INVESTIGATING THE MATHEMATICS BEHIND RSA-BASED ENCRYPTION ATTACKS IN THE FORM OF RANSOMWARE

Math Internal Assessment

FEBRUARY 25, 2018

MARTIN STARKOV
Ressun Lukio

Introduction

In recent years, decreased physical record-keeping has made the safe storage of digital information paramount. Just as physical documentation is at risk of being stolen, it is important to understand that digital data is also vulnerable to theft via cyberattacks. Such attacks exist in many forms, one of which is ransomware – a type of software that locks users out of their computers until a ransom fee is paid. [1] While having existed for over a decade, ransomware attacks have grown exponentially in recent years. [2] Security advisors often describe ransomware campaigns as coordinated operations which rely on one basic principle: the ability to lock users out of their computers using effective encryption algorithms such as the one I will be investigating, known as RSA.

The RSA algorithm, developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman, is a public key encryption algorithm in which the public and private keys used to encrypt and decrypt information, are separate. [20] Their separation allows only the party in control of the private key to decrypt said information. [3]

Three basic principles enable the RSA algorithm to work as effectively as it does:

1. Finding large prime numbers is straightforward

   Numbers are easy to test for primality and therefore the generation and testing of thousands of numbers allows us to easily find large primes. [4]

2. Multiplication is elementary

   Modern computational power enables us to easily perform multiplication of two or more numbers extremely effectively, which facilitates the easy creation of the public keys. [4]

3. Factorizing the product of two prime numbers is complicated

   Prime factorization is an extremely difficult process as there is no direct way of factorizing massive products into their prime factors. [4]

This being said, it is important to note that as quantum computing advances, allowing for the simultaneous processing of multiple operations at once, the effectiveness of the RSA algorithm will fall as large products are factorized quicker. Newer techniques and algorithms should then be investigated.

I feel as though a significant reason for the unawareness around the topic of cyber security is due to the complicated nature of mathematics behind the effectiveness of these attacks. People often do not consider the possibility of an attack happening to them until it is too late. Once infected, users have virtually no option but to pay the ransom fee, as cracking most encryption algorithms, especially the RSA algorithm, can take many years. [4]

I have always found the mathematics behind number theory interesting and think it is important for people to understand how the relationships between prime numbers allow us to create effective algorithms that secure our information. Knowledge of the systems and effectiveness of encryption allows us to respond appropriately and safeguard ourselves from these kinds of attacks in the future.

My aim during this investigation is to introduce and elaborate upon the mathematics behind the RSA algorithm. While at first it may look complicated, I hope to break it down into its component theorems and algorithms. I will begin the investigation by introducing the reader to some new

terminology relating to RSA and attempt to build up a basic understanding behind common divisors and the ways in which they operate.

Some definitions

Plaintext is text or characters that are waiting for encryption by the client or server. An example of plaintext is sensitive information, such as a password. Once information in the form of plaintext has been input into an encryption algorithm, it is considered ciphertext, this is data that has been 'scrambled' and is unreadable without a special key. [5]

Some mathematical notation that might be unfamiliar to an IB student: ∧ means logical conjunction for "and," ∀ means "for any," and ∃ states that "there exists."

Common Divisors

To begin understanding the mathematics behind the RSA algorithm, we must first build some basic knowledge around common divisors and some theorems explaining how they operate.

It is intuitive that if a is fully divisible by b $(b \neq 0)$, the quotient will equal some integer c.

$$\frac{a}{b} = c : c \in \mathbb{Z}$$

An alternative mathematical notation for this operation is $b \setminus a$ meaning a is divisible by b to give some integer number. I will be using this notation as it is more concise.

Linear combination is a principal stating that any integer m can be expressed in terms of any two non-zero integers a and b,

$$m = sa + tb : s, t \in \mathbb{Z}$$

Bézout's identity uses linear combination to show that there exists an integer m that is the greatest common divisor of said integers a and b. [6]

The greatest common divisor of two positive integers is the largest integer that divides them both. For a and b, the greatest common divisor can be denoted as $\gcd(a, b)$. By definition, the greatest common divisor of any integer and 0 is that integer, in other words $\gcd(a, 0) = a$. [7]

Assuming both integers a and b are divisible by c, we can express this as

$$a = cx : x \in \mathbb{Z}$$

$$b = cy : y \in \mathbb{Z}$$

Therefore, via substitution we find that

$$\forall\, s, t \in \mathbb{Z} : sa + tb = scx + tcy = c(sx + ty)$$

Letting $z = sx + ty$, we deduce that

$$\exists\, z \in \mathbb{Z} : sa + tb = cz$$

Meaning that if $sa + tb$ is divided by c, there exists an integer z, therefore the linear combination of a and b is divisible by c. [28]

$$\frac{sa + tb}{c} = z$$

According to Euclid's Division Lemma, dividing one integer a by another b will leave an integer quotient q and a remainder r, such that $0 \leq r \leq b$. [7] For example, $17 = 6 \times 2 + 5$.

In mathematical form, this can be written as

$$a = bq + r$$

A handy trick exists for finding the quotient and remainder of two large numbers. Say we wanted to find the q and r of $a = 3000$ and $b = 197$. We would first divide 3000 by 197 to get a decimal number of approximately 15.2284 ..., looking at this number we see that the integer part is equivalent to 15, therefore $q = 15$, to find r we simply rearrange the equation $a = bq + r$ to get

$$r = 3000 - 197 \times 15 = 45$$

Since $\gcd(a, b)$ is the largest number that divides both a and b separately, it subsequently also divides the integer combination of a and b as discussed earlier in linear combination. [8]

$$\gcd(a, b) \setminus a \wedge \gcd(a, b) \setminus b$$

$$\therefore \gcd(a, b) \setminus (sa + tb)$$

$$\therefore \gcd(a, b) \setminus (a - tb)$$

In this case, $s = 1$ and t is some negative integer. Considering Euclid's Division Lemma form where $a - tb = r$, substituting this into the equation gives

$$\gcd(a, b) \setminus r$$

As $\gcd(b, r)$ is the greatest common divisor of b and r and we have deduced that $\gcd(a, b)$ is some divisor of r, we therefore know that $\gcd(a, b)$ is smaller than or equal to the greatest divisor of r.

$$\gcd(a, b) \leq \gcd(b, r)$$

Repeating the same deductive process for b and r, we get a similar equation in reverse.

$$\gcd(b, r) \setminus b \wedge \gcd(b, r) \setminus r$$

$$\therefore \gcd(b, r) \setminus (qb + r)$$

$$\therefore \gcd(b, r) \setminus a$$

$$\therefore \gcd(b, r) \leq \gcd(a, b)$$

Combining the two equations $\gcd(a, b) \leq \gcd(b, r)$ and $\gcd(b, r) \leq \gcd(a, b)$ gives us

$$\gcd(a, b) = \gcd(b, r)$$

Euclidean Algorithm

Understanding the Euclidean algorithm provides insight on a division process that enables us to quickly and effectively find the greatest common divisor of a and b.

The Euclidean algorithm states that the $\gcd(a, b)$ can be found by completing a series of divisions where the denominator b of the division [9]

$$\frac{a}{b} = q_1 \ (remainder \ r_2)$$

becomes the numerator of the next division, and the obtained remainder r_2 becomes the denominator of the next division

$$\frac{b}{r_2} = q_2\ (remainder\ r_3)$$

This process is repeated until $r_{k+1} = 0$ is eventually reached

$$\frac{r_2}{r_3} = q_3\ (remainder\ r_4) \rightarrow \cdots \rightarrow \frac{r_{k-2}}{r_{k-1}} = q_{k-1}\ (remainder\ r_k) \rightarrow \frac{r_{k-1}}{r_k} = q_k\ (remainder\ r_{k+1} = 0)$$

We proved earlier that $\gcd(a,b) = \gcd(b,r)$. We can extend this proof to the above sequence of divisions with k number of steps and see that when $r_{k+1} = 0$, r_k will be the greatest common divisor of a and b, [22]

$$\gcd(a,b) = \gcd(b,r_2) = \gcd(r_2,r_3) = \ \ldots\ = \gcd(r_{k-2},r_{k-1}) = \gcd(r_k,0) = r_k$$

We may also say that $a = r_0$ and $b = r_1$, thus every division follows the general form

$$r_k = q_{k+1}r_{k+1} + r_{k+2}$$

As an example, let us find the $\gcd(48,18)$.

$$\frac{48}{18} = 2\ (\text{remainder } 12) \rightarrow \frac{18}{12} = 1\ (remainder\ 6) \rightarrow \frac{12}{6} = 2\ (remainder\ 0)$$

Therefore, the $\gcd(48,18) = 6$.

The proof for this algorithm relies on the fact that $r_{k+1} \leq b$. In order to find $\gcd(a,b)$, we divide a finite amount of times and eventually $\gcd(r_k,0)$ is reached. [10] Due to restrictions in space I am not able to show the full proof for the algorithm in this investigation.

Modular Arithmetic

Modular arithmetic is a field of mathematics that allows us to look at numbers in a certain modulo. 12-hour clocks operate in modulo 12, meaning that when they complete a full cycle the counting restarts from 1. Therefore the 13^{th} hour of the clock is often simply referred to as "1 o'clock." This kind of relationship between numbers is called congruence and simply states that two integers are equivalent when counting in a certain modulo. Two integers a and b are said to be congruent in modulo q when their difference $a - b$ is divided by q and gives some integer r, [12]

$$\frac{a-b}{q} = r : r \in \mathbb{Z}$$

From this formula, we see that it is mathematically equivalent to Euclid's Division Lemma. Congruence is however most often expressed in the congruent modulo form

$$a \equiv b\ (mod\ q)$$

as the integer r is often irrelevant when examining the relationship between a and b.

Mathematically speaking, saying "13 is the same as 1 on a clock (in modulo 12)" is written as

$$13 \equiv 1\ (mod\ 12)$$

The relationship between congruence modulo form and Euclid's Division Lemma is extremely important as it is the basis of the Extended Euclidean algorithm (discussed in the following chapter) and is an integral part of the RSA algorithm.

Modular arithmetic equivalence rules state [12] that if $a_1 \equiv a_2 \ (mod\ m)$ and $b_1 \equiv b_2 \ (mod\ m)$ then

$$a_1 \pm\ b_1 \equiv a_2 \pm\ b_2 \ (mod\ m)$$

$$a_1\ b_1 \equiv a_2\ b_2 \ (mod\ m)$$

$$na_1 \equiv na_2 \ (mod\ m)$$

$$a^n \equiv b^n \ (mod\ m)$$

This makes intuitive sense when examining the clock example: adding 2 hours to 13 o'clock is the same as adding 2 hours to 1 o'clock as they are both equivalent in modulo 12. This logic also applies to multiplication and the other mentioned operations.

Extended Euclidean Algorithm

The Extended Euclidean algorithm is an extension to the Euclidean algorithm that, in addition to computing the $m = \gcd(a, b)$, allows us to find coefficients s and t in the subsequent expression, known as Bézout's identity: [13]

$$m = sa + tb : s, t \in \mathbb{Z}$$

When $\gcd(a, b) = 1$, a and b are said to be co-prime, this is definitional. [13] It is important to note however, that a and b do not necessarily have to be prime in order for them to be co-prime.

In cases where a and b are co-prime, meaning $m = 1$, we see that the previous equation can be rearranged in the form

$$\frac{1 - sa}{b} = t$$

By turning this expression into modulo form, it becomes apparent how the algorithm is useful in finding the coefficient, s in the following modular multiplicative inverse [13]

$$s \equiv a^{-1} \ (mod\ b)$$

$$\because sa \equiv aa^{-1} \equiv 1 \ (mod\ b)$$

As we know from before

$$\gcd(a, b) = \gcd(b, r_2) = \gcd(r_2, r_3) = \ \ldots\ = \gcd(r_{k-2}, r_{k-1}) = \gcd(r_k, 0) = r_k$$

Therefore, by the idea of linear combination

$$\gcd(a, b) = r_k = s_k a + t_k b$$

Rearranging the general form for divisions in the Euclidean algorithm and applying the linear combination form gives us

$$r_{k+2} = r_k - q_{k+1} r_{k+1}$$

$$r_{k+2} = s_{k+2} a + t_{k+2} b$$

$$r_k - q_{k+1} r_{k+1} = (s_k a + t_k b) - q_{k+1}(s_{k+1} a + t_{k+1} b) = (s_k - q_{k+1} s_{k+1}) a + (t_k - q_{k+1} t_{k+1}) b$$

Equating coefficients in the second equation and the last one reveals that

$$s_{k+2} = s_k - q_{k+1}s_{k+1} \; and \; t_{k+2} = t_k - q_{k+1}t_{k+1}$$

By letting $s_0 = 1$ and $t_0 = 0$, we are able to obtain s_k and similarly by letting $s_1 = 0$ and $t_1 = 1$, we obtain t. Say $a = 3567$ and $b = 158$, the following table shows the series of operations completed to find s and t:

Step $\boldsymbol{k}$	Quotient $\boldsymbol{q_{k-1}}$	Remainder $\boldsymbol{r_k}$	Coefficient $\boldsymbol{s_k}$	Coefficient $\boldsymbol{t_k}$
0		3567	1	0
1		158	0	1
2	$3567 \div 158 = 22$	$3567 - 22 \times 158 = 91$	$1 - 0 \times 22 = 1$	$0 - 1 \times 22 = -22$
3	$158 \div 91 = 1$	$158 - 1 \times 91 = 67$	$0 - 1 \times 1 = -1$	$1 - (-22) \times 1 = 23$
4	$91 \div 67 = 1$	$91 - 1 \times 67 = 24$	$1 - (-1) \times 1 = 2$	$-22 - 23 \times 1 = -45$
5	$67 \div 24 = 2$	$67 - 2 \times 24 = 19$	$-1 - 2 \times 2 = -5$	$23 - (-45) \times 2 = 113$
6	$24 \div 19 = 1$	$24 - 1 \times 19 = 5$	$2 - (-5) \times 1 = 7$	$-45 - 113 \times 1 = -158$
7	$19 \div 5 = 3$	$19 - 3 \times 5 = 4$	$-5 - 7 \times 3 = -26$	$113 - (-158) \times 3 = 587$
8	$5 \div 4 = 1$	$5 - 1 \times 4 = 1$	$7 - (-26) \times 1 = 33$	$-158 - 587 \times 1 = -745$
9	$4 \div 1 = 4$	$4 - 4 \times 1 = 0$	$-26 - 33 \times 4 = -158$	$587 - (-745) \times 4 = 3567$

Examining the table shows that the linear combination of $\gcd(3567,158) = 33 \times 3567 + (-745) \times 158 = 1$. Therefore, $a = 3567$ and $b = 158$ are co-prime and their linear combination can be written as

$$33a - 745b = 1$$

Rewriting this in modular multiplicative inverse form shows that

$$33 \times 3567 \equiv 1 \; (mod \; 158)$$

This same procedure can be reversed to find the value of k in $\gcd(3567, k)$, given s_k and t_k. [13]

Chinese Remainder Theorem

The Chinese remainder theorem allows us to solve systems of congruences by taking the products of the different modulo and combining everything into one equation to solve for x. Using this theorem in reverse allows us to solve congruences with large modulo by factorizing them down to simpler equations. The theorem asserts that a system of congruences consisting of a set of natural numbers $n_i = \{n_1, n_2, \dots, n_i\}$ that are pairwise co-prime, that is, each positive integer n_i is co-prime to any other number of the set, and given a set of arbitrary integers $z = \{a_1, a_2, \dots, a_i\}$ [14]

$$x \equiv a_1 \; (mod \; n_1)$$

$$x \equiv a_2 \; (mod \; n_2)$$

...

$$x \equiv a_i \; (mod \; n_i)$$

There exists a unique solution for x in modulo $\mathrm{N} = \prod_{i=0}^{i} n_i$ which is the product of all values n_i. I will attempt to generalize this solution for a k amount of congruences. The capital Pi $\prod$ notation seen above is similar to sigma $\sum$ notation, however instead of expressing the sum of a sequence, it expresses the product (multiplication) of a sequence.

First of all, we must understand what is required to combine modulo. If each congruence of the system can be cancelled by taking a separate modulo of it, then we will be able to express the unique solution for x in terms of each separate congruence combined $x \equiv k\ (mod\ N)$. [15]

Through linear combination we can express x as follows

$$x = \sum_{i=1}^{k} a_i y_i z_i$$

In this case, a_i represents each individual arbitrary integer that is given in the system of simultaneous congruences. We know from Bézout's identity that each integer can be expressed as a factor multiplied by a coefficient, z_i. When considering the pairwise co-prime product, N, of the set n_i, we notice that in order to be able to take any modulo n_i, each term must contain a coefficient $y_i = \frac{N}{n_i}$ allowing it to cancel when taking a specific modulo n. Since all n_i are co-prime, meaning their greatest common factor is 1, we may use the Extended Euclidean algorithm to compute z_i, [15]

$$z_i y_i \equiv 1\ (mod\ n_i)$$

To provide some insight on why y_i is required, let us consider an example:

$$x \equiv 2\ (mod\ 3)$$

$$x \equiv 1\ (mod\ 5)$$

In order to be able to rewrite x in terms of both $mod\ 3$ and $mod\ 5$, we will need to express it as two integer parts so that taking $mod\ 3$ on both sides will cancel all terms except 2 and taking $mod\ 5$ will cancel all terms except 1. Let us say

$$x = 2y_1 z_1 + 1y_2 z_2$$

We want the statement $x \equiv 2\ (mod\ 3)$ to hold true, therefore the latter part of the equation needs to equal 0 when taking $mod\ 3$. In order to accomplish this, all other terms must contain a factor of 3. Examining $y_i = \frac{N}{n_i}$ shows us that $y_2 = \frac{3\times5}{5} = 3$ allowing it to cancel when $mod\ 3$ is taken. Similarly, $y_1 = \frac{3\times5}{3} = 5$ allows the first integer part of the equation to equal 0 when taking $mod\ 5$ (since it has a factor of 5). The equation can therefore be expressed as

$$x = 10z_1 + 3z_2$$

Bézout's coefficients z_1 and z_2 can easily be solved using the Extended Euclidean algorithm and the unique solution for x can be found. [15] Due to a lack of space however, I will leave those calculations out of this investigation.

Euler's totient function

Euler's totient function $\phi(n)$ is a useful computational tool for calculating the number of positive integers that are co-prime to an integer n. It therefore expresses the size of the set of positive natural numbers k that are less than n and only share the common factor 1 with n. [16]

Utilizing this function, we are able to calculate how many numbers a computer would have to go through while trying to crack $\gcd(k, \phi(n)) = 1$, the significance of which is discussed later in the RSA algorithm chapter. Applying $\phi(n)$ shows us whether or not the chosen value of k has enough co-prime integers. [26]

Euler's totient function is denoted as

$$\phi(n) = n \prod_{p \setminus n} \left(1 - \frac{1}{p}\right)$$

In Euler's totient function, $p \setminus n$ simply stands for all cases of prime numbers p that divide n. Every positive integer number n can be expressed in terms of its prime factors, such that

$$n = p_1^{k_1} p_2^{k_2} \cdots p_t^{k_t}$$

For example, the prime factors of 50 are 2 and 5 because $50 = 2 \times 5^2$. Applying Euler's totient function gives us the number of co-prime integers to 50

$$\phi(50) = \phi(2 \times 5^2) = 50 \prod_{p \setminus n} \left(1 - \frac{1}{p}\right) = 50 \left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{5}\right) = 50 \times \frac{1}{2} \times \frac{4}{5} = 20$$

By multiplying n by the fractions at which its prime factors do not appear, we effectively eliminate the set of numbers that contain those factors. [16] For example, half of the natural number set is divisible by 2, therefore since 50 is divisible by 2 we can eliminate half of the integers from 0 to 50, resulting in only 25 numbers that do not contain the factor of 2. Every 5^{th} number is a factor of 5 and since 50 is divisible by 5, we can eliminate every 5^{th} number, resulting in only $\frac{4}{5}$ of the 25 possibilities, hence there are only 20 distinct numbers that are co-prime to 50.

Fermat's Little Theorem

Fermat's little theorem allows computers to find large primes using random numbers of large sizes. The theorem states that a natural number a raised to the power of any co-prime prime p will equal a in modulo p, [17]

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

In cases where a is not divisible by p, $a^{p-1} - 1$ is an integer multiple of p [25]

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

It is important to note that $0 \leq a < p$ because any number larger than p in the given $mod\ p$ can be reduced according to modular arithmetic. When testing large numbers for primality, we simply need to raise an integer to the power of the proposed number minus one, and then divide by the proposed number. If the result is an integer number, the proposed number is a prime. [17]

$$\frac{a^{p-1}}{p} = q : q \in \mathbb{Z} \rightarrow \mathrm{p} = \mathrm{prime}$$

When proving the theorem, I will only consider the first form

$$n^p \equiv n \ (mod\ p)$$

The theorem holds when $n = 0$ and since $0 \leq n < p$, we are able to divide both sides by n and obtain the earlier mentioned alternative form $a^{p-1} \equiv 1 \ (mod\ p)$. For the proof, I will be using mathematical induction. [18] The first step is to let $n = 1$

$$1^p \equiv 1 \ (mod\ p)$$

1 to the power of any positive number will always equal 1, therefore the first step holds true.

Secondly, we assume that the result holds true when $n = k$, meaning $k^p \equiv k \ (mod\ p)$.

Lastly, we must prove that the theorem holds for all $n = k + 1$. Therefore, we must prove that $(k+1)^p \equiv k + 1 \ (mod\ p)$ holds true. By adding 1 to both sides of our assumption, we get

$$k^p + 1 \equiv k + 1 \ (mod\ p)$$

The left-hand side (LHS) of this form can be rewritten in the form

$$LHS = k^p + 1 \ \equiv \binom{p}{0} k^{p-0} \times 1^0 + \binom{p}{p} k^{p-p} \times 1^p \ (mod\ p)$$

It appears that only the first and last term of a particular binomial expansion remain when taking $(mod\ p)$. In order to achieve our desired result of $(k+1)^p$, we must find a binomial form that, when taking $(mod\ p)$ on both sides, will make each term that divides p equal to 0. In fact, any term with a binomial coefficient of $\binom{p}{i}$ where $i \neq 0$ and $i \neq p$ will equal to 0 because

$$\binom{p}{i} = \frac{p(p-1)!}{i!\,(p-i)!}$$

This expression shows us that $i!\,(p-i)! \setminus p(p-1)!$, however p is co-prime to $i!\,(p-i)!$ since neither of them share a common factor, $0 < i < p$ and p is a prime. Therefore,

$$i!\,(p-i)! \setminus (p-1)!$$

Hence, when $0 < i < p$, taking $(mod\ p)$ will cause all terms with the coefficient $\binom{p}{i}$ to equal 0. [27] Therefore, the left-hand side (LHS) can be expressed as

$$k^p + 1 \equiv \binom{p}{0} k^{p-0} \times 1^0 + \binom{p}{1} k^{p-1} \times 1^1 + \cdots + \binom{p}{p-1} k^{p-(p-1)} \times 1^{(p-1)} + \binom{p}{p} k^{p-p} \times 1^p \ (mod\ p)$$

In fact, this is the binomial form of $(k+1)^p$ in $(mod\ p)$

$$LHS = k^p + 1 \equiv \sum_{i=0}^{p} \binom{p}{i} k^{p-i} \times 1^i \equiv (k+1)^p \ (mod\ p)$$

Substituting the left-hand side into the original assumption gives us

$$(k+1)^p \equiv k + 1 \ (mod\ p)$$

This is the equation we needed to prove for all $n = k + 1$, therefore by mathematical induction, Fermat's little theorem holds true when $0 \leq n < p$.

Euler's theorem

Euler's theorem provides an alternative method for calculating modular multiplicative inverses using Euler's totient function and generalizes Fermat's little theorem. The theorem states that when two positive integers a and n are co-prime, their relationship can be expressed as [19]

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

This form allows us to easily reduce numbers with large exponents such as for example 3^{234}. Consider this number in $mod\ 10$, in this case $a = 3$ and $n = 10$, they are both co-prime and $\phi(n)$ computes to 4

$$3^4 \equiv 1 \ (mod\ 10)$$

$$3^{234} \equiv 3^{4\times 58+2} \equiv (3^4)^{58} \times 3^2 \equiv 1^{58} \times 3^2 \equiv 9 \ (mod\ 10)$$

We can also prove that

$$a^{\phi(n)^k} \equiv 1 \ (mod\ n)$$

$$\because a^{\phi(n)^k} \equiv a^{\phi(n)} \times a^{\phi(n)^{k-1}} (mod\ n) \equiv a^{\phi(n)^{k-1}} (mod\ n)$$

As this process is repeated k times, $a^{\phi(n)^{k-1}}$ approaches to 1.

Let there exist an integer set $R = \{r_1, r_2, \dots, r_{\phi(n)}\}\ (mod\ n)$, where no two $r_{\phi(n)}$ are congruent in modulo n and the number of elements in the set is equal to $\phi(n)$, we define $\gcd(r_{\phi(n)}, n) = 1$. Multiplying this set by integer a gives [19]

$$aR = \{ar_1, ar_2, \dots, ar_{\phi(n)}\}\ (mod\ n)$$

In modulo n, these two sets aR and R are found to be congruent and hence

$$\prod_{i=1}^{\phi(n)} r_i \equiv \prod_{i=1}^{\phi(n)} ar_i \equiv a^{\phi(n)} \prod_{i=1}^{\phi(n)} r_i \ (mod\ n)$$

By cancelling $\prod_{i=1}^{\phi(n)} r_i$ on both sides we have shown Euler's theorem to hold true

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

The RSA algorithm

Referring back to the introduction, we may recall that the use of the RSA algorithm requires the generation of two public keys and a private key.

The first public key n is generated by finding two large prime numbers p and q and determining their product $n = pq$. The second public key k must be given a value such that it is co-prime to $z = (p-1)(q-1)$, that is, $\gcd(k, (p-1)(q-1)) = 1$. This creates the necessary relationship between the public keys n and k. [21] By examining z, we see that it is equal to Euler's totient function value for n, because

$$\phi(n) = \phi(pq) = pq\left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{q}\right) = pq - p - q + 1 = (p-1)(q-1) = z$$

Using the Extended Euclidean algorithm in reverse for the linear combination of k and $\phi(n)$, we are able to assign a value to s, the server's private key

$$ks - t(p-1)(q-1) = 1$$

$$\therefore s = \frac{1 + t(p-1)(q-1)}{k}$$

Under ordinary circumstances, the public key pair $P = (n, k)$ is published and used by clients to encrypt their information. However, in cases of ransomware attacks these keys are sent through infected programs onto the user's computer to start a background encryption process. The private key pair $S = (n, s)$ remains in the attacker's safekeeping. This private key pair is then sent to the victim once a ransom fee has been paid.

Let P stand for some plaintext message that the attacker is trying to lock. The following formula will encrypt P

$$E \equiv P^k \ (mod\ n)$$

As we see above, all data on a user's computer can be encrypted without the need for a private key. This means that attackers are easily able to encrypt information remotely without attaching a decryption key to the malicious program used for background encryption.

Eventually the victim notices that some or all of their files have been encrypted and locked and is prompted with a message demanding a ransom fee. Once the demanded transaction is made, the attacker sends the victim the private key and decryption algorithm required to unlock their files.

The algorithm used to decrypt E is

$$E^s \equiv P \ (mod\ n)$$

The decryption formula uses both the public and private keys. Attackers assign large values to p and q, resulting in a large public key n and subsequently much larger values of k and s. By examining the math explained during this investigation, it becomes clear that large values of p and q make decryption virtually impossible as there is no direct way to find the prime factors of n. [21]

According to modular arithmetic equivalence rules we are able to raise both sides of the encryption formula to the power of s

$$E^s \equiv P^{ks} \ (mod\ n)$$

Subsequently, rearranging both the encryption and decryption formulas gives us

$$P^{ks} \equiv P \ (mod\ n)$$

This is the relationship between the private key s and the public keys n and k. The proof for this rearranged formula lies in applying the Chinese remainder theorem to the case of n.

Since $n = pq$, we know that there are two separate simultaneous congruences

$$P^{ks} \equiv P \ (mod\ p)$$

$$P^{ks} \equiv P \ (mod\ q)$$

Since p and q are both primes, $\gcd(p, q) = 1$.

Examining the earlier form expressing ks and applying Fermat's little theorem shows us that

$$P^{ks} \equiv P^{1+t(p-1)(q-1)} \equiv P(P^{p-1})^{t(q-1)} \equiv P \times 1^{t(q-1)} \equiv P \ (mod\ p)$$

$$P^{ks} \equiv P^{1+t(q-1)(p-1)} \equiv P(P^{q-1})^{t(p-1)} \equiv P \times 1^{t(p-1)} \equiv P \ (mod\ q)$$

Combining the two simultaneous congruences using the Chinese remainder theorem, we ultimately prove that [21]

$$P^{ks} \equiv P \ (mod\ n)$$

Let us now examine an example where two relatively large primes are chosen.

We will start by assigning values for $p = 83$ and $q = 79$.

Calculating $n = pq$ gives $n = 83 \times 79 = 6557$.

The totient $\phi(n) = (p-1)(q-1) = (83-1)(79-1) = 6396$.

Now we must find a k that is coprime to $\phi(n)$. This is done easily as any prime is co-prime to any number as long as it is not a factor of that number. Letting $k = 101$, a prime, we easily see that $\frac{6396}{101}$ does not give us an integer value, therefore 101 is not a factor of $\phi(n)$ and the two integers are co-prime. Their relationship can be expressed in the form

$$101s \equiv 1 \ (mod\ 6396)$$

The calculation for this step requires lots of space and therefore I will not provide it directly, however applying the Extended Euclidean Algorithm, it can be found that $s = 3293$.

Depending on the context, computers use many different kinds of code languages to convert combinations of letters and text into numbers. In ASCII (American Standard Code for Information Interchange) for example, the word "code" is expressed as a set of numbers "099 111 100 101." For simplicity's sake, let us choose a simple message such as $P = 5$. Applying the encryption algorithm to this message, we find

$$E \equiv 5^{101} \ (mod\ 6557) \equiv 1236$$

The value of E can be computed easily using Euler's theorem. Our plaintext message $P = 5$ has now been converted to ciphertext $E = 1236$. In order to be able to read our original message again, we must have access to the private key and apply the decryption algorithm

$$1236^{3293} \equiv P^{3293 \times 101} \ (mod\ 6557)$$

Once again, using Euler's theorem we arrive at

$$P \equiv 5 \ (mod\ 6557)$$

And there we have it, our decrypted ciphertext message is $P = 5$, which was our original input.

The strength of the RSA algorithm in ransomware relies heavily on the ability to encrypt messages remotely without the need to send a private key to the victim's machine. Factorizing the public keys is difficult, as opening up and rearranging the equation $\phi(n) = (p-1)(q-1)$ gives us

$$\mathrm{p} + \mathrm{q} = \mathrm{n} - \phi(n) + 1$$

By examining the polynomial

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

We notice that the factorized form[4] is $(x-p)(x-q)$, meaning that in order to be able to crack the RSA algorithm, you need to be able to find $p+q$ and $n = pq$, which requires you to find p and q by factorizing n. This is the reason ransomware programs utilize the RSA algorithm: choosing enormous values for p and q will make n virtually impossible to obtain, hence leaving no options for the victim but to pay a ransom fee.

Final reflection

Looking back on the research process, I learned how to independently approach and analyze mathematics beyond the scope of the higher-level classroom. I applied this knowledge in the real world by examining ransomware and how its effectiveness is based on the principles of encryption. While I discussed only the malicious use of RSA as a tool for ransomware, it is important to note that it has alternative applications as a safeguard mechanism for contemporary technology.

Ransomware specifically interested me because of the increasing importance of cybersecurity in our ever-growing digital world. My investigation, however, only covered a small portion of the mathematical theorems and complex algorithms in this vast domain. In the future, by incorporating more encryption algorithms into my investigation, I could draw parallels and comparisons between them and examine the benefits and drawbacks of each one. For example, research into the AES (Advanced Encryption Standard) algorithm could provide useful insight into the necessary consideration of processing power in utilizing encryption. The RSA algorithm requires a high volume of calculations to be completed, which results in a slower speed of encryption and, therefore, limits the length of the private keys generated, making them less secure.

Overall, the investigation successfully showed the core mathematical principles that make RSA an effective encryption algorithm in modern ransomware attacks.

References

1. Symantec. 2016. Special Report: Ransomware and Businesses 2016. Available at: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ISTR2016_Ransomware_and_Businesses.pdf [Accessed 8 May 2017].
2. FBI. 2016. Incidents of Ransomware on the Rise. Available at: https://www.fbi.gov/news/stories/incidents-of-ransomware-on-the-rise [Accessed 8 May 2017].
3. Evgeny Milanov. 2009. The RSA Algorithm. Available at: https://sites.math.washington.edu/~morrow/336_09/papers/Yevgeny.pdf [Accessed 15 May 2017].

4. Burt Kaliski. 2007. The Mathematics of the RSA Public-Key Cryptosystem. Available at: http://www.mathaware.org/mam/06/Kaliski.pdf [Accessed 15 May 2017].
5. Kevin W. 2007. Ciphertext vs. Plaintext. Available at: https://www.denimgroup.com/resources/blog/2007/10/cleartext-vs-pl [Accessed 24 May 2017].
6. Keith Conrad. 2008. Divisibility and Greatest Common Divisors. Available at: http://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/divgcd.pdf [Accessed 9 May 2017].
7. Tkiryl. 2006. Euclid's Lemma. Available at: http://www.tkiryl.com/teaching/aa/les091503.pdf [Accessed 11 May 2017].
8. Proofwiki. 2011. GCD with Remainder. Available at: https://proofwiki.org/wiki/GCD_with_Remainder [Accessed 29 June 2017].
9. Wikipedia. 2017. Euclidean Algorithm. Available at: https://en.wikipedia.org/wiki/Euclidean_algorithm [Accessed 30 June 2017].
10. Proofwiki. 2016. Euclidean Algorithm. Available at: https://proofwiki.org/wiki/Euclidean_Algorithm [Accessed 30 June 2017].
11. Wikipedia. 2017. Modular Arithmetic. Available at: https://en.wikipedia.org/wiki/Modular_arithmetic [Accessed 10 July 2017].
12. Mathworld. 2007. Congruence. Available at: http://mathworld.wolfram.com/Congruence.html [Accessed 15 May 2017].
13. Wikipedia. 2017. Extended Euclidean Algorithm. Available at: https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm [Accessed 11 July 2017].
14. Keith Conrad. 2017. The Chinese Remainder Theorem. Available at: http://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/crt.pdf [Accessed 13 July 2017].
15. Wikipedia. 2017. Chinese Remainder Theorem. Available at: https://en.wikipedia.org/wiki/Chinese_remainder_theorem [Accessed 13 July 2017].
16. Brilliant. 2016. Euler's Totient Function. Available at: https://brilliant.org/wiki/eulers-totient-function [Accessed 9 May 2017].
17. Mathworld. 2006. Fermat's Little Theorem. Available at: http://mathworld.wolfram.com/FermatsLittleTheorem.html [Accessed 13 May 2017].
18. Caroline Laroche Turnage. 2008. Selected Proofs of Fermat's Little Theorem and Wilson's Theorem. Available at: https://sakai.wfu.edu/access/content/group/26f503c9-bec7-4bb7-bdab-82ed8fda9d39/publications/Student/Caroline%20LaRoche%20Turnage%20-%20Thesis.pdf [Accessed 15 May 2017].
19. Wikipedia. 2017. Euler's Theorem. Available at: https://en.wikipedia.org/wiki/Euler's_theorem [Accessed 20 July 2017].
20. Astrometry. 2012. The Art of Computer Programming. Available at: http://broiler.astrometry.net/~kilian/The_Art_of_Computer_Programming%20-%20Vol%201.pdf [Accessed 14 May 2017].
21. Andreas Klappenecker. 2005. The RSA Public-Key Cryptosystem. Available at: http://faculty.cs.tamu.edu/klappi/alg/rsa.pdf [Accessed 18 May 2017].
22. Math.fsu. 2011. Integers and Algorithms. Available at: https://www.math.fsu.edu/~pkirby/mad2104/SlideShow/s5_2.pdf [Accessed 23 May 2017].
23. Michael Calderbank. 2007. The RSA Cryptosystem: History, Algorithm, Primes. Available at:

http://www.math.uchicago.edu/~may/VIGRE/VIGRE2007/REUPapers/FINALAPP/Calderbank.pdf [Accessed 23 May 2017].
24. A. Menezes. 1996. Handbook of Applied Cryptography. Available at: http://cacr.uwaterloo.ca/hac/about/chap8.pdf [Accessed 24 May 2017].
25. Wikipedia. 2017. Fermat's Little Theorem. Available at: https://en.wikipedia.org/wiki/Fermat's_little_theorem [Accessed 28 July 2017].
26. Wikipedia. 2017. Euler's Totient Function. Available at: https://en.wikipedia.org/wiki/Euler%27s_totient_function [Accessed 28 July 2017].
27. Wikipedia. 2017. Binomial Coefficient of Prime. Available at: https://proofwiki.org/wiki/Binomial_Coefficient_of_Prime [Accessed 29 July 2017].
28. Wikipedia. 2017. Common Divisor Divides Integer Combination. Available at: https://proofwiki.org/wiki/Common_Divisor_Divides_Integer_Combination [Accessed 31 July 2017].