RSA Encryption

Anna Maximova and James McNeice

2022 Mathematics Directed Reading Program. Department of Mathematics, University of California, Santa Barbara

Why Public Key Cryptographic systems?

Consider the following situation: A message needs to be sent to someone over a public channel. As the channel is not secure, anyone can look at whatever you send to the other party. The question is, how do you send a message to the other party that without compromising the information contained in the message. This is the crux of the field of cryptography. Public Key systems come into play when there is no way to transmit a key safely. The solution is creating a system where all the necessary information to encrypt a message is available publicly, but decrypting the message is very difficult without some sort of key.

An Introduction To Number Theory

Modular Arithmetic

One of the most basic ideas in number theory is modular arithmetic, which is a system of arithmetic that centers around the remainder after repeated subtraction.

Consider a 12-hour clock. Suppose the hour hand points at 12 when no time has elapsed. When 3 hours pass, the hour hand will point at the 3. When 19 hours elapse, the hand will point at the 7 because the hand will cycle through the 12 hours and then restart its cycle to reach 7. Similarly when 25 hours elapse, the hand will point at the 1. We can represent this using the symbol for congruence $\equiv$ as follows:

$3 \equiv 3 \bmod 12$ $\qquad$ $19 \equiv 7 \bmod 12$ $\qquad$ $25 \equiv 1 \bmod 12$

So $a \equiv b \bmod (n)$ if and only if there exists an integer k such that $(a-b) = nk$.[3]

Modular Exponentiation

Suppose you were asked to find the smallest x such that $x \equiv 3^{173} \bmod 11$. We could multiply 3 by itself 173 times and then subtract 11 until we were left with a remainder between 0 and 11 but that would take too long and we're feeling a little lazy today. Luckily for us there is a very simple process we could employ to help us that relies on modular arithmetic: modular exponentiation. The process is relatively simple. First we find a few other congruences.

$3^1 \equiv 3 \bmod 11$ $\qquad$ $3^{16} \equiv 5^2 \equiv 3 \bmod 11$

$3^2 \equiv 9 \equiv 9 \bmod 11$ $\qquad$ $3^{32} \equiv 3^2 \equiv 9 \bmod 11$

$3^4 \equiv 9^2 \equiv 4 \bmod 11$ $\qquad$ $3^{64} \equiv 9^2 \equiv 4 \bmod 11$

$3^8 \equiv 4^2 \equiv 5 \bmod 11$ $\qquad$ $3^{128} \equiv 4^2 \equiv 5 \bmod 11$

Now we can use smaller powers to get to larger powers based on the fact that $x^m \cdot x^n = x^{m+n}$. So, $3^{173} \equiv 3^{(1+2+2+8+32+128)} \equiv 3 \cdot 9 \cdot 9 \cdot 9 \cdot 5 \equiv 1 \bmod 11$.

Chinese Remainder Theorem

Suppose gcd(m, n) = 1. Given integers a and b, there exists exactly one solution x mod (mn) to the simultaneous congruences: $x \equiv a \bmod (m)$ and $x \equiv b \bmod (n)$. [3]

Fermat's Little Theorem

If p is a prime and p does not divide a, then $a^{p-1} \equiv 1 \bmod (p)$. [3]

Intuition for RSA

Using the classic example in cryptography, suppose Bob wants to send a secret message over an unsecure channel to Alice such that if Eve (the eavesdropper) who is listening in on the channel isn't able to understand the message. Alice would create a lock and a key that only she possesses. She would send the unlocked lock to Bob who would use it to lock his message and send it back to Alice. Finally Alice would unlock the lock with her private key and read the message. Eve would only have information about the unlocked and locked lock and therefore theoretically would not be able to read the message.

RSA

RSA works by first choosing two large prime numbers, p and q, then multiplying them to make N, that is:

$$pq = N$$

This is the value that will serve as the modulus for encryption and decryption. At this point a message can be given a numeric representation, M, such that $0 \leq M \leq N-1$. We now choose some e with the following property

$$\gcd(e, (p-1)(q-1)) = 1$$

We now choose value d such that $ed \equiv 1 \mod (p-1)(q-1)$. The setup is now complete, and (n, e) are released as the public key. A message is encrypted by taking $a \equiv M^e \mod N$, and decrypted by taking $M = a^d \mod N$. [3].

Example [2]

Alice:

1. Chooses two primes: $p = 7$ and $q = 19$.
2. Calculates the product: $N = 7 \cdot 19 = 133$.
3. Calculates the totient: $\phi(N) = (p-1)(q-1) = 6 \cdot 18 = 108$.
4. Selects a public key: $e = 29$
5. Selects a private key: $d = 41$
6. Sends the public key: $(N, e) = (133, 29)$

Bob:

1. Chooses a message: $m_o = 99$.
2. Encrypts the message: $m_e = 99^{29} \bmod 133 = 92$.
3. Sends the encrypted message.

Alice:

1. Decrypts the message: $m_o = 92^{41} \bmod 133 = 99$

Note: The efficiency of RSA lies in the fact that it is significantly faster to multiply two numbers than it is to factor a number of the same size as their product. This means that even if an eavesdropper is able to read a message in its encrypted state, they are unable to understand its content because finding the value of d is difficult. Factoring can be made arbitrarily difficult by choosing sufficiently large numbers. For simplicity's sake, we used very small numbers in our example. However to make the encryption feasible and secure, the primes used are typically 1024 to 2048 bits long, approximately 300 to 600 digits long.

Attacks On RSA

Timing Attack

It was demonstrated in 1995 that by timing the process of decrypting multiple messages a malicious party is able to determine the key. This attack is worth mentioning because it does not attack the fundamental process of encryption[3]. Its more akin to having a storefront tightly locked up, and instead of picking the locks a thief throws a rock through the front window [3].

Fermat Attack

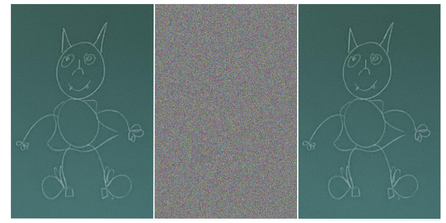
If the primes chosen for encryption are too close to each other then it has been demonstrated that an algorithm can factor N very efficiently. Using the fact that $N = a^2 - b^2 = (a-b)(a+b)$ we can tell that if we find a, b then $(a-b) = p$ and $(a+b) = q$. This is accomplished by taking $\lceil \sqrt{N} \rceil = a$, and determining if $b^2 = a^2 - N$ is an integer. If not, then increment a by one and try again until either a value of b is found, or until 100 or so values of a have been tried [1].

Shor's Algorithm

Shor's Algorithm is a quantum computing algorithm that shatters the security of RSA. It does this taking a 'bad' guess for two numbers that factor some given integer, and spitting out a 'good' guess [3].

Additional Applications of RSA

By coming up with a clever way to express some message many different forms of media can be transmitted via RSA, for instance:



Original Image | Encrypted Image | Decrypted Image

Acknowledgements

We would like to thank the organizers of the of the 2022 Directed Reading Program for the opportunity to learn about cryptography. We would also like to thank our mentor Charles Kulick for his incredible support and mentorship throughout this program.

References

[1] Hanno Böck. *Fermat Attack on RSA*. 2022. URL: https://fermatattack.secvuln.info/.

[2] Ed Harmoush. *RSA Example*. 2021. URL: https://www.practicalnetworking.net/series/cryptography/rsa-example/.

[3] W. Trappe and L.C. Washington. *Introduction to Cryptography: With Coding Theory*. Prentice Hall, 2002. ISBN: 9780130618146. URL: https://books.google.com/books?id=kVU%5C_AQAAIAAJ.