Understanding and Evaluating the Mathematical Techniques used in RSA Cryptography

Research Question: How are discrete mathematical techniques, such as modular arithmetic, Fermat's Little Theorem, Chinese Remainder Theorem, the Euclidean Algorithms and Euler's Totient Function, applied in making RSA a strong and secure cryptosystem?

Subject: Mathematics

Date: 09/09/2020

Word Count: 3880

1.0 Introduction

Since the advent of written communication, mankind has been ever striving to innovatively improve the ways to conceal the information. *Cryptography* is the science that deals with the techniques of secret communications (Gallier and Quaintance, 2020). The sender of the message conceals it, using a process called *encryption*, and the receiver uses a reverse process called *decryption* to decode or decipher it to get the original text sent by the sender (Asdad, 2019). Cryptography is predominantly used by governments, secret services agencies, military agencies and websites that deal with credit cards or personal information exchange (Mumir, 2005). The *RSA cryptosystem* is one such widely used systems developed by Ron Rivest, Adi Shamir and Leonard Adleman in 1976. The strength of the RSA cryptosystem is fundamentally based on the number theory, pertaining to the intractability of large prime number factorization (Mumir, 2005). This extended essay aims to understand and explore the discrete mathematical theorems and principles that are used in the RSA cryptosystem, and analyze and evaluate the role of mathematics in strengthening security. This motivation led to the primary research question, “How are discrete mathematical techniques, such as modular arithmetic, Fermat's Little Theorem, Chinese Remainder Theorem, the Euclidean Algorithms and Euler’s Totient Function, applied in making RSA a strong and secure cryptosystem?”

2.0 Background

This section covers all the background mathematical concepts required to understand and appreciate the RSA cryptosystem. The concepts are also accompanied by their proofs for better understanding.

2.1 Modulus Function

<u>*Definition*</u>:

Let $m \in Z^+$, and $a, b \in Z$, it is possible to write:

$a \equiv b \ (mod\ m)\ if\ m \mid (a - b) \Rightarrow a = km + b\ for\ some\ k \in Z$

This is stated as '*a is congruent to b modulo m*'. In simpler terms, a and b yield the same remainder when divided by m (Keef and Guichard, 2020).

Examples:

$1 \equiv 6 \equiv 21 \equiv 66 \ (mod\ 5)$

$2 \equiv 13 \equiv 24 \equiv 11013 \ (mod\ 11)$

2.1.1 Properties of Modulus

For congruence $modulo\ n$, the following properties are valid: (Keef and Guichard, 2020)

1) $a \equiv a\ for\ any\ a$ (eg: $3 \equiv 3$) <u>Reflexive Property</u>

2) $a \equiv b \Rightarrow b \equiv a$ <u>Commutative Property</u>

   $(17 \equiv 2\ (mod\ 5)\ and\ 22 \equiv 2\ (mod\ 5) \Rightarrow 17 \equiv 22$ (mod 5) and $22 \equiv 17$ (mod 5))

3) $a \equiv b\ and\ b \equiv c \Rightarrow a \equiv c$ <u>Transitive Property</u>

4) $n \mid a \Rightarrow a \equiv 0\ mod\ n$ $\because a = kn + 0$

5) $ka \equiv kb \bmod n$ and $gcd(k, n) = 1, \Rightarrow a \equiv b \bmod n$ Division Property

$72 \equiv 44 \ mod\ 7 \ and\ gcd\ (4, 7) = 1$
$\Rightarrow \frac{72}{4} \equiv \frac{44}{4} \ mod\ 7$
$\Rightarrow 18 \equiv 11 \ mod\ 7 \ \ (True)$

6) $a \equiv b$ and $c \equiv d \Rightarrow (a \pm c) \equiv (b \pm d)$ Modulo Addition and Subtraction

Proof:
$a \equiv b \bmod n \Rightarrow a - b = k_1 n$ $k_1 \in Z$ $[Eq\ 1]$
$c \equiv d \bmod n \Rightarrow c - d = k_2 n$ $k_2 \in Z$ $[Eq\ 2]$
$(a - b) + (c - d) = (n(k_1 + k_2))$ $[Eq\ 1] + [Eq\ 2]$
$\Rightarrow (a + c) - (b + d) = n(k_1 + k_2)$
$(a + c) \equiv (b + d) \bmod n$
Similarly: $(a - c) \equiv (b - d) \bmod n$ $[Eq\ 1] - [Eq\ 2]$
$\Rightarrow (a \pm c) \equiv (b \pm d) \bmod n$

$Q.E.D$ (Gorodnik, 2016)

7) $a \equiv b$ and $c \equiv d \Rightarrow ac \equiv bd$ Modulo Multiplication

Proof:
$a \equiv b \bmod n \Rightarrow a = b + k_1 n$ $k_1 \in Z$ $[Eq\ 1]$
$c \equiv d \bmod n \Rightarrow c = d + k_2 n$ $k_2 \in Z$ $[Eq\ 2]$
$ac = (b + k_1 n) \times (d + k_2 n)$ $[Eq\ 1] \times [Eq\ 2]$
$ac = k_1 k_2 n^2 + k_1 nd + k_2 nb + bd$
$ac = n(k_1 k_2 n + k_1 d + k_2 b) + bd$
$\Rightarrow ac \equiv bd \bmod n$

$Q.E.D$ (Gallier and Quaintance, 2020)

8) $a \equiv b \Rightarrow a^j \equiv b^j$ for any integer j ≥ 1 <u>Modulo Indices Property</u>

Proof:

$a \equiv b \Rightarrow n \mid (a - b)$ $\because a \equiv b \bmod n$

$a^j - b^j = (a - b)(a^{j-1} + a^{j-2}b + \ldots + ab^{j-2} + b^{j-1})$ $\because$ *algebraic property*

$\Rightarrow n \mid (a^j - b^j)$ $\because n \mid (a - b)$

$\Rightarrow a^j \equiv b^j \bmod n$

Q.E.D (Keef and Guichard, 2020)

9) $(a + b) \bmod n \equiv (a \bmod n) + (b \bmod n)$ <u>Modulo Separation Property</u>

Proof:

$Let\ k \equiv a \bmod n,\ l \equiv b \bmod n$

$\Rightarrow k = qn + a$ and $l = pn + b$ $k, l \in Z$

$\Rightarrow k + l = (qn + a) + (pn + b)$ $p, q \in Z$

$\Rightarrow k + l = n(p + q) + (a + b)$

$\Rightarrow (a \bmod n) + (b \bmod n) \equiv (a + b) \bmod n$ $\because n \mid (n(p + q))$

Q.E.D

<u>*Purpose:*</u>

Modulus arithmetic is very useful in number theory providing easier and simple ways to understand divisibility, primality and remainders, which are used extensively in RSA.

2.2 Euclidean Algorithm

Goal:

The Euclidean algorithm is effectively used in finding the greatest common divisor (gcd) of large numbers, for which the factorization techniques become cumbersome.

Proof of Euclid's Algorithm:

<u>Lemma</u>: To first prove $gcd(a,b) = gcd(b, a \bmod b)$ for $a > b$ $\quad a, b \in Z$

If $a > 0$, $a \mid b \Rightarrow gcd(a,b) = a$

If $a \equiv c \bmod b \Rightarrow a = bx + c \quad for\ x \in Z$

$$\Rightarrow c = a - bx \tag{2.2.1}$$

$If\ d \mid a\ and\ d \mid b \Rightarrow d \mid (a - bx) \Rightarrow d \mid c \quad for\ d \in Z \quad \because$ (2.2.1)

When $d = gcd(a,b)$

$\Rightarrow gcd(a,b) \mid c$

$Given\ a > b,\ and\ c\ is\ remainder\ when\ b\ divides\ a \Rightarrow a > c\ and\ b > c$

$Since\ gcd(a,b)\ also\ divides\ c,\ there\ cannot\ be\ any\ other\ larger\ factor\ than\ gcd(a,b)\ that\ can\ divide\ both\ b\ and\ c$

$\Rightarrow gcd(a,b) = gcd(b,c)$

$$\Rightarrow gcd(a,b) = gcd(b, a \bmod b) \tag{2.2.2}$$

$Q.E.D$

<u>Proof:</u> $We\ use\ this\ lemma\ to\ prove\ find\ gcd(a,b),\ when\ a > b\ and\ a,b \in Z$

$gcd(a,b) = gcd(b, a \bmod b) \quad \because$ (2.2.2)

$= gcd(b, r_1) \quad for\ r_1 = a \bmod b \Rightarrow a = q_1 b + r_1,\ for\ q_1, r_1 \in Z$

$= gcd(r_1, b \bmod r_1)$

$= gcd(r_1, r_2) \quad for\ r_2 = b \bmod r_1 \Rightarrow b = q_2 r_1 + r_2,\ for\ q_2, r_2 \in Z$

$= gcd(r_{2,} r_3) \quad for\ r_3 = r_1 \bmod r_2 \Rightarrow r_1 = q_3 r_2 + r_3$

$\ldots$

$$= (r_{i-1}, r_{i-2} \bmod r_{i-1}) \Rightarrow r_{i-2} = q_i r_{i-1} + r_i \tag{2.2.3}$$

$As\ r_0 > r_1 > r_2 > \ldots\ there\ exists\ some\ r_{n+1} = 0$

$\Rightarrow gcd(r_n, r_{n+1}) = gcd(r_n, 0) = r_n$

$\Rightarrow gcd(a,b) = r_n$

Hence through the recursive process of the Euclidean Algorithm, it is possible to find the $gcd(a,b)$

$Q.E.D$ (Keef and Guichard, 2020) (Paar, 2014)

Example of Recursive Application of Euclid's Algorithm:

<u>First Method:</u>

$gcd\,(14736,\ 5232) = (5232,\ 14736\ mod\ 5232) = (5232,\ 4272) = (4272,\ 5232\ mod\ 4272)$

$= (4272,\ 960) = (960,\ 4272\ mod\ 960) = (960,\ 432) = (432,\ 960\ mod\ 432) = (432,\ 96)$

$= (96,\ 432\ mod\ 96) = (96,\ 48) = (48,\ 96\ mod\ 48) = (48,\ 0) = 48$

$\Rightarrow gcd\,(14736,\ 5232) = 48$

<u>Second Method:</u>

$14736 = 5232 \times 2 + 4272$	$a = q_1 b + r_1$
$5232 = 1 \times 4272 + 960$	$b = q_2 r_1 + r_2$
$4272 = 4 \times 960 + 432$	$r_1 = q_3 r_2 + r_3$
$960 = 2 \times 432 + 96$	$r_2 = q_4 r_3 + r_4$
$432 = 4 \times 96 + 48$	$r_3 = q_5 r_4 + r_5$
$96 = 2 \times 48 + 0$	$r_4 = q_6 r_5 + 0$

gcd(14736, 5232) = 48

<u>Purpose:</u>

This algorithm provides a recursive mechanism to find the *gcd* of large numbers, which is well suited for computers. This leads to the proof of the Extended Euclidean Algorithm in Section 2.3.

2.3 Extended Euclidean Algorithm

Goal: $For\ any\ a, b \in Z,\ \exists\ s, t \in Z, such\ that\ gcd\ (a, b)\ = sa + tb$

The gcd of two numbers can be expressed as their linear combination.

(Paar, 2014)

Proof:

By the Euclidean Algorithm,

$$\begin{gathered} a = q_1 b + r_1 \\ b = q_2 r_1 + r_2 \\ r_1 = q_3 r_2 + r_3 \\ \ldots \\ r_{k-1} = q_{k+1} r_k + r_{k+1} \\ r_k = q_{k+2} r_{k+1} + r_{k+2} \\ \ldots \\ r_{n-2} = q_n r_{n-1} + r_n \\ r_{n-1} = q_{n+1} r_n + 0 \end{gathered}$$

Proving the Extended Euclidean Algorithm through mathematical induction:

Initial Step: $For\ k = 1$,
$a = q_1 b + r_1$
$r_1 = a - q_1 b$
$r_1 = a(1) + b(-q_1)$. Hence r_1 can be written as a linear combination of $a\ and\ b$.

Induction Hypothesis: Assume that the proposition is true for $r_{k-1}\ and\ r_k$
$\Rightarrow r_{k-1} = s_{k-1}a + t_{k-1}b$ and $r_k = s_k a + t_k b$ (2.3.1)

Induction Step: Prove that the proposition is true for $(k + 1)$:
$By\ Euclidean\ Algorithm$:
$r_{k-1} = q_{k+1} r_k + r_{k+1}$
$\Rightarrow r_{k+1} = r_{k-1} - q_{k+1} r_k$
Substituting the values of $r_{k-1}\ and\ r_k$ from (2.3.1),
$\Rightarrow r_{k+1} = (s_{k-1}a + t_{k-1}b) - q_{k+1}(s_k a + t_k b)$
$\Rightarrow r_{k+1} = a(s_{k-1} - q_{k+1}s_k) + b(t_{k-1} - t_k q_{k-1})$
$\Rightarrow r_{k+1} = sa + tb$
Hence it has been shown that if the proposition is true for $(k - 1)$ and k, then it is also true for $(k + 1)$. It follows the principle of mathematical induction.

From the Euclidean Algorithm,
$gcd(a, b) = r_n$
$\Rightarrow gcd\ (a, b) = sa + tb$

$Q.E.D$ (Integers and Algorithms, n.d.) (Paar, 2014)

For Example:

$gcd\ (14736,\ 5232) = 48$

$48 = 49(14736) - 138(5232)$ (linear combination of 14736 and 5232)

A full example of the calculation of the coefficients in the Extended Euclidean Algorithm, as used in RSA, is provided in Section 4.2.4.

2.4 Modular Inverses

The modular inverse of a number $'a'$ is a number $'b'$ such that $a \cdot b = 1\ (mod\ n)$. The Extended Euclidean Algorithm, explained above in Section 2.3, can be used to find out the value of the modular inverse of $'a'$ (Penn, 2019).

For a modular inverse of $'a'$ to exist modulo $'n'$, it is stated that $gcd(a, n) = 1$. In other words, a and n must be coprime. Why is this the case?

Proof:
$For\ a \in Z,\ \exists\ b\ such\ that\ a \cdot b \equiv 1\ mod\ n,\ if\ gcd\ (a, n) = 1$
$If\ a \cdot b \equiv 1\ mod\ n$
$\Rightarrow n \mid (ab - 1) \Rightarrow ab - 1 = kn \qquad k \in Z$
$\Rightarrow ab - nk = 1$
$gcd\ (a, n) \mid (ab - nk) \qquad \because (ab - nk)\ is\ a\ linear\ combination\ of\ a\ and\ n,\ by\ Section\ 2.3$
$\Rightarrow m\ gcd(a, n) = 1 \qquad m \in Z$
$gcd\ (a, n) \mid 1$
$\Rightarrow gcd\ (a, n) = 1 \qquad \because\ there\ is\ no\ other\ way\ gcd\ (a, n)\ divides\ 1$
$Hence\ when\ inverse\ exists,\ this\ means\ that\ gcd\ (a, n) = 1$

$Q.E.D$ (Gallier and Quaintance, 2020) (Penn, 2019)

When a, n are coprime, from Extended Euclidean Algorithm, we can write:

$as + nt = 1$, because $gcd\,(a, n) = 1$

This is known as **Bezout's Identity,** and holds true for all $gcd(a, n) = 1$. This property is very useful, especially in the following steps.

$$
\begin{aligned}
&\textit{Inverse of } a = a^{-1} \\
&\Rightarrow a \cdot a^{-1} \equiv 1 \; mod \; n \\
\\
&\textit{For inverse to exist}: \\
&gcd\,(a, n) = 1 \\
\\
&\textit{From Euclid's Extended Algorithm}: \\
&sn + ta = 1 \\
&ta \equiv 1 \; mod \; n \qquad\qquad\qquad\qquad \because n \mid sn \\
&\Rightarrow t = a^{-1} \\
&\Rightarrow \textit{In the linear expression } sn + ta = 1. \textit{ The coefficient of 'a' is the inverse of 'a'.}
\end{aligned}
$$

(Gallier and Quaintance, 2020) (Penn, 2019)

A full example of inverse calculation, as used in RSA, is provided in Section 4.2.4.

2.5 Chinese Remainder Theorem

2.5.1 Generalisation of Theorem for multiple modulo

Theorem:

Suppose $n_1,\ n_2,\ \ldots n_k\ \in Z^+$ *for which* $gcd\,(n_i, n_j) = 1,\ (i \neq j)$ *and* $b_1, b_2,\ \ldots b_k \in Z$.

Then the system of congruences :

$$\begin{gathered} x \equiv b_1 mod\ n_1 \\ x \equiv b_2 mod\ n_2 \\ \ldots \\ x \equiv b_k mod\ n_k \end{gathered}$$

has a unique solution modulo $n_1 \cdot n_2 \cdot n_3 \ldots \cdot n_k$

(Penn, 2019)

Proof: *Let* $N = n_1 \cdot n_2 \cdot n_3 \ldots \cdot n_k$ *and* $N_i = \frac{N}{n_i}$

Claim: $gcd\,(n_i, N_i) = 1$

Suppose $\exists d$ *such that* $d \mid n_i$ *and* $d \mid N_i$. Since all of the n_j are coprime with one another

$\Rightarrow d \mid n_j$ *for some* $j \neq i$. $\quad \because N_i = \frac{N}{n_i}$

$d \mid n_j$ *and* $d \mid n_i \Rightarrow d \mid gcd(n_i, n_j)$ $\quad \because$ *by definition of gcd*

But, $gcd\,(n_i, n_j) = 1$

$\Rightarrow d \mid 1 \Rightarrow d = 1$

$\Rightarrow gcd(n_i, N_i) = 1$

Q.E.D

Assume $\exists\ x_i$ *such that* x_i *is the inverse of* N_i *modulo* n_i :

As per Section 2.4, this inverse exists because $gcd\,(n_i, N_i) = 1$

$\Rightarrow x_i \cdot N_i \equiv 1\ mod\ n_i$ (2.5.1.1)

$\forall\ i \neq j,$ *as* N_j *is the product of all n except* n_j, *which means* $n_i \mid N_j$

$\Rightarrow\ x_j \cdot N_j \equiv 0\ mod\ n_i$ (2.5.1.2)

Consider: $x = x_1 N_1 b_1 + x_2 N_2 b_2 + ... + x_k N_k b_k$

$\equiv (0 + 0 + ... + x_i N_i b_i + 0 + 0) \; mod \; n_i$ $\qquad \because From \; (2.5.1.1) \; and \; (2.5.1.2)$

$\equiv x_i N_i b_i \; mod \; n_i$

$Since \; x_i \cdot N_i \equiv 1 \; mod \; n_i$

$\Rightarrow x_i N_i b_i \; mod \; n_i = b_i mod \; n_i$

$\Rightarrow x \equiv b_i \; mod \; n_i$

$\forall 1 \leq i \leq k, \; x \; satisfies \; the \; system \; of \; congruences \; stated \; in \; Theorem$

Existence of solution has been shown/proved

To prove the **uniqueness** of the solution:

Suppose that x and y are solutions:

$x \equiv b_i \; mod \; n_i$ and $y \equiv b_i \; mod \; n_i$

$x - y \equiv 0 \; mod \; n_i$

$n_i \mid (x - y) \qquad \forall 1 \leq i \leq k$

$As \; gcd(n_i, n_j) = 1, \; and \; n_i \mid (x - y) \; \forall 1 \leq i \leq k$

$\Rightarrow (n_1 \cdot n_2 \cdot n_3 ... \cdot n_k) \mid (x - y)$

$\Rightarrow N \mid (x - y)$

$\Rightarrow x \equiv y \; mod \; N$

$\Rightarrow x \; and \; y \; solutions \; are \; congruent. \; Hence \; uniqueness \; of \; solution \; is \; proved$

$Q.E.D$ (Ruohonen, 2014) (Penn, 2019)

While the above generic Chinese Remainder Theorem is not directly applied in RSA, a variant involving only two prime modulus is used, which is dealt with in the next subsection.

2.5.2 Chinese Remainder Theorem as used in RSA

$If\ a \equiv b\ mod\ m\ , a \equiv b\ mod\ n$
$Then\ a \equiv b\ mod\ (LCM(m,n))$

In other terms,
If $m \mid (a-b),\ n \mid (a-b)$ *then we should prove that* $LCM(m,n) \mid (a-b)$

$Take\ (a-b) = y,\ LCM(m,n) = l$

By definition :
$m \mid l\ , n \mid l$ $\qquad\qquad \because l = LCM(m,n)$

We will prove the above through proof by contradiction :
Assume that $\exists$ 'y' *such that,* $m \mid y,\ n \mid y,\ y > l\ ,\ l \nmid y$
$\Rightarrow y = ql + r$ $\qquad q \in Z,\ 0 < r < l$
$\Rightarrow r = y - ql$
$\Rightarrow m \mid r,\ n \mid r$
$\because m \mid y,\ n \mid y,$ *and* $m \mid l,\ n \mid l$
$r < l$ $\qquad \because r$ *is the remainder when* l *is the divisor*
$\Rightarrow \exists\ r < l,$ *such that,* $m \mid r$ *and* $n \mid r.$
But l *should be the least number such that* $m \mid l$ *and* $n \mid l$ $\qquad \because l$ *is* LCM
Hence this is a contradiction. By proof of contradiction:
$\Rightarrow \forall\ m \mid y, n \mid y \Rightarrow LCM(m,n) \mid y$
$\Rightarrow \forall\ m \mid (a-b), n \mid (a-b) \Rightarrow LCM(m,n) \mid (a-b)$ $\qquad \because y = (a-b)$
$\Rightarrow a \equiv b\ mod\ (LCM(m,n))$

Q.E.D (Proofwiki.org, 2020) (Clark, 1971)

Example:

To find a number, x, that results in 1 as the remainder, when divided by 2, 3, and 5.

$x \equiv 1\ mod\ 2\ , x \equiv 1\ mod\ 3\ ,\ x \equiv 1\ mod\ 5$

Has a unique solution $x \equiv 1\ mod\ (2 \times 3 \times 5) \Rightarrow x \equiv 1\ mod\ 30$

Any $x\ (1 \equiv 31 \equiv 61...)$ which satisfies this congruence, satisfies the above set of congruences.

2.6 Euler’s Totient Function

Euler’s Totient function, $\varphi(n)$, is the number of non-negative integers less than n which are coprime to n (Keef, Guichard, 2020).

For example, finding $\varphi(21)$:

Numbers coprime to 21, less than 21: 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20

$\Rightarrow \varphi(21) = 12$

Claim 1: If p is a prime, then $\varphi(p) = p - 1$.

Explanation: As p is prime there is no number less than p, that has a common factor with p, except for 1, hence there are $(p-1)$ numbers, less than p, that are coprime to p.

The multiplicative property of $\varphi(n)$ is used in RSA.

Claim 2: $\varphi(mn) = \varphi(m) \cdot \varphi(n)$ *when* $gcd\,(m,n) = 1$ **Multiplicative Property of** $\varphi(n)$

Proof of Multiplicative property of $\varphi(\,)$**:**

$$\begin{array}{ccccc}
1 & m+1 & 2m+1 & \ldots & (n-1)m+1 \\
2 & m+2 & 2m+2 & \ldots & (n-1)m+2 \\
3 & m+3 & 2m+3 & \ldots & (n-1)m+3 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
r & m+r & 2m+r & \ldots & (n-1)m+r \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\mathrm{m} & 2m & 3m & \ldots & mn
\end{array}$$

1. The overall mn elements $(1 \ to \ mn)$ can be represented in a table as above, in the form of m rows and n columns. Also given that $gcd\,(m,n) = 1,\ m,n \in Z^{+}$
$\Rightarrow$ *m and n are coprimes.*

2. As indicated in blue, the rth row has elements $r,\ (m+r),\ (2m+r)...((n-1)m+r)$. There are overall '$n$' numbers. It assumes the form of, $(km+r)$, where k ranges from $0\ to\ n-1$.
3. $Let\ d = gcd\ (r,m) \neq 1$.
4. If $d>1$, then in the rth row, $d \mid r\ and\ d \mid m \Rightarrow d \mid (km+r)\ \forall\ k$. No element in the rth row is coprime to mn
5. Out of m rows, the number of rows having numbers coprime to mn will be equal to the number of rows in which r is coprime to m. There are $\varphi(m)$ such rows because $\varphi(m)$ denotes the number of elements less than m, coprime to m.
6. In each such row, there are n numbers out of which only $\varphi(n)$ numbers are coprime to n.
7. From Steps 5 and 6, because m and n are coprime, we can deduce that there are exactly $\varphi(m)\varphi(n)$ numbers that are coprime to mn.

$$\Rightarrow \varphi(mn) = \varphi(m)\varphi(n)$$

$Q.E.D$ (Euler's Phi Function, n.d.)

Taking an example of $\varphi(36)$:

$By\ Multiplicative\ Property,$

$\varphi(36) = \varphi(9) \cdot \varphi(4)$

$\varphi(9) = 6 \qquad \because \{1,2,4,5,7,8\}$

$\varphi(4) = 2 \qquad \because \{1,3\}$

$\Rightarrow \varphi(36) = 6 \times 2 = 12$

$Numbers\ coprime\ to\ 36,\ less\ than\ 36 : 1,5,7,11,13,17,19,23,25,29,31,35 \Rightarrow \varphi(36) = 12$

2.7 Fermat's Little Theorem

Theorem:

$When\ p\ is\ prime\ and\ gcd\,(a, p) = 1\,,$

$$a^{p} \equiv a\ (mod\ p)$$

(Conrad, n.d.)

Proof by Induction:

Initial Step:

$When\ a = 1,$

$\Rightarrow 1^{p} \equiv 1\ mod\ p$

$\Rightarrow 1 \equiv 1\ mod\ p$ $\qquad$ $The\ proposition\ is\ true\ for\ a = 1$

Induction Hypothesis:

$Assume\ proposition\ is\ true\ for\ some\ k,\ k \in Z\,,\ k \neq 1$

$\Rightarrow k^{p} \equiv k\ mod\ p$ $\qquad$ $gcd\,(k, p) = 1,\ k < p,\ k \nmid p$ $\qquad$ $\because\ p\ is\ prime$

Induction Step:

$For\ (k+1),$ by binomial theorem,

$$(k+1)^{p} = k^{p} + \binom{p}{1} k^{p-1} + \binom{p}{2} k^{p-2} + \binom{p}{3} k^{p-3} + \ldots + \binom{p}{p-1} k + 1$$

$$= (k^{p} + 1) + \left(\frac{p!k^{p-1}}{(p-1)!}\right) + \left(\frac{p!k^{p-2}}{(p-2)!2!}\right) + \left(\frac{p!k^{p-3}}{(p-3)!3!}\right) + \ldots + \left(\frac{p!}{(p-1)!}\right)$$

$$= (k^{p} + 1) + p\left(\left(\frac{(p-1)!k^{p-1}}{(p-1)!}\right) + \left(\frac{(p-1)!k^{p-2}}{(p-2)!2!}\right) + \left(\frac{(p-1)!k^{p-3}}{(p-3)!3!}\right) + \ldots + \left(\frac{(p-1)!}{(p-1)!}\right)\right)$$

$(k+1)^{p} \equiv (k^{p} + 1)\ mod\ p$ $\qquad$ $\because p\ divides\ the\ large\ combinatorial\ function\ above$

$(k+1)^{p} \equiv k^{p}\ mod\ p + 1\ mod\ p$ $\qquad$ $\because Modulo\ Separation\ Property,\ Section\ 2.1.1$

$(k+1)^{p} \equiv k\ mod\ p + 1\ mod\ p$ $\qquad$ $\because The\ induction\ hypothesis\ states : k^{p} \equiv k\ mod\ p$

$\Rightarrow (k+1)^{p} \equiv (k+1) mod\ p$ $\qquad$ $\because Modulo\ Separation\ Property,\ Section\ 2.1.1$

Since the proposition is true for $k = 1$, and if it is true for any $k \in Z$, then it is also holds true for $(k+1)$. Hence it follows by the principle of mathematical induction.

$Q.E.D$ (Weisstein, 2020) (Conrad, n.d.)

Further Implications:

If p is prime for any a, $gcd(a,p) = 1$ then $a^p \equiv a \; mod\,(p)$

Dividing both sides by a

$$\frac{a^p}{a} \equiv \frac{a}{a} \; mod\,(p) \qquad \because Modulo\ Division\ Property,\ Section\ 2.1.1$$

$$a^{p-1} \equiv 1 \; mod\,(p)$$

<u>Analysis: Is the converse true?</u>

$Converse : If\ a^{p-1} \equiv 1\ mod\,(p)\ and\ gcd\,(a,p) = 1 \Rightarrow p\ is\ prime$

If the converse is true, Fermat's Little Theorem can be used as a primality test.

Example: Take 3 prime numbers, $x = 3,\ y = 11,\ z = 17$, $p = xyz = 3 \times 11 \times 17 = 561$

By Fermat's Little Theorem for any $a \in Z$,

$a^2 \equiv 1\ mod\ 3$

$a^{10} \equiv 1\ mod\ 11$

$a^{16} \equiv 1\ mod\ 17$

Making all of the exponents equal to 560, (By Modulo Indices Property 2.1.1)

$(a^2)^{280} \equiv (1)^{280}\ mod\ 3$ $\Rightarrow a^{560} \equiv 1\ mod\ 3$	$(a^{10})^{56} \equiv (1)^{56}\ mod\ 11$ $\Rightarrow a^{560} \equiv 1\ mod\ 11$	$(a^{16})^{35} \equiv (1)^{35}\ mod\ 17$ $\Rightarrow a^{560} \equiv 1\ mod\ 17$

By Chinese Remainder Theorem,

$\Rightarrow a^{560} \equiv 1\ (mod\ (3 \times 11 \times 17))$ $\qquad LCM(3, 11, 17) = 3 \times 11 \times 17$

$\Rightarrow a^{560} \equiv 1\ mod\ 561$

$\Rightarrow a^{p-1} \equiv 1\ mod\ p$ for all $a \in Z$.

561 is not a prime number, but it follows Fermat's Little Theorem; this implies that the **converse of Fermat's Little Theorem is not true.**

These types of composite numbers, which satisfy $a^{p-1} \equiv 1\ mod\ (p)$ for all a , $gcd(a, p) = 1$, are called Carmichael numbers (Calderbank, 2007) (Lynn, n.d.).

3.0 Mathematical Analysis of the RSA Cryptosystem

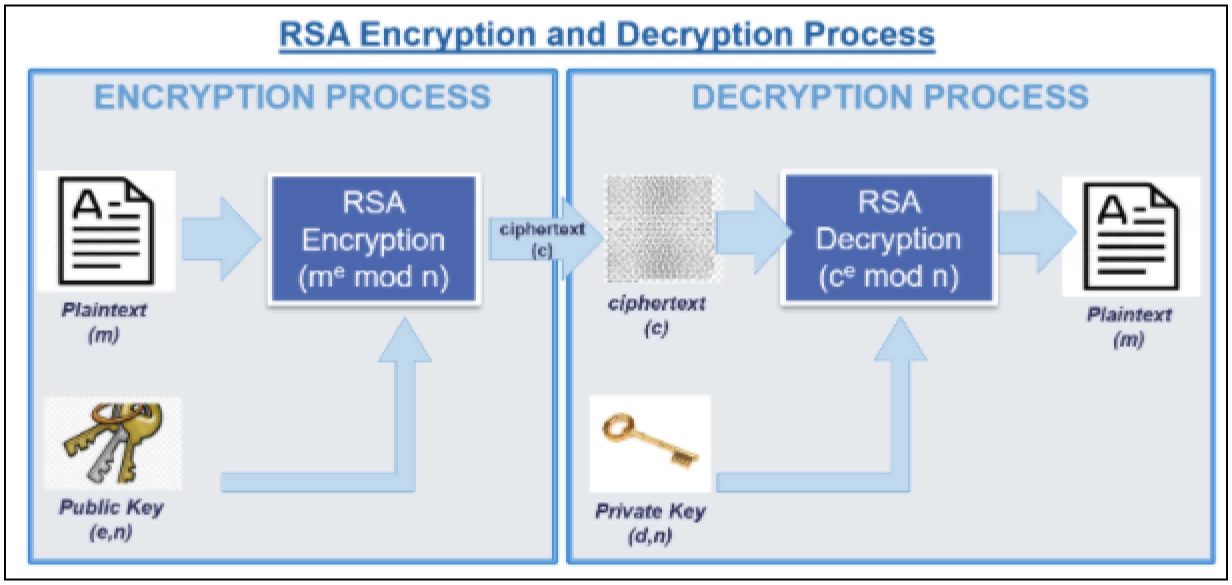
3.1 RSA Cipher

The RSA cipher is used in order to securely communicate a message between two parties without it being understood by an intruder. Such secure instances can be visualised in scenarios such as mission critical covert military operations and sharing sensitive personal information (credit card details) on the internet (Menezes, Van Oorschot and Vanstone, 1997). If a sender, Sam, wants to securely communicate a message *(plaintext, m)* to a receiver, Ryan, Sam will encrypt the *plaintext* using a *'key'* and create an encrypted secret message *(ciphertext, c)*. The encryption key used by Sam is known to all, hence it is referred to as the *public key.* The receiver, Ryan, uses a special key *(private key*) known only to him and Sam, to decrypt the *ciphertext* to obtain the original message (*plaintext*) (Gallier and Quaintance, 2020). This cryptosystem should enable the following characteristics: (Kelly, 2009)

1) It should be easy for Sam to encrypt the message with the *public key*.
2) It should be easy for Ryan to decrypt the message, as long as he has the *private key*.
3) It should be near-impossible for any third party intruder to decrypt the *ciphertext* without the *private key.*

This is known as asymmetric cryptography, as the encryption and decryption keys used are different (Kowalczyk, 2020). The RSA process is pictorially depicted in Figure 1:

Figure 1: RSA Encryption and Decryption Process



The core of the RSA cryptosystem is mathematics-oriented, especially the *public key*, *private key*, *encryption* and *decryption processes*, as further explained in below.

3.2 Mathematical Representation of the RSA Algorithm

Encryption Process:

1) Choose two large prime numbers p, q. Find $n = pq$
2) Find the value of Euler's Totient, $\varphi(n)$
3) Choose a *public key exponent,* e, such that $3 \le e < \varphi(n)$ and $gcd\,(e, \varphi(n)) = 1$
4) Encrypt the plaintext using $E(\,)$, the *public key* (e, n) and obtain *ciphertext*:

   $c = E(m) \ = m^{e} mod\ n$

Decryption Process:

1) Find the *private key exponent,* d, such that $d{\cdot}e \ = \ 1 \ \ (mod\ \varphi(n))$
2) Decrypt the ciphertext using $D(\,)$, the *private key* (d, n) and obtain *plaintext*:

$$m = c^{d} mod\ n$$

(Gallier and Quaintance, 2020) (Kowalcyzk, 2020) (Kaliski, n.d.) (Goshwe, 2013)

The steps mentioned above are mathematically analysed in detail in the subsequent sections.

3.3 Key Generation Process

3.3.1 Prime number selection

Prime numbers play a significant role in enabling security in the RSA cryptosystem (Kaliski, n.d.).

Concept 1: Prime number Factorisation

For two large prime numbers, $p\ and\ q$, it is easy to find $n = pq$, but given n it is hard to recover the prime factors p and q (Kaliski, n.d.). The least complex way to find the prime factors of n, would be to test each of the possible factors one at a time, which would take $\sqrt{n}$ steps (Gallier and Quaintance, 2020). The current methods used are more efficient, but they still are expensive and take a considerable amount of time, "It has been estimated recently that recovering the prime factors of a 1024-bit number would take a year on a machine costing US $ 10 million" (Kaliski, n.d.). This makes it difficult for some intruder to break the RSA cipher.

Concept 2: Sufficiency of Fermat's Primality Test

To find p, q where $n = pq$, it is important to first understand how to test if a number is prime.

By Fermat's Little Theorem, if p is a prime and $gcd(a, p) = 1 \Rightarrow a^{p-1} \equiv 1\ mod\ (p)$.

Converse of this implication is not true:

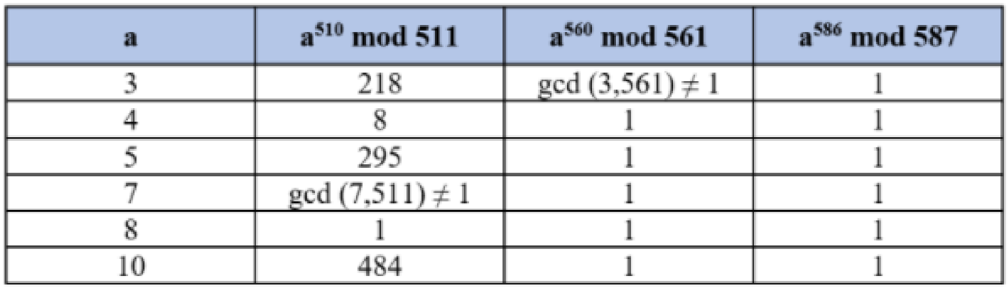
If $a^{p-1} \equiv 1\ mod\ (p)\ and\ gcd(a, p) = 1\ does\ not\ imply\ p\ is\ a\ prime$ (Refer to Section 2.7)

However, the negation of the converse is true:

If $a^{p-1} \not\equiv 1 \ mod\ (p)\ and\ gcd(a,p) = 1 \ \Rightarrow p \ is\ not\ a\ prime$

Testing primality of 511, 561, 587:

Table 1: Primality test of 511, 561, 587



a	a^{510} mod 511	a^{560} mod 561	a^{586} mod 587
3	218	gcd (3,561) ≠ 1	1
4	8	1	1
5	295	1	1
7	gcd (7,511) ≠ 1	1	1
8	1	1	1
10	484	1	1

For 511, there is one instance, $a = 8$, which follows $a^{p-1} \equiv 1 \ mod\ (p)$, however through multiple iterations with different a values, it is clear that $a^{p-1} \equiv 1 \ mod\ p$ is not true for any a which is coprime to 511. For 561, it would appear as though it is prime, as it follows the identity, however it is a Carmichael number as described in Section 2.7. 587 is prime. These values in the table were calculated using a PowerMod calculator online (PowerMod Calculator, n.d.).

Hence Fermat's Little Theorem is a conclusive test for non-primality but not for primality.

<u>Concept 3: Other Alternatives</u>

Since there are several limitations to Fermat's Primality Test, there are other alternative tests used such as the Rabin Miller Primality Test. These tests are probabilistic in nature, and hence are more accurate than Fermat's. However these primality tests are not discussed in this exploration (Rabin Miller Primality Test, n.d.).

<u>Concept 4: How this contributes to the safety and security of RSA</u>

When RSA is realistically used, the size of the prime numbers p and q are 1024-bits, which is around 300 decimal digits long (Collins, 2018). This means that the product n is around 600 digits long. It is difficult to factorise the product of two prime numbers, and at this large size it is nearly impossible to do so. This is the main reason behind the safety of RSA as p and q are used to find the *public key* and *private key* in the later steps. If p and q are found out, then it is very easy to break the RSA cipher as all of the values computed are based around p and q (Kaliski, n.d).

3.3.2 Euler's Totient Function

<u>Concept 5 : Value of</u> $\varphi(n)$

From Section 2.6, by multiplicative property, $\varphi(mn) = \varphi(m) \cdot \varphi(n)$

For $n = pq$,

$\Rightarrow \varphi(n) = \varphi(pq) = \varphi(p) \cdot \varphi(q)$ $\quad\because (p, q) = 1$

$\Rightarrow \varphi(n) = (p - 1)(q - 1)$ $\quad\because$ *p and q are prime* (Section 2.6)

<u>Concept 6 : Why use</u> $\varphi(n)$ *<u>instead of n?</u>*

$\varphi(n)$ is a value which can only be found when p and q are known, so for a specific pair of primes, p and q, $\varphi(n)$ is <u>'unique'</u>. Because n can only be factorised into 2 unique prime factors (Integers and Algorithms, n.d.). The security of RSA is dependent on the difficulty of factorising a large n into two unique prime factors. Hence instead of using n, usage of $\varphi(n)$ increases the security (Kaliski, n.d.) (Gallier and Quaintance, 2020).

Concept 7 : $\varphi(n)$ *is an even number*

- $\varphi(n)$ is an even number, since $(p-1)$ and $(q-1)$ are even numbers as p and q are odd primes.($p, q \neq 2$, since p and q are large primes)

3.3.3 Determining of Public and Private Exponents *(e,d)*

Concept 8: Boundary Condition Analysis

The public encryption exponent , e, is chosen such that $gcd(e, \varphi(n)) = 1$ and $3 \le e < \varphi(n)$

- $e \ge 3$ because taking 1 is trivial, and 2 is not coprime to $\varphi(n)$ as $\varphi(n)$ is even. (Refer *Concept 7*)
- $e \neq \varphi(n)$ as $gcd\,(e, \varphi(n)) = \varphi(n)$ *if* $e = \varphi(n)$. But $gcd\,(e, \varphi(n))$ *must equal* 1.
- $e < \varphi(n)$ as $mod\,(\varphi(n))$ is used in finding d, the *private key exponent.*

Concept 9: How to find the inverse?

The encryption and decryption exponents, e, d, share the property: $d{\cdot}e = 1 \ (mod\,\varphi(n))$. This means that d is the inverse of e modulo $\varphi(n)$. For an inverse of e to exist, $\gcd(e, \varphi(n)) = 1$ as proved in Section 2.4. The inverse of e can be found through the use of both the Euclidean and Extended Euclidean Algorithm. An example inverse calculation is provided later in Section 4.2.4.

Concept 10: For a chosen ‘e’, the modular inverse ‘d’ is unique such that $d \cdot e \equiv 1 \bmod \varphi(n)$

Assume that there is another integer c, less than $\varphi(n)$, which satisfies the above property

$\Rightarrow ed \equiv 1 \bmod \varphi(n)$ *and* $ec \equiv 1 \bmod \varphi(n)$

$\Rightarrow \varphi(n) \mid (ed-1)$ *and* $\varphi(n) \mid (ec-1)$

$\Rightarrow \varphi(n) \mid (ed-1)-(ec-1) \qquad \because a \mid b,\ a \mid c \Rightarrow a \mid (b-c)$

$\Rightarrow \varphi(n) \mid (ed-ec) \Rightarrow \varphi(n) \mid e(d-c)$

As gcd $(\varphi(n), e) = 1$ *by the conditions for inverse,*

$\Rightarrow \varphi(n) \mid (d-c)$

$\Rightarrow d \equiv c \bmod (\varphi(n))$

Hence d and c are congruent modulo $\varphi(n)$. As $c, d < \varphi(n)$, c and d must be equal.

Q.E.D

3.4 Encryption and Decryption

The encryption algorithm and decryption algorithm are both functions of *modulo n* which use the *public* and *private key exponents*.

The encryption process: $E(m) = m^e \bmod n = c$

The decryption process: $D(c) = c^d \bmod n = m$

To show why the encryption and decryption algorithms work, we first need to prove/show that the sequence of encryption and decryption processes leads to the same *plaintext* m.

Concept 11: The sequence of encryption and decryption result in the original plaintext, m.

In other words to prove that the process of $D(E(m)) = m$.

$E(m) = c = m^e \bmod n$

$D(c) = D(m^e \bmod n) = (m^e)^d \bmod n$

$D(c) = m^{de} \bmod n \qquad \because (a^m)^n = a^{mn} \qquad (3.4.1)$

$d \cdot e \equiv 1 \; mod \; \varphi(n)$ *for an inverse to exist*

$\Rightarrow k \, \varphi(n) + 1 = de$ $\quad k \in Z$ (3.4.2)

From (3.4.1), and (3.4.2) above,

$D(c) = m^{de} mod \; n = m^{k \, \varphi(n) + 1} mod \; n$ (3.4.3)

From Fermat's Little Theorem,

$m^{p-1} \equiv 1 \; mod \; p$ $\quad \because p \; is \; prime$

$\Rightarrow ((m^{p-1})^{q-1}) \equiv 1^{q-1} mod \; p$ $\quad \because Modulo \; Indices \; Property$ 2.1.1

$\Rightarrow m^{\varphi(n)} \equiv 1 \; mod \; p$ $\quad \because \varphi(n) = (p-1)(q-1)$

Exponentiating both sides by k :

$\Rightarrow m^{k\varphi(n)} \equiv 1 \; mod \; p$ $\quad \because Modulo \; Indices \; Property$ 2.1.1

Multiplying both sides by m :

$\Rightarrow m^{k\varphi(n)+1} \equiv m \; mod \; p$ $\quad \because Modulo \; multiplication \; property$ 2.1.1 (3.4.4)

Starting from $m^{q-1} \equiv 1 \; mod \; q$, *and following the above steps, the same can be derived for q* :

$m^{k\varphi(n)+1} \equiv m \; mod \; q$ (3.4.5)

From 3.4.4, 3.4.5 above, applying Chinese Remainder Theorem for two factors (Section 2.5)

$\Rightarrow m^{k\varphi(n)+1} \equiv m \; mod \; pq$ $\quad \because p \; and \; q \; are \; co-primes$

$\Rightarrow m^{de} = m \; mod \; n$ $\quad \because From$ 3.4.2, $d \cdot e \equiv 1 \; mod \; \varphi(n)$

$\Rightarrow D(E(m)) = m$

Q.E.D

(Menezes, Van Oorschot and Vanstone, 1997) (Meelu, Malik, Patel and Singh, 2010)

4.0 An illustration of RSA application

In this section, an example message is encrypted and decrypted to showcase all the mathematical concepts and steps explained in the previous sections.

Scenario: In a mission critical covert military operation, an instruction is being relayed securely to the officers on ground to proceed with the attack from the southern direction at 6 o'clock. The *plaintext* intended to be relayed is : **'ATTACK FROM SOUTH AT SIX'**.

4.1 Conversion to Numerical format

Firstly in order to encrypt and decrypt this *plaintext*, we must first convert the message into a numerical format. An easy method of doing so would be through the use of the ASCII codes for the alphabet. Table 2 below lists the conversion between the alphabet and numbers. Hence the *plaintext* can be written as:

65,84,84,65,67,75,32,70,82,79,77,32,83,79,85,84,72,32,65,84,32,83,73,88

Table 2: ASCII values

Alphabet	ASCII value
space	32
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

In real life applications, where the RSA cipher is performed using supercomputers, the whole text is converted into one large number. The above text is represented by a 48-digit number (658484656775327082797732837985847232658432837388). However in this application, for ease of calculation, the 48-digit number is split into 24 blocks, each 2-digits long, each block representing one character.

4.2 Key Generation

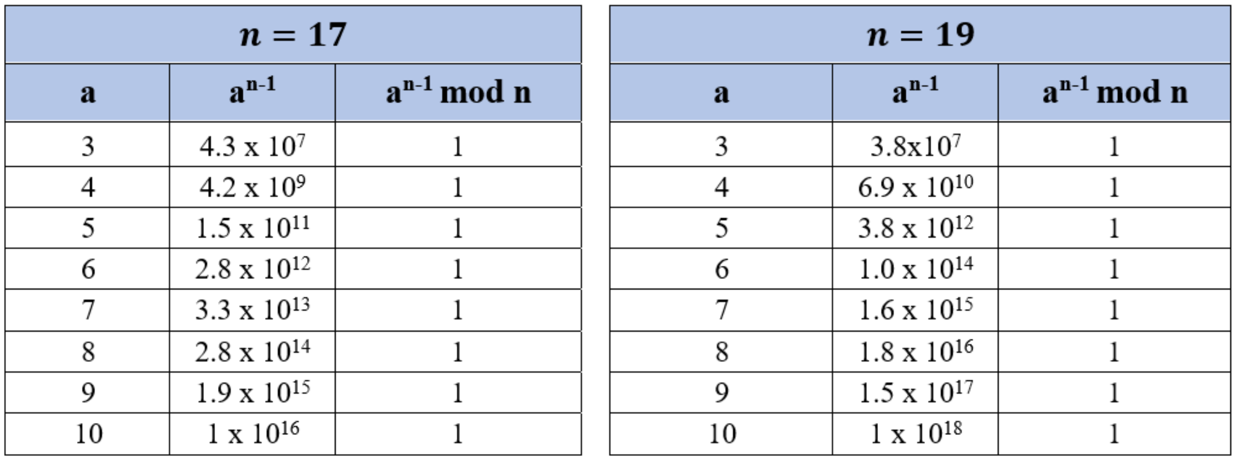
We need to generate the keys, by following the RSA steps:

4.2.1 Choose two prime numbers p, q. Find $n = pq$

In real-life applications of RSA the prime numbers are around 300-digits long (Collins, 2018). Computations involving such large numbers require supercomputing capabilities. For ease of demonstration, two smaller prime numbers, each 2-digits long, are chosen.

Two random numbers $p = 17$ and $q = 19$ are chosen. Though it is known that 17 and 19 are prime numbers, these numbers are subjected to the Fermat Primality Test as shown in Table 3.

Table 3: Test for primality for n=17 and n=19



$n = 17$		
a	a^{n-1}	a^{n-1} mod n
3	4.3×10^{7}	1
4	4.2×10^{9}	1
5	1.5×10^{11}	1
6	2.8×10^{12}	1
7	3.3×10^{13}	1
8	2.8×10^{14}	1
9	1.9×10^{15}	1
10	1×10^{16}	1

$n = 19$		
a	a^{n-1}	a^{n-1} mod n
3	3.8×10^{7}	1
4	6.9×10^{10}	1
5	3.8×10^{12}	1
6	1.0×10^{14}	1
7	1.6×10^{15}	1
8	1.8×10^{16}	1
9	1.5×10^{17}	1
10	1×10^{18}	1

Since 17 and 19 both satisfy the identity, $a^{p-1} \equiv 1 \; mod \; (p)$, for a variety of a values, it is highly likely that they are primes. However as explained in Concept 2, Section 3.3.1, it is not a necessary and sufficient condition for primality.

We then find:

$n = pq = 17 \cdot 19 = 323$

4.2.2 Find $\varphi(n)$.

$\varphi(n) = (p-1)(q-1)$ from Section 2.6

$\varphi(n) = 16 \times 18 = 288$

4.2.3 Choose the public key exponent

The value of 47 is chosen for e. $\because gcd(47, 288) = 1 \; and \; 3 \le 47 < 288$

The key that is publicised and known to the public as the ***public key (47,323)***, and is used in the next step to calculate the *private key* d.

4.2.4 Compute the private key exponent

Find d such that $d \cdot e \equiv 1 \; (mod \; \varphi(n))$.

$$47d \equiv 1 \; (mod \; 288)$$

$$d \equiv 47^{-1} (mod \; 288)$$

Using Euclidean Algorithm, from Section 2.2, on $gcd(288, 47)$

$288 = 6 \times 47 + 6$ $288 \; mod \; 47 \equiv 6$ (4.2.1)

$47 = 7 \times 6 + 5$ $47 \; mod \; 6 \equiv 5$ (4.2.2)

$6 = 1 \times 5 + 1$ $6 \; mod \; 5 \equiv 1$ (4.2.3)

From (4.2.3):

$1 = 6 - 5$ (4.2.4)

From (4.2.2):

$5 = 47 - (6 \times 7)$

Substituting this in (4.2.4):

$1 = 6 - (47 - (6 \times 7))$

$1 = 6 \times 8 - 47 \qquad (4.2.5)$

From (4.2.1):

$6 = 288 - (47 \times 6)$

Substituting this into (4.2.5):

$1 = (288 - (47 \times 6)) \times 8 - 47$

$1 = (288 \times 8) - (47 \times 49)$

$1\ (mod\ 288) \equiv ((288 \times 8) - (49 \times 47))\ (mod\ 288)$

$1\ (mod\ 288) \equiv 47 \times - 49\ (mod\ 288)$

$1\ (mod\ 288) \equiv -49\ e\ (mod\ 288) \qquad \because e = 47$

$1\ (mod\ 288) \equiv (288 - 49)\ e\ (mod\ 288)$

$1 \equiv 239\ e\ (mod\ 288)$

$\Rightarrow d = e^{-1} = 239$

From Euclid's Algorithm and Euclid's Extended Algorithm the value of $d = 239$ has been found.

Public Encryption Key: (47, 323)

Private Encryption Key: (239, 323)

4.3 Encryption

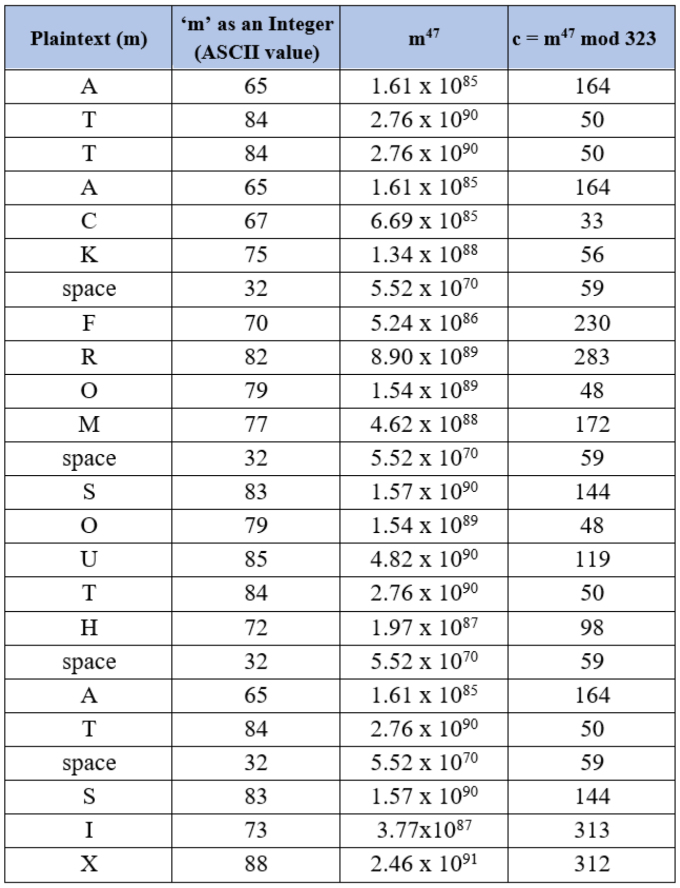
By applying the RSA encryption algorithm on each one letter block of the plain text with the *public key exponent*, the *ciphertext* can be obtained.

For example: applying the encryption process on the first letter A (ASCII value = 65),

$$E\ (m) = m^{e}\ mod\ n$$
$$E\ (65) = 65^{47}\ mod\ 323 = c$$
$$\Rightarrow c = 164$$

The following Table 4 captures all the *cipher text* obtained through the encryption of each letter. As the exponents are large, a specific power mod calculator was used to compute these values (PowerMod Calculator, n.d.).

Table 4: Encryption resulting in ciphertext



Plaintext (m)	'm' as an Integer (ASCII value)	m^{47}	c = m^{47} mod 323
A	65	1.61×10^{85}	164
T	84	2.76×10^{90}	50
T	84	2.76×10^{90}	50
A	65	1.61×10^{85}	164
C	67	6.69×10^{85}	33
K	75	1.34×10^{88}	56
space	32	5.52×10^{70}	59
F	70	5.24×10^{86}	230
R	82	8.90×10^{89}	283
O	79	1.54×10^{89}	48
M	77	4.62×10^{88}	172
space	32	5.52×10^{70}	59
S	83	1.57×10^{90}	144
O	79	1.54×10^{89}	48
U	85	4.82×10^{90}	119
T	84	2.76×10^{90}	50
H	72	1.97×10^{87}	98
space	32	5.52×10^{70}	59
A	65	1.61×10^{85}	164
T	84	2.76×10^{90}	50
space	32	5.52×10^{70}	59
S	83	1.57×10^{90}	144
I	73	$3.77\text{x}10^{87}$	313
X	88	2.46×10^{91}	312

The *ciphertext* sent to the officers would then be,

c = 164,050,050,164,033,056,059,230,283,048,172,059,144,048,119,050,098,059,164,050,059,144,313,312

Leading zeroes are prefixed to each of the 2-digit numbers, so that each character is represented in a uniform three digit code as n (323) is a 3-digit number, hence $(mod\ n)$ values can yield only 3-digit remainders. As mentioned above, all the encrypted characters can be concatenated into one long ciphered text as below.

164050050164033056059230283048172059144048119050098059164050059144313312

4.4 Decryption

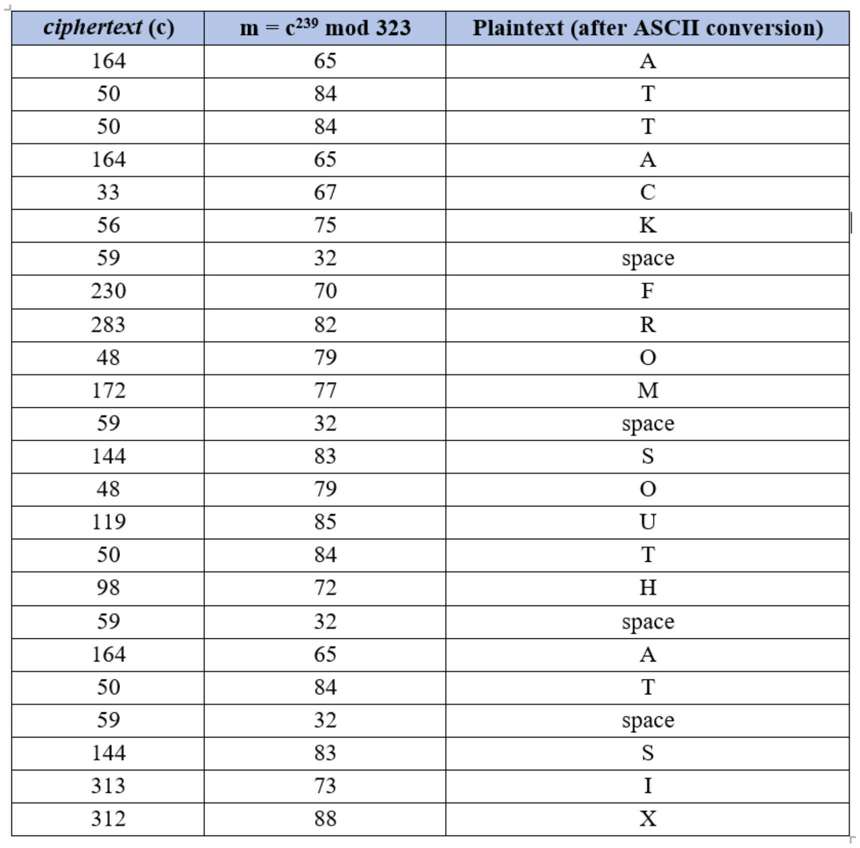
On the receiving end, the *ciphertext*, 164050050164033056059230283048172059144048119050098059164050059144313312, is split into blocks of three digits, and each block is decrypted individually to obtain the original *plaintext*. In real life scenarios, there is no need to break this large number into individual blocks.

Applying the decryption process to the first block,

$$D(c) = c^{d} \, mod \, n = m$$
$$D(164) = 164^{239} \, mod \, 323 = m$$
$$m = 65$$

The results of the decryption process are captured in the below Table 5. As the exponents are large, a specific power mod calculator was used to compute these values (PowerMod Calculator, n.d.).

Table 5: Decryption process resulting in plaintext



ciphertext (c)	m = c^{239} mod 323	Plaintext (after ASCII conversion)
164	65	A
50	84	T
50	84	T
164	65	A
33	67	C
56	75	K
59	32	space
230	70	F
283	82	R
48	79	O
172	77	M
59	32	space
144	83	S
48	79	O
119	85	U
50	84	T
98	72	H
59	32	space
164	65	A
50	84	T
59	32	space
144	83	S
313	73	I
312	88	X

Hence the officers will receive the message “ATTACK FROM SOUTH AT SIX”, after decryption. Since the *plaintext,* m, is the same as the decrypted value of c, this means that the RSA process went successfully. Even though the numbers used in this illustrated application are much smaller for the sake of convenience, compared to RSA algorithms in real life, this is still a realistic depiction of the mathematics involved in the process.

5.0 Evaluation

The best way to evaluate the strength of security behind the RSA cipher is to look at it from the attacker's perspective: how easy or difficult it is for an attacker or intruder to break the 'lock' provided by the public key asymmetric encryption? (Ruohonen, 2014) The following subsections evaluate these implications from a mathematical standpoint.

5.1 Prime numbers and security

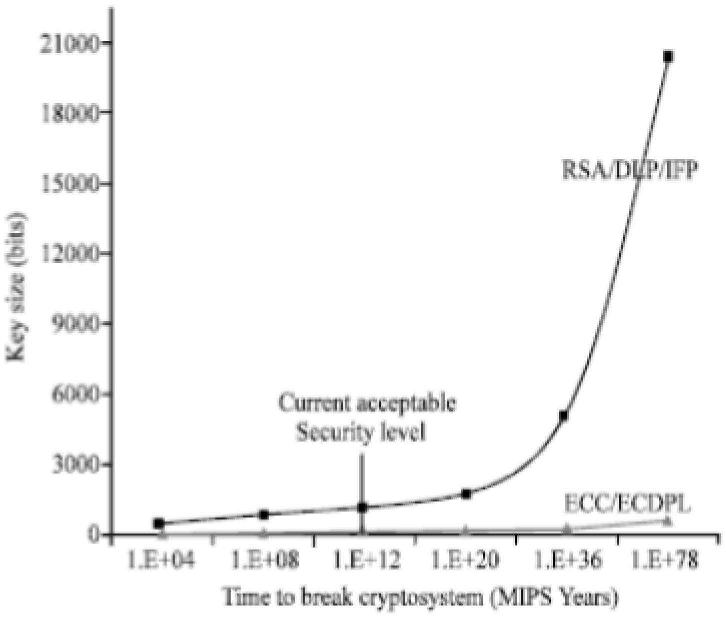
The RSA cipher is in essence based around the two prime factors p and q of n and these factors are used to compute the keys. If these factors are found then it can easily lead to the *ciphertext* being broken. The main way in which RSA ensures that an attacker cannot find p and q is through having an extremely large n. In practical use, the n is at least a 2048-bit number, meaning that n has over 600 decimal digits. This means that p and q are both approximately 300 digits long (Collins, 2018). The difficulty of trying to find out the only two prime factors of a 600+ digit long number is extremely hard making the task almost impossible. This is mainly what makes RSA a cryptosystem that is very hard to attack or break (Kaliski, n.d.).

It is important to note that even though these large numbers provide extra safety they also have several drawbacks. It takes a lot of computing power to find out the modulus of such a large number raised to the power of a *public key* or *private key*. In comparison to other advanced ciphers such as Elliptical Curve Cryptography (Menezes, Van Oorschot and Vanstone, 1997), RSA is a lot slower due to the size of the numbers used in the algorithms. This causes the algorithm to be slow. If one wanted to speed up the algorithm by decreasing the size of n, it would be relatively easier for attackers to find out the factors leading to the whole cipher being rendered useless. This is the compromise one needs to make (Nisha and Farik, 2017).

From the perspective of an attacker there are a few ways in which one can break the RSA cipher. Firstly, finding p and q which in itself is a daunting task for large bit numbers like n. For example it took almost 2 years for 13 researchers with hundred parallel computers to factorise a 768 bit number (Nisha and Farik, 2017). But if this number was a 2048 bit number instead, which is common in today's security algorithms, then the amount of time that this would take would be exponentially larger.

In Figure 2 to the right, the graph of key size (in terms of number of bits) and time taken to break the cryptosystem are graphed. The time is measured in terms of the years taken for a computer running at the capacity of a million instructions per second (MIPS Years). It is an exponential function, which means that the amount of time taken increases at an increasing rate as the key size increases. There are also some other advanced algorithms which use some smaller key sizes, but require higher time to break.

Figure 2: Key size vs. Time taken to break cryptosystem



Rabah, K., 2006. Review of Methods for Integer Factorization Applied to Cryptography. *Journal of Applied Sciences*, 6(2), pp.458-481.

5.2 Further mathematical adaptations to address the limitations of RSA

The main limitation of RSA is the time taken for encryption and decryption key generation, as the numbers which are used are extremely large. While this is desirable in providing security from the attackers, it requires higher computational and monetary resources to generate these secure keys. This is one of the major weaknesses of RSA and it can be diminished to a certain degree through the use of the Chinese Remainder Theorem. The Chinese Remainder Theorem (Section 2.5) states that a unique solution of $mod\ pq$ can be written in terms of $mod\ p$ and $mod\ q$. This property can be used on the encryption and decryption process, to relatively reduce the time taken. The following is an example inspired by Suzuki (Suzuki, 2019):

Ryan creates a RSA system, with $n = 21829\ and\ e = 37$
$21829 = 83 \times 263$
$By\ using\ d \cdot e \equiv 1\ mod\ n,\ d = 11613$

In order to decrypt a message, c, sent to him by Sam,
$m \equiv c^{11613} mod\ 21829$
However this takes a great amount of computing power

Since Ryan knows the factorisation of n, he can use the Chinese Remainder Theorem to improve the efficiency of the computing power, by reducing the computations required and decreasing the size of the modulo.
The Chinese Remainder Theorem states:
Suppose p, q are relatively prime,
$Then\ x \equiv a\ mod\ pq,\ if\ x \equiv a\ mod\ p,\ and\ x \equiv a\ mod\ q$

If Sam sends message $c = 17639$
Ryan needs to find $m \equiv 17639^{11613} mod\ 21829$
Since Ryan knows the factorisation of $21829 = 83 \times 263,\ he\ solves:$
$m \equiv 17639^{11613} mod\ 83$ $\quad$ $m \equiv 17639^{11613} mod\ 263$
$17639 \equiv 43\ mod\ 83 \Rightarrow m \equiv 43^{11613} mod\ 83$
Since 83 is a prime,
$By\ Fermat's\ Little\ Theorem,$
$43^{83-1} \equiv 1\ mod\ 83 \Rightarrow 43^{82} \equiv 1\ mod\ 83$
$43^{82^{141}} \equiv 1^{141} mod\ 83$ $\quad$ $\because Modulo\ Indices\ Property\ Section\ 2.1.1$

$43^{11562} \equiv 1 \; mod \; 83$

$43^{11562} \cdot 43^{51} \equiv 1 \cdot 43^{51} \; mod \; 83$ ∵ Modulo Multiplication Property Section 2.1.1

$\Rightarrow 43^{11613} mod \; 83 = 43^{51} \; mod \; 83$

$\Rightarrow m \equiv 43^{51} \; mod \; 83$

Computing this:

$\Rightarrow m \equiv 58 \; mod \; 83$

Performing the same method on the second equation:

$\Rightarrow m \equiv 18^{85} mod \; 263 \Rightarrow m \equiv 44 \; mod \; 263$

$m = 263k + 44$ $\quad k \in Z$

$m \equiv 58 \; mod \; 83$

$\Rightarrow 263k + 44 \equiv 58 \; mod \; 83$

$14k + 44 \equiv 58 \; mod \; 83$ ∵ $263 \equiv 14 \; mod \; 83$

$14k \equiv 14 \; mod \; 83$

$\Rightarrow k = 1$

$\Rightarrow m = 263(1) + 44$

$\Rightarrow m = 307$

Another adaptation to RSA is to use 3 prime factors, instead of 2 prime factors of n. The Chinese Remainder Theorem is applicable for more than 2 prime modulus, which can be leveraged here. Abd Zaid and Hassan implement a 3-prime RSA adaptation, in which the time taken to generate keys, encrypt and decrypt the plaintext is significantly reduced. This can be seen in the Table 6 below (Abd Zaid and Hassan, 2018).

Table 6: Comparison of time taken using 2 prime factors and 3 prime factors

Key Size	Message Size	Standard RSA time (s)			Modified RSA time (s)		
		Key Gen.	Encryption	Decryption	Key Gen.	Encryption	Decryption
68 bit	3 bit	0.011	0.00015	0.000151	0.008	0.000137	0.00005
340 bit	20 bit	0.228	0.0157	0.0144	0.107	0.01005	0.00096
664 bit	94 bit	5.34	0.219	0.202	0.989	0.213	0.00643

6.0 Conclusion

The objective of this mathematical exploration was to analyze and discuss how discrete mathematical theorems are effectively employed to ensure the security and safety of the RSA cryptosystem. In this extended essay, all the mathematical theorems and concepts such as Modular arithmetic, Euclidean and Extended Euclidean algorithm, Chinese Remainder Theorem, Euler's Totient functions and Fermat's Little Theorem are examined in detail to understand their application in RSA. It has been established that the intractability of the prime number factorization is indeed the core component contributing to the security of the RSA algorithm. The functioning of the encryption and decryption processes, and the uniqueness in obtaining the original plaintext are comprehensively substantiated through the proofs and a detailed step-by-step example. The objective of this exploration has been successfully accomplished.

7.0 Bibliography

Asdad, S., 2019. *The RSA Algorithm*. [ebook] Available at: <https://www.researchgate.net/publication/338623532_The_RSA_Algorithm> [Accessed 5 March 2020].

Bodur, H. and Kara, R., 2015. *Secure SMS Encryption Using RSA Encryption Algorithm On Android Message Application*.

Calderbank, M., 2007. *The RSA Cryptosystem: History, Algorithm, Primes*. [ebook] Available at: <https://www.math.uchicago.edu/~may/VIGRE/VIGRE2007/REUPapers/FINALAPP/Calderbank.pdf> [Accessed 27 March 2020].

Clark, A., 1971. *Elements Of Abstract Algebra*. New York: Dover Publications.

Collins, J., 2018. *Get Smart Maths*.

Conrad, K., n.d.. *Fermat's Little Theorem*. [ebook] Available at: <https://kconrad.math.uconn.edu/blurbs/ugradnumthy/fermatlittletheorem.pdf> [Accessed 8 March 2020].

Galbraith, S., 2018. *Mathematics Of Public Key Cryptography. Version 2.0*.

Gallier, J. and Quaintance, J., 2020. *Mathematical Foundations And Aspects Of Discrete Mathematics Second Edition In Progress*.

Gorodnik, A., 2016. *Lecture 3: Congruences*. [ebook] Available at: <https://www.math.uzh.ch/gorodnik/nt/lecture3.pdf> [Accessed 4 February 2020].

Goshwe, N., 2013. Data Encryption and Decryption Using RSA Algorithm in a Network Environment. *International Journal of Computer Science and Network Security*, Vol.

13(No. 7).

Guichard, D. and Keef, P., 2020. *An Introduction To Higher Mathematics*. [ebook] Available at: <https://www.whitman.edu/mathematics/higher_math_online/higher_math.pdf> [Accessed 12 April 2020].

Kaliski, B., n.d. *The Mathematics Of The RSA Public-Key Cryptosystem*.

Kelly, M., 2009. *The RSA Algorithm: A Mathematical History Of The Ubiquitous Cryptological Algorithm*. [ebook] Available at: <https://www.sccs.swarthmore.edu/users/10/mkelly1/rsa.pdf> [Accessed 22 March 2020].

Kowalczyk, C., 2020. *Crypto-IT*. [online] Crypto-it.net. Available at: <http://www.crypto-it.net/eng/asymmetric/index.html> [Accessed 4 February 2020].

Lynn, B., n.d. *Number Theory - Carmichael Numbers*. [online] Crypto.stanford.edu. Available at: <https://crypto.stanford.edu/pbc/notes/numbertheory/carmichael.html> [Accessed 22 January 2020].

M. Abd Zaid, M. and Hassan, D., 2018. Lightweight Rsa Algorithm Using Three Prime Numbers. *International Journal of Engineering & Technology,* [online] 7(4.36), p.293. Available at: <https://www.researchgate.net/publication/330636701_Lightweight_Rsa_Algorithm_Using_Three_Prime_Numbers> [Accessed 9 April 2020].

Meelu, P., Malik, S., Patel, R. and Singh, B., 2010. RSA and its Correctness through Modular Arithmetic.

Menezes, A., Van Oorschot, P. and Vanstone, S., 1997. *Handbook Of Applied Cryptography*. Boca Raton, Fla.: CRC Press.

Mumir, W., 2005. [online] Available at: <https://www.researchgate.net/publication/276317739_Cryptography> [Accessed 3 April 2020].

Mtholyoke.edu. n.d. *Powermod Calculator*. [online] Available at: <https://www.mtholyoke.edu/courses/quenell/s2003/ma139/js/powermod.html> [Accessed 24 April 2020].

n.d. *2. Integers And Algorithms*. [ebook] Florida State University. Available at: <https://www.math.fsu.edu/~pkirby/mad2104/SlideShow/s5_2.pdf> [Accessed 6 April 2020].

n.d. *Euler's Phi Function*. [ebook] Loyola University Chicago. Available at: <http://gauss.math.luc.edu/greicius/Math201/Fall2012/Lectures/euler-phi.article.pdf> [Accessed 15 May 2020].

n.d. *The Rabin-Miller Primality Test*. [ebook] Available at: <http://home.sandiego.edu/~dhoffoss/teaching/cryptography/10-Rabin-Miller.pdf> [Accessed 4 February 2020].

Nisha, S. and Farik, M., 2017. RSA Public Key Cryptography Algorithm – A Review. *International Journal of Scientific & Technology Research*, Vol. 6 (Issue 07).

Paar, C., 2014. *Lecture 11: Number Theory For PKC: Euclidean Algorithm, Euler's Phi Function & Euler's Theorem*. [video] Available at: <https://www.youtube.com/watch?v=fq6SXByItUI&ab_channel=Introductionto

CryptographybyChristofPaar> [Accessed 5 March 2020].

Paar, C., 2014. *Lecture 12: The RSA Cryptosystem And Efficient Exponentiation*. [video] Available at: <https://www.youtube.com/watch?v=QSlWzKNbKrU&ab_channel=IntroductiontoCryptographybyChristofPaar> [Accessed 5 February 2020].

Penn, M., 2019. *Number Theory | Chinese Remainder Theorem Proof*. [video] Available at: <https://www.youtube.com/watch?v=EolotL9HN8k&ab_channel=MichaelPenn> [Accessed 18 March 2020].

Penn, M., 2019. *Number Theory | Modular Inverses*. [video] Available at: <https://www.youtube.com/watch?v=uPFh9_nLw1c&ab_channel=MichaelPenn> [Accessed 15 April 2020].

Proofwiki.org. 2020. *LCM Divides Common Multiple - Proofwiki*. [online] Available at: <https://proofwiki.org/wiki/LCM_Divides_Common_Multiple> [Accessed 15 April 2020].

Rabah, K., 2006. Review of Methods for Integer Factorization Applied to Cryptography. *Journal of Applied Sciences*, 6(2), pp.458-481.

Ruohonen, K., 2014. *Mathematical Cryptology*. [ebook] Available at: <http://math.tut.fi/~ruohonen/MC.pdf> [Accessed 4 February 2020].

Suzuki, J., 2019. *Decrypting RSA Using The Chinese Remainder Theorem*. [video] Available at: <https://www.youtube.com/watch?v=NcPdiPrY_g8&ab_channel=JeffSuzuki> [Accessed 9 April 2020].

Weisstein, E., n.d. *Fermat's Little Theorem -- From Wolfram Mathworld*. [online] Mathworld.wolfram.com. Available at:<https://mathworld.wolfram.com/FermatsLittleTheorem.html> [Accessed 4 February 2020].