RSA Cryptography and Number Theory

Chase Wiederstein

2022-10-31

1 Introduction

Cryptography has been around for centuries and is simply a form of communication. The most simple definition of cryptography is the art of creating and interpreting secret messages. To create a secret message is called "encrypting" and to interpret a secret message is called "decrypting."

2 Ancient History of Cryptography

2.1 Caesar's Cipher

One of the oldest and most famous methods of cryptography is known as the Caesar's Cipher used in the first century B.C. by Julius Caesar[1]. The Caesar's Cipher is quite simple and is known as a substitution cipher. It is just a shifting of the alphabet. First, assign the numbers 0-25 to the alphabet respectively. Then, the encrypter will choose a shift. Suppose the shift is $n = 2$. Each letter in the alphabet will shift 2 units to the right. A will become 2, B will become 3, C will become 4, and so on. Using modular arithmetic, this can be explained using $E_n(x) = x + n(mod 26)$ where x is the numerical value of the letter and n is the shift value.

After encrypting the message and sending it, decrypting it is just as simple. All the receiver needs is the key to unlocking the message, the shifting value, n. To decrpyt, you simply use the function $D_n(x) = x - n(mod 26)$.

Suppose I want to encrpyt the word "CRYPTOGRAPHY" to send to a fellow classmate using a shift of $n = 5$. First, I plug the numerical number value of each individual character in the string "CRYPTOGRAPHY" into the encrypting equation, $E_n(x) = n + x(mod 26)$.

Now, the encrypted message will read the character string of "HWDUYTLWFUMD." To decrypt, the fellow classmate will use the key of $n = 5$ along with the decrypting function $D_n(x) = x - n(mod 26)$ to reverse the shift and read "CRYPTOGRAPHY." However, if the message is intercepted, cracking the encrypted message would take no time at all even

Table 1: Caesar's Cipher 5 Unit Shift

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Caesar	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U

without the key. There are only 26 possible combinations of what the shift could be. Over time, this process of creating trickier encryption methods has become much more complex, yet the Caesar's Cipher is still a foundational model of how cryptography works.

2.2 Spartan Scytale

The Spartan Scytale is a type of transposition cipher dating back to fifth century B.C. used by the ancient Greeks in times of war.[2] A permutation cipher is one in which the ordering of the message is changed by a certain rule. The spartan scytale was created using a wooden shaft with a thin strip of leather wrapped around it. The sender would then write his message down onto the leather. Once he unwrapped the leather with the new message on it, the message would become encrypted. Once received, the receiver would then wrap the strip of leather around a wooden shaft of the same width to decrypt the message.

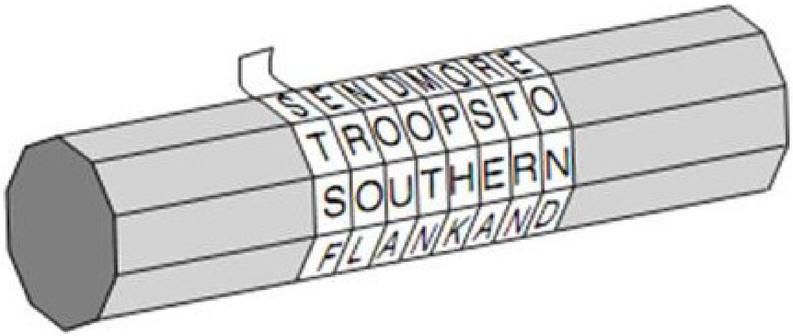

Figure 1: Spartan Scytale[2]

3 Contemporary Cryptography

3.1 Asymetric public-private Keys

Since the origination of secret messages, the challenge was for the sender and receiver of the message to have the same key for encoding and decoding a message. According to Auguste Kerckhoffs, one of the main principles of cryptography is the security of a cryptosystem should rely on the secrecy of a small parameter, the secret key.[1] The process of “key distribution” required the delivery of the key to the receiver of the message. If the receiver did not have the key value for the Caesar’s Cipher, they could not read what it said. In modern times, all countries rely heavily on the security of information via cryptography. If one can find your key, they can read your information. As trade between countries increased post World War II, it required payments to be made across borders and sometimes at great distances. The cost associated with the delivery of keys became prohibitive and banks, in particular, began to investigate how to make the process more efficient. The solution that had evaded discovery for centuries was first proposed by two Americans: Whitfield Diffie and Martin Hellman in 1976. Their idea was to have an “asymmetric public-private key.” This discovery is considered “to be the greatest cryptographic achievement since the invention of the monoalphabetic cipher”[3], just like Caesar’s Cipher, over two thousand years ago.”

3.2 Diffie-Hellman

Whitfield Diffie was born in 1944 and graduated from the Massachusetts Institute of Technology in 1965. He then took a series of jobs in computer-security roles. Diffie recognized that the key distribution problem was a riddle for the ages. Diffie considered the situation where two people on the internet need to keep a message (e.g. e-mail) or a transaction private. The two parties need to share a key, but exchanging the key securely became a problem. The volume of messages and transactions made the physical delivery of the key impractical. Without secure encryption, communication would be impossible because messages and credit card information could be discovered.

At a talk at IBM, Diffie learned that a Stanford University professor was also interested in the problem, Martin Hellman. In 1974, he received a call from Whitfield Diffie and agreed to a half-hour meeting at his office. The half-hour meeting turned into hours with Diffie leaving Hellman’s home at midnight. The two formed a partnership with another cryptography fanatic, Professor Ralph Merkle, to address the key distribution problem.

3.3 A Simple Example

Examples from cryptography involve a traditional cast of characters: Alice, Bob, and Eve. Alice (A) and Bob (B) wish to exchange a secure message Eve wishes to eavesdrop and steal the message. Before Diffie-Hellman, Alice and Bob would have to meet to exchange keys. If they were exchanging lots of messages or wanted to change encryption methods, they might have to meet weekly. The system breaks down when one of them gets sick. Couriers could help but they also add complexity and potential insecurity to the process.

Let's envision a situation where Alice puts her message in a box and then padlocks it with a key. Alice keeps the key. When Bob receives the message, he has no way to open the padlock. Bob needs the key that Alice retained. This is essentially the age-old problem but restated. Diffie-Hellman changed the way the problem was conceived. Instead, they described this situation. Alice locks the message in a box with her lock, retaining the key, and sends the box to Bob. Bob then takes the box, adds his lock with his key, and returns the box to Alice. Alice receives the box but it now has two locks. She takes her lock off the box, leaving only Bob's lock. She then sends the box back to Bob and Bob receives the box with only his lock. He removes the lock with the key he retained and can finally read the message. No padlock key was exchanged.

Diffie-Hellman's contribution was to propose a theoretical method whereby an encrypted message could be exchanged without the need for exchanging the keys. The use of keys is asymmetric in that Alice's padlock key (the key that encrypted the message) is different from Bob's padlock key (the key that decrypted the message). The discovery was "revolutionary." Their research left unresolved how the encryption was to occur. They needed a mathematical function that was easy to encrypt but impossible for an eavesdropper to decipher.

3.4 Trapdoor Encryption

To turn the concept of asymmetric ciphers into a practical invention, a mathematician needed to create a function that acted as a lock. Some mathematical functions are considered to be two-way functions in that they are easy to do and undo. They act like a light switch where the effort to turn the switch on is the same as turning it off. For example, doubling a number takes the same effort as dividing a number by two. Diffe and Hellman were not interested in two-way functions, but instead, were looking for a one-way or "trapdoor" function. A trapdoor function is one where it was possible to apply a function to a number but much harder to reverse. These kinds of functions are sometimes described as "Humpty-Dumpty" functions because it's easy to get Humpty-Dumpty up on the wall, but hard to put him back together after he falls. Diffe-Hellman-Merkle would attempt to find a solution in modular arithmetic, but the one-way mathematical encryption would ultimately be solved by another group of mathematicians.

A workable trapdoor mathematical function would be discovered by Ron Rivest, Adi Shamir, and Leonard Adleman. (The initials of their surnames form the acronym "RSA".) The three researchers worked together at an MIT Laboratory for Computer Science and applied different approaches to the problem for a year. One evening after some Manischewitz wine, Rivest stumbled upon a possible approach and wrote the equivalent of a scientific paper overnight. He showed the solution to his partners and, after a friendly dispute on the order of authors, they submitted the paper. Adelman thought at the time that the paper would be "the least interesting paper" that he ever authored. The system "went on to become the most influential cipher in modern cryptography." [2]

RSA was first announced in August 1977 in *Scientific American.* The article, “A New Kind of Cipher that Would Take Millions of Years to Break,” was authored by Martin Gardner. Gardner issued a challenge to the public to decode a message and issued the public key. The message was decoded seventeen years later and required six hundred volunteers.

3.5 Dual Discoveries

Asymmetric public key encryption was discovered earlier in Great Britain by James Ellis who worked for the Government Communications Headquarters, but knowledge of the discovery was delayed because of national security. Ellis was a pack rat of academic articles and discovered the idea in a trove of old telecommunications articles. A paper proposed the idea of adding noise to the telephone line to make the signal unintelligible. The recipient would then remove the noise to hear the message. Lacking the mathematical background to create a padlock, Ellis’ discovery languished until a new mathematician joined the team. Clifford Cocks, a recent graduate of Cambridge with a specialization in number theory, discovered a one-way function. In recalling his epiphany, he said “it was natural to think about one-way functions, something you could do but not undo. Prime numbers and factoring was a natural candidate.” From start to finish, Cocks estimated that it took him no more than half an hour to solve. [3]

4 The RSA Algorithm

Cryptography has become more important as we communicate globally with the internet. The sharing of credit card numbers, social security numbers, names, etc. online is an everyday occurrence and security for our information is a must. RSA cryptography is one solution protecting us from those who wish to steal our information. The RSA algorithm can be broken up into 3 sections: key generation, encryption, and decryption.

Step 1: Find two large primes, p_0 and p_1, such that their product, n, is of the required bit length. The length of n will usually be of the standard 1024, 2048, or 3027 bits.[4]

Step 2: Calculate $n = p_0 \cdot p_1$ and use Euler’s Totient Function to calculate $\phi(n) = (p_0 - 1) \cdot (p_1 - 1)$.

Step 3: Choose some integer e such that $(e, \phi(n)) = 1$ where $1 < e < \phi(n)$.

Step 4: Calculate d, the multiplicative inverse of $e(mod\, \phi(n))$. i.e. $d \equiv e^{-1}(mod\, \phi(n))$

Step 5: Key generation is finished. The “public key” is now (n, e) and the “private key” is (n, d).

Step 6: Encrypt the message using $C \equiv M^e(mod\, n)$ to calculate the least residue $(mod\, n)$. C stands for the ciphertext and M stands for the original message.

Step 7: Decrypt the ciphertext using $M \equiv C^d(mod\, n)$ to calculate the least residue $(mod\, n)$.

5 Finding Large Primes(prob of prime 1/4, etc. ,)

Calculating large primes for p_0 and p_1 is one of the “easier” parts of the key generation algorithm. The goal is to calculate n, the product of p_0 and p_1. p_0 and p_1 are very large: at least 512 digits, but 1024 digits is considered safe.[5]

A common method of primality testing relies on the “Rabin-Miller primality test.” However, the Rabin-Miller primality is a *probability* test. The test calculates numbers that are high probable candidates for being prime.The Rabin-Miller test is one of the most efficient algorithms for calculating large primes for RSA cryptography and is used widely. Another popular primality test is the “Fermat primality test” relying upon Fermat’s Little Theorem.

Lemma 5.1:

If a and m are relatively prime integers, then the least residues of $a, 2a, 3a, ..., (m-1)a(modm)$ are equivalent to $1, 2, 3, ..., (m-1)(mod\, m)$, up to reordering.

Theorem 1: Fermat’s Little Theorem

If p is a prime and a is an integer such that $p \nmid a$, then $a^{p-1} \equiv 1(modp)$.

5.0.0.0.1 Proof of Fermat’s Little Theorem: {theorem} Let p be a prime and $a \in \mathbb{Z}$ such that $p \nmid a$. Thus a and p are relatively prime, i.e. $(a, p) = 1$. Thus, by the Lemma 1 $a, 2a, 3a, ..., (p-1)a(mod\, p)$ have the least residues of $1, 2, 3, ..., p-1(mod\, p)$ up to reordering.

Hence,

$$\begin{aligned} a \cdot 2a \cdot 3a \cdots (p-1) \cdot a &\equiv 1 \cdot 2 \cdot 3 \cdots (p-1)(modp) \\ &\equiv (p-1)! \\ &\equiv -1(mod\, p) \quad \text{by Wilson's Theorem.} \end{aligned}$$

Note that $a \cdot 2a \cdot 3a...(p-1) \cdot a = a^{p-1} \cdot (p-1)!$. Thus, $a^{p-1} \cdot (p-1)! \equiv a^{p-1} \cdot (-1) \equiv a \cdot 2a \cdot 3a \cdots (p-1)a \equiv (-1)(mod\, p)$.

By definition of multiplicative inverse, $(-1)^{-1} \equiv (-1)(modp) \equiv (p-1)(modp)$. Hence,

$$\begin{aligned} 1 &\equiv (-1) \cdot (-1) \\ &\equiv (-1) \cdot a^{p-1} \cdot (p-1)! \\ &\equiv (-1) \cdot a^{p-1} \cdot (-1) \\ &\equiv a^{p-1}(mod\, p). \end{aligned}$$

5.0.1 Theorem 2: Fermat Primality Test

An odd positive integer n is composite if there exists a positive integer a such that $\gcd(a, n) = 1$ and $a^{n-1} \not\equiv 1(mod\, n)$. [6]

5.0.2 Definition 1: Fermat Pseudoprime

A Fermat pseudoprime to a base a is a composite number n such that $a^{n-1} \equiv 1(modn)$. [7]

5.0.3 Definition 2: Strong Pseudoprime

A strong pseudoprime to a base a is an odd composite number n with $n-1 = d \cdot 2^s$ where d is odd for which either $a^d \equiv 1(mod\, n)$ or $a^{d \cdot 2^r} \equiv -1(mod\, n)$ for some $r = 0, 1, 2, ..., s-1$. [8]

5.0.4 Lemma 2:

If p is a prime number, then the complete solution set to $x^2 \equiv 1(mod\, p)$ is $x \equiv \pm 1(mod\, p)$.

5.0.5 Lemma 3:

Let p be a prime such that $p = 2^s q + 1$ where q is odd and let a be an integer that is nonzero modulo p. Exactly one of the following holds:
(i) $a^q \equiv 1(mod\, p)$;
(ii) $a^{2^i q} \equiv -1(mod\, p)$, for some $0 \leq i < s$. [9]

5.0.6 Theorem 3:

Let n be an odd prime, and write n in the form of $1 + 2^s \cdot d$ where d is an odd. Then the Rabin-Miller sequence

$$a^d, a^{2d}, a^{4d}, ..., a^{2^{s-1}d}, a^{2^s d}(mod\, n)$$

ends with 1. If a^d is not congruent to $1(mod\, n)$, then the value directly preceding the first appearance of 1 is $n-1$. [6]

5.0.7 The Rabin-Miller Algorithm

Given an odd integer n, find an integer s so that $n - 1 = 2^s q$ for some odd q. Then,
1. Pick a random integer a such that $1 \leq a \leq n-1$.
2. If $a^q \equiv 1(mod\, n)$, then n could be prime and n passes the test.
3. Otherwise, for $i = 0, 1, ..., s-1$, see if $a^{2^i q} \equiv -1(mod\, n)$. If so, then n could be prime and n passes the test.
4. Otherwise, n is composite. [10]

5.0.7.1 Rabin-Miller Test Example: Suppose we wish to determine whether $n = 645$ is prime. Then, $n - 1 = 644 = 2^2 \cdot 161$.
1. Let $a = 2$.
2. $2^{161} \equiv 257(mod\, 645)$. Since 2^{161} is not congruent to $1(mod\, n)$, we move on to step 3.
3.

$$257^2 \equiv 259(mod\, 645)$$
$$259^2 \equiv 1(mod\, 645)$$

But 257^2 and 259^2 are not congruent to $-1(mod\, 645)$. Thus, n is composite.

6 2.2 Using Euler's Totient Function

Once two primes are found using a primality test, multiplying them together yields n, the modulus used in each of the key pairs. The reason for doing this is to create the "trapdoor function." A trapdoor function that multiplies these two primes takes little time, but going backwards (the inverse) to find the original primes takes much longer.

For example, say the two prime numbers I picked were 257 and 331. Then, computing $n = (257)(331) = 85,067$ takes no time at all. Now consider going the other direction. If I gave you the product of 85,067 to begin with and asked you to find the two primes I multiplied together to get this number, then this process becomes much more time consuming. Going one direction was easy, but going the other direction was much more difficult. The same difficulty goes for computers. This is the beauty of RSA encryption. Algorithms can be ranked by their efficiency. In the example above, 257 and 331 are incredibly small primes relative to the primes used for RSA encryption. Multiplying two of these large numbers still takes a small amount of time and computational resources. However, factoring n requires a much more complex algorithm. In fact, recovering a prime number of 1024 bits would require a year's worth of work on a $10 million machine, and recovering a prime number of 2048 bits would require several billion times more work.[11] However, this process is theoretically possible due to the Unique Factorization Theorem below. The prime factorization of 1024 bit number (308 digits) may still be an incredibly long factorization.

6.0.1 Theorem 4: Unique Factorization Theorem

Every natural number $n > 1$ can be uniquely expressed as a product of primes. i.e. $n = (p_1^{e_1})(p_2^{e_2})(p_3^{e_3})...(p_k^{e_k})$ for distinct primes p_i and positive integers e_i with $1 \leq i \leq k$.

6.0.2 Theorem 5: Correspondence Theorem for Divisors

Let $a, b \in \mathbb{Z}^+$ such that a and b are relatively prime. Then every positive divisor of $a \cdot b$ can be uniquely expressed in the form $d \cdot e$ where d is a positive divisor of a and e is a positive divisor of b. Any number of the form $d \cdot e$ such that $d|a$, $d > 0$, $e|b$, and $e > 0$ is also a divisor of $a \cdot b$.

6.0.3 Euler's Totient Function

If $n \in \mathbb{Z}^+$, $\phi(n)$ is defined to be the number of positive integers less than or equal to n and relatively prime to n.[10]

6.0.4 Theorem 6:

Let n have the prime power factorization $(p_1^{e_1})(p_2^{e_2})(p_3^{e_3})...(p_k^{e_k})$ for distinct primes p_i and positive integers e_i with $1 \leq i \leq k$. Then,

$$\phi(n) = \prod_{i=1}^{k} \phi(p_i^{e_i}) = \prod_{i=1}^{k} p_i^{e_i-1}(p_i - 1).$$

6.0.5 Lemma 4

For a prime p and a positive integer k,

$$\phi(p^k) = p^{k-1} \cdot (p-1).$$

6.0.5.1 Proof of Lemma 4: Let p be a prime and let $k \in \mathbb{Z}^+$. There are p^k positive integers that are less than or equal to p^k. Of these positive integers, the only numbers not relatively prime to p^k are the multiples of p. i.e. $(1)p, (2)p, (3)p, ..., (p^{k-1})p$. Thus, there are p^{k-1} positive integers less than or equal to p^k that are not relatively prime to p^k. Therefore,

$$\begin{aligned}\phi(p^k) &= p^k - p^{k-1}\\ &= p^{k-1}(p-1).\end{aligned}$$

6.0.6 Corollary 1:

The Euler's totient function is multiplicative.

6.0.6.1 Proof of Corollary 1: Suppose $n \in \mathbb{Z}^+$. Then n has a unique prime power factorization by the Unique Factorization Theorem. Let $n = (p_1^{e_1})(p_2^{e_2})(p_3^{e_3})...(p_k^{e_k})$ for distinct primes p_i and positive integers e_i with $1 \leq i \leq k$. Let $a = (p_1^{e_1})(p_2^{e_2})(p_3^{e_3})...(p_l^{e_l})$ and $b = (p_{l+1}^{e_{l+1}})(p_{l+2}^{e_{l+2}})...(p_k^{e_k})$ where $l \in \mathbb{Z}$ and $1 \leq l < k$. Hence, $n = ab$ and $(a, b) = 1$. Then

$\phi(a) = \prod_{i=1}^{l} p_i^{e_i-1}(p_i - 1)$ and $\phi(b) = \prod_{i=l+1}^{k} p_i^{e_i-1}(p_i - 1)$ by Theorem. Thus,

$$\begin{aligned}
\phi(a) \cdot \phi(b) &= \prod_{i=1}^{l} p_i^{e_i-1}(p_i - 1) \cdot \prod_{i=l+1}^{k} p_i^{e_i-1}(p_i - 1) \\
&= \prod_{i=1}^{l} p_i^{e_i}(1 - \frac{1}{p_i}) \cdot \prod_{i=l+1}^{k} p_i^{e_i}(1 - \frac{1}{p_i}) \\
&= a \prod_{i=1}^{l}(1 - \frac{1}{p_i}) \cdot b \prod_{i=l+1}^{k}(1 - \frac{1}{p_i}) \\
&= ab \prod_{i=1}^{k}(1 - \frac{1}{p_i}) \\
&= n \prod_{i=1}^{k}(1 - \frac{1}{p_i}) \\
&= \prod_{i=1}^{k} p_i^{e_i}(1 - \frac{1}{p_i}) \\
&= \prod_{i=1}^{k} p_i^{e_i-1}(p_i - 1) \\
&= \phi(n) \\
&= \phi(ab).
\end{aligned}$$

Therefore, Euler's totient function is multiplicative by definition.

For the RSA algorithm, we are asked to calculate $\phi(n)$ where n is the product of two primes p_0 and p_1. Since Euler's totient function is multiplicative, $\phi(p_0 \cdot p_1) = \phi(p_0) \cdot \phi(p_1)$. Also, p_0 and p_1 have the unique prime power factorization of themselves. i.e. $p_0 = p_0^1$ and $p_1 = p_1^1$. Thus, $\phi(p_0) = (p_0 - 1)$ and $\phi(p_1) = (p_1 - 1)$. Therefore, $\phi(n) = (p_0 - 1) \cdot (p_1 - 1)$.

7 2.3 Encryption and Decryption

After calculating $\phi(n)$ for the RSA algorithm, the next step is to find an some integer e between 1 and $\phi(n)$ such that e and $\phi(n)$ are relatively prime. e is known as the "encryption exponent" that is used for the sender generating public key. The public key is the pair of positive integers (e, n). Once e is found, the sender then uses the congruence $C \equiv M^e (modn)$ to generate the ciphertext, C. Ciphertext is the encrypted messaged that hides the true meaning from anyone, other than the receiver, wishing to read the message.

In order for the receiver to decrypt the message, the receiver needs to have the private key, the pair of positive integers (d, n). d can be derived by the receiver because d is the multiplicative inverse of e. i.e. $d \equiv e^{-1} (modn)$. With the private key, the receiver can then use the congruence $C \equiv M^d (modn)$.

8 RSA Example

One method of implementing the RSA algorithm would be using the "ASCII keyboard," the American Standard Code for Information Interchange. In this example, we will just focus on A-Z with their prescribed decimal values on the ASCII keyboard.

Table 2: ASCII: A-Z

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

Suppose I wish to send the message "RSA" to a classmate. First, it is important to represent each value of "RSA" by a number between 0 and $n-1$. [12] For this example, we will use a message for each character. After conversion, R will now be $M_1 = 82$, S will be $M_2 = 83$, and A will be $M_3 = 65$.

First, I will select two primes (much smaller than the standard RSA). Suppose I pick $p_0 = 7$ and $p_1 = 13$.

Next, calculating $n = p_0 \cdot p_1$ yields $n = 7 \cdot 13 = 91$. Then, with Euler's Totient Function, $\phi(91) = (6) \cdot (12) = 72$.

Now, choose some integer e such that $(e, \phi(n)) = 1$ where $1 < e < \phi(n)$. Supposed I pick $e = 23$.

Next, I create the ciphertext with $C \equiv M^e (mod n)$.

$$C_1 \equiv 82^{23} \equiv 10 (mod 91)$$
$$C_2 \equiv 83^{23} \equiv 34 (mod 91)$$
$$C_3 \equiv 65^{23} \equiv 39 (mod 91)$$

Where C_1, C_2, and C_3 are the least residues (mod 91) of their respective messages. These are the messages that will be sent so that no one else can read what they say.

Once received, the receiver will then decipher the messages with their private key (n, d) using the congruence $M \equiv C^d (mod n)$. First, to find d, the receiver needs to calculate the multiplicative inverse of $23 (mod 72)$. Since $23 \cdot 47 \equiv 1 (mod 72)$, 47 is the multiplicative inverse.

Then, to decipher, the receiver will use the congruence $M \equiv C^d (mod n)$.

$$M_1 \equiv 10^{47} \equiv 82 (mod 91)$$
$$M_2 \equiv 34^{47} \equiv 83 (mod 91)$$
$$M_3 \equiv 39^{47} \equiv 65 (mod 91)$$

Done! Now we have arrived back to our original numbers with their respective character values of RSA.

9 Next Generation in Encryption Standards

Because of advancements in computer technology, the next generation of encryption standards is currently under consideration by the National Institute of Standards and Technology (“NIST”).

9.1 NIST

The NIST is the premier, standard-setting organization in the U.S. According to its website, the mission of the NIST is to “promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life.” Originally founded in 1901, the NIST is a part of the U.S. Department of Commerce and has led the promulgation of encryption standards. NIST technology secures tablets, cellphones, and ATMs; encrypts international transactions on the web; and protects US federal information including those with national security implications.

NIST’s first encryption standard was known as the Data Encryption Standard (DES) and was adopted in 1977. The DES standard, its definition, and processing standards were withdrawn in 2005. The NIST published its second standard, the Advanced Encryption Standard (AES), in 1997 after a 4 1/2-year process. The AES was officially adopted in 2001. The encryption algorithm is known as the “Rijndael” algorithm and was developed by Belgium scientists Joan Daemen and Vincent Rijmen. (The algorithm is pronounced as “rain doll”). The Rijndael algorithm was picked as the best out of the 15 originally submitted algorithms. NIST is in the process of adopting its third standard because of advances in quantum computing.

9.2 Quantum Computing

Quantum computing is the next generation in computer technology and harnesses the power of quantum mechanics. According to wikipedia, “while quantum computers provide no additional advantages over classical computers in terms of computability, quantum algorithms for certain problems have significantly lower time complexities than corresponding known classical algorithms. Notably, quantum computers are believed to be able to quickly solve certain problems that no classical computer could solve in any feasible amount of time—a feat known as”quantum supremacy.”

NIST’s annual report, “2021 Cybersecurity and Privacy Annual Report”, described the state of quantum computing and cryptography as follows:

> [T]here has been steady progress in building quantum computers – machines that exploit quantum mechanical phenomena to solve problems that are difficult or intractable for conventional computers. When the capacity to build large-scale quantum computers exists, they will be able to break many of the public-key cryptosystems currently in use. This weakness would seriously compromise the confidentiality and integrity of digital communications online and elsewhere.

An ominous prediction was offered within the report that current public key encryption systems will be obsolete within the next 20 years.

McKinsey and Company releases a quarterly report entitled, “Quantum Technology Monitor.” In June 2022, McKinsey documented $1.24 billion in private investment start-ups and $1.9 billion of announced government funding for the development of quantum technology. China increased its quantum technology patent activity and originated more than half globally. “Public and private funding continues to **skyrocket** around the world, with North America still investing the most.”

9.3 Standardized Algorithm

The next generation of encryption standards is being considered by NIST and deemed a priority because of advances in quantum computing. To meet this new challenge, NIST began by evaluating 15 new algorithms. The algorithms were evaluated in rounds with the third round victors being announced in 2022 and a fourth round to be held for the remaining algorithms. According to the latest news release on July 5, 2022, NIST will recommend for most use cases the CRYSTALS-KYBER for key establishment and CRYSTALS-Dilithium for digital signatures. The current status of the standard is maintained on the NIST Post-Quantum Cryptography Project website.

In describing CRYSTALS-KYBER’s performance, the NIST declared the public and cipher key size “in the order of a thousand bytes which should be acceptable for most applications.” The conclusion was that “the security of KYBER has been thoroughly analyzed and is based on a strong framework of results in lattice-based cryptography. KYBER has excellent performance overall in software, hardware, and many hybrid settings.”

10 Conclusion(Note: Still need)

11 Notes:

1. More explanation of finding large primes
2. Ask about a better flow in “Euler’s Totient Function.” Include Euler’s theorem, but show it’s similar to Fermat’s Little Theorem. Also, need another word/title for “Lemma.”
3. Include definitions page for smaller elements like multiplicative inverse, gcd, relatively prime, etc.
4. (Ask) Could need to add Chinese Remainder Theorem, Euclidean Algorithm, etc.

12 References

1. Lefton P (1991) Number Theory and Public-Key Cryptography. *The Mathematics Teacher* 84: 54–63.
2. Singh S (2011) The code book: The science of secrecy from ancient Egypt to quantum cryptography, Anchor.
3. Singh S (1999) The code book, Doubleday New York.
4. National Institute of Standards and Technology *NIST*.
5. Steyn B How RSA Works With Examples, *Doctrina-How RSA Works With Examples.*
6. McGregor-Dorsey ZS (1999) Methods of primality testing. *MIT Undergraduate Journal of Mathematics* 1: 133–141.
7. Weisstein EW Fermat Pseudoprime.
8. Weisstein EW Strong Pseudoprime.
9. Sutherland A (2017) Lecture Notes 12 Primality Proving: 18.783 Elliptic Curves.
10. Number Theory - Primality Tests.
11. Jahan I, Asif M, Rozario LJ (2015) Improved RSA cryptosystem based on the study of number theory and public key cryptosystems. *American Journal of Engineering Research (AJER)* 4: 143–149.
12. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21: 120–126.

13 Appendix

Table 3: First 500 Primes

2	3	5	7	11	13	17	19	23	29
31	37	41	43	47	53	59	61	67	71
73	79	83	89	97	101	103	107	109	113
127	131	137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223	227	229
233	239	241	251	257	263	269	271	277	281
283	293	307	311	313	317	331	337	347	349
353	359	367	373	379	383	389	397	401	409
419	421	431	433	439	443	449	457	461	463
467	479	487	491	499	503	509	521	523	541
547	557	563	569	571	577	587	593	599	601
607	613	617	619	631	641	643	647	653	659
661	673	677	683	691	701	709	719	727	733
739	743	751	757	761	769	773	787	797	809
811	821	823	827	829	839	853	857	859	863
877	881	883	887	907	911	919	929	937	941
947	953	967	971	977	983	991	997	1009	1013
1019	1021	1031	1033	1039	1049	1051	1061	1063	1069
1087	1091	1093	1097	1103	1109	1117	1123	1129	1151
1153	1163	1171	1181	1187	1193	1201	1213	1217	1223
1229	1231	1237	1249	1259	1277	1279	1283	1289	1291
1297	1301	1303	1307	1319	1321	1327	1361	1367	1373
1381	1399	1409	1423	1427	1429	1433	1439	1447	1451
1453	1459	1471	1481	1483	1487	1489	1493	1499	1511
1523	1531	1543	1549	1553	1559	1567	1571	1579	1583
1597	1601	1607	1609	1613	1619	1621	1627	1637	1657
1663	1667	1669	1693	1697	1699	1709	1721	1723	1733
1741	1747	1753	1759	1777	1783	1787	1789	1801	1811
1823	1831	1847	1861	1867	1871	1873	1877	1879	1889
1901	1907	1913	1931	1933	1949	1951	1973	1979	1987
1993	1997	1999	2003	2011	2017	2027	2029	2039	2053
2063	2069	2081	2083	2087	2089	2099	2111	2113	2129
2131	2137	2141	2143	2153	2161	2179	2203	2207	2213
2221	2237	2239	2243	2251	2267	2269	2273	2281	2287
2293	2297	2309	2311	2333	2339	2341	2347	2351	2357
2371	2377	2381	2383	2389	2393	2399	2411	2417	2423
2437	2441	2447	2459	2467	2473	2477	2503	2521	2531

Table 3: First 500 Primes *(continued)*

2539	2543	2549	2551	2557	2579	2591	2593	2609	2617
2621	2633	2647	2657	2659	2663	2671	2677	2683	2687
2689	2693	2699	2707	2711	2713	2719	2729	2731	2741
2749	2753	2767	2777	2789	2791	2797	2801	2803	2819
2833	2837	2843	2851	2857	2861	2879	2887	2897	2903
2909	2917	2927	2939	2953	2957	2963	2969	2971	2999
3001	3011	3019	3023	3037	3041	3049	3061	3067	3079
3083	3089	3109	3119	3121	3137	3163	3167	3169	3181
3187	3191	3203	3209	3217	3221	3229	3251	3253	3257
3259	3271	3299	3301	3307	3313	3319	3323	3329	3331
3343	3347	3359	3361	3371	3373	3389	3391	3407	3413
3433	3449	3457	3461	3463	3467	3469	3491	3499	3511
3517	3527	3529	3533	3539	3541	3547	3557	3559	3571

14 Appendix

Table 4: US-ASCII Printable Characters

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32	40	20	100000	NA	&#32;	NA	Space
33	41	21	100001	!	&#33;	NA	Exclamation mark
34	42	22	100010	"	&#34;	&quot;	Double quotes (or speech ma
35	43	23	100011	#	&#35;	NA	Number
36	44	24	100100	$	&#36;	NA	Dollar
37	45	25	100101	%	&#37;	NA	Per cent sign
38	46	26	100110	&	&#38;	&amp;	Ampersand
39	47	27	100111	’	&#39;	NA	Single quote
40	50	28	101000	(	&#40;	NA	Open parenthesis (or open b
41	51	29	101001	)	&#41;	NA	Close parenthesis (or close b
42	52	2A	101010	*	&#42;	NA	Asterisk
43	53	2B	101011	+	&#43;	NA	Plus
44	54	2C	101100	,	&#44;	NA	Comma
45	55	2D	101101	-	&#45;	NA	Hyphen
46	56	2E	101110	.	&#46;	NA	Period, dot or full stop
47	57	2F	101111	/	&#47;	NA	Slash or divide
48	60	30	110000	0	&#48;	NA	Zero
49	61	31	110001	1	&#49;	NA	One
50	62	32	110010	2	&#50;	NA	Two
51	63	33	110011	3	&#51;	NA	Three
52	64	34	110100	4	&#52;	NA	Four
53	65	35	110101	5	&#53;	NA	Five
54	66	36	110110	6	&#54;	NA	Six
55	67	37	110111	7	&#55;	NA	Seven
56	70	38	111000	8	&#56;	NA	Eight
57	71	39	111001	9	&#57;	NA	Nine
58	72	3A	111010	:	&#58;	NA	Colon
59	73	3B	111011	;	&#59;	NA	Semicolon
60	74	3C	111100	<	&#60;	&lt;	Less than (or open angled br
61	75	3D	111101	=	&#61;	NA	Equals
62	76	3E	111110	>	&#62;	&gt;	Greater than (or close angle
63	77	3F	111111	?	&#63;	NA	Question mark
64	100	40	1000000	@	&#64;	NA	At symbol
65	101	41	1000001	A	&#65;	NA	Uppercase A
66	102	42	1000010	B	&#66;	NA	Uppercase B
67	103	43	1000011	C	&#67;	NA	Uppercase C

Table 4: US-ASCII Printable Characters *(continued)*

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
68	104	44	1000100	D	&#68;	NA	Uppercase D
69	105	45	1000101	E	&#69;	NA	Uppercase E
70	106	46	1000110	F	&#70;	NA	Uppercase F
71	107	47	1000111	G	&#71;	NA	Uppercase G
72	110	48	1001000	H	&#72;	NA	Uppercase H
73	111	49	1001001	I	&#73;	NA	Uppercase I
74	112	4A	1001010	J	&#74;	NA	Uppercase J
75	113	4B	1001011	K	&#75;	NA	Uppercase K
76	114	4C	1001100	L	&#76;	NA	Uppercase L
77	115	4D	1001101	M	&#77;	NA	Uppercase M
78	116	4E	1001110	N	&#78;	NA	Uppercase N
79	117	4F	1001111	O	&#79;	NA	Uppercase O
80	120	50	1010000	P	&#80;	NA	Uppercase P
81	121	51	1010001	Q	&#81;	NA	Uppercase Q
82	122	52	1010010	R	&#82;	NA	Uppercase R
83	123	53	1010011	S	&#83;	NA	Uppercase S
84	124	54	1010100	T	&#84;	NA	Uppercase T
85	125	55	1010101	U	&#85;	NA	Uppercase U
86	126	56	1010110	V	&#86;	NA	Uppercase V
87	127	57	1010111	W	&#87;	NA	Uppercase W
88	130	58	1011000	X	&#88;	NA	Uppercase X
89	131	59	1011001	Y	&#89;	NA	Uppercase Y
90	132	5A	1011010	Z	&#90;	NA	Uppercase Z
91	133	5B	1011011	[	&#91;	NA	Opening bracket
92	134	5C	1011100	\	&#92;	NA	Backslash
93	135	5D	1011101	]	&#93;	NA	Closing bracket
94	136	5E	1011110	ˆ	&#94;	NA	Caret - circumflex
95	137	5F	1011111	_	&#95;	NA	Underscore
96	140	60	1100000	‘	&#96;	NA	Grave accent
97	141	61	1100001	a	&#97;	NA	Lowercase a
98	142	62	1100010	b	&#98;	NA	Lowercase b
99	143	63	1100011	c	&#99;	NA	Lowercase c
100	144	64	1100100	d	&#100;	NA	Lowercase d
101	145	65	1100101	e	&#101;	NA	Lowercase e
102	146	66	1100110	f	&#102;	NA	Lowercase f
103	147	67	1100111	g	&#103;	NA	Lowercase g
104	150	68	1101000	h	&#104;	NA	Lowercase h
105	151	69	1101001	i	&#105;	NA	Lowercase i

Table 4: US-ASCII Printable Characters *(continued)*

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
106	152	6A	1101010	j	&#106;	NA	Lowercase j
107	153	6B	1101011	k	&#107;	NA	Lowercase k
108	154	6C	1101100	l	&#108;	NA	Lowercase l
109	155	6D	1101101	m	&#109;	NA	Lowercase m
110	156	6E	1101110	n	&#110;	NA	Lowercase n
111	157	6F	1101111	o	&#111;	NA	Lowercase o
112	160	70	1110000	p	&#112;	NA	Lowercase p
113	161	71	1110001	q	&#113;	NA	Lowercase q
114	162	72	1110010	r	&#114;	NA	Lowercase r
115	163	73	1110011	s	&#115;	NA	Lowercase s
116	164	74	1110100	t	&#116;	NA	Lowercase t
117	165	75	1110101	u	&#117;	NA	Lowercase u
118	166	76	1110110	v	&#118;	NA	Lowercase v
119	167	77	1110111	w	&#119;	NA	Lowercase w
120	170	78	1111000	x	&#120;	NA	Lowercase x
121	171	79	1111001	y	&#121;	NA	Lowercase y
122	172	7A	1111010	z	&#122;	NA	Lowercase z
123	173	7B	1111011	{	&#123;	NA	Opening brace
124	174	7C	1111100	|	&#124;	NA	Vertical bar
125	175	7D	1111101	}	&#125;	NA	Closing brace
126	176	7E	1111110	~	&#126;	NA	Equivalency sign - tilde
127	177	7F	1111111	NA	&#127;	NA	Delete

15 Appendix - Acronyms

The list of acronyms was taken from NIST's *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms.*

Table 5: NIST Cryptography Acronyms

acronym	defined
AES	Advanced Encryption Standard; specified in FIPS 1977
ANS	American National Standard
ANSI	American National Standard Institute
ASC	Accredited Standards Committee
CA	Certification Authority
CBC	Cipher Block Chaining mode; specified in SP 800-38A8
CFB	Cipher Feedback mode; specified in SP 800-38A
CKMS	Cryptographic Key Management System
CP	Certificate Policy
CPS	Certification Practice Statement
CRL	Certificate Revocation List
CTR	Counter Mode; specified in SP 800-38A
DES	Data Encryption Standard; originally specified in FIPS 46; now provided in
DH	Diffie-Hellman algorithm
DNSSEC	Domain Name System Security Extensions
DRBG	Deterministic Random Bit Generator; specified in SP 800-90A10
DSA	Digital Signature Algorithm; specified in FIPS 18611
ECB	Electronic Codebook mode; specified in SP 800-38A
ECDSA	Elliptic Curve Digital Signature Algorithm
EMC	Electromagnetic Compatibility
FCKMS	Federal Cryptographic Key Management System
FIPS	Federal Information Processing Standard
FISMA	Federal Information Security Management Act
GCM	Galois Counter Mode; specified in SP 800-38D12
HMAC	Keyed-Hash Message Authentication Code; specified in FIPS 19813
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
EMI	Electromagnetic Interference
INCITS	International Committee for Information Technology Standards
IPSEC	Internet Protocol Security
ISO	International Organization for Standardization
IT	Information Technology
KMAC	KECCAK Message Authentication Code; specified in SP 800-18514

Table 5: NIST Cryptography Acronyms *(continued)*

acronym	defined
MAC	Message Authentication Code
MQV	Menezes-Qu-Vanstone algorith; specified in SP 800-56A15
NRBG	Non-deterministic Random Bit Generator
NIST	National Institute of Standards and Technology
OFB	Output Feedback mode; specified in SP 800-38A
OMB	Office of Management and Budget
OTAR	Over-the-Air-Rekeying
PKI	Public Key Infrastructure
RA	Registration Authority
RBG	Random Bit Generator
RFC	Request for Comment
RSA	A public key algorithm attributed to Rivest, Shamir, and Adleman
ROTs	Roots of Trust
SHA	Secure Hash Algorithm
SP	Special Publication
SSH	Secure Shell protocol
TCG	Trusted Computing Group
TDEA	Triple Data Encryption Algorithm; specified in SP 800-67
TLS	Transport Layer Security
TPM	Trusted Platform Module