THE APPLICATION OF NUMBER THEORY IN CRYPTOGRAPHY

Sean McCaffrey

April 2025

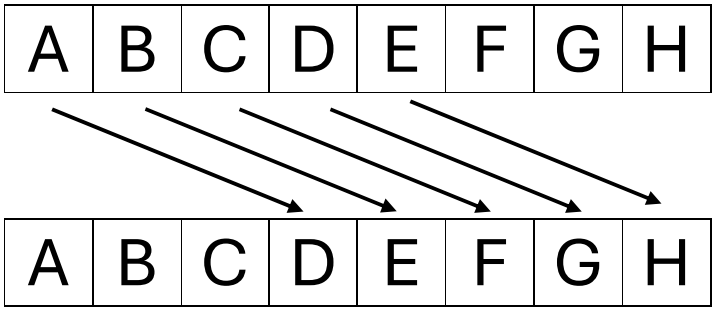
1. What is cryptography?

Cryptography is the study of techniques for securing information and data. It comes from the Greek roots “crypt” – meaning hidden, and “graphy” – meaning writing. It is where pure mathematics and computer science meet and make way to a beautiful and intricate science, and one that affects our daily lives much more than you would think. It is used everywhere in the cyber world, from secure communication of what’s for dinner all the way to war plans, and in protecting online banking transactions.

Cryptography has evolved from mechanical means, like the Romans using a stick and length of leather to encode messages, all the way to the enigma machine of the Nazi’s, and into the digital world using systems like the *Diffie-Hellman key exchange* and *RSA*. It will inevitably continue to evolve through the generation of *quantum computers*

Picture yourself sitting in school, bored out of your mind (probably English class), and your friend is sitting across the room. Let’s come up with a way to communicate with them! As they are much too far away to verbally communicate, we will have to rely on written communication, however simply recording our thoughts will not suffice as we run the risk of the teacher intercepting the message, and then getting into trouble! We will have to devise a method of making our conversations much more secretive…

A very simple but effective method we could try is a shift cipher which replaces each letter in our message by another which is a set number of positions along in the alphabet, for example, if we shift each letter right by 3, we would develop the following encoding:



In this case, A becomes D, B becomes E and so forth. This would result in a word, such as HELLO, transforming into KHOOR. This forms the basis of cryptography, and this particular method is dubbed the “*Caesar cipher”*. It is one of the most basic forms of encryption. Encryption is the process of converting our *plaintext* into *ciphertext*. This prevents unauthorized parties accessing the information. Decryption is the reverse. We take the ciphertext and undo the transformations to restore our valuable information.

Now we can successfully send messages in secret! However, this method is not very secure. It will fall apart if the unauthorized party discovers the method of encryption, which could be very easily done by brute force, as then they could easily undo the encryption. To protect more valuable data, we'll need a more secure method of encryption! To develop more secure techniques, we will require some rather complex mathematics, so let's delve into the world of *number theory* for more tools.

2. What is number theory?

Well that seemed pretty easy right? And where does the *actual* maths even come into this? Fasten up because its only getting harder from here…

Number theory is the study of integers and arithmetic functions. Simple! I know all the numbers: 1,2,3,4….99,100… a million. And isn't arithmetic just addition, multiplication and the likes? Well, that is correct… however it goes much, much deeper than that. It is in all essence, *pure,* as 20th century number theorist G.H Hardy would call it. However, he also claimed that it would have no real-world applications for good *or* evil, and he was wrong on that. Number theory explores the connections and relationships between integers (whole numbers), for example prime numbers, which make up a great portion of number theory.

Number theory plays a critical role in cryptography and serves as a tool for encrypting and decrypting valuable information. It gives us clues into the world of prime numbers, rational numbers and algebraic integers (relates to complex numbers). It yields many methods of encrypting our sensitive information to protect it from prying eyes. Here's where the actual mathematics begin. We need to develop the building blocks we'll use to construct our encryption.

2.1 Prime numbers

Prime numbers will form our simplest yet most invaluable tool in navigating the world of cryptography. A prime number is a number with only two factors – itself and 1. They are the most basic form of numbers and are the building blocks of mathematics as a whole.

They range from 2,3,5… to 317 all the way to $2^{136279841}$-1(a number with 41,024,320 digits when written in base 10) and beyond!

“317 is a prime, not because we think so, or because our minds are shaped in one way rather than another, but because it is so, because mathematical reality is built that way.” – G.H Hardy

Prime numbers are special in more ways than one, but one of the most familiar and relevant properties is the ability to rewrite any number in terms of its prime factors, and that its prime factors are completely unique to it. A number which is not prime is dubbed a *composite* number, and this is because it can be written as the product of specific primes, like 18 can be written as 2 x 3 x 3, or how 32244727 ≡ 4159 x 7753. As numbers grow larger and larger they become increasingly difficult to split into its prime factors and this will form the basis for later methods of cryptography.

In fact, it has been shown that there are an infinite number of primes to be found. A form of Euclid’s proof beautifully demonstrates this:

• Let's suppose there are a finite number of primes; $p_1, p_2, p_3 \ldots p_n$.

• Let P be the product of all these primes.

• Let q = P + 1, and q will either be prime or not.

If *q were* not prime, then it would divide by some prime, so let’s attempt to rewrite q as a product of its prime factors. We may attempt this by dividing q by all of our finite list of primes to find which ones are factors. In this case let p denote one of our many finite primes:

$$\frac{q}{p} = \frac{\boldsymbol{P}}{p} + \frac{1}{p}$$

As any of our primes divide P, we will be left with an integer quotient but with a remainder of 1, no matter what value of p we select from our finite set. It is clear q will leave remainder 1 when divided by any of these numbers, and because any number can be written as the product of its prime factors, this indicates one of two things:

1) If q is not prime there exists a prime larger than p_n , that will divide q, or

2) That q is prime

This shows there is at least one more prime number than we originally thought, and so p_n is not the largest prime. This can be scaled infinitely to show that there will always be another, larger prime.

This is useful in exploring cryptography as frequently huge prime numbers are used. This, alongside the fact that any non-prime can be factorised by a set of prime numbers, is key to beginning to understand cryptography.

2.2 Modular arithmetic

The next tool we will have to explore is modular arithmetic. Despite sounding so alien it is actually a basic concept to grasp and in fact we use it every single day…

Take a twelve-hour clock, and let's say the time reads *9 o'clock*. The question is simple; What will the time be in 6 hours time? Fret not as this is not a trick question – the answer is *3 o'clock*! This is the basis of modular counting. We take a set number, which we will call our *modulus,* and we consider what the remainder will be when we divide any chosen number by it. In this case, 3 is the remainder of 15 (9 + 6) given a modulus of 12.

When we are first taught division, we are told one number divided by another will produce a *quotient* and a *remainder (*which may be 0), however, as we move through school, we abandon this and are told to instead consider the result in decimal or fractional form. But number theory is the study of *integers* so we must use whole numbers and so return to the quotient and remainder.

In the case of modular arithmetic, we really focus in on the remainder of the result. If we were to consider the modulus of *4,* we could picture it in a diagram, like a circle sliced into four:

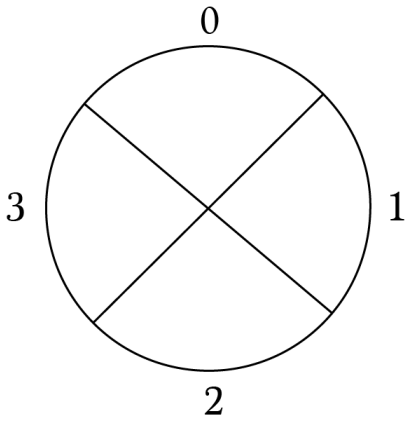


Diagram from medium.com

By imagining starting at *0* and taking a set number of steps *(n)* round the clock, we will end up at the remainder we would get from dividing *n* by the modulus. Take *5,* for example, if we were to start at zero and moved round the clock 5 steps, we would end up at *1,* precisely the remainder we'd get when dividing 5/4. We may also consider a larger number, *19*. This time we would go round the clock 4 times and end up at three, which is congruent with 19/4 which equals 4 remainder 3.

The formal notation for this is $a \equiv b \pmod{m}$, where *a* is the number we are dividing, *b* is the remainder, and *m* is the modulus, which is the number of hours on the clock that we may imagine, like the one above with a modulus of *4*. So this means that $5 \equiv 1 \pmod 4$ and $19 \equiv 3 \pmod 4$. Modular arithmetic will prove very useful and has some rather unique properties.

Consider a clock which has a prime number of hours, *p*. When we take a number, *a* and raise it to the power of *p*, we will find that when we move this number of hours round the clock, we will end back up at *a*. This means $a^p \equiv a \pmod p$. We can test this using numbers like *a=2* and *p=5;* $2^5=32$. When 32 divided by five we get a remainder of *2*. In fact, as one increases the exponent the result on the clock seems to map out a pattern:

$2^1 = 2 \equiv 2 \pmod 5$ $\quad$ $2^6 = 64 \equiv 4 \pmod 5$

$2^2 = 4 \equiv 4 \pmod 5$ $\quad$ $2^7 = 128 \equiv 3 \pmod 5$

$2^3 = 8 \equiv 3 \pmod 5$ $\quad$ $2^8 = 256 \equiv 1 \pmod 5$

$2^4 = 16 \equiv 1 \pmod 5$ $\quad$ $2^9 = 512 \equiv 2 \pmod 5$

$2^5 = 32 \equiv 2 \pmod 5$ $\quad$ $2^{10} = 1024 \equiv 4 \pmod 5$

Clearly, we can see a *pattern* emerge. Every time we increase our exponent by 1, we seem to follow a pattern of which hours we hit on the clock before we end up right back where we started, at *2*. Interestingly this occurs when our base is raised to the modulus, *p*, and when *p* is a prime number, which is congruent with the above equality: $a^p \equiv a \pmod p$. What was also strange was that the remainder seemed to go to *1* at a certain power, then the next would equal *a,* and the pattern would repeat. We can see this with other numbers like *a=3* and *p=13*: $3^{12} \equiv 1 \pmod{13}$, $3^{13} \equiv 3 \pmod{13}$. This will also work for any prime *p* number of hours on the clock.

This was discovered by *Pierre de Fermat* and is expanded upon into Fermat's Little Theorem.

2.3 Fermat's Little theorem

In the 17th century an incredibly famous mathematician by the name of *Pierre de Fermat* wrote a letter to his compatriot, Frénicle de Bessy, that he had discovered something rather interesting. He essentially claimed;

"If p is a prime and a is an integer not divisible by p, then $a^{p-1} - 1$ will then be divisible by p"

Remarkably, he refused to elaborate further due to a lack of space to explain his statement (not unlike him), and instead it was left up to Euler a century later to publish a proof. It is dubbed "Fermat's Little Theorem" so as not to confuse it with his other theorem he also refused to elaborate due to a lack of space, "Fermat's Last Theorem".

However, the statement above is only true if p and a are *coprime*, meaning that the only common divisor to them both is 1. There is a similar generalised statement that will work for any value of p and a;

"If p is prime and a is an integer, then $a^p - a$ is an integer multiple of p"

This can be rewritten is our modular form;

$$a^p \equiv a \pmod{p}$$

This is because if $a^p - a$ is divisible by p, if we add another a to it the remainder will then also equal a. Truly it is fascinating that this happens, and it is one of the many wonders of number theory. The proof is out of the scope of this essay, but is nonetheless very intriguing. While this is interesting, we still have a bit of manipulation to do before we can exploit it, and who else was better to do that then the great Leonard Euler, the Swiss mathematician so great he has his own number! This can all be explained by Eulers Theorem.

2.4 Euler's Theorem

A century later, Euler picks up where Fermat left off, and he graced us with the amazing Euler's Theorem. Before we explore that we will have to understand Euler's Totient Function.

Euler's Totient Function, which is denoted by $\phi(n)$ (pronounced 'phi of n'), essentially counts all the integers between 1 and n, that are *coprime* to n, meaning that the only common divisor they share is 1. For example $\phi(10) = 4$, as the integers between 1 and 10 coprime to 10 are 1,3,7,9. What is important is that when n is prime, which we will call p, then $\phi(p)$ will be one less than the prime, as every number less than p will be coprime to p, and we exclude the number itself, so $\phi(p) = p-1$.

Where this truly was able to come together is Euler's theorem, and he claimed when *a* and *n* are coprime:

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

This generalisation could account for values of n that weren't prime as Fermat's original statement showed, however, when we plug in a prime value of *n*, we get back out Fermat's Little Theorem. Yet again we can develop on this further, and let *N* equal the product of two primes, p & q, therefore $\phi(N)$ will be equal to (q-1)(p-1). This happens because *N* only has two factors (excluding one and itself), which are *p* and *q*, meaning it will share a common divisor with every multiple of *p* up to and including *N*, and every multiple of *q* up to and including *N*. There are *p* multiples of *q* up until *N*, and *q* multiples of *p*, so we can rewrite the total totient function when n is prime as; *pq-p-q+1*. The plus one will account for the overlap between the *-p* and *-q* as they both include *N* as their multiples. Factorising this will result in (*p-1*)(*q-1*). So finally, we can see that when *N* is the product of two primes, *p* & *q*, and when *a* and *N* are coprime, that:

$$a^{(p-1)(q-1)} \equiv 1 \pmod{N}$$

We can now relate this back to an interesting property of modular arithmetic that as we increase the exponent of the base, the hour on the clock appears to map out a pattern. Euler was able to discover that when our clock has *N* hours (*N= p x q),* when our base *a* is raised to *(p-1)(q-1)+1*, then the remainder will equal *a*. So; $a^{(p-1)(q-1)+1} \equiv a \pmod{N}$

$$a^{(p-1)(q-1)+1} \equiv a \pmod{N}$$

There we have it. One of the most invaluable theorems in cryptography and will act as the foundations of one of the most widely used cryptographic systems - RSA.

2.5 Modular multiplicative inverse

The modular multiplicative inverse of any number *a*, with respect to a modulus *m*, will be the number, *x ,* that when multiplied with *a* is congruent with 1 (mod *m*):

$$ax \equiv 1 \pmod{m}$$

A modular multiplicative inverse will only exist if both *a* and *m* are coprime. The inverse of a given number *a* with respect to modulus m, may be found using the *Extended Euclidean Algorithm*.

This is a simple but invaluable tool in the process of generating keys for the RSA cryptosystem.

2.6 Factorising big numbers

Factorising a number can be simple. Take a number, *n*, and suppose we want to factorise it. We could try dividing it by some primes, let's start at 2,3,5,7… When we find one that works, we record that prime and then repeat the process again until the quotient itself results in a prime number. This works very well on small numbers, and we would have no problem finding its prime factors. But what if our number has >100 digits? We would be here for quite a while. Even the most powerful computers on earth would struggle with a number such as this, so we must find a new method.

Well, to put it simply, we *haven't* yet produced an efficient algorithm for factorising a large number, but it hasn't been proved that such an algorithm does not exist. But this is not a bad thing, see the presumed difficulty of this problem forms the backbone for why cryptosystems like *RSA* work, and in fact, if an efficient algorithm were to be constructed it could completely ravage our digital world as a whole – and this is where quantum computing poses a threat.

The hardest type of number to factorise are *semiprimes*, an example of which is *N* from Euler's theorem above. That is the product of two primes, which are randomly selected, large enough and are of similar size (but not too close or else they may be factorised by special means), will be so difficult to factorise by modern classical computers that it is impractical to do as it takes so much time. Given enough time, a computer may be able to factorise any number, but it will take so long that it just isn't *feasible*. In fact, as the digits of the primes increase, the time taken to factorise their product increases *drastically*.

So far, the most efficient algorithm produced to factorise numbers larger than 10^{100} is the *General Number Field Sieve*. It is our fastest method but still far from perfect.

Now we have all of our number theory techniques to help us in our quest to understanding cryptosystems. Cryptography and computer science as a whole is inarguably rooted in pure mathematics, and it is extremely intriguing.

2.7 Number theory summary

Before we delve into the complex world of cryptosystems, lets summarize our number theory tools:

Prime number: a number with only two divisors – itself and *1*. Any number that isn't prime can be split into its prime factors, and the larger you go the harder it is to factorise a number.

Modular notation: $a \equiv b \pmod{m}$, where b is the remainder of dividing a by m.

Fermat's Little theorem: If p is a prime and a is an integer not divisible by p, then $a^{p-1} - 1$ will then be divisible by p. This may be expressed in two ways depending on a and p:

1) When a and p are coprime: $a^{p-1}-1 \equiv 1 \pmod{p}$

2) When p is prime then for any integer a: $a^p \equiv a \pmod{p}$

Euler's Theorem:

1) For values when a and n are coprime integers: $a^{\phi(n)} \equiv 1 \pmod{n}$

2) When p and q are prime, and $N = p \times q$, $a^{(p-1)(q-1)} \equiv 1 \pmod{N}$, which will lead to: $a^{(p-1)(q-1)+1} \equiv a \pmod{N}$

Modular Multiplicative Inverse: For any coprime integers a and m, the modular multiplicative inverse of a with respect to modulus m, will be x, so that when a is multiplied by x it is congruent to $1 \pmod{m}$: $ax \equiv 1 \pmod{m}$

3. Cryptography revisited

We know that "cryptography is the study of techniques for securing information and data". Cryptography allows us to transmit secure information over insecure channels. It takes on many forms and applications, such as symmetric and asymmetric cryptography.

The symmetric type is what we explored at the beginning, where we transform our data in a specific way to make it unreadable, then we put it through the opposite transformation to get our valuable data back. This method is quick easy but unreliable in the digital world due to the sheer force of computers.

Asymmetric cryptography on the other hand uses two different keys; a *public key*, and a *private key*. Anybody has access to the public key, and it is used to encode the data, but here's the catch – this time, this same key won't decrypt the data. Instead, the receiving party will have a key, the *private key*, which is specific to the algorithm used to encrypt the data and can decrypt the data. This forms the basis for *public key cryptography*, which involves methods like RSA. We will look at one form of modern symmetric cryptography and one form of modern asymmetric cryptography.

3.1 Diffie – Hellman Key Exchange

The “key” in a cryptosystem determines how the data is to be transformed by the system. In a symmetric system the same is used to encrypt and decrypt the data, so two parties will have to meet in person and agree on a key they will use to transmit data. However, this simply is not feasible in the huge online world, and so a method of creating a secure key both parties have was needed, but without ever sharing it online. This is where the Diffie – Hellman Key Exchange comes in;

Whitfield Diffie, when he published his 1976 paper “New directions in cryptography” originally used this analogy to describe the Diffie – Hellman key exchange:

Picture two people, Alice and Bob, who together want to make a secret colour of paint. They both will publicly agree on what colour to start with, say *yellow*. They will both also have their own *secret colour*. In this case let's say Alice has *red* and Bob has *blue*. The first step to creating their shared secret colour is to mix their private colour with their public colour, giving Alice orange and Bob green. They will then publicly swap their new paints, and once again will add their private colour to the mix. Now they will both end up with a common colour, a *dark green/brown*. Although the original colour was public, only Alice and Bob are aware of what colours were added to it and so the final colour remains “secret”. However, this analogy has its flaws, because the intermediate colours of orange and green will be public. By using basic colour theory one many deduce what colours have been mixed with yellow, as one should know what mixed with yellow gives red.

The actual process does not have this flaw however (thankfully). It relies on *modular* arithmetic and its properties surrounding exponentiation. The true process goes like this:

-Alice and Bob pick set parameters, say a modulus of 23 and a base of 5

-Alice will then pick a random integer, say $a=4$, and will compute $5^4 \pmod{23}$ which is equal to four, which we will set equal to A, $A=4$.

-Bob will then pick his secret number, let's say $b=3$, and so he will compute $5^3 \pmod{23}$, which will give 10 and we will set equal to B, $B=3$.

-Bob and Alice will now publicly trade B and A

-Alice can now calculate the secret key using $B^4 \pmod{23}$ which when B=10 will give 18.

-Bob can also calculate the secret key with $A^3 \pmod{23}$ which when A=4, gives 18.

Et viola, they both have a secret key they can use for encryption and decryption without ever meeting face to face. This works because of an *interesting* equivalence:

When g and p are the set parameters, being the base and modulus respectively, and when A and B are the intermediate keys:

$$A^b \pmod{p} = g^{ab} \pmod{p} = g^{ba} \pmod{p} = B^a \pmod{p}$$

This works because of strange rules when it comes to modular exponentiation. Much larger values will need to be used, as the small numbers used here would be easy to crack in no time, however the base is only ever needed to be a small integer. The Diffie – Hellman Key Exchange is invaluable for the secure creation of keys to then be used in the secure transmission of data.

3.2 RSA

Now on to the next example of a popular cryptosystem, this time one that employs an *asymmetric* means of encryption. It was developed in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman. RSA is an acronym of the creators' names – Rivest, Shamir, and Adleman. It is asymmetrical public-key cryptosystem that is widely employed in protecting our data. As it is an asymmetrical system, there is a public key, used to encrypt the data, and a private key only the recipient will have, and it is the only method to decrypt the data.

RSA is primarily based on the difficulty of factorising big numbers, as it is relatively easy to multiply two primes together but given the result it is extremely difficult to derive the primes.

A great analogy for how the RSA cryptosystem works appears in Marcus du Sautoy's book, *The Music of the Primes*. It goes as follows:

Suppose we are keen to encrypt a credit card number. The RSA cryptosystem will act rather like a magician performing a card trick, however it is not so simple, as our deck of cards will contain over *100 digits* of cards in decimal ($>10^{100}$). The customer's credit card will be one of these playing cards. They will place their card at the top of the deck, and the website will shuffle the pack, so the location of the credit card seems to be completely lost, and any hacker is faced with the impossibly task of finding the single card from the entire pack. However, the website has a trick up its sleeve. By exploiting *Fermat's Little Theorem*, it knows the exact sequence of shuffles it can use to make the costumer's card magically reappear at the top of the pile. The way the deck is shuffled the first time depends on the public key, but the original card (the credit card number) will determine where in the deck it ends up. The website will know how to make the card reappear at the top of the deck, and this is determined by our *private key*.

The shuffling of the deck is determined by a mathematical calculation, but this isn't any normal calculation – instead it uses *modular arithmetic*. This means the initial calculation will be easy to perform, but super difficult to undo, unless you have access to the secret key.

Fermat and Euler were able to determine that given any base a and a modulus n, there will be a specific exponent to raise our base to for it to spit back out a. RSA splits this exponentiation into two parts. Our base is first raised to an encrypting number, E, which

transforms it in such a way making it very difficult to undo. We when then raise this to our decrypting number, D, and we get back out our base.

We will continue to look at the example of a customer making a purchase with their credit card number, *C*:

When a company wants to develop a public and private key to use the RSA cryptosystem, it must go through the following steps; First, it will need to choose a specific number of hours to use on the modular clock, *N*. The company picks this number by multiplying two huge primes (around 60 decimal digits), *p* and *q*. *N* will now have over 120 decimal digits. This number is made public, but *p* and *q* are kept private. The difficulty of factorising large numbers means that *p* and *q* will remain private for a very long period of time. To create the public and private keys, *p* and *q will* be used so this is why it is important they are kept secret.

An encoding number, *E,* needs to be developed. This number will be what *C* is raised to in order to encrypt it. The number *E* will be the same for everyone and will act as our *public key*. *E* is usually selected as a relatively small prime number and must satisfy two conditions; it must be a positive integer smaller than $\phi(N)$, and it must also be coprime to $\phi(N)$. *E* is chosen to be as computationally efficient as possible, so 65537 is often chosen. This is because $65537 = 2^{16}+1$, giving it a short bit length and a low hamming weight (the larger the hamming weight the more bits set to *1* in a string of binary, so $2^{16}+1$ has a hamming weight of 2).

Now when a costumer makes a purchase online, in order for the website to encrypt their credit card number, *C*, the website will take *C*, and raise it to the power of *E* with respect to modulus *N*. Let this encrypted number equal *m*:

$$C^E \pmod{N} \equiv m$$

The number *E* and *N* are both public to everybody, and *m* will travel through insecure channels so it will also be seen by unauthorised parties, however even given *E* and *N*, it is nearly impossible to calculate what *C* might have been. This is because of the nature of exponentiation when it comes to modular arithmetic, as the result does not continue to grow like it does with normal exponentiation. The original numbers 'tracks' are seemingly covered up behind it each time it makes a loop of the modular clock. The only feasible way to make *C* appear again will be to create a number, *D*, so that when *m* is raised to the power of *D,* we get *C* back out again.

Now for the website to create its decrypting number, *D*. It will need to take advantage of of Euler's Theorem that $a^{\phi(n)} \equiv 1 \pmod{n}$. This should also works for any multiple of $\phi(n)$, and we will want it in the form $a^{\phi(n)+1} \equiv a \pmod{n}$, where the base *a* will be our credit card number and $\phi(N)+1$ will equal $E \times D$, so; $\phi(N) + 1 = E \times D$. Rearranging this we get $E \times D - 1 = \phi(N)$, therefore thanks to Fermat's Little Theorem we can conclude:

$$E \times D \equiv 1 \pmod{\phi(N)}$$

Hence *D* is the modular multiplicative inverse of *E,* and so the company can be able to calculate it using the Extended Euclidean algorithm. Even though both *E* and *N* are public, the difficulty of calculating $\phi(N)$ is what makes the decrypting number so secure. The website may calculate it easily as it knows the secret primes, *p* and *q*, so it

can work out $\phi(N) = (p-1)(q-1)$. However, for someone who only knows N, which may have around 120 decimal digits, calculating $\phi(N)$ will be next to impossible, and so D remains secure.

After our encrypted credit card number, m, reaches the website across the insecure channel, the website will now decrypt it my raising m do the power of D with respect to modulo N:

$$m^D \pmod{N} \equiv C$$

And there we have it. The credit card number will have been securely transmitted over an insecure channel and has been successfully protected from unauthorised parties.

If we sub in $C^E \pmod{N} \equiv m$ we can see more clearly what is going on across the whole process:

$$C^E \pmod{N} \equiv m$$

$$m^D \pmod{N} \equiv C$$

$$\therefore (C^E)^D \pmod{N} \equiv C \Rightarrow C^{E \times D} \pmod{N} \equiv C$$

And there we have it. RSA is an intrinsically secure and wonderfully complex cryptosystem that is still widely employed today. Today it differs slightly from how I explained, for example in the original RSA paper they used Euler's Totient Function to calculate *E and* D, however nowadays *Carmichaels Totient Function* is used, denoted by $\lambda(n)$. This works because $\phi(n)$ is always divisible by $\lambda(n)$, but using $\lambda(n)$ is far more optimised.

The difficulty of calculating D is what makes this method so secure and so it is wholly reliant on the difficulty of factorising huge numbers, and therefore if an efficient means of factorising large numbers becomes available, then the cryptography world will be forced to evolve.

4. Conclusion

It is not hard to find a use for mathematics in any science, but none truly takes advantage of *pure* mathematics as much cryptography does. What once was a study with little real-world implications, *number theory* has stolen the spotlight in the world of late 20th and early 21st century mathematics and without its wonders the cyber world would not be the same as it is today. The field which sparked interest in many great mathematicians like Gauss, Fermat, Euler and G.H Hardy finally has a powerful and irreplaceable use, and we will need our best minds to keep pushing forward into the future as the game changes, and we know that it will. Quantum technology will change everything.

4.1 Implications of quantum technology

Quantum computers are being developed at an alarming rate, and it has been thought that they will completely change the game in medicine and engineering, and they will be able to accurately simulate many real-world interactions including at a molecular scale. They work by relying on two processes – *super position*, and *quantum entanglement*. This allows quantum computers to be able to perform some calculations *significantly* faster than classical computers can, and one of those calculations may be factorising large numbers.

In 1994 Peter Shor published a paper claiming that he had created a *quantum algorithm* that would be able to factorise large numbers in a time incomparable to classical computers. It has been theorised that a quantum computer using Shor's algorithm may be able to break an RSA key of 2048 bits (approximately 600 decimal digits) in mere hours or days, where it would've taken a classical computer *millions* of years. It is clear that quantum computers may completely undermine many cryptosystems such as RSA and Elliptic Curve Cryptography, as well as negatively affecting many others.

However, thankfully experts claim that quantum computers are developing at such a rate that they will not be a threat in the near future, and there are many extremely clever mathematicians and computer scientists working tirelessly to advance techniques that could keep our data secure in a post quantum world.

4.2 Personal thoughts

I believe that number theory is truly a fascinating field, and I look up to many of the greats that have contributed to it. Cryptography is such a beautiful application of the field, and it is interesting to be able to understand such a science that affects our daily lives so significantly yet doesn't seem to get the credit for it. I have thoroughly enjoyed researching and writing about it, and I would suggest readers to look deeper into other aspects of number theory and other cryptosystems such as Elliptic Curve Cryptography as well as the Advanced Encryption Standard. Looking forward into the future it will be compelling to observe the developments made by the greatest minds of our generation and the next.

References:

-A mathematician's apology – G.H Hardy

-The Music of the primes – Marcus du Sautoy

-New directions in cryptography - Whitfield Diffie and Martin E. Hellman

-A Method for Obtaining Digital Signatures and Public-Key Cryptosystems - R.L. Rivest, A. Shamir, and L. Adleman

-https://www.geeksforgeeks.org/fermats-little-theorem/

-https://www.geeksforgeeks.org/eulers-theorem/