“The RSA Cryptographic Algorithm and its Proof”

Joel Richards

April 2018

Contents

Introduction

In this technologically-rich age, keeping data safe is becoming more and more crucial. With almost everything done in life being shared on the internet in some form, no byte of data can be considered safe while unencrypted. This is where the RSA encryption algorithm comes into effect. RSA is an encryption algorithm that depends on the difficulty of factoring large numbers, often being hundreds of digits in length. My research question is, “Why is the RSA encryption algorithm so secure, and how is it used in today’s society?”

In answering my research question, I will explain why the RSA algorithm is secure by nature, and why the RSA problem (the problem that arises when trying to break the algorithm with brute force) still exists to this day. In explaining this, I will show the steps necessary to securing data using the RSA encryption algorithm and will prove the mathematics behind the algorithm itself. I will describe best uses for the algorithm, along with areas and applications in which RSA is not the best choice for keeping data safe from others.

Math is commonly seen as a discipline that has little applicative value outside of the mathematical field itself (Sigmon, 2002). The RSA algorithm, and mathematics-based data encryption as a whole, is an area that takes some of the fundamentals of mathematics and applies it in a way that is used in daily life, but in a way that is hidden from the average person. For this reason, data encryption is an area that could be taught to students to show that mathematics is something that does have real life applications outside of rudimentary arithmetic.

Basics of the RSA Algorithm

The RSA algorithm was developed in the late 1970's by Ron Rivest, Adi Shamir and Leonard Adleman. The name RSA came from the first letter of each of their last names (Ireland, RSA Algorithm, 2016).

RSA is an asymmetrical, or one-way trapdoor algorithm (The RSA Cryptosystem, n.d.). In this context, these terms mean that there are two keys, one of which is known publically, and the other is kept private. These keys are inverse in nature, and one key can reverse the effects of the other.

The public key consists of two numbers. One is the product of two prime numbers, called the modulus. The other is a value called the public key exponent. Typically, data is encrypted using this public key. The private key is what is used to decrypt the messages in a timely manner, and is made up of the same product of primes and a different secondary value. This second auxiliary value is called the private key exponent. The issue of performing a private key-based task (decrypting data) using only the public key is known as the "RSA Problem", which remains unsolved to this date (Rivest, 2003). Figure 1 outlines the process of encrypting and decrypting a message sent from one person to another.

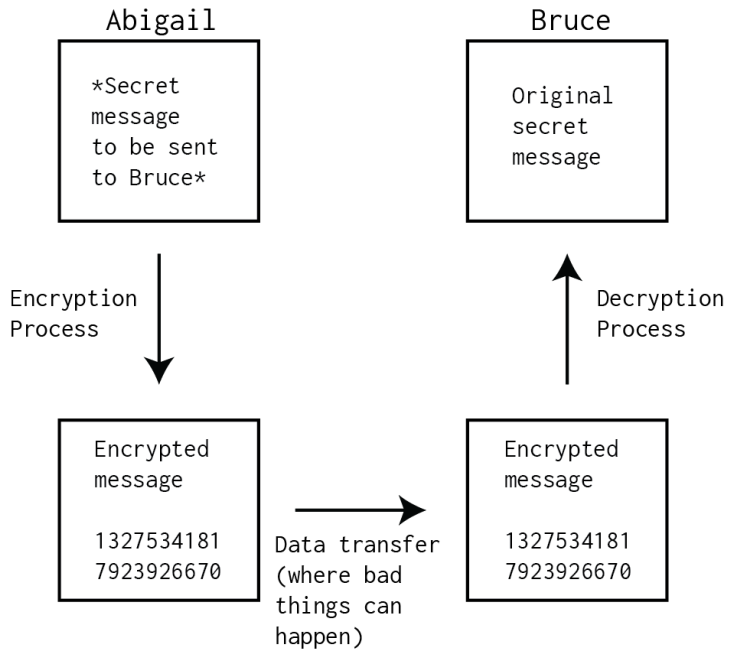

Figure 1. A message transfer.

The RSA Problem and RSA Assumptions

The RSA problem, by definition, is the problem that arises when given an encrypted piece of information and the key that is used to encrypt a message, without having the other parts necessary to decrypt the message (Rivest, 2003). To solve the problem, one must be able to reverse, or “break” the RSA algorithm in its entirety, something that to this day has not been efficiently done. To do this would mean following the path shown in Figure 1 in the opposite direction, which is nearly impossible due to the mathematics being done on the information in the encryption and decryption processes. It is theoretically possible to computationally break RSA encryption that is used today, when given unrealistic amounts of time, but due to the way computers process information, the process could take an average computer thousands of years to break a key that is considered safe by today’s standards.

The RSA Problem is commonly equated to integer factoring in terms of its difficulty (Rivest, 2003). If the main number used in an instance of the algorithm, called the modulus, is factored into the two primes it is made up of, then one can easily compute the decryption key by following the steps that are readily available as public knowledge. This process involves the use of mathematical formulas and theorems that will be shown later.

Development toward solving the RSA problem was encouraged by RSA Laboratories in the 1990s and 2000s (Jajodia, 2011). This was done by offering cash prizes to people or groups that successfully factored large semiprimes, numbers that are product of two prime numbers, also known as RSA numbers. Some of the smaller RSA numbers were factored the same year the challenge was issued, but there still remains countless that have not been factored. Rapid development in computing and awareness of large prime generation and factoring lead the challenge to be closed in 2007 (Jajodia, 2011).

The RSA Assumption, based on the RSA Problem, states that the problem and the algorithm itself are infinitely-difficult to break when the modulus is large enough and is truly randomly generated (Rivest, 2003). It also assumes that the message to be encrypted is random in nature due to being contained in the set $0 < m < (n-1),\ m;\ n \in \mathbb{Z}^+$, where m is the message being encrypted and n is the modulus (Rivest, 2003). As shown, the size of the modulus dictates the size of the number set for the message. The size of the set is a very important part of this problem, as a small number set would enable a stranger to be able to try all possible values of n and see if encrypting each individual possible value of m results in the encrypted message.

The RSA Strong Assumption, a variation of the standard Assumption, was first derived by Baric and Pfitzmann in 1997 (Rivest, 2003). It assumes that even if someone who is trying to break the RSA algorithm is able to choose one of the two keys, RSA remains difficult to solve. Being hard to solve in this context means that there is no efficient way to break the encryption; it must be done through trial and error (Kelly, 2009). Having one known key is actually common in practice, as the numbers 3, 5, 17, 257 and 65537 have been arbitrarily chosen by the encryption community to be used as public key exponents due to them being one greater than a power of two (Ireland, RSA Algorithm, 2016).

The Math behind RSA Encryption

RSA Keys

As previously mentioned, two separate keys are generated and are needed for the RSA algorithm to work successfully. One key is called the public key, while the other is called the private key. Each key is made up of a pair of integers that are transferred together.

For the purposes of simplicity and ease of understanding, the section entitled “Key Generation” will run through generating an example set of keys. The process will be shown using numbers that should never be used for encrypting data in the real world, due to their small size. However, using these small numbers is advantageous because they will be easy to deal with so that one can follow along with the process. Typical RSA keys that are considered to be safe are either 1024 or 2048 bits in length. Each bit can be

either a 1 or a 0, which means there are 2^{1024} possible 1024-bit keys, or 2^{2048} possible 2048-bit keys. To represent this value as a standard decimal number, it is necessary to follow steps to find the appropriate amount of digits.

Since binary values by nature have two possible values per digit, there are 2^{2048} possibilities. In a decimal number, there are 10 possibilities per digit (0, 1, 2…10). To convert binary to decimal, using logarithms is the most straight-forward approach. This can be done by doing the following:

$$2^{2048} = 10^n$$

$$2048\,(\log_{10} 2) = n(\log_{10} 10) \qquad \text{(Eliminating the base 2.)}$$

$$2048(\log_{10} 2) = n = 616.5 \approx 617 \qquad \text{(Solving for } n\text{, then rounding.)}$$

As shown, a 2048-bit RSA key is made up of 617 decimal digits, unlike the name seems to imply. This is still just as secure, as the added complexity of each digit makes up for the shorter length of characters (Thomas, 2013).

Key Generation

This section of the paper will outline the process for generating keys necessary for RSA encryption. All that is needed to begin is a pair of prime numbers. The numbers used in this example are orders of magnitude too small, and should never be used to encrypt any amount of data that should not be considered public. In this process, new mathematical functions and concepts are needed. They are called the modulus function and Euler's Totient function.

The modulus, otherwise known as the modulo function, is used to find the remainder after a division of one number by another. The modulo function is most commonly expressed in the form $a \, mod \, b = c$, with a, b and c representing the dividend, the divisor, and the remainder, respectively. For example, $14 \, mod \, 3 = 2$ because after dividing 3 into 14, the quotient is 4 and a remainder of 2 is left. The quotient has no effect in the modulus function (Weisstein, Modulus, n.d.). The modulus function is not related in any way to the absolute (modulus) value of a number.

Euler's totient function outputs the number of positive integers less than a number, n, that are coprime, or relatively prime to n. Being coprime to a number means that the two numbers do not share any common factors. Euler's totient function is represented by the Greek letter phi (ϕ), and is represented as $\phi(n)$ (Weisstein, Totient Function, n.d.). For example, $\phi(16) = 8$ because 1, 3, 5, 7, 9, 11, 13 and 15 all do not share a common factor with 16. These coprime numbers are called totatives (Weisstein, Totient Function, n.d.). One important part about coprime numbers is that 1 is coprime to all numbers.

When calculating $\phi(n)$, if n is a prime number, then $\phi(n) = (n - 1)$. This is due to the nature of prime numbers, as they by definition do not have integer factors other than 1 and n. For example, $\phi(7) = 6$ because 1, 2, 3, 4, 5 and 6 do not share factors with 7 (Weisstein, Totient Function, n.d.).

The steps needed to generate RSA keys are as follows, using 7 and 19 as the primes needed:

Let $p = 7$ and $q = 19$	(Two prime numbers are needed to start.)
$n = pq$ $= (7)(19)$ $= 133$	(Calculate n, which will be used as the modulus.)
$\phi(n) = (p - 1)(q - 1)$ $= (6)(18)$ $= 108$	(Calculate the totient of n, $\phi(n)$ using Euler's Totient function.)
Let $e = 13$	(Find e, a number coprime to $\phi(n)$, and is contained in the set $1 \leq e \leq \phi(n)$; $e \in Z^+$. e is used as the public key exponent.)

Now that one of the two keys has been found for RSA, its inverse must be found as well. These two keys, when combined, must satisfy one certain relationship called a modular congruence. A modular congruence is a type of mathematical relationship that compares the remainders of divisions. Modular congruence relations use a congruence (≡) symbol instead of the traditional equivalence (=) symbol (Weisstein, Congruence, n.d.). They are most often shown in the form of $a \equiv b \, mod \, n$. In this example, it is stating that both a

and b output the same remainder after being divided by n. Another characteristic of the congruence is that $(a-b)\,mod\,n = 0$. This means that the difference between the two integers is always perfectly divisible by n (Weisstein, Congruence, n.d.). For example, $63 \equiv 133\,mod\,5$ because both 63 and 133 give the same remainder when divided by 5. Also, their difference, 70, is divisible by 5, thus having a remainder of 0 (Weisstein, Congruence, n.d.).

The congruence that must be satisfied by the two keys is $de \equiv 1\,mod\,\phi(n)$ (The RSA Cryptosystem, n.d.). With e being known, its inverse, d, can be calculated by using the Euclidean and Extended Euclidean algorithms. The totient of n is used as the modulus for the exponents instead of n itself because of Euler's theorem, which will be explained when showing the proof of RSA. If n was used instead of the totient, the two exponents would not cancel each other out, making the algorithm useless, as it would not be possible to decrypt any data.

These two algorithms are primarily used to find the greatest common divisor of two numbers, but can be adapted to find the modular multiplicative inverse, needed for RSA. The modular multiplicative inverse of a number, a, is another number that when multiplied by a, gives a remainder of 1 when being divided by a third number. The general form for this is $ax \equiv 1\,(mod\,m)$ (Hobson, Nichol, & Weisstein, n.d.).

The Euclidean algorithm is used to find the greatest common divisor (gcd) of two numbers, j and k. Each step of the algorithm divides k into j, and then outputs a remainder, r. Next, k and r shift to become the new j and k,

and this process is repeated until the remainder cannot be divided any further (Lynn, n.d.). The last non-zero remainder is the greatest common denominator of the two numbers (Lynn, n.d.). Figure 2 illustrates this process.

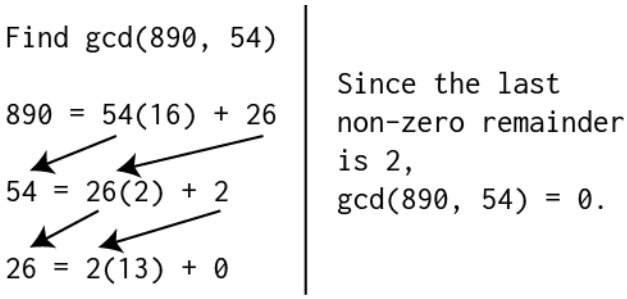

Figure 2. The Euclidean Algorithm.

The Euclidean algorithm is used here to verify that $\gcd(e, \phi(n)) = 1$, which must be true for the Extended Euclidean algorithm to work as intended. Running the Euclidean algorithm on 108 and 13 is as follows:

$108 = 13(8) + 4$	(Expressing 108 as a sum of a group of 13 with remainder.)
$13 = 4(3) + 1$	(Shifting the expression to express 13 as a sum of a group of 4 with a remainder. At this point you can use the Extended Euclidean algorithm because of the remainder of 1.)

Now the Extended Euclidean algorithm must be used. The Extended Euclidean algorithm builds upon the Euclidean algorithm in the opposite direction, working back up to a point where the greatest common divisor of a and b is expressed as $ax + by = \gcd(a, b)$, where x and y are integers yet to be determined (Guevara-Vasquez, 2011).

In terms of the use of the Extended Euclidean algorithm in RSA, finding the modular multiplicative inverse of the public exponent gives the private

exponent which is used when decrypting information. When using the Extended Euclidean algorithm with RSA, the a value is the modulus. Since $ax + by = \gcd(a,b)$, adding the use of a being a modulus enables the simplification of the expression to $by \equiv \gcd(a,b) \,(mod\ a)$, since the modulus function eliminates all multiples of a. Once again, the values for a and b in RSA are e and $\phi(n)$, respectively.

Running the Extended Euclidean algorithm on the numbers chosen goes as follows:

$1 = 13 + 4(-3)$	(Rearranging the final equation in the Euclidean algorithm process in terms of 1.)
$4 = 108 + 13(-8)$	(Rearranging the first equation in terms of 4.)
$1 = 13 + (108 + 13(-8))(-3)$	(Substituting the second equation into the first.)
$(13)(25) + (108)(-3) = 1$	(Collecting like terms and simplifying.)

Now that both algorithms have been run completely, and the equation is in the form $ex + \phi(n)y = \gcd(e,\phi(n))$, the step involving adding the use of modulus to the expression and making it a congruence once again can be completed. In this case, the modulus 108 eliminates all multiples of 108.

$(13)(25) + (108)(-3) \equiv 1 \,(mod\ 108)$	(The congruence that came from the final equation.)
$1 \equiv (13)(25) \,(mod\ 108)$	(Eliminating the multiples of 108 due to the modulus.)

Let $d = 25$ (d is equal to 25, since it is the value multiplied by e, 13.)

This congruence is true, as $[(13)(25) - 1]\, mod\, 108 = 0$.

(Ireland, The Euclidean Algorithm and the Extended Euclidean Algorithm, 2010).

To complete the key generation process, it is necessary to pair the integers needed for the public key (e, n) and the private key (d, n). The public key in this case is (13,133), and the private key is (25,133) (Ireland, RSA Algorithm, 2016).

Encrypting a Message

Let the message needed to be encrypted be the number 9. This means that $m = 9$. To encrypt m and get the cyphertext, c, one must do the following using the public key:

$c = m^e\, (mod\, n)$ (Raising the message to the public exponent.)

$c = 9^{13}\, (mod\, 133)$ (Filling in known values.)

At this point, a technique called modular exponentiation is used to compute c, as 9^{13} is not expressible on the calculators available. Modular exponentiation is a technique to simplify finding the remainder of a number when raised to a high power. The first part of it involves using the laws of exponents to break the large power into smaller, more manageable pieces, as shown below. The easiest way to do this is to break the exponent into a sum of powers of two. For this example, $13 = 8 + 4 + 1$. Once it has been broken down,

the modular arithmetic rule that states if $e \equiv f\ (mod\ m)$, then $e^n \equiv f^n\ (mod\ m)$, as well as the one that states if $n = (a + b + c \ldots)$, $e^n \equiv [(e^a\ mod\ n)(e^b\ mod\ n)(e^c\ mod\ n) \ldots]\ mod\ n$ can be used to compute the remainder as follows:

$c = m^e\ mod\ n$ (Raising the message to the public exponent.)

$c = 9^{13}\ mod\ 133$

$c = 9^{8+4+1}\ mod\ 133$

$c = [(9^8\ mod\ 133)(9^4\ mod\ 133)(9\ mod\ 133)]\ mod\ 133$

$c = 44$ (The encrypted message.)

(Ireland, RSA Algorithm, 2016; Art of Problem Solving)

The cyphertext can now be sent to the intended recipient, along with the private key so that they can decrypt the message. Anyone who does not have access to the private key has no easy way of getting the original input (9) from the cyphertext (44) and public key (13,133), also known as the RSA Problem.

Decrypting a Message

To decrypt the cyphertext, c, with access to the private key, (25,133) the recipient must do the following, once again using modular exponentiation:

$m = c^d \ mod \ n$ (Raising the cyphertext to the private exponent.)

$m = 44^{25} \ mod \ 133$

$m = 44^{16+8+1} \ mod \ 133$

$m = [(44^{16} \ mod \ 133)(44^8 \ mod \ 133)(44 \ mod \ 133)] \ mod \ 133$

$m = 9$ (The original message, unencrypted.)

(Ireland, RSA Algorithm, 2016; Art of Problem Solving)

As evident by the mathematics, the decryption process was successful using the private key. With both integers that make up the private key, it is possible to take the cyphertext and work back to the original message.

Proof of the RSA Algorithm

When working with modulus, it is possible for multiple expressions to equal each other. For example, $24 \equiv 3 \ (mod \ 7)$, and $31 \equiv 3 \ (mod \ 7)$. This is because as long as multiples of 7 are added, the same remainder will be the outcome. The set of numbers that always has a remainder of 3 after being divided by 7 can be expressed as $7k + 3; \ k \in \mathbb{Z}^+$ (Weisstein, Modulus, n.d.).

Since in RSA $de \equiv 1 \ mod \ \phi(n)$, it can be expressed as $de \equiv 1 + k\phi(n) \ (mod \ \phi(n))$, or $de \equiv 1 + k\phi(pq) \ (mod \ \phi(n))$. This can be done because any

multiple of $\phi(n)$ is ignored because of the modulus being applied. Therefore, one can keep adding multiples of $\phi(n)$ and always get the same result.

To put in the context of RSA, c^{ed} is being computed, which can be expressed in many ways. Using the laws of exponents and the way of expressing the combination of private and public exponents, the expression can be rearranged as follows:

$$de \equiv 1 + k\phi(n)\ (mod\ n)$$

$$c^{de} \equiv c^{1+k\phi(n)}\ (mod\ n)$$

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

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

At this point, it is necessary to use Euler's theorem, an expansion of Euler's totient function and Fermat's Little theorem. Fermat's Little theorem deals with properties of prime numbers, specifically in relation to the modulus function. It states that if a is any number and b is a prime number, $a^b \equiv a\ (mod\ b)$ (Weisstein, Fermat's Little Theorem, n.d.). Also, if a is not a multiple of b, then $a^{b-1} \equiv 1\ (mod\ b)$ (Weisstein, Fermat's Little Theorem, n.d.). Applying the work of Euler and his Totient function in the context of RSA, $a^{(p-1)(q-1)} \equiv 1\ (mod\ pq)$, or $a^{\phi(n)} \equiv 1\ (mod\ n)$ (Conrad). Now the equation can be manipulated as follows:

$c^{de} \equiv c^{1}\left(c^{\phi(n)}\right)^{k}\ (mod\ n)$ (The congruence as found previously.)

$c^{de} \equiv c^{1}1^{k}$ (Applying $c^{\phi(n)} \equiv 1\ (mod\ n)$ and simplifying.)

Since any power of 1 is equal to itself, and 1 expressed to any power is equal to 1, only the original cyphertext is left over, as shown:

$$c^{de} = c^{1+k\phi(n)} = c^1 1^k = c$$

This proves the RSA algorithm's key pair to be inverse with respect to each other, and confirms that they can decrypt information that was encrypted using the other key. This same process applies if the private key is used to encrypt, and the public key is used to decrypt (Steyn, 2012).

Real-Life Uses of RSA Encryption

The RSA algorithm is mostly useful in two processes: encryption and decryption, as well as identity verification. As shown throughout the exploration, the RSA algorithm can be used to encrypt data and have only the intended recipient(s) be able to decrypt it. The issue with using RSA to encrypt files and documents is that it, along with other asymmetric algorithms, are much slower than symmetric ones (Ireland, RSA Algorithm, 2016). There is no real mathematical reason why this is true other than the larger keys, but based on years of anecdotal evidence, that conclusion has been reached. The decryption process takes much longer than the encryption process, as the numbers being processed are much larger, meaning more modular exponentiation is needed.

The RSA algorithm is useful in identity verification because if one needs to prove where data came from a process called digital signing is used. Much like signing a document in real life on paper, a digital document or file can be signed by attaching a "signature", or hash, to a file. This hash can

then be encrypted using the RSA algorithm. The hash can be used to verify the integrity of the file, as it if has not been tampered with in any form, the recipient could then decrypt it and be left with the original hash. Having a signed file is very important in protecting end users from downloading compromised or malicious files and applications that could do damage to a computer or network (Ireland, RSA Algorithm, 2016).

The identification aspect of the RSA algorithm is also used in most internet traffic today. If browsing the web securely, the web address at which someone is visiting will include the prefix https://, which means that any data transfer between the web host and the recipient's local computer and network is encrypted. It is also used for other internet-based use cases, such as connecting mail servers to a mail client, or when a user inputs their personal information to a web form, such as credit card information or social security details. The technology behind this is called SSL, or Secure Socket Layer cryptography, which almost always uses the RSA encryption algorithm. SSL typically uses 1024-bit or 2048-bit RSA keys, as opposed to the simple ones calculated earlier. This is because certificates that are smaller than 2048 bits are generally considered not safe due to increased computational power. The average present-day computer would take approximately 14 billion years to crack a 2048-bit SSL certificate (Behind the Scenes of SSL Cryptogrphy, 2013). While this is more than strong enough to keep our data secure and safe, the issue with using RSA is that its keys are much longer than that of other cryptographic methods. For example, when using a symmetric cryptographic system, unlike RSA, keys are typically 128 or 256 bits in length. This cuts

down on the amount of time to transfer the key, as well as lessening the computational load put upon the computers involved.

As shown, the RSA algorithm, is widely used in today's technology-rich age, most of the time without even being seen by a standard user. Being an asymmetrical algorithm, RSA is not always the best choice, but having the freedom to choose what is best is always better when it comes to keeping data safe.

Conclusion

The RSA algorithm and the RSA Problem rely on the difficulty of factoring very large numbers. If the number sets of a RSA-breaking workload is sufficiently large, they will better secure the data being encrypted. RSA is a relatively simple algorithm in practice, but because it is a one-way trapdoor algorithm, breaking it in an efficient manner remains to be a major feat in both mathematics and computer science.

The mathematics community still has no straightforward way of reversing the work done by the encrypting function, despite numerous attempts. Even after encouragement of the development of methods to factoring large semiprimes, issued by RSA Laboratories, the algorithms still has held its ground (Jajodia, 2011).

Because of its resistance to the ease of factoring, the RSA algorithm is safe to use for encrypting private data. While it may not be as safe as other algorithms, with a long enough key, it will keep others away from the sensitive data for a sufficiently long enough time. The issue of much longer keys when compared to symmetrical encryption remains to be its main weakness, increasing encryption time, decryption time, as well as the load placed upon the computers and networks being used.

Further Investigation

To add to the research done in this paper, or to get a better grasp of the fundamentals of encryption, looking into simpler encryption algorithms and techniques, such as letter shifting and turning letters to numbers, would supply someone with a deeper understanding of the field.

To do more research into the topic of RSA-based encryption in particular, one could explore deeper into the contributions of Ron Rivest, Leonard Adleman and Adi Shamir in mathematics and computer science, so that one could get a clearer image of how RSA came to exist. Besides this, research in other encryption algorithms both from the time period from which RSA was created, as well as newer algorithms such as elliptic curve encryption, which uses a more geometric-based approach to encrypting data can be done.

General computer science research on computer-based encryption and algorithms would also prove to be useful in understanding the technological side of the topic, especially implementing it into the web and other applications. Other computer science topics, such as processor efficiency and how the hardware and software layers interact with each other, would give insight about the limitations of the algorithms.

References

Adleman, L. R. (n.d.). *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.*

Art of Problem Solving. (n.d.). *Modular Arithmetic/Introduction.* Retrieved from Art of Problem Solving: http://artofproblemsolving.com/wiki/index.php/Modular_arithmetic/Introduction

Behind the Scenes of SSL Cryptogrphy. (2013). Retrieved from DigiCert: https://www.digicert.com/ssl-cryptography.htm

Conrad, K. (n.d.). *Euler's Theorem.*

Guevara-Vasquez, F. (2011). *The Euclidean Algorithm and Multiplicative Inverses.* Retrieved from The University of Utah: https://www.math.utah.edu/~fguevara/ACCESS2013/Euclid.pdf

Hobson, N. &. (n.d.). *Modular Inverse*. Retrieved from Wolfram MathWorld: http://mathworld.wolfram.com/ModularInverse.html

Hobson, N., Nichol, R., & Weisstein, E. W. (n.d.). *Modular Inverse*. Retrieved from Wolfram MathWorld: http://mathworld.wolfram.com/ModularInverse.html

Ireland, D. (2010, August 14). *The Euclidean Algorithm and the Extended Euclidean Algorithm*. Retrieved from DI Management: https://www.di-mgt.com.au/euclidean.html

Ireland, D. (2016, December 2016). *RSA Algorithm*. Retrieved from DI Management: https://www.di-mgt.com.au/rsa_alg.html

Jajodia, S. &. (2011). *Encyclopedia of Cryptography and Security.* Springer.

Kelly, M. D. (2009, December 7). The RSA Algorithm: A Mathematical History of the Ubiquitous Cryptological Algorithm.

Khan Academy. (n.d.). *Modular Exponentiation*. Retrieved from Khan Academy: https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/modular-exponentiation

Lynn, B. (n.d.). *Euclid's Algorithm*. Retrieved from Stanford Applied Cryptography Group: https://crypto.stanford.edu/pbc/notes/numbertheory/euclid.html

Rivest, R. L. (2003, December 10). RSA Problem.

Sigmon, N. &. (2002, Winter). RSA ENCRYPTION WITH THE TI-82. *Mathematics & Computer Education, 36*(1), 13-23.

Steyn, B. (2012, September 3). *Why RSA Works: Three Fundamental Questions Answered*. Retrieved from Doctrina: http://doctrina.org/Why-RSA-Works-Three-Fundamental-Questions-Answered.html

The Extended Euclidean Algorithm. (n.d.). Retrieved from University of Colorado Mathematical and Statistical Sciences: http://www-math.ucdenver.edu/~wcherowi/courses/m5410/exeucalg.html

The RSA Cryptosystem. (n.d.). Retrieved from Mathematics at Dartmouth: https://math.dartmouth.edu/archive/m19w03/public_html/Section2-3

Thomas. (2013, June 23). *How long is a 2048-bit RSA key?* Retrieved from Stack Exchange: https://security.stackexchange.com/questions/37907/how-long-is-a-2048-bit-rsa-key

Weisstein, E. W. (n.d.). *Congruence*. Retrieved from Wolfram MathWorld: http://mathworld.wolfram.com/Congruence.html

Weisstein, E. W. (n.d.). *Fermat's Little Theorem*. Retrieved from Wolfram MathWorld: http://mathworld.wolfram.com/FermatsLittleTheorem.html

Weisstein, E. W. (n.d.). *Modulus*. Retrieved from Wolfram MathWorld.

Weisstein, E. W. (n.d.). *Totient Function*. Retrieved from Wolfram MathWorld: http://mathworld.wolfram.com/TotientFunction.html