

An Exploration of Image Encryption

Travis McVoy

January 30, 2024

Contents

1	RSA	2
1.1	Totient Function and PIE	2
1.2	Euler's Theorem	5
1.3	Pseudo Code	5
2	Image Encryption via RSA	6
2.1	Integer-Representation of Pixels	6
2.2	Scrambling vs. No Scrambling	7
2.3	Results	7
3	Conclusion	8
	References	8

1 RSA

In general, when we encrypt a message we wish to reversibly alter the message to the point that the encrypted message is meaningless to anyone that does not know the necessary decryption method. There are many ways to achieve such a goal, some more secure than others. RSA uses number theory to map an integer to a different integer and then back to the original integer. So, to encrypt a message with RSA we might first assign “A” to 0, “B” to 1, and so on to get

This is a message \rightarrow 197818 818 0 124181864.

From there, we may pad each number, combine all the digits to make one large number, or in some other way employ a defensive technique that ensures a message can actually be decrypted and is still secure (more on that later). For demonstrational purposes, I will simply encrypt each word as is. Doing so yields

This	\rightarrow	197818	\rightarrow	52935883302
is	\rightarrow	818	\rightarrow	5974917121
a	\rightarrow	0	\rightarrow	0
message	\rightarrow	124181864	\rightarrow	76083038779

where our encrypted message is now

52935883302 5974917121 0 76083038779.

We now wish to understand where these numbers come from and how we can convert them back to their original unencrypted form.

1.1 Totient Function and PIE

Euler’s Totient function, $\phi(n)$, is defined to be a function $\phi(n) : \mathbb{N} \rightarrow \mathbb{N}$ such that for some positive integer input n , $\phi(n)$ is the number of positive integers that are both less than and coprime to n .

We can derive a general expression for $\phi(n)$ using some clever counting. Namely, we recognize that if the number of positive integers less than or equal to n that are *not* coprime to n is m , then the number of positive integers less than and coprime to n is simply $n - m$.

To see that $n - m = \phi(n)$, consider some set

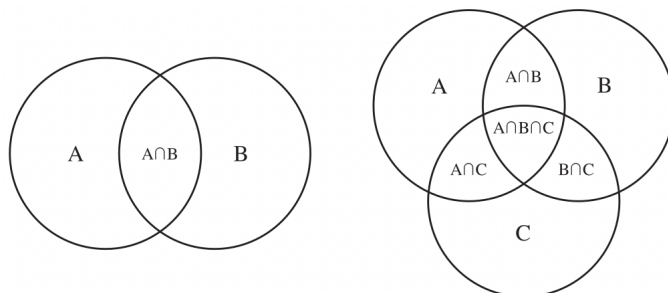
$$S_n = \{1, 2, \dots, n\}.$$

Let the set S_m be a subset of S_n such that S_m is the set of all elements of S_n that are not coprime to n . That is,

$$S_m = \{m_i : 0 < m_i \leq n \text{ and } \gcd(n, m_i) > 1\}.$$

Clearly, $|S_n| = n$, $|S_m| = m$, and $|S_n - S_m| = n - m = \phi(n)$ as desired.

We will now use the Principle of Inclusion-Exclusion (PIE) to find $|S_m|$. We can gain an intuitive understanding of PIE via venn diagrams. Let the diagrams below be for sets S_1 and S_2 where $A \cup B = S_1$ and $A \cup B \cup C = S_2$.



Source: WolframAlpha Venn Diagram Article

From the venn diagrams, we can see that

$$|A \cup B| = |A| + |B| - |A \cap B|$$

and

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|.$$

One may guess that we can chain this sort of thinking together for arbitrarily many subsets, and in fact, that is exactly the case (assuming the super set is finite).

Theorem. If $(A_i)_{1 \leq i \leq n}$ are finite sets, then

$$\left| \bigcup_{i=1}^n A_i \right| = \sum_{i=1}^n |A_i| - \sum_{i < j} |A_i \cap A_j| + \sum_{i < j < k} |A_i \cap A_j \cap A_k| - \dots + (-1)^{n-1} |A_1 \cap \dots \cap A_n|.$$

Proof. The point of PIE is to ensure that every element in the union of a collection of sets has been counted once and only once. Hence, if we can show that the above formula counts some arbitrary element a only once, then we can conclude it is valid. To do so, we must consider the individual sets that a is an element of. Namely, suppose a is in exactly k of the A_i sets where $1 \leq k \leq n$.

Since a is in k of the A_i sets, a is counted k times in the first sum. In the second sum, we consider an intersection between two sets, so the second sum contributes a negative count of $\binom{k}{2}$. Likewise, the third sum contributes a count of $\binom{k}{3}$ and so on until we get to the intersection of k sets which contributes a count of $(-1)^{k+1}$. In total, then, we see that a gets counted

$$\binom{k}{1} - \binom{k}{2} + \binom{k}{3} - \dots + (-1)^{k+1} \binom{k}{k}$$

times. The above expression reminds us of the Binomial Theorem [3] but it's missing a $\binom{k}{0}$ term. To fix this, we can simply add it and subtract it. Then, we can represent the above in the much simpler form $(1 - 1)^k + 1 = 1$. Thus, a is indeed counted only once and we may now use PIE to derive a formula for Euler's Totient Function.

Theorem. If n has the prime factorization $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ then

$$\phi(n) = n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right).$$

Proof. Since it is easier to count integers that are *not* coprime to n , we start by doing just that. We observe that there are $\frac{n}{p_i}$ integers less than or equal to n that share the factor p_i . Likewise, there are $\frac{n}{p_i p_j}$ positive integers less than or equal to n that share the factor $p_i p_j$, and so on. It follows from PIE that the number of positive integers less than or equal to n that share a factor greater than 1 is

$$\begin{aligned} m &= \frac{n}{p_1} + \frac{n}{p_2} + \cdots + \frac{n}{p_k} \\ &\quad - \frac{n}{p_1 p_2} - \frac{n}{p_1 p_3} - \cdots - \frac{n}{p_{k-1} p_k} \\ &\quad + \frac{n}{p_1 p_2 p_3} + \frac{n}{p_1 p_2 p_4} + \cdots + \frac{n}{p_{k-2} p_{k-1} p_k} \\ &\quad - \cdots \\ &\quad + (-1)^{k-1} \frac{n}{p_1 p_2 \cdots p_k}. \end{aligned}$$

Now that we know the number of positive integers less than or equal to n that are *not* coprime to n , we can find the positive integers less than n that *are* coprime to n using complementary counting. We have

$$\begin{aligned} \phi(n) &= n - m \\ &= n - \frac{n}{p_1} - \frac{n}{p_2} - \cdots - \frac{n}{p_k} \\ &\quad + \frac{n}{p_1 p_2} + \frac{n}{p_1 p_3} + \cdots + \frac{n}{p_{k-1} p_k} \\ &\quad - \frac{n}{p_1 p_2 p_3} - \frac{n}{p_1 p_2 p_4} - \cdots - \frac{n}{p_{k-2} p_{k-1} p_k} \\ &\quad + \cdots \\ &\quad + (-1)^k \frac{n}{p_1 p_2 \cdots p_k}. \end{aligned}$$

If we were to expand the equation stated in the theorem, we would find that the expansion is identical to the above. Thus, the proof is finished.

1.2 Euler's Theorem

Euler's Theorem provides us with the mathematical tools to implement RSA encryption. We state the theorem and provide a group theoretic proof below.

Theorem. (Euler) Let $a, m \in \mathbb{Z}$ with $m > 0$. If $\gcd(a, m) = 1$, then

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

Proof. Let G be the group formed between multiplication modulo m and the set $\{a_1, a_2, a_3, \dots, a_n\}$ such that each a_i is relatively prime to m . Then, $1 \pmod{m}$ is in G and more importantly, is clearly the identity. Notice that from our proof of $\phi(n)$ we must have $|G| = \phi(m)$. Now, consider some cyclic subgroup of G , $H = \langle a_h \rangle$. Let the order of a_h be k . Then, $|H| = k$ and $a_h^k \equiv 1 \pmod{m}$. From Lagrange's Theorem [4], we know that $|H|$ divides $|G|$ so it follows that

$$a_h^{\phi(m)} = a_h^{k\ell} = (a_h^k)^\ell \equiv 1^\ell \equiv 1 \pmod{m}.$$

Our selection of H was completely arbitrary, and indeed, one such H will exist for each a_i in G . For proof, observe that by definition G is finite. Since G is a finite group, we must have that a_i^k is in G for all $i \in \mathbb{Z}$ such that $1 \leq i \leq n$ and all $k \in \mathbb{Z}$. Thus, we can form the desired H for any such a_i and we have now established that for $\gcd(a, m) = 1$, we have $a^{\phi(m)} \equiv 1 \pmod{m}$ as desired.

1.3 Pseudo Code

To implement RSA, one needs a private and public key. A public key can be used by anyone to encrypt a message, whereas the private key is known only to the recipient of the message. The idea is to use exponentiation and reduction under a modulus to convert a number to a new number and then back to the original, as we saw in the beginning of the paper. Notice from Euler's Theorem, that if $(a, n) = 1$ then we have

$$a^{\phi(m)+1} \equiv a \pmod{m}.$$

Our first goal, then, is to find a modulus m such that any message a that we send is coprime to m . Our second goal is to find encryption and decryption exponents e and d such that

$$ed \equiv 1 \pmod{\phi(m)}.$$

Under such conditions, our encrypted message is simply $a^e \pmod{m}$ and our decrypted message is

$$(a^e)^d = a^{ed} = a^{k\phi(m)+1} = (a^{\phi(m)})^k a \equiv 1^k a \equiv a \pmod{m}.$$

We call the pair (e, m) a public key, which is available to anyone. The private key (d, m) is a secret kept by an individual with a public key. Then, when one wishes to securely

communicate with someone else, they use the recipient's public key to encrypt a message, and the recipient uses the private key to decrypt, as seen above. To generate keys, we follow a series of steps:

1. Choose two large and distinct primes p and q and let the modulus m be $m = pq$. Notice that if $a > m$, when we reduce mod m , information will be lost. So, at a minimum, we want the condition $a < pq$ to hold. Ideally, though, we would like primes large enough such that $a < p$ and $a < q$ for any message a we wish to send. Under such conditions, a will always be coprime to m and Euler's Theorem will therefore always hold (and consequently, so will the encryption/decryption scheme).
2. Choose an encryption exponent e such that $(e, \phi(m)) = 1$.
3. Find the inverse of e under mod m and let that be the decryption exponent d .

In practice, though, there may be extra steps one takes that are specific to their own goals. As we will soon see, our implementation requires some specific padding.

2 Image Encryption via RSA

We now wish to ask, can we encrypt images using an RSA scheme? And if so, what is the best way to do it? To start, we first must understand how images are stored in a machine.

When we look at an image on a screen, we are looking at a grid of colors. The more squares our grid has, the finer our image can be (much like how a Riemann sum is more and more accurate as we have more and more rectangles). In the grid, any given square is a pixel, and each pixel has what's called an RGB value. RGB stands for red, green, blue. We can think of an RGB value as a triple (r, g, b) where $r, g,$ and b denote the intensity of the color. One has some flexibility in how they wish to choose the scale of the intensity of r, g, b . For simplicity, we have opted to adjust the scale to represent percentages. That is, $0 \leq r, g, b \leq 100$ where a value of 100 corresponds to 100% and 0 corresponds to 0%. An RGB value of $(100, 0, 0)$ would then be a bright red pixel whereas $(50, 0, 50)$ would be a shade of purple.

2.1 Integer-Representation of Pixels

In practice, most images will likely have hundreds of thousands or even millions of pixels. Consequently, we will save a lot of computational effort by storing RGB values not as a triple, but as one single integer. For example, instead of $(12, 34, 56)$ we would use 123456. An astute reader may notice that a single digit representation may cause confusion if we are not careful. Consider the integer 12345. Does that correspond to the triple $(1, 23, 45)$ or

does it correspond to a different triple such as (12,3,45) or (12,34,5)? We can avoid confusion by adding some rules to ensure every integer representation of a pixel is always of the same form. To do so, we must handle three cases: the case where any value of the triple is a single digit, the case where the first value is zero, and the case where a value of 100 occurs. We can fix the first two with some padding and the last case with “truncation”. Suppose we want to represent (0, 0, 100) as a single six digit integer. Though it will lose a little precision, it is sufficient to first represent 100 as 99. Then, we would like to have 000099, but if that were stored in a computer as an integer, output would yield only 99. Such a loss of information is clearly not acceptable, but is easily fixed with the padding of a nonzero integer in the front. We made the choice of padding with 1. In general, we can represent each pixel with the form $1rrggbb$ where the first r, g, b denotes a padded zero if necessary (single digit case).

2.2 Scrambling vs. No Scrambling

When we encrypt an image, we may choose to scramble the pixels or not. If we scramble the pixels, we need a way of restoring orientation. A simple fix is to simply add the index in between the padded 1 and the six digit RGB value. So, if have some pixel at index i then that pixels generic representation is $1irrggbb$. Once we have decrypted the list, we can sort it using a merge sort algorithm.

2.3 Results

While this encryption scheme works, it is, regrettably, either quite slow or quite lacking in security (and even then it's still slow). If we were to use primes with 50 digits, decrypting a picture with only 1 million pixels will take roughly four minutes. If we were to attempt to speed up the process, we would need to go beyond simple algorithmic analysis and examine the performance of Python's `pow()` function. While `pow()` is *significantly* faster than exponentiating and reducing using `**` and `%`, it's still not fast enough to handle the scale of a secure RSA system. It is not clear to me which aspect slows down `pow()` the most. Ultimately, I believe the size of the primes are the culprit, as they determine not only the size of the encryption and decryption exponent, but also the size of the modulus with which we reduce. If our primes are really large, our modulus is also large, and so our encrypted values will also be quite large (around 50 digits). So, even if we somehow could find small encryption and decryption exponents, we may still have problems due to the size of the base we are exponentiating.

3 Conclusion

While I certainly wouldn't recommend this encryption scheme as a pragmatic solution to any real world problem, I still learned a lot while working on this project. In particular, I have a much, much greater appreciation of the importance of a proper literature search, and the speed at which one conducts a literature search. There came a point at which I became impatient which resulting in coding more, and reading less. Had I been more patient, I may have recognized in advanced that the scheme wouldn't be practical, and then could move forward from there.

References

- [1] *Elementry Number Theory* by James K. Strayer
- [2] *Cryptological Mathematics* by Robert Edward Lewand
- [3] *Intermediate Counting and Probability* by David Patrick
- [4] *Abstract Algebra* by Dummit and Foote