know that $C = (aP + b)\ mod\ N^2$, we can now substitute each corresponding digraph pair into this equation in an effort to deduce the enciphering key by means of simultaneous equations.

$675a^{'} + b^{'} \equiv_{729} 134$

$216a^{'} + b^{'} \equiv_{729} 512$

$238a^{'} + b^{'} \equiv_{729} 721$

We can eliminate $b^{'}$ by subtracting any two of the equations above, but it turns out is that if we subtract the first two we obtain that $459a^{'} = 351 \ mod \ 729$. There are 27 different solutions for $a^{'}$. A better approach would be to subtract the first and third congruence, obtaining that $437a^{'} = 142 \ mod \ 729$. By multiplying $437^{-1} \ mod \ 729$ on both sides we get that $a^{'} = 437^{-1} \cdot 142 \ mod \ 729$. In order to calculate $437^{-1} \ mod \ 729$, we need to make use of the Euclidean algorithm, a process which we will demonstrate below:

$729 = 437 \cdot 1 + 292$

$437 = 292 + 145$

$292 = 2 \cdot 145 + 2$

$145 = 72 \cdot 2 + 1$

Next, our goal is to write 1 as a linear combination of 437 and 729, in other words to obtain $1 \equiv_{729} B \cdot 437 + C \cdot 729$. This is because $(C \cdot 729) \ mod \ 729 = 0$, so this is the same as saying that $1 \equiv_{729} B \cdot 437$, which implies that B is the multiplicative inverse of 437 *mod* 729. Recall that the multiplicative inverse of B is the number B^{-1} such that $B \cdot B^{-1} \equiv_{729} 1$. We already have all the pieces that we need in order to write 1 as a linear combination of 437 and 729. We simply use the equations above in order to isolate 1. We do so by writing these equations in terms of their remainders. This process is as follows:

$1 = 145 - 72 \cdot 2 = 145 - 72(292 - 2 \cdot 145)$

$1 = 145 \cdot 145 - 72 \cdot 292$

$1 = 145(437 - 292) - 72 \cdot 292$

$1 = 145 \cdot 437 - 217 \cdot 292$

$1 = 145 \cdot 437 - 217(729 - 437)$

Therefore $1 = 362 \cdot 437 - 217 \cdot 729$. As we had mentioned above, any multiple of 729 *mod* 729 equates to 0, and so this is the same as saying that $1 \equiv_{729} 362 \cdot 437$.

Therefore we conclude that $a^{'} = 362 \cdot 142 \equiv_{729} 374$ which in turn allows us to calculate $b^{'} \equiv_{729} 647$. Now we have the deciphering key and applying the deciphering transformation to our digraphs "ND", "XB", and "HO" produces the integers 365, 724 and 24 respectively. This gives us that $365 = 13 \cdot 27 + 14, 724 = 26 \cdot 27, 24 = 0 \cdot 27 + 24$ which allows us to conclude that the message reads "NO WAY" [4], pages 59-60.

3 Enciphering Matrices

In the previous section, we saw how we could represent digraphs as numerical values by letting each digraph take the value of $xN + y$ such that $x, y \in \mathbb{Z}/N\mathbb{Z}$. Doing so gave us a 1-1 correspondence between each digraph and its numerical equivalent in $\mathbb{Z}/N^2\mathbb{Z}$. Ultimately this gave us a mapping from $\mathbb{Z}/N^2\mathbb{Z}$ to $\mathbb{Z}/N^2\mathbb{Z}$. With enciphering matrices, we also produce a mapping of digraphs, however this time we do so by means of vectors. Specifically, we let each digraph represent a vector

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

where $x, y \in N$. For example the digraph "HI" corresponds to the vector

$$\begin{pmatrix} 7 \\ 8 \end{pmatrix}.$$

This type of correspondence allows us to visualize all possible digraphs as vectors in $\mathbb{Z}/N\mathbb{Z} \times \mathbb{Z}/N\mathbb{Z}$. This is similar to $\mathbb{R}^2$ however each axis in our plane, $\mathbb{Z}/N\mathbb{Z}$ can only take discrete integers values. This allows us to interpret an enciphering transformation as a rearrangement of each vector within the plane.

Let us review some basic principles of linear algebra in an attempt to make the

upcoming details on enciphering matrices more clear. Recall that

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

such that a, b, c and $d \in \mathbb{R}$ as well as $x, y \in \mathbb{R}$.

Additionally, if we want to solve the equation $AX = B$, where A and X represent the matrix and vector mentioned above respectively, and B represents some arbitrary vector, we simply need to multiply both sides of the equation by A^{-1} assuming that it exists. Note that the inverse of a matrix exists if its determinant is not 0. Then the solution to this system becomes $X = A^{-1}B$. The determinant of a matrix A is given by $D = det(A) = ad - bc$, and its inverse, assuming its determinant is non-zero is given by

$$\begin{pmatrix} D^{-1}d & -D^{-1}b \\ -D^{-1}c & D^{-1}a \end{pmatrix}.$$

3.1 Linear Algebra Modulo N

In this section, we will cover enciphering transformations of the form $C = aP$ where $a^{-1} \in \mathbb{Z}/N\mathbb{Z}$. This represents a linear transformation where a is an enciphering matrix made up of components in $\mathbb{Z}/N\mathbb{Z}$.

We will commence our exploration of enciphering matrices by providing a description of the type of matrices that we need in order to produce enciphering/deciphering keys. As a start, the matrices we want will have components which are elements of a ring R. Recall that a ring R is a field except it doesn't require the existence of multiplicative inverses and multiplicative commutativity. A field is a set which forms an Abelian group under addition, as well as an Abelian group under multiplication excluding the additive identity element. For example, the real numbers form a field, while the $\mathbb{Z}/N\mathbb{Z}$ forms a field if N is prime. This is because each element in $\mathbb{Z}/N\mathbb{Z}$

will be relatively prime to N if it is prime, and so each element (excluding 0) will have a multiplicative inverse.

Let R represent an arbitrary ring. We define $M_2(R)$ to be the set of all 2×2 matrices with entries in R and call it the "matrix ring over R" [4], page 68. Note that while R itself forms a commutative ring, $M_2(R)$ forms a ring as well, but commutativity doesn't hold as the order by which we multiply matrices matters.

Let's suppose that we have an arbitrarily chosen ring R and an arbitrarily chosen matrix

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in M_2(R).$$

Then we have a similar situation when taking the inverse of this matrix to the case where we have matrices over the real numbers. Let's assume that $D = det(A) = ad - bc$ is non-zero in R. Then by definition its inverse $D^{-1} \in R^*$, that is the set of all invertible elements in R. This is equivalent to saying that the $gcd(D, N) = 1$. Then we find that the product of the matrix multiplication

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} D^{-1}d & -D^{-1}b \\ -D^{-1}c & D^{-1}a \end{pmatrix}$$

is equivalent to the matrix

$$\begin{pmatrix} D^{-1}(da - bc) & -0 \\ 0 & D^{-1}(-cb + ad) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Multiplying in the opposite order gives us the same result. Thus, by definition, the inverse matrix of A is given by the same formula as taking the inverse of matrices over the real numbers. To summarize, we have found that if $det(A) \in R^*$ then A is an invertible matrix so naturally we can use it as an enciphering key, and its inverse as a deciphering key.

Now that we have confirmed that taking the inverse of a matrix over a ring R takes the same structure as that over the reals, let's consolidate our understanding with an example.

Example 4. Find the inverse of

$$\begin{pmatrix} 9 & 3 \\ 5 & 2 \end{pmatrix}$$

modulo 26. The determinant is given by $D = det(A) = ad - bc = 9(2) - 5(3) \equiv_{26} 3$. By the Euclidean algorithm we can say that $26 = 3(8) + 2$ and that $3 = 2 + 1$, which confirms that $gcd(26, 3) = 1$. This implies that the matrix A has an inverse. Now by iterating backward through the Euclidean algorithm we can find D^{-1}. We then get that $1 = 3 - 2 = 3 - (26 - (3)(8)) = 3 - 26 + 3(8) = 3(9) - 26$ all modulo 26

Since multiples of $k \cdot 26 \equiv_{26} 0$ such that $k \in \mathbb{Z}$ we can conclude that $1 \equiv_{26} 3(9)$. Thus we have that $D^{-1} \equiv_{26} 9$. Now that we have D^{-1} we know that A^{-1} must be

$$\begin{pmatrix} 9(2) & -9(3) \\ -9(5) & 9(9) \end{pmatrix} = \begin{pmatrix} 18 & -27 \\ -45 & 81 \end{pmatrix}.$$

Taking this matrix modulo 26 gives us

$$\begin{pmatrix} 18 & 25 \\ 7 & 3 \end{pmatrix}.$$

Now let's check that this is actually the inverse matrix. By taking the product of AA^{-1}, we get that

$$\begin{pmatrix} 9 & 3 \\ 5 & 2 \end{pmatrix} \begin{pmatrix} 18 & 25 \\ 7 & 3 \end{pmatrix}$$

modulo 26. This is congruent to

$$\begin{pmatrix} 183 & 234 \\ 104 & 131 \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Proposition 5.

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in M_2\mathbb{Z}/N\mathbb{Z}, D = ad - bc$$

Then the following statements are equivalent

(*a*) *gcd(D,N) = 1*

(*b*) *A has an inverse matrix*

(*c*) *if x and y are not both 0 in* $\mathbb{Z}/N\mathbb{Z}$, *then* $A \begin{pmatrix} x \\ y \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

(*d*) *A gives a 1-to-1 correspondence of* $\mathbb{Z}/N\mathbb{Z}$ *with itself.*

This proposition can be found on [4], page 70.

Proof. We have already proven that $(a) \implies (b)$ a few paragraphs up. We now wish to show that $(b) \implies (d) \implies (c) \implies (a)$

Suppose that (b) is true. We know that a function is $1-\text{to}-1$ *if and only if* it has an inverse, and since the inverse exists, we conclude that the inverse mapping exists from

$$\begin{pmatrix} x' \\ y' \end{pmatrix}$$

to

$$\begin{pmatrix} x \\ y \end{pmatrix}.$$

Next, we wish to show that $(d) \implies (c)$. Since A gives us a $1 - to - 1$ mapping

from $\mathbb{Z}/N\mathbb{Z}$ to $\mathbb{Z}/N\mathbb{Z}$, and since

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

maps to

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

we conclude that no other vectors map there as well. Thus if x, y are not both zero, then

$$A \begin{pmatrix} x \\ y \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Now we wish to show that $(c) \implies (a)$. We will prove this via the contrapositive. That is, if (a) is not true, then (c) is not true. So let's suppose that (a) is false, and set $m = gcd(D, N) > 1$ and let $m' = N/m$. We then have three possible cases:

$Case(i)$. If all four entries of A are divisible by m, set

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -bm' \\ am' \end{pmatrix}$$

which will give us a contradiction to (c)

Proof: Suppose that a, b, c, d are divisible by m, set

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} m' \\ m' \end{pmatrix}.$$

So

$$A \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} m' \\ m' \end{pmatrix} = \begin{pmatrix} am' + bm' \\ cm' + dm' \end{pmatrix} = \begin{pmatrix} m'(a+b) \\ m'(c+d) \end{pmatrix}.$$

Since $m^{'} = \frac{N}{m}$, this is equivalent to the vector

$$\begin{pmatrix} N\frac{a+b}{m} \\ N\frac{c+d}{m} \end{pmatrix}.$$

Since $m|a$ and $m|b$, $m|(a+b)$, so $\frac{a+b}{m} = K$ where K is an integer. Thus $\frac{N(a+b)}{m} = NK$ and we know that NK is equivalent to 0 mod N

$Case(ii)$. If a, b are not both divisible by m, set

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -bm^{'} \\ am^{'} \end{pmatrix}.$$

It is important to note here that since if a and b are both divisible by m, then this is equivalent to the zero vector modulo N. Therefore we are not dealing with the zero vector in this case.

Then

$$A\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ab \\ cd \end{pmatrix}\begin{pmatrix} -bm^{'} \\ am^{'} \end{pmatrix} = \begin{pmatrix} -abm^{'} + bam^{'} \\ -cbm^{'} + dam^{'} = m^{'}(da - bc) = m^{'}D \end{pmatrix} = \begin{pmatrix} 0 \\ Dm^{'} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The top component of the vector equals zero under subtraction, while for the second component, we know that $m|D$, therefore it follows that $mm^{'}|Dm^{'}$ and since $N = mm^{'}$, we have that $N|Dm^{'}$. This implies that $Dm^{'} = KN$ such that K is an integer, which is equivalent to 0 modulo N.

$Case(iii)$ If we have a situation in which a and b are both divisible by m, but either c or d is not, set

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} dm^{'} \\ -cm^{'} \end{pmatrix},$$

and simply proceed in the same manner as $Case(ii)$ as our proof. Therefore we have

proven that all these statements are all equivalent (Koblitz, 1994, p. 70). □

The importance of proposition 1 cannot be stressed enough. It is the heart of enciphering matrices. To summarize, it says that we are able to produce an enciphering transformation (as well as a deciphering transformation) by obtaining a matrix $A \in M_2(\mathbb{Z}/N\mathbb{Z})$ whose determinant is relatively prime to N. If $det(A) = D \in (\mathbb{Z}/N\mathbb{Z})^*$, then its inverse exists. Therefore we have that

$$A \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

gives us an enciphering transformation from plaintext message units to ciphertext message units, while

$$A^{-1} \begin{pmatrix} x' \\ y' \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix}$$

gives us our deciphering transformation from ciphertext to plaintext. In this way, we can write the enciphering transformation as $C = aP \ mod \ N$ where a represents our invertible A matrix and P are plaintext digraph vectors which map to ciphertext vectors C. Writing it in this way allows us to see the similarities between enciphering matrices and linear transformations that we encountered in earlier sections. We will investigate the enciphering process here, however it is not difficult to see that the deciphering process takes place in the same way.

Suppose that we have finished writing our message. We then divide the characters in the message into digraphs. Suppose that there end up being n digraphs. Then our plaintext can be made into a $2 \times n$ matrix consisting of each digraph, that is $P = P_1P_2P_3...P_n$. We can encipher all of this in a single step by multiplying our enciphering matrix A, which is necessarily a 2×2 matrix, by our matrix P. In other words $C = AP$ yields an output matrix which is also $2 \times n$ made up of the ciphertext digraphs corresponding to each plaintext digraph.

Let's look at a simple example of how we would encipher a string of characters.

Example 6. Suppose that we have the enciphering matrix

$$A = \begin{pmatrix} 3 & 7 \\ 1 & 4 \end{pmatrix}$$

where R represents the ring $\mathbb{Z}/27\mathbb{Z}$. In this case $D = 5$ and since $gcd(D, 27) = 1$, this is a justifiable matrix as it is invertible. We wish to encipher the string of characters "CODE RED" which we will send as a message to one of our allies. We then split the message into its digraphs, those being "CO", "DE", " R", and "ED" and turn these digraphs into vectors where the first component represents the first letter of each digraph and the second component represents the second letter. Note that blank character has the numerical equivalent 26. Doing so gives us the vectors

$$\begin{pmatrix} 2 \\ 14 \end{pmatrix}, \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \begin{pmatrix} 26 \\ 18 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

which we concatenate into the single larger matrix

$$P = \begin{pmatrix} 2 & 3 & 26 & 4 \\ 14 & 4 & 18 & 3 \end{pmatrix}.$$

Then our ciphertext matrix becomes

$$\begin{pmatrix} 3 & 7 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} 2 & 3 & 26 & 4 \\ 14 & 4 & 18 & 3 \end{pmatrix} = \begin{pmatrix} 104 & 37 & 204 & 33 \\ 58 & 19 & 98 & 16 \end{pmatrix}$$

which we then *mod* by $N = 27$, resulting in the matrix

$$\begin{pmatrix} 23 & 10 & 15 & 6 \\ 4 & 19 & 17 & 16 \end{pmatrix}.$$

These digraphs then correspond to the ciphertext message "XEKTPRGQ".

Example 7. Now let's consider a more advanced problem. Suppose we have intercepted a ciphertext message from our adversary, which we know uses a 27-letter alphabet, where A-Z represents the values 0-25 and 26 represents "!". The message reads "TGLZOUR!LT". Let's assume we have found out that the last 4 letters represent a special character as well as our adversary's signature, namely "!TED". Therefore we know that the ciphertext digraphs "R!" and "LT" correspond to the plaintext digraphs "!T" and "ED". In other words we know that the matrix P is

$$\begin{pmatrix} 26 & 4 \\ 19 & 3 \end{pmatrix},$$

while C is

$$\begin{pmatrix} 17 & 11 \\ 26 & 19 \end{pmatrix}.$$

We wish to use this information to decrypt the entire message. Recall that $A^{-1} = PC^{-1}$. This implies that

$$A^{-1} = \begin{pmatrix} 26 & 4 \\ 19 & 3 \end{pmatrix} \begin{pmatrix} 17 & 11 \\ 26 & 19 \end{pmatrix}^{-1}.$$

First we need to find C^{-1} in order to calculate A^{-1}. Recall that the inverse of a

matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

is given by

$$\begin{pmatrix} D^{-1}d & -D^{-1}b \\ -D^{-1}c & D^{-1}a \end{pmatrix}.$$

We know that the $D_c = det(C) \equiv_{27} (17 \cdot 19) - (11 \cdot 26) \equiv_{27} 10$. The next step is to find D_c^{-1} which we will calculate using the Euclidean algorithm in the next paragraph.

Finding 10^{-1} *mod* 27 using the Euclidean Algorithm:

$27 = 10 \cdot 2 + 7$

$10 = 7 \cdot 1 + 3$

$7 = 3 \cdot 2 + 1$

Next we will step backwards using this information in order to write 1 as a linear combination of 10 and 27. This goes as follows

$1 = 7 - 3 \cdot 2 = 7 - 2(10 - 7) = 3 \cdot 7 - 2 \cdot 10$

This is equivalent to $3(27 - 10 \cdot 2) - 2 \cdot 10 = 3 \cdot 27 - 8 \cdot 10$ all *mod* 27

Since we are working modulo 27 and any multiple of 27 *mod* 27 is equal to 0, we can simplify the above expression down to $-8 \cdot 10 \equiv_{27} 1$.

In other words we have found that -8 is the multiplicative inverse of 10 under modulo 27. We know that $-8 \mod 27$ is congruent to $-8 + 27 = 19 \mod 27$.

Therefore we finally have that $D_c^{-1} \equiv_{27} 19$.

This tells us that

$$C^{-1} = \begin{pmatrix} 19 \cdot 19 & -19 \cdot 11 \\ -19 \cdot 26 & 19 \cdot 17 \end{pmatrix}$$

modulo 27 which equals

$$\begin{pmatrix} 10 & 7 \\ 19 & 26 \end{pmatrix}.$$

We now have that

$$A^{-1} = PC^{-1} = \begin{pmatrix} 26 & 4 \\ 19 & 3 \end{pmatrix} \begin{pmatrix} 10 & 7 \\ 19 & 26 \end{pmatrix} = \begin{pmatrix} 12 & 16 \\ 4 & 22 \end{pmatrix}$$

So now we have the deciphering key A^{-1}, and so to recover the plaintext, all we need to do is compute

$$A^{-1}C = \begin{pmatrix} 12 & 16 \\ 4 & 22 \end{pmatrix} \begin{pmatrix} 19 & 11 & 14 & 17 & 11 \\ 6 & 25 & 20 & 26 & 19 \end{pmatrix} = \begin{pmatrix} 0 & 19 & 2 & 26 & 4 \\ 19 & 0 & 10 & 19 & 3 \end{pmatrix},$$

which reads "$ATTACK!TED$".

4 Public Key

Before investigating public key cryptography, it is a good idea to recall some of the definitions that we spoke about in the previous sections. Firstly, we remind the reader that a *cryptosystem* refers to a 1 to 1 mapping between the set of all plaintext message units and ciphertext message units. The mapping that we speak of refers to the enciphering transformation which takes in plaintext message units as inputs in the domain and outputs ciphertext message units in the range, a process in which we scramble the alphabet of characters in our message to produce a new message which appears meaningless.

More generally, cryptosystems can be thought of as families. For example, consider the affine cryptosystem defined as $C \equiv_N aP + b$ where $P = C = \mathbb{Z}/N\mathbb{Z}$. Additionally, a and b are parameters such that $a \in (\mathbb{Z}/N\mathbb{Z})^*$, the set of invertible elements in $\mathbb{Z}/N\mathbb{Z}$, and $b \in \mathbb{Z}/N\mathbb{Z}$. For some fixed N, which we can think of as our alphabet, the family of affine cryptosystems represent all of the possible enciphering transformations which can be created for different values of a and b, as well as their corresponding

deciphering transformations. In other words, a single enciphering transformation can be described by two pieces of information, namely an algorithm as well as its parameters. The algorithm refers to the general process by which the enciphering and deciphering takes place, whereas the parameters refer to the variables within the enciphering transformation, in our case a, b. The parameters of our cryptosystem form the enciphering keys, K_D as well as the deciphering keys K_E. What this tells us is that knowing the keys and general algorithm allows us to both encrypt and decrypt the cryptosystem.

Recall some of the simple cryptosystems which we have already spoken about in this paper. The first types of cryptosystems which we spoke of were *shift transformations* defined by the function $C = P + b \ mod \ N$. This is the most basic type of cryptosystem that exists and can be easily be broken by a characteristic of language known as frequency analysis. Recall that frequency analysis allows us to match the most frequently occurring letters or groupings of letters in a message (such as digraphs), to the most frequently occurring letters or grouping in the specific language. This matching then allows us to substitute corresponding plaintext and ciphertext pairs into our enciphering transformation and solve for the unknown parameters which give us our enciphering and deciphering keys.

The second type of cryptosystem that we spoke of was affine cryptosystem. This system, which is similar to a shift transformation but also includes a multiplication of our plaintext message units as part of the scrambling of message units, can also be easily solved using frequency analysis.

What we notice in each of these cryptosystems is that if we know the enciphering key, we can easily figure out the deciphering key. For shift transformations, knowing the enciphering shift parameter a allows us to easily determine the deciphering shift parameter, $-a$. For affine transformations, knowing the enciphering key, $K_E = (a, b)$, allows us to easily determine the deciphering key, $K_D = (a^{-1}, -a^{-1}b)$. For encipher-

ing matrices, knowing the enciphering matrix A allows us to easily determine the deciphering matrix A^{-1}. If it is not apparent enough, with these systems knowledge of how to encipher is equivalent to knowledge of how to decipher. In fact, it was actually considered foolish to think that a person who could decrypt a system by finding the deciphering key was not able to figure out the corresponding enciphering key [4], page 84. For this reason, both the enciphering key and deciphering key needed to be kept hidden from the general public, and therefore these cryptosystems are referred to as *private key cryptosystems*.

This is where *public key cryptography* comes into play. By definition, public key cryptosystems are cryptosystems in which we cannot use the enciphering key in order to find the deciphering key. In other words, public key cryptosystems make use of a function $f : P \to C$ which is easy to compute with knowledge of the enciphering key, K_E , while it's inverse $f^{-1} : C \to P$ is extremely difficult to compute despite knowing K_E. This type of cryptosystem, dissimilar to private key cryptosystems in that we can make the enciphering keys available to the public, was first developed by W. Diffie and M.Hellman in 1976 and introduced to the world in their book entitled 'New directions in cryptography'.The special function which we mentioned above is referred to as a trapdoor function. A trapdoor function is a mathematical function which is very easy to compute, while its inverse is almost impossible to compute without access to additional pieces of information (such as the deciphering key). Therefore we can think of a trapdoor function as a function which is 'non-invertible'.

A trapdoor function is very similar to the idea of a 'one-way function' which has also been used in cryptography. This too is a function which is easy to compute while its inverse is difficult to compute, however the difference between the two lies in the fact that with trapdoor functions, having access to additional information can make it a lot easier to decrypt the cryptosystem while one-way functions are not made any easier to decrypt by having access to more information. In order to develop a better

understanding of a one-way function, let's consider a historical example of a time in which it was used. This example derives from a book written by the author Wilkes in 1968 entitled 'Time-Sharing Computer Systems'. In this book, Wilkes speaks about a 'one-way' cipher used by R.M. Needham which allowed computers to store the enciphered versions of user's passwords instead of the passwords themselves, so that an adversary could not attempt to log in as a user's identity. So how did this system work? Whenever a user of the system chooses his/her password, it is immediately scrambled into an enciphered form. It is in this enciphered form in which passwords are stored in the computer's database. When a user types their password into a computer on the system, the password undergoes enciphering and this result is then compared against the enciphered version of the password stored on the database in order to see if they match [4], page 86.

For these reasons, it would be meaningless if an adversary were to get their hands on the list of enciphered passwords, because she would still need to decipher these passwords in order to use them, and to decipher them, she would need specific details about the deciphering and enciphering algorithm. Furthermore, even if she were to have access to this specific information, it would still take her ages to decipher. In returning to trapdoor functions which are at the heart of public key cryptography, it is important to recognize that the definitions that we have provided are not clear-cut from a mathematical point-of-view. When we say that a trapdoor function $f : P \to C$ is not invertible, we do not mean that its inverse does not exist or that it cannot be calculated. Instead we mean that f is not invertible in terms of *realistic computability* [4], page 86. In actuality, the inverse does exist and can be calculated, however it cannot be calculated in a 'realistic' amount of time. Therefore, trapdoor functions are closely related to the 'notion of information that can be accessed through feasible computation' [5]. In short, it takes too long to compute. In trying to compute the inverse, we exceed the window of time available to us to decrypt the system.

After this time has elapsed, the parameters that make up the decryption keys could have changed, taking us back to square one. In other words, a trapdoor function is a function which is computationally non-invertible even with knowledge of the encryption key.

While we can think of a trapdoor function as non-invertible in terms of modern computers, as technology and algorithms continue to develop over time it is possible that we can reach a place technologically in which the inverse can be found fairly quickly. Therefore it is possible that a function which is considered trapdoor in 2007 can lose its trapdoor title in 2021. So while public key cryptosystems perform successfully for us today, it is possible that these cryptosystems will be as weak to us in the future as affine transformations and enciphering matrices are for us today.

If it has not been made apparent enough thus far, the reason that public key cryptosystems have their name, is because their trapdoor mechanism means that their enciphering key, K_E, can be made public (publicly accessible or known to the public), without anybody being able to decipher the cryptosystem and gain access to private information. So why is this helpful? Let's suppose that we have a group of users of a specific public key cryptosystem. Each user of the system wants to be able to send and receive information to and from other users without a middle-man being able to intercept and read their messages. Conveniently public key cryptosystems, because of their trapdoor mechanism, allow for a central organization to collect each user's public key and publish all of these keys in a book similar to a phone-book [4], page 87. In this book we can find the name of any user of the cryptosystem and their corresponding public key.

Then if some user, let's call her Alice, wants to send a message to another user, let's call him Bob, all she needs to do is look up Bob's public key in the phone book and use the general enciphering algorithm of the cryptosystem along with Bob's enciphering key, K_B in order to scramble the message so that it is incomprehensible

to an adversary. Then only the intended recipient of the message, Bob, can use his private key in order to unlock the message (as he is the only person with access to the deciphering key).

To summarize what we have discussed so far, a classical cryptosystem, also known as a private key cryptosystem, is a cryptosystem in which if we know the enciphering key, the process of deciphering can take proportionately how long it took to encipher. In each of the cryptosystems that we have discussed so far, the process of enciphering and deciphering are more or less the same. Of course there exist private key cryptosystems in which the deciphering process takes significantly longer than the enciphering process. Consider Enciphering Matrices as an example of this. Taking the inverses of matrices is crucial to the deciphering process, and finding the multiplicative inverse of the determinant is crucial to finding the inverses. We previously learnt that finding the multiplicative inverse of numbers under modular arithmetic requires us to make use of the Euclidean algorithm, a process which can take a long time. This tells us that the deciphering process, assuming that we have knowledge of the enciphering key, can take a lot longer than enciphering process when working with certain cryptosystems that aren't considered public key. The main point here is that despite taking a longer amount of time, the deciphering process still takes place in a 'realistic' amount of time as opposed to public key cryptosystems which take far too long to be feasible. In other words, the amount of time, despite being long, doesn't prevent somebody from being able to decipher the message. Generally, if the time it takes to decipher is many orders of magnitude greater than the time it takes to encipher then we are dealing with a public key cryptosystem.

5 Authentication

Authentication in cryptography is a way of verifying that the message that has been received is really from the claimed sender (the person whose name is signed on the top or bottom of the sent document). The name of the sender is referred to as a signature, but could more appropriately be thought of as a digital signature. This digital signature also often consists of the date and time at which the message was sent as further evidence that the message comes from the sender.

Public key cryptography allows users to very easily identity oneself in such a way that nobody could be pretending to be you [4], page 88. Let's take a look at how authorization works using public key cryptography. Let's suppose that Alice is in the process of sending a message to Bob and wants to mark her message with a digital signature. Suppose that f_A represents the enciphering transformation that Alice uses in order to send Bob the message. We will also suppose that S represents Alice's signature. It would be insufficient for Alice to send Bob the message $f_B(S)$, since anybody that knows Bob's public key is capable of encrypting S and sending to Bob. Instead, Alice should send the message $f_B f_A^{-1}(S)$. Then Bob receives the message and applies f_B^{-1}. Once he has done this, all of the message apart from a small portion of gibberish (in the place of the Alice's signature) has been converted to plaintext. Specifically, this section of gibberish is $f_A^{-1}(S)$, since $f_B^{-1} f_B f_A^{-1}(S) = f_A^{-1}(S)$. In order to unscramble this gibberish, Bob uses Alice's public key and applies the transformation f_A which results in $f_A f_A^{-1}(S) = S$. The reason that this works is because Alice is the only person that could have hidden her signature,S, by applying the transformation $f_A^{-1}(S)$ (as one can only apply the inverse function if they have access to the deciphering key). Therefore, Bob knows that this message has to have come from Alice herself, and not somebody pretending to be her.

6 RSA

The RSA cryptosystem is one of the most famous and oldest public key cryptosystems that is still in use today. The acronym RSA stands for Rivest, Shamir and Adleman who were the original creators of the cryptosystem. So what is it about RSA that makes it so successful? It turns out that embedded within the RSA trapdoor function is a very simple mathematical idea that makes the function $f : P \to C$ very easy to compute, while making the reverse process, namely $f^{-1} : C \to P$ nearly impossible to compute (in terms of *realistic computability*). As we had mentioned in the previous section, the hallmark of a public key cryptosystem is in its inability to be inverted and with RSA, there is a long history of evidence demonstrating people's difficulty doing so [4], page 92. However, it is not only the security that RSA provides which makes it so successful. It is also, relative to other cryptosystems as well as in terms of mathematical concepts, very simple. Take note that when we say simple, we refer not only to the mathematical concepts underlying RSA, but also in terms of its implementation. So what is this simple concept that we keep on talking about? We are referring to an ancient concept in number theory; the difficulty in finding the prime factorization of a composite number given that these primes are not known in advance. In fact, it is this difficulty of factoring that is at the heart of RSA's success and technological stamina.

6.1 How RSA Works

The first thing that happens is the user, let's call her Alice, needs to choose two extremely large prime numbers (over hundreds of digits long) which we will call p and q. It is important to point out that the numbers p and q were randomly chosen via a random number generator. Let's take a look at how this works. The generator first randomly picks, using statistical properties of a truly random sequence, a number

r. If r is even, then it is obviously not a prime as it contains the factor of 2, so this number is incremented to be $r + 1$, an odd number. The computer then conducts primality tests on this number in order to determine whether or not it is prime. If it is not, then we increment the number $r+1$ by 2 in order to obtain the next odd number and then once again conduct primality tests on that number. If it not prime, then we continue to repeat this process until we run into a prime number. Specifically, primes occur once every $log(r)$ numbers away from r itself and so we would expect to test $log(r)$ numbers at worst before finding the closest prime to r [4], page 92.

Once Alice has obtained these primes, she multiplies them together to get $n = pq$. This composite number n which is made up of p and q will be very difficult to split up into its prime components once they have been multiplied together (assuming that one doesn't know them in advance). This is analogous to mixing 2 obscure colors of paint together and then asking somebody looking at this strange mixture, without knowing which colors were mixed in the first place, which two colors have been mixed together.

Alice then proceeds to makes use of a special mathematical function known as the *phi* function which is written as $\phi(n)$. This function takes in a natural number n as an input and calculates how many natural numbers less than n are relatively prime to it. Recall that a number which is relatively prime to n shares a greatest common divisor of 1 with it. While it appears difficult on the surface to calculate how many natural numbers less than n are relatively prime to it, it is given by the simple formula $\phi(n = pq) = (p - 1)(q - 1)$. By factoring this out, we get that $\phi(n) = pq - p - q + 1 = n - p - q + 1$ which is an alternative way in which this function is expressed. It is not hard for the reader to recognize that by knowing the primes p, q which make up n, implementing this function is very easy, but without knowing p, q, it is extremely difficult to calculate $\phi(n)$ (for very large values of n). If we don't know p and q, unsurprisingly, we have to manually iterate through all

of the numbers in the set $\mathbb{Z}/N\mathbb{Z} = \{0, 1, 2, 3, \ldots, N-1\}$ and count all the numbers $a \in \mathbb{Z}/N\mathbb{Z}$ such that $gcd(a, n) = 1$.

The next thing that Alice needs to do is to find the number e such that $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$. She needs this number to be relatively prime to $\phi(n)$ because that guarantees that it has a multiplicative inverse which we will call d that acts as her decryption key. To clarify, we have that if $gcd(e, \phi(n)) = 1$, then we have that $e \cdot d \equiv 1 \ mod \ \phi(n)$. This implies that $ed \equiv k \cdot \phi(n) + 1$ where k is an integer, a property which is crucial to the RSA cryptosystem. We will elaborate on its importance a little later, but for now we need to remember if e being relatively prime to $\phi(n)$ ensures that a decryption key, d, exists and the existence of a decryption key ensures the existence of an inverse function $f^{-1} : C \to P$. In other words, it ensures us of a decryption algorithm.

Alice's encryption key, K_e, is then given by (e, n) and is made known to the public. Furthermore, the encryption algorithm $f : P \to C$ is defined as $f(P) = P^e = C \ mod \ n$ where P is the set of all plaintext message units and C is the set of all ciphertext message units. On the other hand, the deciphering algorithm is given by $f^{-1}(C) = C^d \ mod \ n$. Let us take a detailed look at why these two functions are inverses of one another. We know that for an arbitrary function g, g^{-1} is the inverse function such that $gg^{-1} = I$, where I is the identity mapping. In our case we have that $ff^{-1} = (P^e)^d = P^{ed} \ mod \ n$ and since $ed \equiv k\phi(n) + 1$, this implies that $P^{k\phi(n)+1} = P^{k\phi(n)} \cdot P \ mod \ n$. A general rule is that $X^{\phi(n)} \equiv 1 \ mod \ \phi(n)$. This tells us that $P^{k\phi(n)} \cdot P \ mod \ \phi(n) = (P^{\phi(n)})^k \cdot P = 1^k \cdot P = P \ mod \ n$. Therefore, we conclude that $ff^{-1}(P) = P$ for all values of P and thus we have achieved the identity mapping ,proving that $f = P^e \ mod \ n$ and $f^{-1} = C^d \ mod \ n$ are inverse functions. So once Alice has calculated e and consequently d, its multiplicative inverse, she officially has both an encryption and decryption algorithm. Below we shall consider a basic example in order for the reader the develop a better understanding of how encryption and

decryption works using RSA.

Example 8. Suppose that Alice wishes to send Bob the message "WIN". Additionally, we shall suppose that she is working in the standard 26 letter alphabet, in other words that $N = 26$ and that the plaintext message units are trigraphs while ciphertext message units are four graphs. Note that in addition to guiding the reader through the encyrption and decryption process, this example will also make a case for why the ciphertext message units are necessarily longer than the plaintext message units as expressed in the previous sentence.

Alice first needs to encode "WIN" into its numerical equivalent. To do this, she uses the $N = 26$ trigraph labeling convention, namely $26^2X + 26Y + Z$ such that X, Y, Z are the letters in the trigraph. She then obtains $26^2 \cdot 22 + 26 \cdot 8 + 13 = 15093$. In other words, she gets that $P = 15093$. In order to encipher P, Alice needs to make use of Bob's public key in order to address the message to Bob. His key, $K_B = (17623 = e_B, 39203 = n_B)$. The enciphering transformation is given by $f(P) = P^{e_B} \ mod \ n_B$ which is $15093^{17623} \ mod \ 39203 = 18675 = C$. Take note that the set of all trigraph message units, namely $\mathbb{Z}/N^3\mathbb{Z} = \{0, 1, 2, 3, \ldots, 26^3 - 1 = 17575\}$.

In other words, the ciphertext message unit, 18675 is too big to fit in this set which poses a problem. For this reason, Bob needs to structure his cryptosystem in such a way that it allows ciphertext message units to fall within a range larger than $N^{k=3}$, and so he sets the range to be $N^{L=4}$. This range represents the set of all possible four-graphs. In other words, since the enciphered version of "WIN" cannot be expressed as a trigraph, we have to express it as a larger string of numbers, a four-graph. Therefore when Bob is setting up his public key, he needs to compute n_B in such a way that $N^k < n_B = pq < N^L$ where $K < L$ are natural numbers in order to account for the fact that some $K - graphs$ will have to map to larger $L - graphs$. It also turns out that structuring his cryptosystem in this way, ensures that each $k - graph$ has a unique representation as an $L - graph$, whereas neglecting such a

structure runs the risk of mapping mapping multiple $k-graphs$ plaintext message units to the same ciphertext message unit. If this were the case, our enciphering transformation would be a many-to-1 mapping and so it wouldn't have an inverse, thus it wouldn't have a deciphering transformation. Therefore Bob has ensured a one-to-one mapping between plaintext and ciphertext message units and in turn ensured that an enciphering algorithm along with it's corresponding deciphering algorithm exist. In short, he has ensured that he has a cryptosystem.

Let us return to the problem at hand. Alice has obtained that $C = 18675$. As a four-graph this is represented alphabetically as $26^3 \cdot 13 + 26^2 \cdot 8 + 26 \cdot 22 + 22 =$ "NIWW" (it is a coincidence that the ciphertext contains the same letters as the plaintext). Once Bob has received the scrambled message, "NIWW", he takes its numerical equivalent, 18675 and then uses his private key, $K_{D_B} = (21583 = d_B, 39203 = n_B)$ and applies the deciphering transformation $f^{-1} = C^{d_B} \ mod \ n_B = 18675^{21583} \ mod$ 39203 and gets 15093 as a result. He then converts this to a trigraph, and finds that it corresponds to the message "WIN".

7 Summary

We began this paper by defining cryptography and speaking about some of its common applications in today's world. We then proceeded by examining the mathematics behind the most fundamental private key cryptosystems such as shift transformations and affine transformations and explored the vulnerability of these systems to the simple cryptanalytic technique known as frequency analysis. We then explained how these fundamental principles behind affine transformations can be translated to another type of private key cryptosystem known as enciphering matrices. Furthermore, we explained that public key cryptosystems were different to private key cryptosystems in that we could make the user's public key available to the public and

that public key cryptosystems use a trapdoor mechanism that makes the decryption process extremely difficult. We concluded this paper by analysing one of the most common private key cryptosystems in use today known as RSA and saw how it uses authentication in order to ensure that a message or information is from the sender.

References

[1] Ochoche Abraham and Ganiyu O Shefiu. An improved caesar cipher (icc) algorithm. *International Journal Of Engineering Science & Advanced Technology (IJESAT)*, 2:1198–1202, 2012.

[2] Tony M Damico. A brief history of cryptography. *Inquiries Journal*, 1(11), 2009.

[3] Hans Delfs, Helmut Knebl, and Helmut Knebl. *Introduction to cryptography*, volume 2. Springer, 2002.

[4] Neal Koblitz. *A course in number theory and cryptography*, volume 114. Springer Science & Business Media, 1994.

[5] Andrew C Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 80–91. IEEE, 1982.