Abstract

This report discusses the implementation of RSA-based encryption and decryption keys using a range of data platforms, including numbers, text, images, voice, and video. Finally, we investigated the influence of noise in cipher text and possible solutions. Objectively, we intend to experiment with numerous feasible ways to implement encryption and decryption techniques depending on the original message type, such as number, text, image, voice, and video.

Introduction

RSA (Rivest-Shamir-Adleman) is the most popular public-key cryptography in data communication systems [1]. It's an asymmetric algorithm, and the computational difficulty relies on their chosen prime numbers. In RSA, commonly known as a dual-key system such as public and private keys, the public key is used for encrypting the original message and the private key is used for decrypting (Figure 1). There is no need to share a secret key as required in secret key (symmetric) cryptography [2, 3]. This RSA cryptography mechanism protects sensitive data during data storage and transmission from unauthorised access [4].

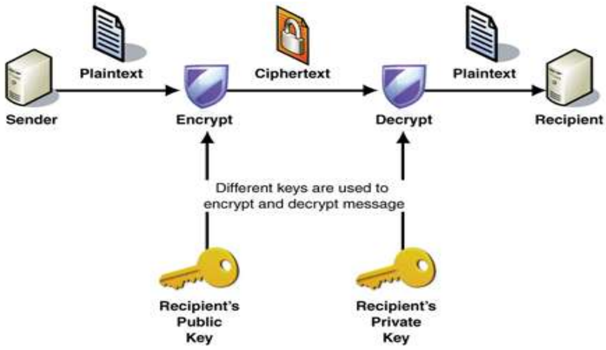
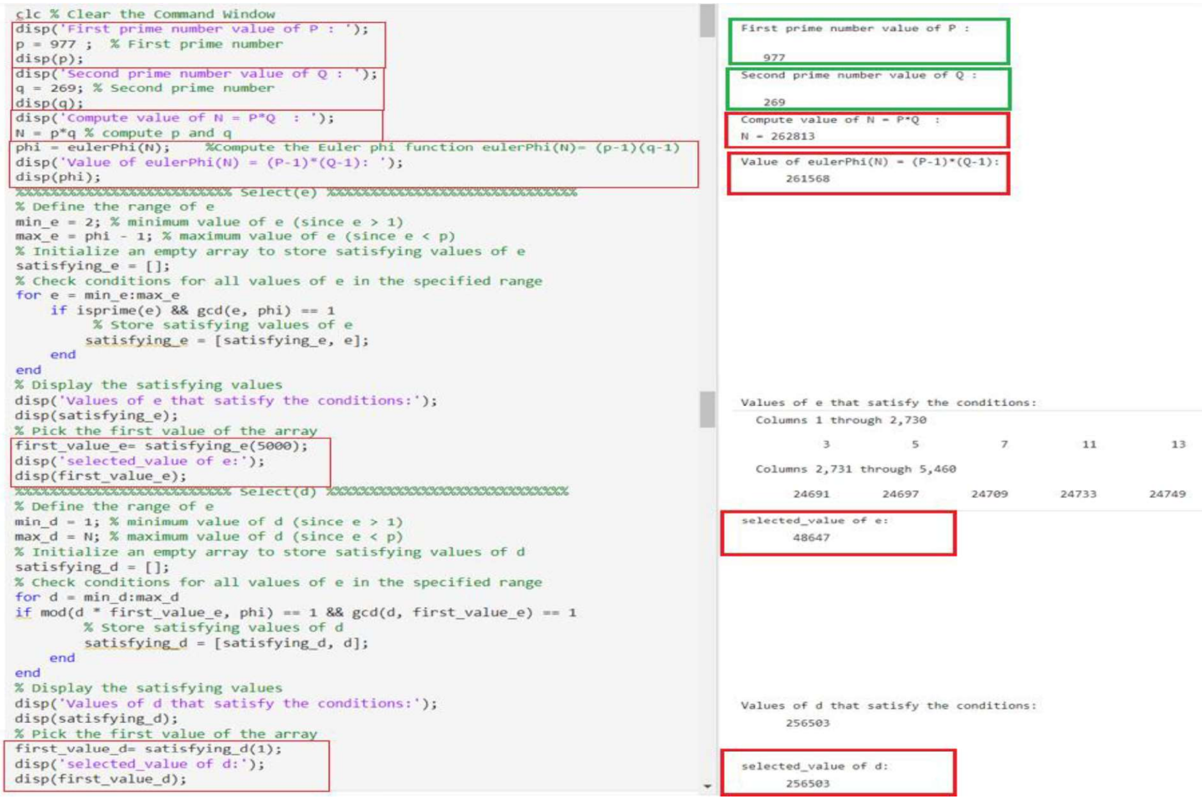


Figure 1: Encryption and Decryption [4]

Results and MATLAB

1. Creating encryption and decryption keys

Public and private keys are essential for RSA encryption and decryption key generation. Initially, we chose the two large prime numbers, p and q, for our programming operation. After computing the values of p and q, store the results in "N" and use "Euler's totient function" to obtain the value of phi(N) (phi(N) = (p-1) * (q-1)). The next step is determining the value of the private key (e) using the condition 1< e < phi(N), where e and phi(N) must be coprime, and the greatest common divisor of e and phi(N) is equal to 1. Similarly, using the equation d.e. mod phi(N) = 1, find the value of e as well (figure 2).



```
clc % Clear the Command Window
disp('First prime number value of P : ');
p = 977 ;  % First prime number
disp(p);
disp('Second prime number value of Q : ');
q = 269; % Second prime number
disp(q);
disp('Compute value of N = P*Q  : ');
N = p*q % compute p and q
phi = eulerPhi(N);    %Compute the Euler phi function eulerPhi(N)= (p-1)(q-1)
disp('Value of eulerPhi(N) = (P-1)*(Q-1): ');
disp(phi);
%%%%%%%%%%%%%%%%%%%%%%%%%%% Select(e) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define the range of e
min_e = 2; % minimum value of e (since e > 1)
max_e = phi - 1; % maximum value of e (since e < p)
% Initialize an empty array to store satisfying values of e
satisfying_e = [];
% Check conditions for all values of e in the specified range
for e = min_e:max_e
    if isprime(e) && gcd(e, phi) == 1
         % Store satisfying values of e
        satisfying_e = [satisfying_e, e];
    end
end
% Display the satisfying values
disp('Values of e that satisfy the conditions:');
disp(satisfying_e);
% Pick the first value of the array
first_value_e= satisfying_e(5000);
disp('selected_value of e:');
disp(first_value_e);
%%%%%%%%%%%%%%%%%%%%%%%%%%% Select(d) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define the range of e
min_d = 1; % minimum value of d (since e > 1)
max_d = N; % maximum value of d (since e < p)
% Initialize an empty array to store satisfying values of d
satisfying_d = [];
% Check conditions for all values of e in the specified range
for d = min_d:max_d
if mod(d * first_value_e, phi) == 1 && gcd(d, first_value_e) == 1
        % Store satisfying values of d
        satisfying_d = [satisfying_d, d];
    end
end
% Display the satisfying values
disp('Values of d that satisfy the conditions:');
disp(satisfying_d);
% Pick the first value of the array
first_value_d= satisfying_d(1);
disp('selected_value of d:');
disp(first_value_d);
```

```
First prime number value of P :

   977
Second prime number value of Q :

   269
Compute value of N = P*Q  :
N = 262813
Value of eulerPhi(N) = (P-1)*(Q-1):
      261568

Values of e that satisfy the conditions:
  Columns 1 through 2,730

           3           5           7          11          13

  Columns 2,731 through 5,460

       24691       24697       24709       24733       24749

selected_value of e:
       48647

Values of d that satisfy the conditions:
      256503

selected_value of d:
      256503
```

Figure 2. Encryption and Decryption key generation

2. Encrypting and decrypting messages and images

2.1 Number

In this number encryption and decryption operation, we are using the equations of $C = M^{e} \bmod N$ (encryption) and $M = C^{d} mod\ N$ (decryption) where M is the original message (number) or plain text, C is the ciphertext, 'e' and 'd' are public and private keys, and N is the modulus. The input number (1st red box), ciphertext (2nd green box), and decrypted number (3rd red box) in Figure 3.

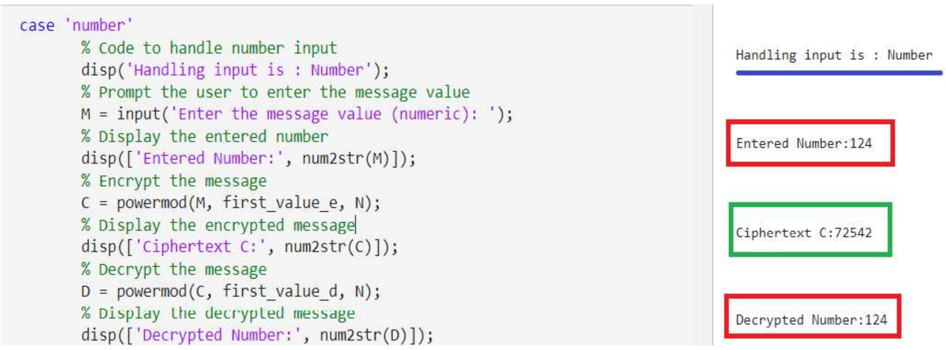


Figure 3. Number-Based Encryption and Decryption

2.2 Text

Likewise, the input is in text format; in this case, each character is translated to its corresponding ASCII code (green underline code (figure.6) for computation purposes. Finally, using the same encryption and decryption formula to get the original text (3rd red box) in figure 4.

Figure 4. Text-Based Encryption and Decryption

2.3 Images

We can't execute image encryption and decryption directly because of the two dimensions, so we convert to a one-dimensional array to make encryption easier. The one-dimensional picture array is encrypted and decrypted with its respective equations. The subplot clearly shows the cryptographic approaches to image processing operations to improve data security and privacy (Figure 5).

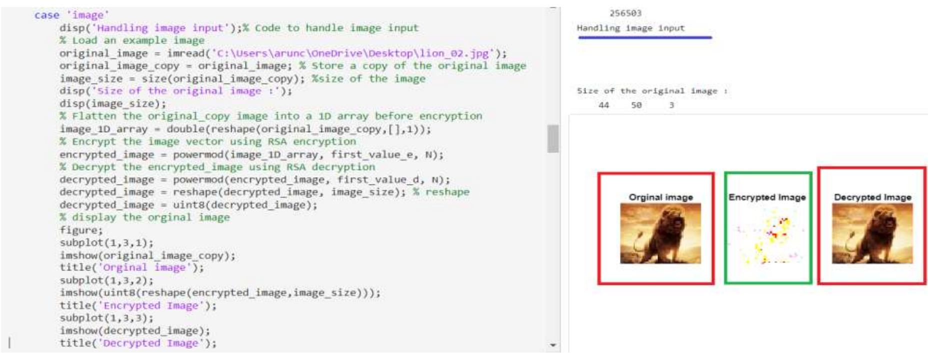


Figure 5. Image-Based Encryption and Decryption

3. Encrypting and decrypting multimedia (voice and videos)

3.1 Voice

First, we create a short audio file in waveform audio format (.WAV) and store it in a certain location. The audio file is then loaded using MATLAB's 'audioread' function, which provides both the audio data and the sampling frequency. The audio data is then analysed, converted to integers ranging from 0 to 255, and translated into a 1D array with double precision. After decryption, audio is automatically saved in a predetermined path folder. See figure 6.

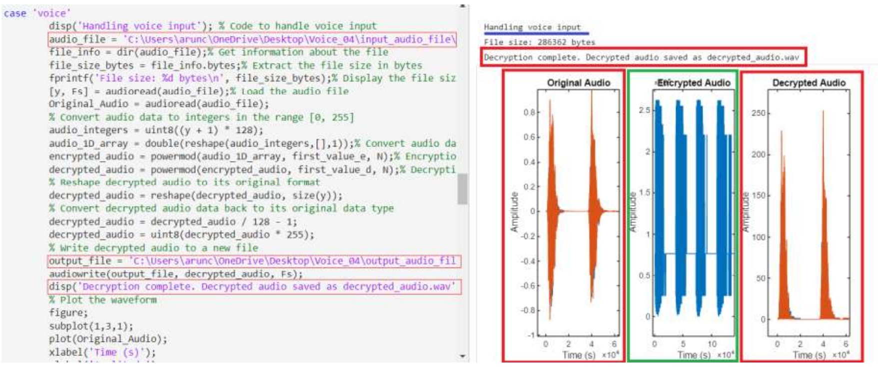


Figure 6. Voice-Based Encryption and Decryption

3.2 Video

In the case of video, the file is read as binary data, encrypted with RSA using a public key, and then decrypted with the associated private key. The 'fread' function in Matlab is used to read data in binary format, which is more efficient than frame-by-frame reading. The decrypted video data is subsequently written to a new file with a predetermined path. See Figure 7.

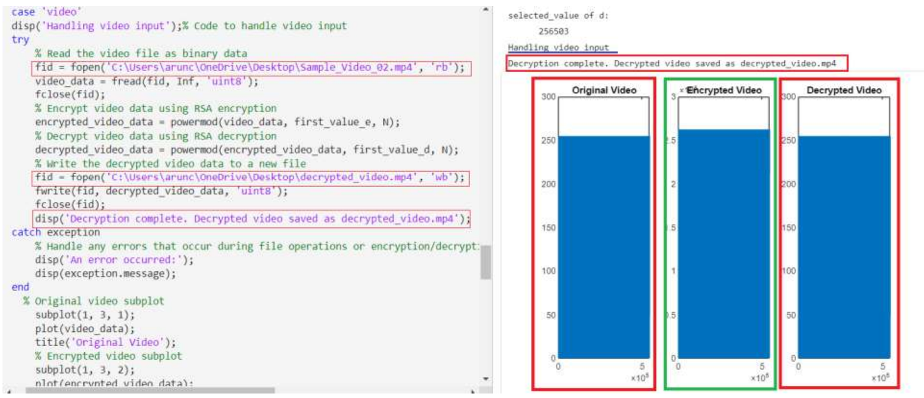


Figure 7. Video-Based Encryption and Decryption

4. Investigation of the effect of noise on encrypted data

4.1 Noise

In this scenario, we simply apply a random noise signal to the cipher text and attempt to decrypt the original message. The noisy ciphertext is presented, followed by decryption with the associated private key. The extracted number is displayed, and the noise is subtracted. We tried to reveal the original message value, but an error is still presented. See Figure 8.

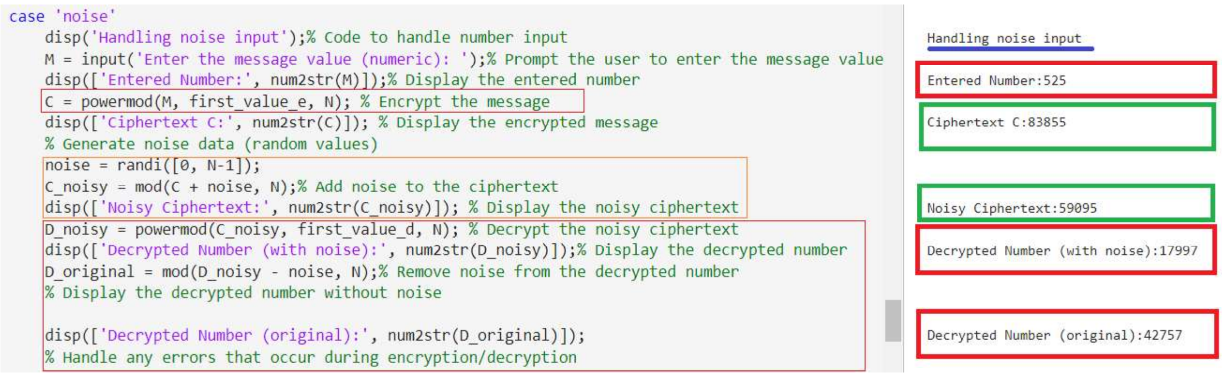


Figure 8. Noise-Based Encryption and Decryption

4.2 Noise with CRC (Error detection)

The program uses Cyclic Redundancy Check (CRC) to detect errors in encrypted image data, comparing original and noisy decrypted images. If no errors are detected, the data's authenticity is confirmed, otherwise, it's corrupted (Figure 9).

Figure 9. Noise with CRC (Error detection)

4.3 Hamming code generation (Error detection and correction)

This part handles error detection and correction using the Hamming code formula ($2^n \geq m + n + 1$). Finally, the transmitted bit pattern with updated Hamming code bits is displayed, along with a remark indicating the ASCII values (48) for '0' and (49) for '1'. See Figure 10.

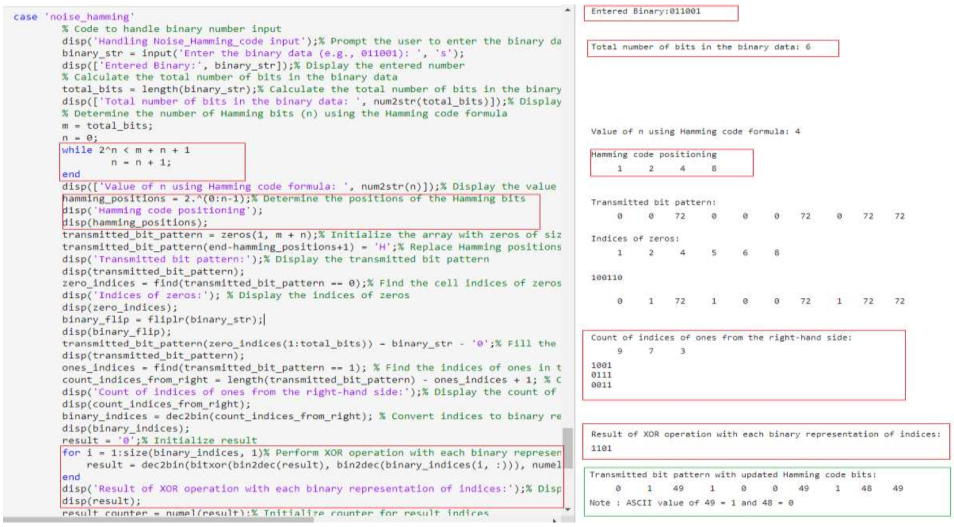


Figure 10. Noise with Hamming code generation (Error correction)

Discussion

During the RSA encryption and decryption process, we understood that the weakness of the RSA algorithms is directly dependent on some factors, such as the chosen values of p and q and their computational result N and capability of noise separation. For this limitation, we would like to explain it with two case studies. Firstly, the relation between the values of p, q, and N and message size. Secondly, the value of p, q, and N with message security.

Firstly, we noticed some issues during the execution of the RSA program: if the size of the message is greater than the value of N, the result will be incorrect. To overcome this issue, the value or size of the message (number, text, voice, image, and video) should be less than N. Secondly, the smaller values of p and q in the RSA algorithm make it vulnerable to attacks like brute force attacks, and moreover, the generation of key combinations is limited.

To overcome this issue, there are possible solutions, such as using high values of prime numbers p and q, programmatically splitting the data into smaller blocks, parallel RSA algorithm [5], and encrypting them separately and using better encryption methods like symmetric encryption algorithms like AES [6,7]. In the case of data noise suppression, we recommend using more efficient algorithm like RSA-VMD-DNN [8].

Conclusion

RSA encryption and decryption keys were successfully developed, and the security of various data streams, including numbers, text, images, voice, and video, was verified. Using RSA key generation methods, we examined some parameters that have a direct impact on data security, such as the initial prime number values of p and q. Our investigation into noise effects on cypher data yielded error detection using CRC and error correction via hamming code generation.

We observed limitations in our RSA methods, including processing speed and computational efficiency, particularly when dealing with multimedia data like video, as well as the noise impact of cypher text without using the CRC, which corrupted the original data. These issues indicate the need for efficient RSA optimisation and testing on many platforms.

References

[1] Wikipedia. *RSA (cryptosystem)*. [Online]. Available online at: Wikipedia (Accessed May 12, 2024).

[2] G. C. Kessler, An Overview of Cryptography, Boca Raton: Auerbach Publications, 2017.

[3] Yogeshwaran, M., et al. *RSA Public Key Cryptography Algorithm – A Review*. DOI: 10.13140/RG.2.2.18221.44004 [Accessed May 12, 2024].

[4] https://www.ssl2buy.com/wiki/author/wikis2buserjason, "What is Encryption and Decryption in SSL?," *SSL2BUY Wiki - Get Solution for SSL Certificate Queries*, Jan. 12, 2017. https://www.ssl2buy.com/wiki/what-is-encryption-and-decryption.

[5]. S. Saxena and B. Kapoor, "An efficient parallel algorithm for secured data communications using RSA public key cryptography method," *2014 IEEE International Advance Computing Conference (IACC)*, Gurgaon, India, 2014, pp. 850-854, doi: 10.1109/IAdCC.2014.6779433.

[6]. S. Liu, Y. Li and Z. Jin, "Research on Enhanced AES Algorithm Based on Key Operations," *2023 IEEE 5th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, Dali, China, 2023, pp. 318-322, doi: 10.1109/ICCASIT58768.2023.10351719.

[7] Wang Y. AES Encryption and Decryption Algorithm and Its Security Analysis. Network Security Technology and Applications, 2022(09).

[8]. F. Yan, S. Pi, Q. Yu and J. Lin, "Transient Electromagnetic Data Noise Suppression Method Based on RSA-VMD-DNN," in *IEEE Geoscience and Remote Sensing Letters*, vol. 21, pp. 1-5, 2024, Art no. 7500105, doi: 10.1109/LGRS.2023.3334293.

Recommend books for RSA cryptography.

1. Ferguson, N., Schneier, B., & Kohno, T. (Cryptography Engineering: Design Principles and Practical Applications). Wiley Publishing, Inc. (2010).

2. Katz, J., & Lindell, Y. (2014). *Introduction to Modern Cryptography* (2nd ed.). Chapman and Hall/CRC.

3. Handbook of Applied Cryptography by Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone

Appendix

MATLAB Program Execution methodology

For ease of processing, the complete program is being divided into two sections. The primary part is RSA key generation for encryption and decryption keys. The second part is the switch statement, which controls the activities of the subsequent part, including number, text, voice, image, video, noise, noise_crc, and noise_hamming. The reason behind this separation is that each subpart is efficiently controlled by a switch statement condition that is managed through the command window and keeps the same keys for all subsequent parts. (See figure 11 for operation selection using the command window.)

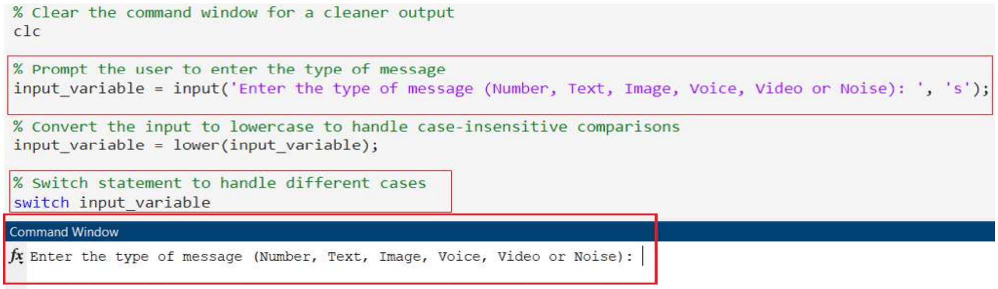


Figure 11. Operation selection using command window.

Note: Before entering the input message (e.g., a number, text, image, voice, video, noise, noise_crc, or noise_hamming), press the two-times run button to clear both windows.