RSA Cryptography

Carolyn Zhang, Tara Zhan, and Ryan Denomey
Team Pascal

Supervisor: Dr. Liprandi

May 2021

1 Introduction and History

RSA is a form of public key cryptography that is very commonly used today—from network communications to data transmissions, encryption is necessary to ensure the security and privacy of important information. This encryption must be incredibly robust and unbreakable because it must protect sensitive data from the interception of malicious third parties. But, how can data be encrypted so securely? And, perhaps more importantly, how can authorized users then decrypt the data and rightfully gain access to it?

Through this paper, our objective is to thoroughly explore both the origins and current applications of RSA, as well as how its algorithm encrypts and decrypts data by making use of complex mathematics, whose properties ensure the highest levels of security.

1.1 History of Cryptography

The need for secrecy in the transmission of information has existed for as long as communication itself. Throughout history, cryptography has played a pivotal role in many ways, ranging from day-to-day communication to alterations in the outcomes of battles and wars.

The first recorded uses of cryptography date back to Ancient Egypt (circa 2000 BC), when scribes encrypted the messages of the kings with hieroglyphs [1]. Hieroglyphs are pictograms that are representative of larger ideas and concepts, but their meaning was only understood by a select few individuals [2].

As writing evolved and progressed, mono-alphabetic substitutions were used to encrypt messages. Notably, around 500 BC, Julius Caesar used the Caesar Shift Cipher to encode the messages he sent. This method of cryptography consisted of shifting the individual letters of a message by three positions in the Roman alphabet. To decrypt this message and obtain the original version, the recipient would shift the letters back three positions [1].

By the European Renaissance, cryptosystems had diversified to encompass a wide variety of techniques and algorithms. For instance, Vigenere Coding, an advanced form of alphabetic substitution in which different letters were shifted to a different number of places, was created in the 15th century [1]. One of the most recognizable earlier forms of cryptography is the Enigma rotor machine, which was widely used by Nazi Germany during World War II. Once the Allies managed to discover the cipher used by the Germans, they were able to make sense of the knowledge transmitted through the German military—something that significantly altered the course of the war.

The field of cryptography continued undergoing changes, improvements, and new additions throughout the 20th century.

1.2 Public-Key Cryptography

Up until the 20th century, sending messages through encryption faced a major problem— key distribution. In symmetric encryption (also known as private-key encryption) the same key is used for encrypting and decrypting the message. This key needed to be communicated between the sender and receiver, but there was no way to securely send it without the chance of a third-party intercepting it, gaining the ability to decrypt any ciphertexts that were encrypted with that same key. The first idea to solve this issue came from an American cryptographer, Whitfield Diffie. He came up with the concept of asymmetric (public-key) cryptography, a method that would employ two different keys— the private key, and the public key. The public key would be used to encrypt the message, whereas the private would be used to decrypt.[3] Public-key encryption reduces cybersecurity risks because users never need to transmit their private key.[4]

The main advantage of asymmetric cryptography is that it uses a one-way function— meaning that it is easy to compute in one direction, but near impossible in the opposite (inverse) direction without a special key. In cryptography terms, this is called a trapdoor function. A quick example of a trapdoor is trying to find two prime numbers while only knowing the product of the two. If the product was very large, then finding the answer without knowing either of the prime factors would be impossible with our current computing technology— but if one knew the value of either of the two prime factors, finding the other would be simple. However, secure trapdoor functions are remarkably hard to find, as Diffie would realize. He (along with his partner, Hellman) was only able to propose the concept. It would be a year before a group of scientists managed to create the first public-key algorithm, known as RSA.[3]

1.3 The Origins of RSA Cryptography

Named after its three inventors, RSA is a form of public-key cryptography that was developed by Ronald Rivest, Adi Shamir, and Leonard Adleman at the Massachusetts Institute of Technology (MIT) in 1977 [2].

The greatest obstacle faced by Diffie and his idea of using asymmetric keys was his inability to create a suitable one-way function that could be used as a cipher. Building off of Diffie's work, Rivest, Shamir, and Adleman sought to develop a one-way function to ensure the security of encrypted data transmission. Rivest and Shamir used their background in computer science to propose mathematical solutions, while Adleman, a mathematician, analyzed and reviewed their solutions [5]. Over the course of one year, Rivest and Shamir had generated 42 failed attempts at creating a satisfactory one-way function, since Adleman identified flaws and refuted each one [6]. Finally, in 1977, the trio discovered what would become known as the RSA algorithm—a solution to the conundrum faced by their predecessors.

As mentioned earlier, the main advantage of using any form of public-key cryptography is the absence of a key exchange—a high-risk operation, as non-intended recipients may be able to intercept it. [5] RSA, a form of public-key cryptography, further surpasses its predecessors as a result of the security associated with its algorithm. Any interceptions by third parties will not reveal any information, since the data is indecipherable without the private key [6].

The general idea of RSA cryptography can be described as follows: Alice and Bob, two friends, wish to pass a treasure chest amongst themselves. This chest is secured with a padlock that only opens with a specific key. Alice and Bob each have their own padlock and the corresponding key. Suppose Alice wants to send Bob some treasure one day. Using the concept of the RSA algorithm, Alice first asks Bob to give her his opened padlock. Then, Alice places the treasure inside the chest and locks it with Bob's padlock. She sends the chest back to Bob, who is able to open it, since he owns the key. Because neither Alice or Bob's personal keys are ever exchanged throughout the process, a third party cannot open the chest, even if they manage to intercept it [5].

2 RSA Math

Now, we must carry out this relatively simple concept with a complex, robust, and secure mathematical implementation. The main mathematical principles used in RSA cryptography are:

- Modular arithmetic (and its properties)
- Euler's totient function
- Euler's theorem

The following is a summary of the RSA algorithm:

1. Receiver creates public key: $n = pq$; e is relatively prime to $\varphi(n)$
2. Receiver creates private key: $de \equiv 1 \pmod{\varphi(n)}$
3. Sender encrypts their message: m is obtained through some conversion process (e.g. ASCII); $c \equiv m^e \pmod{n}$
4. Receiver decrypts c to obtain m: $m \equiv c^d \pmod{n}$

We will now explore each step in detail...

2.1 Public Key

The public key is the component of the encryption that is publicly distributed and available—that is, anyone can use the public key to send its owner a message. The public key consists of two parts:

1. $\boldsymbol{n}$, the product of two large prime numbers, p and q
2. $\boldsymbol{e}$, a number which is relatively prime to $\varphi(n)$

2.1.1 n

n is quite straightforward to compute, as it is obtained through the simple operation of multiplication, performed on prime numbers p and q. However, it would be quite difficult to determine p and q from n if it is sufficiently large (tens or even hundreds of digits). Hence, we consider this a one-way function: an operation that is simple to execute one way, yet nearly impossible the other way.

To illustrate, let's take a look at a brief example:

Example 1a: Prime factor 813473 by hand.

You probably started testing whether or not it is divisible by each prime, working up from 2, to 3, to 5, to 7, and so on. You also probably gave up pretty quickly, since this task is simply too time consuming (and draining!).

Example 1b: Multiply 859 and 947 by hand.

Using your preferred method of long multiplication, you most likely obtained the correct answer of 813473. This was not nearly as daunting of a task, and it was exponentially easier and faster.

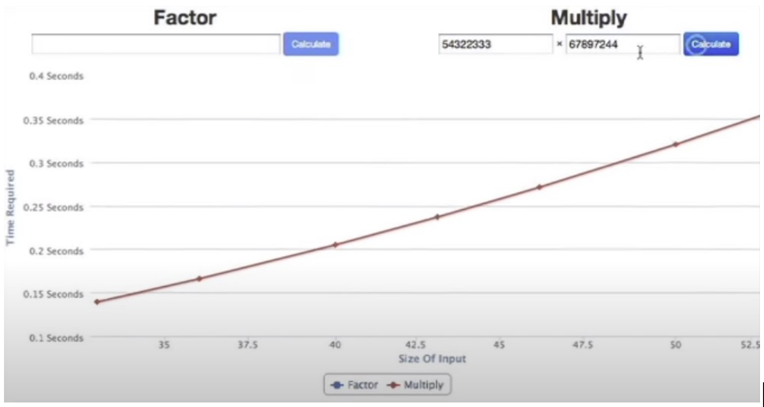
A computer would concur with this conclusion—even for a computer, prime factoring quickly becomes an incredibly time-consuming task as the number increases in size. Conversely, the task of multiplying two numbers together is significantly faster.

The security and reliability of the RSA algorithm is founded on the notion that it is incredibly difficult to deduce the prime factorization of a very large number. You may be wondering, "Why does the prime factorization of n matter?"

To answer this, we must understand Euler's totient function, $\varphi(n)$, which is an essential component of the private key used in the message decryption process. Specifically, for a very large n value, $\varphi(n)$ can only be efficiently calculated knowing n's two factors, p and q.

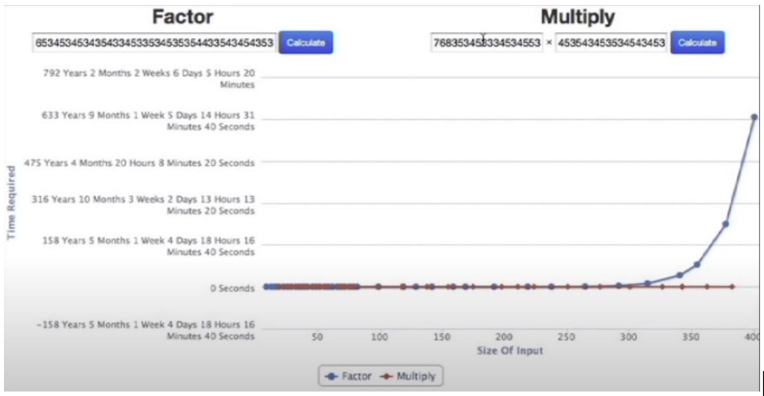
2.1.2 e and Euler's totient function

As mentioned before, the second component of the public key, $\boldsymbol{e}$, is relatively prime to $\varphi(n)$, where n is the first component of the public key.



[7]

Figure 1: The time required to complete a multiplication operation increases linearly as the size of input increases.



[7]

Figure 2: The time required to complete a prime factorization increases exponentially as the size of input increases, quickly reaching several years or even decades.

In general, for a positive integer $k, k \in \mathbb{Z}^+$, $\varphi(k)$ returns the number of integers between 1 and k that are relatively prime to k—that is, the number of m values where $m < k$ and $\gcd(m, k) = 1$.

Example 2a: What is $\varphi(5)$? What is $\varphi(6)$?

1. $\varphi(5) = 4$, since $\gcd(1, 5) = \gcd(2, 5) = \gcd(3, 5) = \gcd(4, 5) = 1$
   Notice that for a prime number, r, $\varphi(r)$ = r - 1, since every number less than r is relatively prime to it.
2. $\varphi(6) = 2$, since $\gcd(1, 6) = \gcd(5, 6) = 1$
   Notice here that $6 = 2 \times 3$, and $\varphi(6) = \varphi(2) \times \varphi(3) = (2-1) \times (3-1) = 2$. This is *Euler's totient function*, which we will explore next.

Now, to answer the previously posed question, "Why does the prime factorization of n matter?":

A property of Euler's totient function is that $\varphi(k) = \varphi(a)\varphi(b)$ if $k = ab$ and $\gcd(a, b) = 1$ (i.e. a and b are coprime).

Additionally, as mentioned in the solution for ***Example 2a***, for a prime number r, $\varphi(r) = r - 1$. So, putting this together, if we choose a and b to be prime factors of k, then we can rewrite the totient function as $\varphi(k) = (a-1)(b-1)$.

Here is where the prime factorization of n matters: given that $n = pq$, we can calculate $\varphi(n)$ with the equation $\varphi(n) = (p-1)(q-1)$.

e is selected to be relatively prime to $\varphi(n)$. Its randomness is not incredibly important because it is publicly available, so, in practice, e is generally chosen to be $2^{16} + 1 = 65537$. This value is large enough to improve security, but not so large as to impair the efficiency of the encryption. [8]

Although $\varphi(n)$ is relevant for the selection of the e value, it plays an even more pivotal role in the calculation of the receiver's private key—our next subject of exploration.

2.2 Private Key

The receiver's private key, $\boldsymbol{d}$, is only known to themselves and is never publicly distributed. As was explained before, RSA is a form of asymmetric cryptography—the public key and private key are asymmetric, or non-identical. Thus, if a third party were to intercept a message, they would be unable to decipher its original form without the private key, which they are unable to obtain.

$\boldsymbol{d}$ is the *modular multiplicative inverse* of $e \pmod{\varphi(n)}$. Simply put, this means that $de \equiv 1 \pmod{\varphi(n)}$.

With that, let's explore the properties and applications of modular arithmetic!

Modular arithmetic is a way to find the remainder when one number is divided by another. If $x \equiv y \pmod{z}$, then x and y have the same remainder when divided by z. Here, x and y are said to be congruent (we read the congruence statement as "x is congruent to y mod z"). For example, $36 \equiv 8 \pmod{7}$, since both 36 and 8 have a remainder of 1 when divided by 7. Negative numbers can also be part of congruence statements, such as $15 \equiv -9 \pmod{12}$, since both 15 and -9 have a remainder of 3 when divided by 12 (when subtracting 3 from either 15 or -9, the result is divisible by 12).

Here are a few basic properties of modular arithmetic:

If $x \equiv y \pmod{z}$, then

1. $y \equiv x \pmod{z}$
2. $x = kz + y$, where $k \in \mathbb{Z}$
3. $x - y \equiv 0 \pmod{z}$, since x and y have the same remainder when divided by z

Example 3a: Determine a value of y that satisfies $34 \equiv y \pmod{7}$.
Solution: Using property 1, this means that $y \equiv 34 \pmod{7}$, or $y \equiv 6 \pmod{7}$. We could rewrite this into an equation using property 2: $y = 7k + 6$. Substituting in integer values for k, we obtain $y = ... - 8, -1, 6, 13...$
Example 3b: Determine a value of z that satisfies $36 \equiv 10 \pmod{z}$.
Solution: Using property 3, we can rewrite this as $(36 - 10) \equiv 0 \pmod{z}$. In other words, 26 is divisible by z, or z is a factor of 26. We know that $26 : 1, 2, 13, 26$, so z could take on any of these values (although observe that $z = 1$ would be a bit pointless).

Normal mathematical operations can be used when dealing with modular arithmetic too. Let's take a look at some properties of common operations in modular arithmetic:

<u>Addition</u> [9]

1. If $a + b = c$, then $a \pmod{N} + b \pmod{N} \equiv c \pmod{N}$
2. If $a \equiv b \pmod{N}$, then $a + k \equiv b + k \pmod{N}$, $k \in \mathbb{Z}$
3. If $a \equiv b \pmod{N}$ and $c \equiv d \pmod{N}$, then $a + c \equiv b + d \pmod{N}$

Example 4: What is the remainder when $123 + 234 + 32 + 56 + 22 + 12 + 78$ is divided by 3?
Solution: Essentially, the question is asking for x, where

$123 + 234 + 32 + 56 + 22 + 12 + 78 \equiv x \pmod 3$. Using property 1, each term can be replaced with its (mod 3) value:

$$\begin{aligned} x &\equiv 123 + 234 + 32 + 56 + 22 + 12 + 78 \pmod 3 \\ &\equiv 0 + 0 + 2 + 2 + 1 + 0 + 0 \pmod 3 \\ &\equiv 5 \pmod 3 \\ &\equiv 2 \pmod 3 \end{aligned}$$

$\therefore$ The remainder is 2.

Multiplication [9]

1. If $a \times b = c$, then $a \pmod N \times b \pmod N \equiv c \pmod N$
2. If $a \equiv b \pmod N$, then $ka \equiv kb \pmod N$, $k \in \mathbb{Z}$
3. If $a \equiv b \pmod N$ and $c \equiv d \pmod N$, then $ac \equiv bd \pmod N$

Example 5: What is the remainder when $124 \times 134 \times 23 \times 49 \times 235 \times 13$ is divided by 3?
Solution: Similarly to ***Example 4***, we can rewrite the question and solve for x, where $124 \times 134 \times 23 \times 49 \times 235 \times 13 \equiv x \pmod 3$. Again, using property 1 of multiplication in modular arithmetic, each term can be replaced with its (mod 3) value:

$$\begin{aligned} x &\equiv 124 \times 134 \times 23 \times 49 \times 235 \times 13 \pmod 3 \\ &\equiv 1 \times 2 \times 2 \times 1 \times 1 \times 1 \pmod 3 \\ &\equiv 4 \pmod 3 \\ &\equiv 1 \pmod 3 \end{aligned}$$

$\therefore$ The remainder is 1.

With a better idea of how modular arithmetic works, we return to the explanation of RSA.
Recall that $de \equiv 1 \pmod{\varphi(n)}$. We can isolate for d as follows:

$$\begin{aligned} de &= 1 + k\ \varphi(n)\ ,\ k \in \mathbb{Z} \\ de - k\ \varphi(n) &= 1 \end{aligned}$$

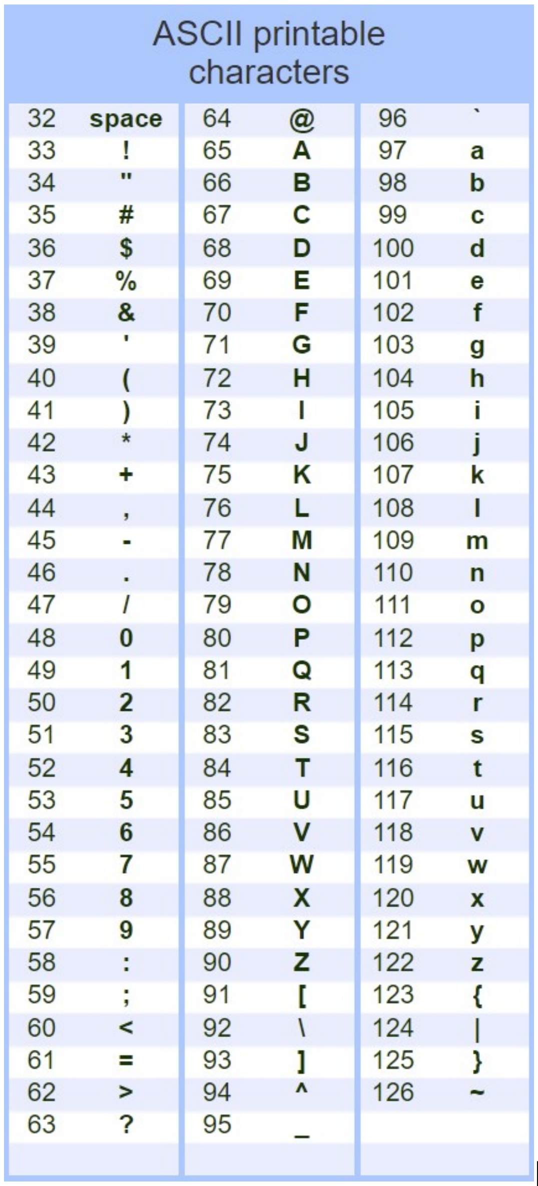
The receiver knows the values of $\varphi(n)$ and e, but does not yet know d at this point. Once again, owing to the properties of modular arithmetic, more than one d value can satisfy the above equation, given that k is an integer. We can make use of the **extended Euclidean algorithm** to efficiently compute a d value. This algorithm and its importance in RSA cryptography will be covered in **APPENDIX A**.

We have now generated the two critical components of RSA cryptography: the **public key** and the **private key**. This means that the sender may encrypt messages with the public key, and the receiver may decrypt the sender's messages with their private key. Next, we will take a look at the specifics of the encryption and subsequent decryption processes.

2.3 Message Encryption

Following the generation of both the public and private keys, the sender must encrypt their message, then send it to the receiver.

Evidently, sending a message in plaintext with letters and symbols is incompatible with the RSA algorithm, which needs numerical input! So, the sender's message must first be converted to a number, $\boldsymbol{m}$. The only condition is that $m < n$, so that the message is fully preserved when (mod n) is taken at a later step. However, this is generally ensured by initially selecting p and q to be very large prime numbers to ensure the size of n.


ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

[10]

Figure 3: There are many ways in which the m value can be obtained from regular text. One common way is using the ASCII alphabet. Pictured above is a table wit the ASCII numbers for printable characters.

Example 6: Using the above ASCII table, encode the message "H3LLO!".
Solution: 'H'=72, '3'=51, 'L'=76, 'L'=76, 'O'=79, '!'=33

Next, m must be converted into $\boldsymbol{c}$, the *ciphertext*, to further increase the security of the encryption. This is done with the following formula:

$\mathbf{c \equiv m^e \pmod{n}}$

Now, the message has been transformed into c and is ready to be sent to the receiver.

In addition to the difficulty of prime factoring n, another layer of security is introduced here. The sender will send the receiver the ciphertext, but the risk of interception still exists. If such an interception were to occur, an unauthorized individual would have access to c, as well as e and n, which are publicly available. This leaves m as the only unknown variable, which would seem to suggest that we could calculate it. However, because of the exponentiation and the (mod n) that are applied, the interceptor would need to use modular logarithms to determine m. This is an incredibly difficult problem in mathematics, and there are no known ways to efficiently solve it—even the most advanced algorithms are not fast enough to complete the calculation in relevant time frames. Because of this, the one and only way to obtain the sender's message, m, is through the decryption formula that uses d, which we will cover next.

2.4 Ciphertext Decryption

The formula that is used by the receiver to decrypt the sender's message is

$\mathbf{c^d \equiv m \pmod{n}}$, which can be rewritten as

$\mathbf{m \equiv c^d \pmod{n}}$

Since the receiver obtains c from the sender, has access to d (their private key), and knows the value of n (component of their public key), they are able to apply the formula and accurately decrypt m. Once the receiver obtains m, they can easily reconvert the numerical m into plaintext by using the agreed upon conversion method (e.g. the ASCII table).

Conversely, if a third party were to intercept c, they do not have the value of d, and are thus unable to accurately decrypt m.

In order to show that the formula $m \equiv c^d \pmod{n}$ does indeed work to decrypt the sender's message, m, we must rely on two equations:

- Ciphertext encryption formula: $c \equiv m^e \pmod{n}$
- Private key formula: $de = 1 + k\ \varphi(n)$, $k \in \mathbb{Z}$

- *Euler's theorem* (see **APPENDIX B**): $m^{\varphi(n)} \equiv 1 \pmod{n}$

Using substitution and knowledge about modular arithmetic from **2.2 Private Key**:

1. Starting from the ciphertext encryption formula, raise both sides to the exponent of d so the left side corresponds with the left side of the decryption formula:

$$\begin{aligned} c &\equiv m^e \pmod{n} \\ c^d &\equiv (m^e)^d \pmod{n} \\ c^d &\equiv m^{de} \pmod{n} \end{aligned}$$

2. Substitute in the private key formula for de and simplify:

$$\begin{aligned} c^d &\equiv m^{1+k\ \varphi(n)} \pmod{n} \\ c^d &\equiv m^1 \times m^{k\ \varphi(n)} \pmod{n} \\ c^d &\equiv m \times (m^{\varphi(n)})^k \pmod{n} \end{aligned}$$

3. Substitute in Euler's theorem for $m^{\varphi(n)}$:

$$\begin{aligned} c^d &\equiv m \times 1^k \pmod{n} \\ c^d &\equiv m \pmod{n} \\ m &\equiv c^d \pmod{n} \end{aligned}$$

Thus, by raising c to the power of d and taking the modulus n, the receiver can decrypt the message, m.

2.5 Summary and the Security of the RSA Algorithm

Now, let's try to run through the RSA algorithm (using small values!):

Example 7a: Determine the receiver's public and private key if $p = 11$, $q = 17$, and $e = 3$.

Solution - public key

$$\begin{aligned} n &= pq \\ &= 187 \\ e &= 3 \end{aligned}$$

Solution - private key (*please see **APPENDIX A** to learn more about the extended Euclidean algorithm, which makes the process of determining d much more efficient)

$$\begin{aligned} de &\equiv 1 \pmod{\varphi(n)} \\ 3d &\equiv 1 \pmod{(11-1)(17-1)} \\ 3d &\equiv 1 \pmod{160} \\ 3d &= 1 + 160k \ , \ k \in \mathbb{Z} \\ 3d - 160k &= 1 \\ d &= 170, 267, 427, ... \end{aligned}$$

We could choose $d = 107$.

Example 7b: Continuing with the public and private key calculated in ***Example 7a***, encrypt and decrypt the message "MATH" using the ASCII table. This will be done one letter at a time for easier calculations, but keep in mind that in its real world use, such simplifications would not occur.
(Recall that $n = 187$, $e = 3$, $\varphi(n) = 160$, $d = 107$)

Solution - "M":
To encrypt the message, we need to transform m into c, using the ASCII table provided in *Figure 3*.

$$\begin{aligned} m &= 77 \\ c &\equiv m^e \pmod{n} \\ c &\equiv 77^3 \pmod{187} \\ c &= 66 \end{aligned}$$

To decrypt the message, we need to raise c to the power of d and take the modulus n.

$$\begin{aligned} m &\equiv c^d \pmod{n} \\ m &\equiv 66^{107} \pmod{187} \\ m &= 77 \end{aligned}$$

We find that through the decryption process, the original message does indeed remain intact! To convert this number into meaningful text, all that needs to be done is a quick reference to the ASCII table to recover our original letter, "M".

You will notice that computing large exponents, like 66^{107}, cannot be reasonably done on a regular calculator. While computers have greater computational power, using "brute force" to calculate 66^{107} is still excessively slow and wasteful. Thus, computers make use of *fast modular exponentiation*, which is covered in **APPENDIX C**. By using the algorithm explained there, you will find that you are, in fact, able to compute $66^{107} \pmod{187}$ on your average calculator with minimal effort!

Now, we repeat the encryption and decryption processes for the remaining letters in "MATH".

Solution - "A":
Encryption:

$$\begin{aligned} m &= 65 \\ c &\equiv m^e \pmod{n} \\ c &\equiv 65^3 \pmod{187} \\ c &= 109 \end{aligned}$$

Decryption:

$$\begin{aligned} m &\equiv c^d \pmod{n} \\ m &\equiv 109^{107} \pmod{187} \\ m &= 65 \end{aligned}$$

Solution - "T":
Encryption:

$$\begin{aligned} m &= 84 \\ c &\equiv m^e \pmod{n} \\ c &\equiv 84^3 \pmod{187} \\ c &= 101 \end{aligned}$$

Decryption:

$$\begin{aligned} m &\equiv c^d \pmod{n} \\ m &\equiv 101^{107} \pmod{187} \\ m &= 84 \end{aligned}$$

Solution - "H":
Encryption:

$$\begin{aligned} m &= 72 \\ c &\equiv m^e \pmod{n} \\ c &\equiv 72^3 \pmod{187} \\ c &= 183 \end{aligned}$$

Decryption:

$$\begin{aligned} m &\equiv c^d \pmod{n} \\ m &\equiv 183^{107} \pmod{187} \\ m &= 72 \end{aligned}$$

As with any cryptosystem, the most important aspect of its functionality is the security of its algorithm. For RSA, the security lies in the difficulty of finding $\varphi(n)$ without knowing the large prime factors of n, in addition to the inherently impossible task of computing modular logarithms, which was discussed in **2.3** .

In our final decryption congruence statement, $m \equiv c^d \pmod{n}$, the value of n is publicly known and the value of c can be captured through interception. However, d, the receiver's private key, may only be computed through $d = (1 + k\ \varphi(n)) \div e$, which requires the value of $\varphi(n) = (p-1)(q-1)$. If a computer tried to prime factor n, it would take decades or even centuries due to the large size of n, as explained in **2.1 Public Key**. Thus, without knowing the original prime factors, p and q, one cannot determine the value of $\varphi(n)$, leading to an inability to compute a value for d and thus an inability to decode the ciphertext.

3 RSA In Everyday Life[11]

RSA allows for communication without the need to share a secret key. It is used today to store information, money, passwords, fingerprints and more valuable data that must not be stolen by unauthorized users. It can also be used for things such as:

Time Stamping: Time stamping can be used to certify that an electronic document or communication was delivered at a certain time. RSA encryption is used in this.

Electronic Money: Money can be transferred electronically through credit cards or banks, and this can be made anonymous. RSA encryption helps to conceal signatures, fingerprints, names and other data that can be taken from the transactions.

Secure Network Communications: Secure Socket Layer: There is a public key protocol that has been developed called SSL or secure socket layer for securing data layered between internet based communications (TCP/IP), and application protocols. This is good for data encryption, and authentication for TCP/IP connection. This system uses RSA's public key encryption for authentication.

Anonymous Remailers: An anonymous remailer is a free service that many people use. It removes the identity of the mailer, and lets only the content be sent. Because the remailer can keep your identity, many people use more than one remailer. To ensure the safety of their message, people encrypt their message using a public key from the final remailer. That way, the person's identity is lost, and the message cannot be taken until after it reaches the final remailer, and only the first remailer has the person's identity.

How You Can Get Involved: To make yourself more knowledgeable of RSA encryption, you can participate in an ethical hacking contest. I participated in one that had some RSA problems that I cracked called PicoCTF (Capture The Flag), and there are many others like it. PicoCTF helped me understand RSA encryption better because there were sample problems, and if you needed help, you could ask the people who were running the contest to give you hints.

4 Breaking RSA: Quantum Computing

In the world of cryptography, quantum computing presents a possible threat to many encryption systems our digital society is built on. An ordinary computer uses bits to store data in the forms of 0s and 1s.[12] All digital information such as numbers, text, and images are represented with a combination of millions of these bits. Quantum computers use qubits, which, in contrast to bits, can be set to 0 and 1 at the same time, or even a combination of the two. This is called a superposition, and it is the main principle that could make quantum computing significantly faster than the world's best computers.[13]

For example, think of a bit as a coin. A normal computer can only process the coin as facing either heads or tails. But a qubit would be like a spinning coin— it can be heads, or tails, or either, until one stops and measures it. Qubits allow for uncertainty and possibility, whereas normal computers can only tackle problems through brute force. If one were to ask a regular computer to navigate through a maze, it would test every possible pathway before eventually finding the correct one. Quantum computers can test every pathway at once and simply choose the correct option. [13]

4.1 Shor's Algorithm

In 1994, a mathematician named Peter Shor developed an algorithm for quantum computers that could potentially break encryption methods based on factoring like RSA. In short, Shor's algorithm breaks down the prime factorization of any integer into 3 parts. With the use of a theoretical quantum computer, these 3 steps could be done in polynomial time— breaking the defense of exponential time that RSA is centered around.[14]

However, quantum computers have not become developed enough to outperform classical computers in prime factoring. To crack an RSA 2048-bit standard encryption scheme, a quantum system would require 20 million working qubits and 8 hours of computing time. The current best quantum computers in the world have around 60-70 stable qubits.[15] As of right now, quantum computers have yet to outperform classical computers in prime factoring. It is estimated that the large-scale quantum computers required to break RSA will not exist for another decade or two.[16]

4.2 Solutions

In the case that RSA does become ineffective, there are numerous options for secure encryption that even quantum computers cannot crack. The two main proposals include quantum-resistant algorithms and quantum key distribution.[16]

Many quantum-resistant algorithms (also known as post-quantum cryptography) already exist in use around the world. They operate similar to RSA by using public-key encryption, meaning they could easily replace it while maintaining our current digital framework, but the main issue falls in the fact that none of these algorithms have been proven to be unbreakable by quantum computers. With still so little known about quantum mechanics, it is impossible to say whether such an algorithm could even theoretically exist. As of right now, security experts note that bit lengths could simply be increased to keep pace with the rapidly progressing technology of quantum computers.[16]

Another option is to use quantum mechanics against itself. In the world of quantum physics, subatomic particles behave in such a way that simply observing them will change their structure. Quantum key distribution (QKD) sends photons through a fiber optic line to generate a key. If a third-party hacker attempts to intercept this key exchange, the photons are affected and the recipient will know that the key isn't secure. QKD is unbreakable in theory but would require expensive new infrastructure such as satellites or new fiber optics, and is also extremely slow compared to standard encryption methods.[17] New technology may be needed in the future to combat quantum computing, but for now, the falling apart of our encryption methods remains a mere possibility.

5 Conclusion

This report has covered the origins of public-key cryptography, the applications of RSA, and the mathematical functions that make it secure enough to be implemented in global digital security. RSA represents an example of how math can be used in groundbreaking ways to benefit society. Its simple concept yet ingenious execution of a trapdoor function is arguably the first and best of its kind, continuing to be one of the most widely used cryptosystems to date. While the development of quantum computing in our rapidly progressing technology could render the algorithm futile, RSA will remain as a key milestone in cryptography.

6 Appendices

APPENDIX A: The Extended Euclidean Algorithm [18]

In the RSA algorithm, the private key is determined by the formula $de \equiv 1 \pmod{\varphi(n)}$, since d and e are modular multiplicative inverses. The receiver evidently has access to e, a component of the public key, and can compute $\varphi(n)$, since they know the values of p and q, the prime factors of n. However, with the given formula, it is not exceptionally simple or efficient to determine the value of d, as some form of trial and error would need to be used.

A solution to this problem is the **extended Euclidean algorithm**, which is regarded as an extension of the standard Euclidean algorithm because it is executed in reverse.

A.1 The Euclidean algorithm

First, let's review the *Euclidean algorithm*—an algorithm which is used for computing the greatest common divisor (gcd) of two integers, A and B. $\gcd(A, B)$ will return the largest integer that divides both A and B.

If $A = 0$, then $\gcd(A, B) = B$. Likewise, if $B = 0$, then $\gcd(A, B) = A$. Thus, when either A or B becomes 0, the algorithm can end. The algorithm executes as follows: [19]

1. A can be written in the form $A = BQ + R$, where Q is the quotient and R is the remainder of $A \div B$.
2. From the previous equation, $A = BQ + R$, repeat step 1 for $\gcd(B, R)$. In other words, replace the former A with B, and the former B with R: $\gcd(A, B) = \gcd(B, R)$.

Example: Use the Euclidean algorithm to determine the gcd of 72 and 54.
Round 1: $A = 72$, $B = 56$

$$\begin{aligned} A &= BQ + R \\ 72 &= 56(1) + 16 \end{aligned}$$

Round 2: $A = 56$, $B = 16$

$$\begin{aligned} A &= BQ + R \\ 56 &= 16(3) + 8 \end{aligned}$$

Round 3: $A = 16$, $B = 8$

$$\begin{aligned} A &= BQ + R \\ 16 &= 8(2) + 0 \end{aligned}$$

$A = 8$, $B = 0$
$\therefore \gcd(72, 56) = 8$

A.2 The extended Euclidean algorithm

The extended Euclidean algorithm is used to find integers x and y, such that $Ax + By = \gcd(A, B)$.

Now, having executed all the steps in the Euclidean algorithm, we work backwards from the end to determine x and y. If we isolate for the R value at each step before the 0 was reached, and repeatedly substitute these R values into each previous step, we essentially "unravel" the effects of the regular Euclidean algorithm to obtain x and y.

Example: Compute x and y such that for $A = 72$ and $B = 56$, $Ax + By = \gcd(A, B)$.
Substituting in known values, we need to determine x and y for 72x + 56y = 8. First, we return to the solution to the previous example and isolate for the R value in each step before 0 was reached:
Round 1: $16 = 72 - 56(1)$
Round 2: $8 = 56 - 16(3)$
Now, working backwards, from *Round 2* to *Round 1*, we make our substitution and rearrange:

$$\begin{aligned} 8 &= 56 - 16(3) \\ 8 &= 56 - (72 - 56(1))(3) \\ 8 &= 56 - 72(3) + 56(3) \\ 8 &= 56(4) + 72(-3) \end{aligned}$$

$\therefore x = -3$, $y = 4$

However, one solution for x and y is not the only possible solution. In fact, we could generate an infinite number of pairs (x, y) that satisfy the equation if we add cB to x and $-cA$ to y, $c \in \mathbb{Z}$:

$$\begin{aligned} Ax + By &= Ax + cAB - cAB + By \\ &= A(x + cB) + B(y - cA) \end{aligned}$$

For our previous example, if we choose $c = 2$, we could find another solution for x and y:

$$\begin{aligned} x &= -3 + (2)(56) \\ &= 109 \end{aligned}$$

$$\begin{aligned} y &= 4-(2)(72) \\ &= -140 \end{aligned}$$

If you substitute these new x and y values into $72x + 56y = 8$, you will find that they are an equally valid solution to the equation.

Returning to our original conundrum, the computation of d (the receiver's private key), we first rewrite the formula for d in the format of $Ax + By = \gcd(A, B)$:

$$\begin{aligned} de &\equiv 1 \pmod{\varphi(n)} \\ de &= 1 + \varphi(n)\ k\ ,\ k \in \mathbb{Z} \\ ed - \varphi(n)\ k &= 1 \end{aligned}$$

Recall from **2.2 Private Key** that, by definition, e is *relatively prime* to $\varphi(n)$, meaning $\gcd(e, \varphi(n)) = 1$. Thus, ***ed - φ(n) k = 1*** is in the form of $Ax + By = \gcd(A, B)$. As a result, we can apply the Euclidean algorithm to e and $\varphi(n)$, then work in reverse to obtain d and k from the extended Euclidean algorithm. Take a look at the following solution to ***Example 7a*** using the extended Euclidean algorithm:

Example: Determine the receiver's private key if $p = 11$, $q = 17$, and $e = 3$.

$$\begin{aligned} \varphi(n) &= (11-1)(17-1) \\ &= 160 \end{aligned}$$

Substituting this into the rearranged private key equation:

$$\begin{aligned} ed - \varphi(n)k &= 1 \\ 3d - 160k &= 1 \end{aligned}$$

Performing the Euclidean algorithm for gcd(160, 3):

$$\begin{aligned} 160 &= 3(53) + 1 \\ 3 &= 1(3) + 0 \end{aligned}$$

Isolating for the remainder and performing the extended Euclidean algorithm:

$$\begin{aligned} 1 &= 160 - 3(53) \\ 1 &= 3(-53) - 160(-1) \end{aligned}$$

Thus, $d = -53$ and $k = -1$ is one solution. However, for the RSA algorithm, we need a d value that is positive, so we must find an alternate solution using $A(x + cB) + B(y - cA) = \gcd(A, B)$. In this case,

$$3(-53 + 160c) + 160(-1 - 3c) = 1$$

As a general formula, d could mathematically be any value satisfying $-53 + 160c$, $c \in \mathbb{Z}$. Since we need to find a positive d value, we can simply choose $c = 1$.

$$\begin{aligned} d &= -53 + 160(1) \\ &= 107 \end{aligned}$$

APPENDIX B: Euler's Theorem [20]

Euler's theorem, $m^{\varphi(n)} \equiv$ ***1 (mod n)***, is one of the key facts that allows the RSA decryption formula, $c^d \pmod{n}$, to correctly return the message m. To further verify the correctness of the decryption formula, let us prove the theorem:

Let $\{r_1, r_2, r_3, ..., r_{\varphi(n)}\}$ represent the set of integers that are less than n and are relatively prime to n—that is, $\gcd(r_i, n) = 1$.

Choose m such that $m \in \{r_1, r_2, r_3, ..., r_{\varphi(n)}\}$. For any r_i in the set, there exists another element r_j such that $r_j \equiv m \times r_i \pmod{n}$. This is because m and r_i are both relatively prime to n, so their product, mr_i, must also be relatively prime to n—an integer r_j.

As a result, the set $\{mr_1, mr_2, mr_3, ..., mr_{\varphi(n)}\} \pmod{n}$ is a *permutation*, or rearrangement, of the set $\{r_1, r_2, r_3, ..., r_{\varphi(n)}\}$. In other words, both these sets contain the same elements.

Because of this, the product of all the elements in each set will be equal:

$$\prod_{k=1}^{\varphi(n)} r_k \equiv \prod_{k=1}^{\varphi(n)} mr_k \pmod{n}$$

Using the properties of products, we can take the constant m out of the left side of the congruence statement, in the form of $m^{\varphi(n)}$:

$$\prod_{k=1}^{\varphi(n)} r_k \equiv m^{\varphi(n)} \prod_{k=1}^{\varphi(n)} r_k \pmod{n}$$

We can cancel out $\prod_{k=1}^{\varphi(n)} r_k$ from both sides because it is relatively prime to n,

leaving us with $1 \equiv m^{\varphi(n)} \pmod{n}$, or $m^{\varphi(n)} \equiv$ ***1 (mod n)***.

APPENDIX C: Fast Modular Exponentiation

With the encryption and decryption processes, very large numbers are being handled as a result of exponentiation. Specifically, in the encryption process, m is raised to the power of e ($c \equiv m^e \pmod{n}$), and in the decryption process, c is raised to the power of d ($c^d \equiv m \pmod{n}$). With large c, d, and e values, computing the value of these powers is incredibly time-consuming and wasteful.

As a result, computers generally do not use a "brute force" method to compute the powers. Instead, the operation can be simplified using *fast modular exponentiation*, which is made possible by the fact that we are taking the modulus n. In essence, the exponent is broken down into smaller multiplication operations (as exponents are repeated multiplication) and the (mod n) is taken after each step.

Recall the following property of multiplication with modular arithmetic from **2.2 Private Key**:
$a \times b \pmod{n} = a \pmod{n} \times b \pmod{n}$

The most efficient way to break down a large exponent is to use base two, or *binary exponentiation.* Computers naturally operate in binary—0's and 1's—so the quickest way to perform exponentiation is with the use of base two.

The steps for the binary exponentiation of $x^y \pmod{z}$ are as follows:

1. Convert the exponent, y, into base two
2. Start from $x^0 = 1$ as the first number
3. Working from left to right along the base two number, at each digit, square the previous number, starting with $(x^0)^2 = 1$
4. - If the digit in the base two number at that place value is a 1, multiply the previous number by x: $(x^0)^2 \times x = x$
   - Otherwise, if the digit in the base two number at that place value is a 0, do nothing.
5. Repeat steps 3 and 4 until the end of the base two number is reached. At each step, take the (mod z), as permitted by the property of modular multiplication outlined earlier.

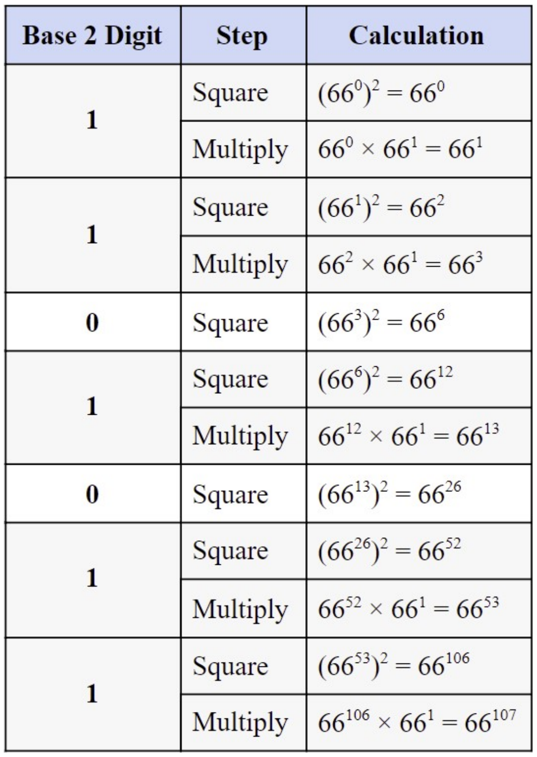
Because the (mod z) is taken at frequent intervals, the computations will never exceed z^2.

This process may seem complicated, but going through an example may help make this algorithm more clear:

Example: Calculate the value of $66^{107} \pmod{187}$. You may recognize this from ***Example 7b*** earlier!

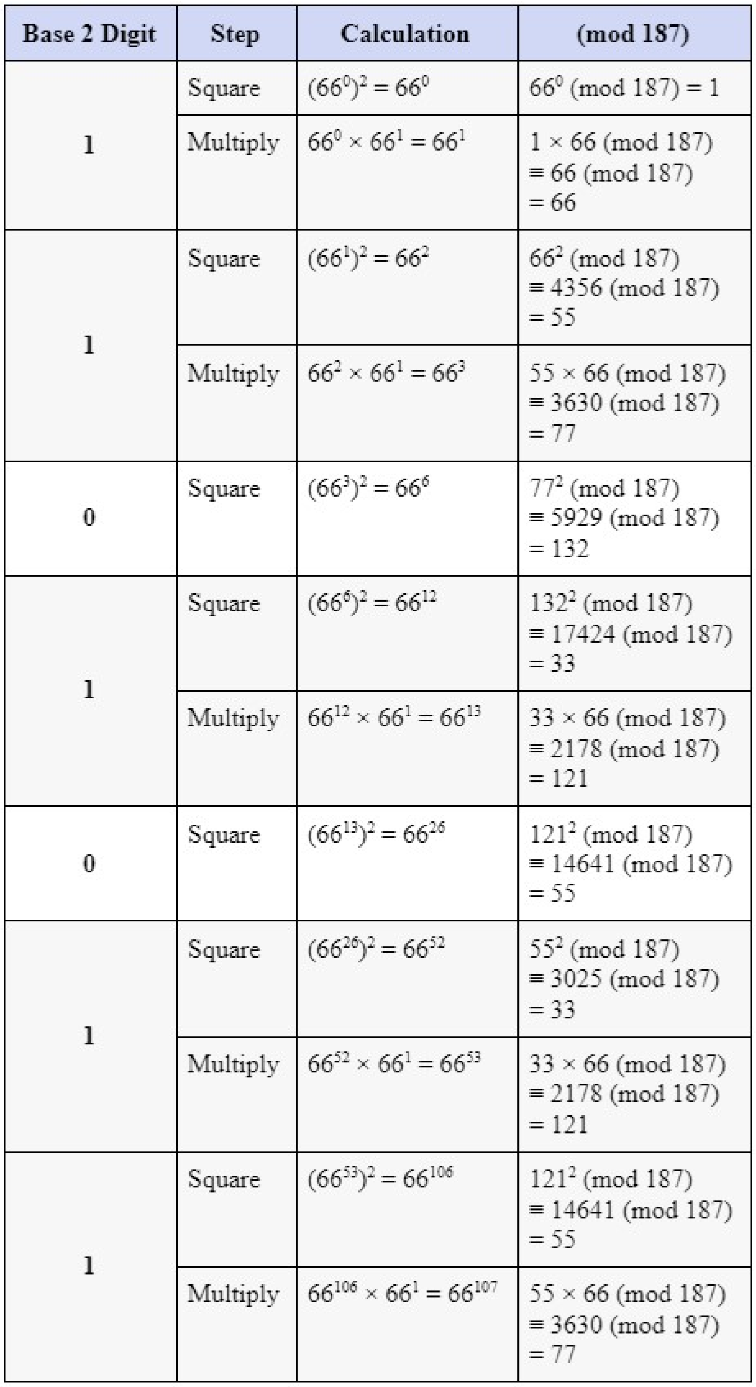
Converting the exponent, 107, to base two: $107_{ten} = 1101011_{two}$

First, let us show that the "square and multiply" method from above works to reach 66^{107}:



Base 2 Digit	Step	Calculation
1	Square	$(66^0)^2 = 66^0$
	Multiply	$66^0 \times 66^1 = 66^1$
1	Square	$(66^1)^2 = 66^2$
	Multiply	$66^2 \times 66^1 = 66^3$
0	Square	$(66^3)^2 = 66^6$
1	Square	$(66^6)^2 = 66^{12}$
	Multiply	$66^{12} \times 66^1 = 66^{13}$
0	Square	$(66^{13})^2 = 66^{26}$
1	Square	$(66^{26})^2 = 66^{52}$
	Multiply	$66^{52} \times 66^1 = 66^{53}$
1	Square	$(66^{53})^2 = 66^{106}$
	Multiply	$66^{106} \times 66^1 = 66^{107}$

Figure 4: You will see that we have indeed reached 66^{107} by following this method. Now, applying the (mod 187) to the result of each preceding step, the process will look like this:



Base 2 Digit	Step	Calculation	(mod 187)
1	Square	$(66^0)^2 = 66^0$	66^0 (mod 187) = 1
	Multiply	$66^0 \times 66^1 = 66^1$	1×66 (mod 187) $\equiv 66$ (mod 187) $= 66$
1	Square	$(66^1)^2 = 66^2$	66^2 (mod 187) $\equiv 4356$ (mod 187) $= 55$
	Multiply	$66^2 \times 66^1 = 66^3$	55×66 (mod 187) $\equiv 3630$ (mod 187) $= 77$
0	Square	$(66^3)^2 = 66^6$	77^2 (mod 187) $\equiv 5929$ (mod 187) $= 132$
1	Square	$(66^6)^2 = 66^{12}$	132^2 (mod 187) $\equiv 17424$ (mod 187) $= 33$
	Multiply	$66^{12} \times 66^1 = 66^{13}$	33×66 (mod 187) $\equiv 2178$ (mod 187) $= 121$
0	Square	$(66^{13})^2 = 66^{26}$	121^2 (mod 187) $\equiv 14641$ (mod 187) $= 55$
1	Square	$(66^{26})^2 = 66^{52}$	55^2 (mod 187) $\equiv 3025$ (mod 187) $= 33$
	Multiply	$66^{52} \times 66^1 = 66^{53}$	33×66 (mod 187) $\equiv 2178$ (mod 187) $= 121$
1	Square	$(66^{53})^2 = 66^{106}$	121^2 (mod 187) $\equiv 14641$ (mod 187) $= 55$
	Multiply	$66^{106} \times 66^1 = 66^{107}$	55×66 (mod 187) $\equiv 3630$ (mod 187) $= 77$

Figure 5: As you may recall, the final answer of 77 is identical to the answer computed in Example 7b. Try out the fast modular exponentiation algorithm with the other letters in ***Example 7b*** and see if you arrive at the right answer!

References

[1] Tutorialspoint. *Origin of Cryptography.* URL: `https://www.tutorialspoint.com/cryptography/origin_of_cryptography.htm`.

[2] Gustavus J. Simmons. *RSA Encryption.* July 2009. URL: `https://www.britannica.com/topic/RSA-encryption`.

[3] Michael Calderbank. *The RSA Cryptosystem: History, Algorithm, Primes.* Aug. 2007. URL: `https://www.math.uchicago.edu/~may/VIGRE/VIGRE2007/REUPapers/FINALAPP/Calderbank.pdf`.

[4] *What is Public-key Cryptography?* URL: `https://www.globalsign.com/en/ssl-information-center/what-is-public-key-cryptography`.

[5] Noa Bar-Yosef. *Understanding Public Key Cryptography and the History of RSA*. Feb. 2012. URL: `https://www.securityweek.com/understanding-public-key-cryptography-and-history-rsa`.

[6] *Leonard Adleman.* 2018. URL: `https://www.invent.org/inductees/leonard-adleman`.

[7] ArtOfTheProblem. *Public Key Cryptography: RSA Encryption Algorithm.* July 2012. URL: `https://www.youtube.com/watch?v=wXB-V_Keiu8`.

[8] Josh Lake. *What is RSA encryption and how does it work?* Mar. 2021. URL: `https://www.comparitech.com/blog/information-security/rsa-encryption/`.

[9] Brilliant Math Science Wiki. *RSA Encryption.* URL: `https://brilliant.org/wiki/rsa-encryption/`.

[10] *The complete table of ASCII characters, letters, codes, symbols and signs.* URL: `https://theasciicode.com.ar/`.

[11] Sarah Simpson. *Cryptography in Everyday Life.* Apr. 1997. URL: `https://www.laits.utexas.edu/~anorman/BUS.FOR/course.mat/SSim/life.html`.

[12] YK Sugi. *What is a quantum computer?* Oct. 2018. URL: `https://www.freecodecamp.org/news/what-is-a-quantum-computer-explained-with-a-simple-example-b8f602035365/`.

[13] Amit Katwala. *Quantum computing and quantum supremacy, explained.* May 2020. URL: `https://www.wired.co.uk/article/quantum-computing-explained`.

[14] John Loeffler. *How Peter Shor's Algorithm Dooms RSA Encryption to Failure*. May 2019. URL: `https://interestingengineering.com/how-peter-shors-algorithm-dooms-rsa-encryption-to-failure`.

[15] Andreas Baumhof. *Breaking RSA Encryption – an Update on the State-of-the-Art*. June 2019. URL: `https://www.quintessencelabs.com/blog/breaking-rsa-encryption-update-state-art/`.

[16] Dan Garisto. *Quantum computers won't break encryption just yet.* May 2020. URL: https://www.protocol.com/manuals/quantum-computing/quantum-computers-wont-break-encryption-yet.

[17] Maria Korolov and Doug Drinkwater. *What is quantum cryptography? It's no silver bullet, but could improve security.* Mar. 2019. URL: https://www.csoonline.com/article/3235970/what-is-quantum-cryptography-it-s-no-silver-bullet-but-could-improve-security.html.

[18] Ben Lynn. *Number Theory - Euclid's Algorithm.* URL: https://crypto.stanford.edu/pbc/notes/numbertheory/euclid.html.

[19] *The Euclidean Algorithm.* 2014. URL: https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/the-euclidean-algorithm.

[20] Brilliant Math Science Wiki. *Euler's Theorem.* URL: https://brilliant.org/wiki/eulers-theorem/.