The Mathematics of RSA

Michael Hoefler

November 2022

Contents

1 Introduction

In our current day and age, cryptography and privacy are increasingly important matters. With an ever-increasing number of cyber-criminals looking to intercept sensitive data, we must ensure that our encryption schemes are up to par. But with quantum technology quickly evolving, many of our classic mathematical techniques in cryptography are under threat. In the following discussion we will inspect the mathematics behind the most common encryption mechanism: RSA.

2 Asymmetric Cryptography

Before we start discussing RSA, we must first understand Symmetric and Asymmetric Cryptography. Up until the late 20th century, cryptography consisted solely of symmetric algorithms.

Definition: Symmetric Cryptography

Symmetric Cryptography uses the same key to encrypt and decrypt messages. Suppose k is a secret key, M is a message, C is the ciphertext, and E and D are encryption and decryption functions, respectively.

$$E(M, k) = C$$

$$D(C, k) = M$$

To put this definition into perspective, consider the Caeser cipher, which takes letters of the alphabet and shifts them according to a key. The $k = 13$ case is a popular example, since applying the shift twice decrypts the message. So essentially $E = D$ for this algorithm, and it is often referred to as ROT13.

Clearly ROT13 is not a safe or secure encryption scheme, but there are many widely used symmetric algorithms. Several of these include AES, DES, and Blowfish. While these are very interesting examples, we will focus our discussion on Asymmetric cryptography.

Definition: Asymmetric Cryptography

Rather than using one secret key, Asymmetric Cryptography uses a pair of related keys for encryption and decryption. If we suppose E is an encryption method which uses some public key e, and D is a decryption method which uses a private key, d, we should have the following properties.

$$D(E(M)) = M$$

$$E(D(M)) = M$$

Also, E and D should be easy to compute, but revealing e should not disclose an easy way to compute d.

So what gives an asymmetric scheme an advantage over a symmetric scheme? If a user wants to communicate with lots of third-parties using a symmetric scheme, the user would need to have a separate key for every single communication (otherwise, one recipient could decrypt another recipient's communications). For large corporations like banks, the amount of keys to keep track of quickly gets out of hand.

With asymmetric encryption, the corporation can have one "public key" which they distribute to every client they want to talk to. The clients can use this key to encrypt their communications, and the bank uses their "private key" to decrypt them. With asymmetric encryption, each party only needs one key-pair to communicate securely!

Another advantage of asymmetric cryptography is digital signatures. If Alice wants to send a verified message, M, they can simply pass it into their decryption function, D. When Bob receives $D(M)$, they can use Alice's public encryption function, E, to recover the message: $E(D(M)) = M$. Since Bob can be sure that no one else could have used Alice's decryption function to generate $D(M)$, Bob can be sure that the message really came from Alice!

Now that we understand the rationale of asymmetric cryptography, it is time to look at a real example: RSA!

3 The RSA Algorithm

RSA [1] is named after its three creators, R. Rivest, A. Shamir, and L. Adleman, who came up with the method in 1977.

Figure 1: Shamir, Rivest, and Adlemen in 1977

The RSA algorithm uses modular math and number theory to achieve the goals of the asymmetric encryption scheme. The algorithm works as follows.

The RSA Algorithm

1. First, pick two large distinct primes p and q.
2. Compute $n = p \cdot q$, and $\phi = (p-1)(q-1)$.
3. Choose e such that $1 < e < \phi$ and $gcd(e, \phi) = 1$.
4. Compute d where $e \cdot d \equiv 1 \pmod{\phi}$
5. Then (e, n) is the public key, and (d, n) is the private key.
6. To encrypt a message, m, the user computes $c = m^e \pmod{n}$.
7. To decrypt a message, c, the user computes $m = c^d \pmod{n}$.

3.1 Correctness of the Algorithm

We will start off by showing that this method is correct.

3.1.1 Euler's Totient Function

You can see ϕ showing up multiple times in the algorithm. This is actually Euler's totient function evaluated at n.

Euler's Totient Function

We define $\phi : \mathbb{N} \to \mathbb{N}$ as follows.

$$\phi(n) = \#\{1 \leq m < n : gcd(m, n) = 1\}$$

The # symbol denotes the number of elements in the set. Note that by definition $\phi(p) = p - 1$ when p is a prime.

In *step 2*, we calculate $\phi(n)$ by taking $(p-1) \cdot (q-1)$. This comes directly from the following result.

Theorem: Multiplicity of Euler's Totient Function

Suppose we have p, q which are distinct prime numbers. Then we have the following.

$$\phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$$

Proof: Notice that $\phi(pq)$ is the number of elements in the following set which are relatively prime to pq.

$$\begin{Bmatrix} 1, & 2, & \cdots, & p, \\ p+1, & p+2, & \cdots & 2p, \\ \vdots & \vdots & \ddots & \vdots \\ (q-1)p+1, & (q-1)p+2, & \cdots & qp \end{Bmatrix}$$

The elements in the set which are *not* relatively prime to n (not including n itself) are in the following sets.

$$P = \{p, 2p, 3p, \cdots, (q-1)p\}$$

$$Q = \{q, 2q, 3q, \cdots, (p-1)q\}$$

We will show that these sets are disjoint. Suppose $a \in P \cap Q$. Thus $a \in P$ and $a \in Q$. Therefore $a = jp = kq$ where $j, k \in \mathbb{Z}$. By Euclid's Lemma, $p|a$ implies $p|k$ or $p|q$. Clearly $p \nmid q$ (since p and q are distinct primes), so $p|k$. But $0 < k < p$, so this is a contradiction. The sets P and Q must be disjoint.

So clearly $\phi(n) = pq - (|Q| + |P|) - 1$. Note that the extra -1 is accounting for n itself not being counted in Q or P.

$$\begin{aligned}\phi(n) &= pq - (p - 1 + q - 1) - 1 \\ &= pq - p - q + 1 \\ &= (p-1)(q-1) = \phi(p)\phi(q)\end{aligned}$$

Therefore $\phi(n)$ is multiplicative over the product of two distinct primes.

3.1.2 Modular Multiplicative Inverses

In *step 4* of the RSA Algorithm, we choose d such that $e \cdot d \equiv 1 \pmod{\phi(n)}$. We will show that such a d exists and is unique.

3.1.3 Bezout's Identity

Before we can prove d exists, we must show an important result relating linear combinations and greatest common divisors - Bezout's Identity.

Theorem: Bezout's Identity

Suppose $a, b \in \mathbb{Z}$. Then $\exists x, y \in \mathbb{Z}$ s.t.

$$ax + by = gcd(a, b)$$

Proof: Suppose $a, b \in \mathbb{Z}$ with $g = gcd(a, b)$. Notice that by definition $g|a$ and $g|b$. Thus $a = ga_0$ and $b = gb_0$ where $a_0, b_0 \in \mathbb{Z}$. So we get the following where $x, y \in \mathbb{Z}$.

$$ax + by = ga_0a + gb_0b = g(a_0a + b_0b)$$

Therefore $g|(ax+by) \rightarrow ax+by = l \cdot gcd(a, b)$ where $l \in \mathbb{Z}$. Now consider the set $S = \{ax+by | x, y \in \mathbb{Z}, ax+by > 0\}$. Notice that $a+b \in S$, so $S \neq \varnothing$. By the Well-Ordering Principle, this set has a smallest element, d. Suppose B.W.O.C.

that $d \nmid a$. So by the division algorithm, $\exists q, r \in \mathbb{Z}$ s.t. $a = dq + r$ and $0 < r < d$. Now notice that $dq = q(ax_0 + by_0) = a - r$. But then $r = a(1 - qx_0) - b(qy_0)$. So $r \in S$ and $r < d$, and d was chosen as the minimum element in S, so this is a contradiction. WLOG we also have $d|b$.

Finally, suppose we have $c \in \mathbb{N}$ s.t. $c|a$ and $c|b$. Clearly this implies $c|(ax+by)$ for all $x, y \in \mathbb{Z}$, including $x = x_0$ and $y = y_0$. Thus $c|(ax_0 + bx_0) \rightarrow c|d$. Therefore d is the greatest common divisor of a and b.

3.1.4 Existence of Inverses Modulo n

Now we will show the result which allows us to pick d such that $e \cdot d \equiv 1 \pmod{\phi(n)}$ when $gcd(e, \phi(n)) = 1$.

Theorem: Existence of Modular Multiplicative Inverses

Let $a, n \in \mathbb{N}$. Then $gcd(a, n) = 1$ iff there exists $a^{-1} \in \mathbb{Z}$ which we call the modular multiplicative inverse of a.

$$a \cdot a^{-1} \equiv 1 \pmod{n}$$

Proof: Suppose $a, n \in \mathbb{Z}$ s.t. $gcd(a, n) = 1$. By Bezout's Lemma, we must have $x, y \in \mathbb{Z}$ with the following property.

$$ax + ny = gcd(a, n) = 1$$

Taking this whole relation modulo n, we get the following.

$$ax + ny \equiv ax \equiv 1 \pmod{n}$$

Therefore $x = a^{-1}$. Now suppose we have an inverse of a.

$$a \cdot a^{-1} \equiv 1 \pmod{n}$$

Thus $n|(a \cdot a^{-1} - 1)$. So $\exists k \in \mathbb{Z}$ s.t. $a \cdot a^{-1} - 1 = nk$. We can rearrange this into a nicer form.

$$a \cdot a^{-1} + n(-k) = 1$$

Notice that since we have a linear combination of a and n, this must be some multiple, $l \in \mathbb{Z}$, of $gcd(a, n)$ by Beozout's Lemma.

$$l \cdot gcd(a, n) = 1$$

Since we are concerned with positive integers a and n here, we see that $l = 1$, and thus $gcd(a, n) = 1$. Finally, suppose that $ay \equiv 1 \pmod{n}$ but $y \not\equiv a^{-1}$. But $ay \equiv a \cdot a^{-1} \pmod{n} \rightarrow y \equiv a^{-1} \pmod{n}$. So clearly, the multiplicative inverse must be unique. And this is an important property for a cryptographic private key.

3.1.5 Euler's Theorem

We have shown that steps (1)-(5) work out. Now it is only left to show that $(m^e)^d \equiv m^{e \cdot d} \equiv m \pmod{n}$.

Euler's Theorem

Suppose $gcd(a, n) = 1$. Then we have the following relationship.

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

Proof: Consider the set of numbers k less than n which are relatively prime to n. By definition, there will be $\phi(n)$ of them.

$$K = \{k_1, k_2, \cdots, k_{\phi(n)}\}$$

Now suppose $a \in K$. Notice that $a \cdot k_i$ will be a member of K since both a and k_i are relatively prime to n (for any i). Thus it is clear that $aK = \{ak_1, ak_2, \cdots, ak_{\phi(n)}\} \pmod{n}$ is a permutation of K. This leads to the following fact.

$$\prod_{i=1}^{\phi(n)} k_i = \prod_{i=1}^{\phi(n)} ak_i$$

Reducing modulo n and factoring out a, we get the following.

$$\prod_{i=1}^{\phi(n)} k_i \equiv a^{\phi(n)} \prod_{i=1}^{\phi(n)} k_i \pmod{n}$$

Finally, since the products will be relatively prime with n, they will have modular multiplicative inverses (from the previous result). Therefore we can essentially cancel them out, to yield the final result: $a^{\phi(n)} \equiv 1 \pmod{n}$.

3.2 Example

It is easy to create a basic example in Python. The following short script successfully generates an RSA key-pair according to the algorithm, and encrypts/decrypts a message.

```
from Crypto.Util.number import long_to_bytes, bytes_to_long
from Crypto.Util.number import getStrongPrime, inverse

# step 1: generate our primes p and q
p = getStrongPrime(512)
q = getStrongPrime(512)

# step 2: calculate n = p * q and phi(n)
phi = (p - 1) * (q - 1)
n = p * q

# step 3: choose an e less than phi and relatively prime to n
# this value is a commonly chosen as the public exponent
e = 65537

# step 4: calculate d as the inverse of e modulo phi(n)
d = inverse(e, phi)

# step 5: declare the public and private keys
print(f"[+] Public Key: ({n}, {e})\n")
print(f"[+] Private Key: ({n}, {d})\n")

# step 6: perform encryption and decryption
m = bytes_to_long(b"This is a secret message!")

# c = m^e (mod n)
c = pow(m, e, n)

print(f"[+] Encrypted Message in Hex: {hex(c)}")

# always holds due to euler's theorem :)
assert(m == pow(c, d, n))
```

Here is some sample output from the script. We can see the (rather large) keypairs and the result of the encryption.

```
[+] Public Key: (137636178308258524865973987271282171307145802466330554433
690481444283303374135318767371831258690902593227194297229104694902520100000
447028892702694497532572611339664917836593071051825760629638233598180931111
040534097561258619430809368646660624905576027412278360277658648698474531520
990174803506805705525589111541, 65537)

[+] Private Key: (137636178308258524865973987271282171307145802466330554433
690481444283303374135318767371831258690902593227194297229104694902520100000
447028892702694497532572611339664917836593071051825760629638233598180931111
040534097561258619430809368646660624905576027412278360277658648698474531529
901748035068057055255891115 41, 79865819687882622673117853455904141059850
889146468173010037312035104593497032089314499952255012532381980209224244710
963920349373377939188818620133519487635137303404347221564923910831433983182
637994658164274979113543601465959552931442199877394789015811835953808090530
473839991327519431261907574433915885826077)

[+] Encrypted Message in Hex: 0x69a78c5ad991377db7fe16c9d2d81dea4fd600ad1488
f58d14cabb3cf8448562441c74c9228ad768e9f2d5cdfadeb052ba697109a95667392eaaccd
1ae825dbce00515bd64c2e60e4186547e68887204866df523150d9d06c74609ba6093023a5
0169d4d9d081713dd62070763adc12ac11834ca86f025b68a4c4447e27e7b0e
```

For computers, storing numbers in base 10 is not always practical. Since computers use binary (base-2) at the lowest level, it becomes more convenient to use bases which are powers of 2. This is why hexadecimal and base-64 are so common in computing (plus, they also take less bytes to store larger numbers). For example, my public GPG key is below. As you can see, it is stored in base-64. This is what you could use if you wanted to send me an encrypted (and/or signed) email. Since it uses 2048 bit keys, it should be secure for the time being. *Good luck factoring this n!*

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBGOIPe4BCADP5W5CcULozcP3XRXfIfXdYd3EI3kOGtse37OhfAzoxg+uxubF
ladGltgsHMZrNFk5M7uUCw9o+BfC04aTokWxtACx1IfFRRmsDJqv/rGoAaeIylcC
QwgA+gyEN3uyZjnHsa1mRlZl1aqbBm4JwPkgXGmJi0R0vddolSmvg9Ob8jmHfH0r
gUp/iBbB3RUXV/tIMtF9Jaynvwuw6iclO0wIIltZY43oOUfoH9929mYSjLLgXd9m
0OKUs0dylxBi24kbfJTISp/gfuur7SS3i+Gh3XtMX4A/Ly7ycoLdGQhl4Fc+gdeO
8tFQbEHkRWBnoYSmHyw5eXwwNFgQOec3YaJPABEBAAG0J01pY2hhZWwgSG9lZmxl
ciA8aG9lZmxlcjIwMDJAZ21haWwuY29tPokBVAQTAQgAPhYhBHI2hn9y+XNg6Ktr
zxV6sA1uV6q8BQJjiD3uAhsDBQkA7U4ABQsJCAcCBhUKCQgLAgQWAgMBAh4BAheA
AAoJEBV6sA1uV6q8wG8IAKXGYIa1VDNlaWYDOp0udEuAEdcbRj+hpcFBpHtX2SpR
1x3pif6k1C+XvWNuiiJ48In75DuvZicnrR2UijUrlFnyZJF8ZHYS9lRx5ZWjPbL5
wBmU8c/PiIGkhW7InfElIMg4twor3JgSQsG8I0EDsMlJaSW/sjtxbCK/kj7zOEDs
4DdkCMTXuBQSmLYEhs5qocaZZIKNsj19I8zX/JExjRYwlUuQKxQ9K9U/DyCTClmp
dADwyH5lreH/flHPWn+mo8beA+WReo6mmAmHtJzVscbXbau8QBoanV+z3Ar+tCpI
/Xp3UbX8OiBnIEDvQ8Dp6KjlL2TbqM+pEM6nLK5qkuS5AQ0EY4g97gEIALPov/D+
asw4M3uzq8M/TZncX+gNr54RInPa1HMW+xm2Sj/HyXOBbb43dklMr/gt59ZgBeUh
WxoqGxVq9hOAxBrs97wiTIDxQoXue2IN9EHNQNo06ONUFNM3cHXXaowhxW/kn3UC
D86ZCi4Dsy96hVzyRJxnaJ1KOtpPyQah8Lf/n01lZaeJ2i60zvxKdAQwFHWQtAAa
1G3zNnTWsUyworHfAVyLgvbVAaRBogmbjKXS3v+86+GAZzkQidICAxEwE2INIfbc
FqC9c+QXsdrl5rBBFWraooq7sFw2ez03X80NKfsMx7HtgUu+vX1Np4xyN/r6ebgg
zQgBhHVdrarLVhEAEQEAAYkBPAQYAQgAJhYhBHI2hn9y+XNg6KtrzxV6sA1uV6q8
BQJjiD3uAhsMBQkA7U4AAAoJEBV6sA1uV6q8S+QH/j744I9DJkYqpxVhdowWem1o
1jIGv9DV8JhLT8Iml4phIAcIzUFMB+QwVTPH/MN4+6Wnpd7OgIDeGB6DwAaOhSpx
Zhvh4Z0I/SJWYit6oDxbnSfL3+GQmTnMOAxqzI1fUbmu6prWxpJZ1PSZep3T//KP
A46TqiRRBG7KTg3+S+9IzLbexck5zL36urgi3pRpDQBhpTpeQdp0qasoiZVEtq4J
AiDllYE/wMp+f7GifuxT4E1ZkODDwpC72yBC1bPsxFHFRnpu0GEg5ISBTzwSz2CF
8LaQBMaNda+94Fk0BV8lAfp1fCUbZ35eTzQ6T23wkkvWGio34BVI5JwzE/yp6CQ=
=83s3
-----END PGP PUBLIC KEY BLOCK-----
```

4 Conclusions

We have shown the correctness of the popular RSA cryptosystem and have explored some of the number theory lying behind the algorithm. Due to recent advances in Quantum computing, this method might not be around for much longer. Either way, it offers a very interesting application for number theory concepts, which will be explored for years to come.

References

[1] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.