EE423 Design Project- Encrypted Messaging

Louis Dwyer

dept. of Engineering

Maynooth University

Maynooth Ireland

***Abstract*-In this document we will be discussing an encrypted messaging platform. We will be going into detail about the different types of encryption, the advantages of certain types, why encryption is so important Additionally, this paper will explore the evolving challenges facing encryption, including the implications of advancing computational capabilities and emerging cybersecurity threats that may shape the future landscape of secure communication.**

***Key Terms* — (**key words**)**

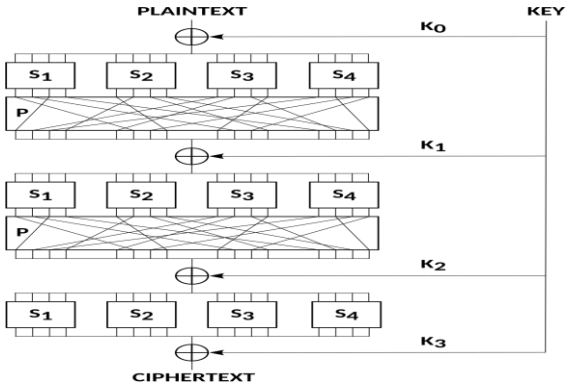
I. INTRODUCTION

The earliest written evidence of an encrypted message is actually found nearly 4000 years ago, in ancient Egypt on the tomb of Khnumhotep II, who had a script detailing his life engraved within his tomb, surprisingly some of the hieroglyphics were not the usual ones used, they were new ones made up. This was to obscure the true meaning of the text as in ancient Egypt knowledge was sacred as education was a privilege limited to the highest circles of society and was also a way to show one's skills in writing. It was also used for religious reasons, for example, to discuss taboos. Encryption then was then swiftly twisted to be used for military purposes as around 500 BC. The scytale, a device utilized in Spartan encryption, made it possible to send and receive hidden messages. A strip of parchment was wound around the device, and the text was written along the length of the device. Once the strip was wound off the cylinder, the text became unreadable unless the reader possessed an identical cylinder. This was the first time the concept of a common key, seen even today in modern cryptographic technologies, was used for both encryption and decryption. Of course encryption today is more complex than simply writing on a piece of parchment around a cylinder. The reason encryption is so important today is because it protects sensitive data from unauthorized access, ensuring privacy and security in our digital world. With huge amounts of personal, financial, and business data transferred digitally, encryption helps safeguard against cyber threats like identity theft and data breaches. By converting information into unreadable code accessible only to authorized parties, encryption helps maintain the integrity and confidentiality of data across mediums. This is critical as more and more people work remotely, use cloud services, and engage in online transactions, where the risk of data exposure is high.

II. TYPES OF ENCRYPTION ALGORITHMS

A. AES

AES, also known as the Advanced Encryption Standard is a type of encryption protocol, it was first established by the U.S National Institute of Standards and Technology in 2001. AES is included in the ISO 18033-3 standard. AES is the only publicly accessible cipher approved by the NSA. AES is based on a principle known as substitution permutation network.



Substitution permutation network

AES uses a 4x4 array of 16 bytes(b_0,b_1, b_2 ... b_{15}) arranged as follows.

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

The key size used for an AES cypher specifies the number of transformation rounds that convert the input, called the plaintext, into the final output, called the cyphertext. The number of rounds are as follows:

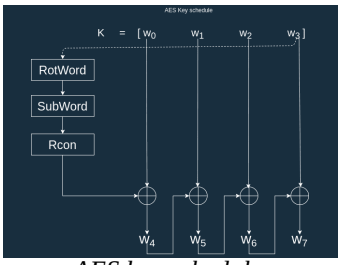
- 10 rounds for 128-bit keys.
- 12 rounds for 192-bit keys.
- 14 rounds for 256-bit keys.

A.1 How AES works

AES has a number of steps involved

1. Key Expansion – round keys are derived from the cipher key using the AES key schedule.

A.1.1 AES key schedule



AES key schedule

The round constant $rcon_i$ for round i of the key expansion is the 32-bit word $rcon_i = [rc_i \quad 00_{16} \quad 00_{16} \quad 00_{16}]$, where rc_i is defined as:

$$r_{ci} = \begin{cases} 1 & \text{if } i = 1 \\ 2 * r_{ci-1} & \text{if } i > 1 \text{ and } r_{ci-1} < 80_{16} \\ (2 * r_{ci-1}) \oplus 11B_{16} & \text{if } i \leq 1 \text{ and } r_{ci-1} < 80_{16} \end{cases}$$

Where the constants are given in Hex

- *N is the length of the key*
- *K_0, K_1, ... K_{N-1} are the 32-bit words of the original key*
- *R as the number of round keys needed*
- *W_0, W_1, ... W_{4R-1} as the 32-bit words of the expanded key*

The rot word is defines as a one bit left shift.

$$RotWord([b_0 \quad b_1 \quad b_2 \quad b_3]) = ([b_3 \quad b_2 \quad b_1 \quad b_0])$$

The Sub word is an application of the AES S-box to each of the bytes.

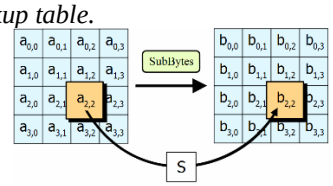
The S-box is created based on two mathematical operations over a finite field (specifically GF(2^8)):

1. Multiplicative Inversion: First, take each byte of input (in hexadecimal, 00 to FF) and find its multiplicative inverse in GF(2^8). If the byte is 0x00, it maps to itself (0x00) because it has no inverse.
2. Affine Transformation: After finding the multiplicative inverse, XOR the byte with a fixed constant and then transforming each bit according to a specific matrix.

2. For 9, 1 1or 13 rounds

2.1 Initial round key addition, *each byte of the state is combined with a byte of the round key using bitwise xor*

2.2 SubBytes step- *A step wherein each byte in the data block is replaced with a sub-byte using an 8-bit lookup table.*



2.3 ShiftRows step- a step where the last 3 rows are shifted cyclically. The first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three.

2.4 MixColumns step- a mixing step where it combines the four bytes in each column

3. Final round

3.1 SubBytes

3.2 ShiftRows

3.3 AddRoundkey

A.2 Security

AES is considered highly Secure and is commonly used by: Governments, Banks and for general data protection worldwide.
AES offers three sizes, 128,192 and 256 bits. Each level provides a higher level of security, with 256-bit AES being the most secure and resistant to brute-force attacks. Breaking AES-256 with brute force would take longer than the age of the universe, under current computing power.

We also need to keep in mind that AES is Symmetric Encryption because it uses the same key for both encryption and decryption, where the key is determined by either the user or system implementing AES encryption or it could be randomly generated, If AES is used in a scenario where the key needs to be shared between two parties (such as for communication), a secure method for key exchange must be used. This is typically done using asymmetric encryption to securely exchange the symmetric AES key.

However, in terms of quantum computing resistance AES security could be reduced effectively in half with the introduction of Grovers algorithm

A.2.1 Grovers Algorithm

Grovers algorithm is a quantum algorithm which could be used to sort through an unsorted database a lot more efficiently than “traditional” sorting algorithms, in “typical” algorithms, if you had a large list of N terms and you wish to find a specific term, using “traditional” algorithms you would, on average need to search through $\frac{N}{2}$ items to find the item you are looking for. With Grovers Algorithm that can be reduced to $\sqrt{N}$ steps.
Grovers algorithm works by: utilizing superposition, a state where a quantum system can exist in multiple states at once. In Grover's algorithm, the quantum system is initialized to be in a superposition of all possible items in the database. This means the system "represents" all possible database entries simultaneously.
Grover’s algorithm revolves around an iterative procedure using two main operations:

- **Oracle**: This is a quantum operation that "marks" the solution. It flips the phase (sign) of the state corresponding to the correct item, giving it a unique identity in the quantum state. This is the quantum equivalent of checking if F(x)=1
- **Diffusion**: The idea behind this step is to increase the Amplitude of the correct answer (P{right})

while reducing the amplitude of incorrect answers. The diffusion operator acts like a reflection, amplifying the probability amplitude of the correct state in each iteration.

Grovers algorithm repeats the Oracle and Diffusion steps approximately $\sqrt{N}$ times, with each iteration increasing (P{right}) until the probability of finding the target is maximised, where on average the number of iterations is: $\frac{4}{\pi}\sqrt{N}$. After performing the Grover iterations, the quantum state has a high probability of being in the state that represents the correct item. When you measure the system, the superposition collapses into the correct state, hopefully giving you the correct answer.

A.3 Advantages and disadvantages of AES encryption

Advantages:

Efficiency AES is highly efficient, especially when implemented in hardware. It is capable of encrypting large volumes of data quickly, making it ideal for applications requiring fast encryption, such as encrypting entire disk drives.

Non Resource Heavy: AES operates well on both small devices and high-performance servers due to its low computational requirements, especially compared to other encryption algorithms for bulk encryption.

Multiple Key Sizes: AES supports 128, 192, and 256-bit key sizes, allowing users to choose the level of security they need based on the sensitivity of their data. AES-128 is sufficient for most purposes, while AES-256 provides an extra layer of security for highly sensitive information.

Widespread Adoption: AES is used globally. Its universal adoption means that there is a high level of confidence in its security and reliability.

Disadvantages:

Key Distribution: Since AES is a symmetric encryption algorithm, both the sender and receiver need to securely share and store the same secret key. Ensuring secure key exchange and storage can be challenging, especially in large-scale or distributed systems.

No Integrity or Authentication: AES by itself only provides confidentiality and does not guarantee data integrity or authenticity. Without additional mechanisms, such as message authentication codes (MACs) there is no way to ensure that the encrypted data has not been altered or tampered with during transmission.

Key Storage: Protecting AES keys is crucial to the security of the system. If the key is compromised, all data encrypted with it can be decrypted. Managing keys securely across different devices and platforms is an issue to be considered when implementing AES encryption.

B. RSA

RSA, is a type of encryption protocol used for secure data transmission. It was named off of the initials of its founders, Rivest–Shamir–Adleman algorithm, It was developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman and remains widely used for secure communications, particularly in public-key cryptography. Unlike AES, which is a symmetric encryption standard, RSA is an asymmetric encryption algorithm which means that it uses a pair of keys—a public key for encryption and a private key for decryption. RSA's security relies on the mathematical difficulty of factoring large prime numbers, making it ideal for protecting data during transmission over the internet. The RSA algorithm has four main steps: key generation, key distribution, encryption, and decryption. The key idea behind RSA is to find three large numbers: *e*, *d*, and *n*, where *n* and *e* comprise the public key, *d* represents the private key, and *m* represents the message. So that for any number *m*, (where $0 \leq m < n$), encrypting and then decrypting *m* gives back the same value. This means:

1. When m is raised to the power of e (encryption) and then to the power of d (decryption), the result is still m.
2. This works because the calculations are done using a property of modular arithmetic, where numbers "wrap around" after dividing by n.

$$(m^e)^d = m \quad (modulus\ n)$$

This means that when given only *e* and *n*, it is extremely difficult to find *d*.

Another neat feature of RSA is that the public and private key can be swapped if we swap the two exponents.

B.1 How RSA works

RSA generates keys by:

1. Choosing *p* and *q*, two large prime numbers. (if we want to make RSA harder to break ideally *p* and *q* should be chosen at random, be big numbers and have a huge difference), typically these prime numbers are chosen by picking a random number an then doing a trial and error script to choose a prime number close to the one you chose.
2. Calculate *n=pq*, where *n* is the modulus of both the public and private keys. Its length is known as the key length(number of bits in a key)
3. Release *n* and keep *p* and *q* hidden
4. Calculate the Carmichael function of *n*,
   Since $n = pq$, $\lambda(n) = lcm(\lambda(p), \lambda(q))$, and since *p* and *q* are prime, $\lambda(p) = \varphi(p) = p - 1$, and $\lambda(q) = q - 1$. Therefore $\lambda(n) = lcm(p - 1, q - 1)$.

   4.1 The Carmichael function $\lambda(n)$ of a positive integer *n* is the smallest positive integer *m* such that
   $$a^m = 1 \quad (mod\ n)$$
   holds for every integer *a* is mutually prime to *n*.(the only positive integer that is a divisor of both of them is 1.),
   For example, if we take *n=8*, the set of numbers mutually prime to 8 is *{1,3,5,7}* and *Euler's totient (φ)* has a value of 4, this means λ has to be a divisor of 4 $\lambda(8) = 2$ because when we square each coprime number, the results all "loop back" to 1 modulo 8:

$1^2 \equiv 1 \bmod 8$
$3^2 \equiv 1 \bmod 8$
$5^2 \equiv 1 \bmod 8$
$7^2 \equiv 1 \bmod 8$
$\therefore \lambda(8) = 2$, because squaring any coprime number gives 1 modulo 8.

4.2 Euler's Totient Function ($\phi(n)$) is a function that tells us how many numbers are coprime to *n* (they don't share any common factors with *n* except 1), from 1 up to $(n - 1)$. For example $\phi(8)$, would be 4, because there is 4 numbers coprime to 8 *{1,3,5,7}*

5. Calculate $gcd(e,\lambda(n))=1$, meaning *e* and λ(n) are coprime. To make encryption more efficient, *e* is often selected to have a short bit-length and a small Hamming weight (fewer 1s in its binary representation). The most commonly used value for *e* is $2^{16}+1=65537$. While the smallest possible value for *e* is 3, it is generally considered less secure, however a lot faster. The value of *e* is made publicly available as part of the public key.
6. To determine *d*, we calculate it as the modular inverse of *e* modulo λ(n), which means *d* satisfies the equation $d * e \equiv 1(mod\lambda(n))$. This means that when *d* is multiplied by *e*, the result leaves a remainder of 1 when divided by $\lambda(n)$. The value of *d* can be found efficiently using the extended Euclidean algorithm, as *e* and $\lambda(n)$ is coprime. This equation is a specific case of Bézout's identity, where *d* is one of the coefficients. Once computed, *d* is kept secret and serves as the private key exponent in the RSA encryption scheme.

   6.1 Bézout's identity states that if two numbers *a* and *b* are coprime, there are numbers that exist *x* and *y* such that:
   $a * x + b * y = 1$, in our case *x* is *d*, and the equation is rewritten in terms of modulo:
   $d * e \equiv 1 \pmod{\lambda(n)}$

The public key includes the modulus *n* and the public (encryption) exponent *e*. The private key consists of the private (decryption) exponent *d*, which is kept secret. Additionally, *p*, *q* and $\lambda(n)$ must also be kept secret, as they can be used to derive *d*. Once *d* is calculated, *p*, *q*, and $\lambda(n)$ are no longer needed.

B.2 RSA key Distribution

If Bob wants to send a message (*M*)to Alce using RSA encryption he can do so by first turning the message not an Integer *m*, using a reversible padding scheme that both he and Alice agree on. This should satisfy the following:
$0 \leq m < n$
Bob then computes the ciphertext *c* using Alice's public key *e* with the formula:

$$c \equiv m^e \quad (mod\ n)$$

This calculation is efficient even for very large numbers. Once *c* is calculated, Bob sends it to Alice.

Alice can recover *m* from *c* by using her private key exponent *d* by computing

$$c^d \equiv (m^e)^d \equiv m \quad (mod\ n)$$

Given *m*, she can recover the original message *M* by reversing the padding scheme.

B.3 RSA Example

Step 1:
Choose 2 prime numbers, given by a random generate prime numbers online website .

67 and *89*

Step 2:
Calculate *n*:

$$n = p * q = 67 * 89 = 5963$$

Step 3:
Compute Carmichaels Function $\lambda(n)$

$$\lambda(n) = lcm(p - 1, q - 1) = lcm(66,88)$$

We can calculate lcm using the Euclidean algorithm
The gcd of (*66,88*) is *22*

$$lcm = \frac{66 * 88}{22} = 264$$

$\therefore \lambda(n)=264$

Step 4:
Choose *e*, for this example we will use $e = 65537$ as it is the most common value of *e* used.

Step 5:
Calculate *d*
The private key exponent *d* is the inverse modular inverse

$$d * 65537 \equiv 1(mod 264)$$

To solve this we must use Bézout's identity.

$$65537 \div 263 \approx 248$$
$$65537 - 264 * 248 = 65537 - 65472 = 65$$

This means we have:

$$65537 \equiv 65(mod 264)$$

$$65 * d \equiv 1(mod\ 264)$$

We must find *d* so that the above is true, we can do this using the Extended Euclidean Algorithm.
The Extended Euclidean Algorithm allows us to find numbers such that:

$$65 * d + 264 * a = \gcd(65,264)$$

gcd of (65,264) is 1, as they are coprime, this means we can shorten the equation to:

$$65 * d + 264 * a = 1$$

We now apply the Extended Euclidian Algorithm:

1. 264=4×65+4 (Divide 264 by 65, remainder is 4)
2. 65=16×4+1 (Divide 65 by 4, remainder is 1)
3. 4=4×1+0 (Divide 4 by 1, remainder is 0)

From Step 2: $1 = 65 - 16 * 4$
From Step 1: $4 = 264 - 4 * 65$

$$1 = 65 - 16(264 - 4 * 65)$$
$$1 = 65 - 16 * 264 + 64 * 65$$
$$1 = 65(1 + 64) - 16 * 264$$
$$1 = 65 * 65 - 16 * 264$$

We can gather from above:

$$65 * 65 \equiv 1 \ (mod\ 264)$$

This proves that $d = 65$, the multiplicative mod inverse

The public key is ($n = 5963,\ e = 65537$). For a padded message *m*, the encryption function is:

$$c(m) = m^e \ \ (mod\ n)$$
$$= m^{65537} (mod\ 5963)$$

The private key is (n = 5963, d = 65). For an encrypted text *c*, the decryption function is:

$$m(c) = c^d \ \ (mod\ n)$$
$$= c^{65} \ \ (mod\ n)$$

To encrypt $m = 3$ the correct calculation is:

$$c = 3^{65537}\ (mod\ 5963) \approx 849$$

To decrypt $m = 849$ the correct calculation is:

$$m = 3^{65} \ \ (mod\ 5963) \approx 3$$

B.4 RSA Secutity

RSA is a highly secure encryption algorithm widely used for secure communication, digital signatures, and key exchange. Its security is based on the computational difficulty of factoring large prime numbers, a problem that becomes exponentially harder as key sizes increase. Modern RSA implementations typically use key sizes of 2048 or 4096 bits to ensure the best security against brute-force attacks. Breaking RSA encryption by factoring the large modulus with current technology would require an obscene amount of time, even for the most powerful supercomputers. For example, RSA 4096-bit encryption relies on the modulus *n*, which is the product of two large prime numbers, each approximately 2048 bits in size. Factoring *n* requires testing numbers up to 2^{2048}, which is a number so astronomically large that its bigger than the total number of atoms in the observable universe ($about\ 10^{80}$).However, RSA's security could be threatened by quantum computers in the future, as they are theoretically capable of efficiently solving the factoring problem, specifically Shor's Algorithm

B.4 Shor's Algorithm

Shor's algorithm is a quantum algorithm that can efficiently factor large numbers, a task that is extremely difficult for traditional computers to perform for very large inputs. Factoring is at the core of RSA, which relies heavily on the difficulty of this problem for its security, as discussed above. In classic algorithms, finding the prime factors of a number *N* requires time that grows exponentially depending on the size of *N*, the larger *N* is the longer it will take. Shor's algorithm reduces this time significantly.
Shor's algorithm works by utilizing quantum properties like *superposition*, which we discussed in Grovers algorithm above, and *entanglement*, which is a unique property of quantum mechanics where two or more particles become connected, so the state of one particle is directly related to the state of the other, no matter how far apart they are. For example, if two particles are entangled, and their spins are linked. If one particle is measured and found to have a spin "up," the other particle will *instantly* have a spin "down," even if they are separated by a big distance. This happens because their states are not independent, they exist as a single combined system until measured. Shor's algorithm reduces the factoring problem into a simpler "period finding" problem. It does this by:

1. **Choosing a Random Base**: Select a random number *a* smaller than *N* and check if it already shares a common factor with *N*
2. **Quantum Period Finding**: If *a* and *N* are co-prime, the problem of factoring *N* is translated into finding the period *r* of the function $f(x) = a^x$. Shor's algorithm uses quantum mechanics to find this period extremely efficiently. The quantum system is initialized in a superposition of all possible values of *x*, and through the application of a quantum Fourier Transform, the period *r* is obtained.
3. **Classical Post-Processing**: Once *r* is found, it can be used to compute the factors of *N*. If *r* is even, the factors can be calculated using

$$x = a^{\frac{r}{2}} \ \ (mod\ n)$$

taking the greatest common divisor of $x - 1$ or $x + 1$.

Shor's algorithm is important because it shows how much quantum computers outperform "typical" ones for specific tasks, particularly those involving large-scale computations like breaking RSA encryption. Shor's algorithm can break RSA in $O((logN)^3)$ time as opposed to the typical algorithms, such as the General Number Field Sieve algorithm, which would typically be used in cases like this, which could break RSA in $O\ (e^{c(logN)^{\frac{1}{3}}\ (\log(\log\ (N)))\text{^}2/3}\ )$

**Note, this is done in polynomial time due to the fact that the larger the size of the input the amount of time grows exponentially.*

B.5 Digital Signatures using RSA

A digital signature using the RSA algorithm is a cryptographic technique that ensures the authenticity and trustworthiness of digital data. When we use this process the signer generates a pair of keys a private and a public key, the private key is kept hidden and the public key can be shared. To create a digital signature, the data is first hashed using a hash function to generate a summary of a specific size. The signer then encrypts this hash with their private key, creating the signature. To verify the signature, the recipient uses the signer's public key to decrypt it and

compares the result with a hash of the original data. If the two match, it confirms that the data has not been altered and that the signature was created by the corresponding private key. RSA-based digital signatures are widely used in secure communication protocols like TLS and email encryption standards like PGP.

*A.*3 *Advantages and disadvantages of RSA encryption*

Advantages:

Public-Key Infrastructure (PKI)*:* RSA uses a public and private key pair, allowing communication without the need to share secret keys in advance. This is particularly useful for secure data exchanges over a network.

Versatility*:* RSA can be used for both encrypting data and authenticating the identity of sender/receiver through digital signatures, making it useful for secure communications.

Widely Adopted

RSA is one of the most widely used encryption methods in the world. Its history, standardization, and support in numerous cryptographic libraries and protocols (e.g., SSL/TLS for internet communication), this broad adoption ensures compatibility across different systems and platforms.

Disadvantages:

Computationally expensive: RSA is more computationally intensive compared to, for example a symmetric encryption method (AES). The encryption and decryption processes require significant processing power, especially with larger key sizes, which in turn makes it less efficient for large-scale data encryption.

Key Length Requirement: To ensure security against future threats, i.e. advancements in computational power, RSA requires large key sizes which can slow down performance.

Not Suitable for Large Data Encryption: RSA is inefficient for encrypting large volumes of data directly. It is typically used to encrypt small data or keys (e.g., in hybrid encryption schemes where RSA encrypts a symmetric key, which then encrypts the actual data).

III. Choosing the correct encryption Algorithm for my Project:

I chose RSA encryption over AES for several reasons for my specific use-case, the main reason being the availability of the RSA library through a pip installation:

```
pip3 install RSA
```

which made it a lot easier to integrate with Python. Unlike AES, a symmetric encryption algorithm requiring both parties to securely exchange and manage a shared key beforehand, RSA uses a simpler method by using a public-private key pair. In a chat application, where secure communication between two parties is, obviously the most important thing, RSA's method of sharing a public key for encryption means that only the intended recipient, (who has the private key) can decrypt the message. This advantage not only makes the key exchange process a lot more simple but also increases security especially in situations where sharing a symmetric key beforehand might expose vulnerabilities. RSA is particularly suitable for secure communications at the application layer, where message confidentiality and the ease of implementation are important.

Another thing that swayed my decision to go with RSA encryption was the ability to implement the chat app using Python's socket library, makes it easy to establish a TCP connection across networks. By including RSA in this TCP link, I was able to implement a secure "handshake process", which basically means rules are established for communication when a computer attempts to communicate with another device. In this "handshake process" public keys are exchanged at the start of the connection using my *"exchange_keys"* function. This means that all following communication is encrypted *e2e*. This approach removes the need for a separate key distribution, which would have been required if AES had been used. Additionally RSA's asymmetric encryption provides more flexibility for one-to-many scenarios, which could be a future expansion for my chat app, where a server could broadcast encrypted messages to multiple clients without sharing its private key.

From an implementation standpoint, using RSA encryption aligns with the requirements for simplicity and security in small message exchanges. RSA falls short, and AES would be a lot better for large scale data transmission over a network, however for my needs RSA more than suffices. Messages are encrypted using the receivers public key and decrypted by their private key, meaning that even if the information was intercepted during transmission, without the private key there is not much that could be done with the information. For larger data transfers I had to write two functions, *send_with_length* and *recv_with_length* to interpret message length, means solid data transfer over sockets, even when handling larger encrypted information. I also used threading to manage simultaneous sending and receiving of messages, using Python's threading module to ensure a real-time user experience. This design prevents the application from stopping a message being sent while waiting for input or network responses or vice versa, making the chat app both efficient and user-friendly.

From a theoretical perspective, RSA offers distinct advantages over AES in specific use cases. For example RSA supports digital signatures, which could allow my chat app to verify the authenticity of messages, which could be more future work I could investigate into, which would be much harder to implement if I chose AES. What's more, because RSA is asymmetric, it avoids the downfalls that could happen if a shared AES key is intercepted, all encrypted communications could be decrypted, whereas RSA's separation of public and private keys mitigates this risk. While RSA is computationally more intensive than AES, it is more suited for encrypting small packages, such as short text messages, making it a practical choice for this project.