Image Encryption Using RK-RSA Algorithm in Aadhaar Card

R. Felista Sugirtha Lizy\textsuperscript{a}, V. Joseph Raj\textsuperscript{b}

\textsuperscript{a}Research Scholar of Computer Science, Kamaraj College, Thoothukudi – 628 003, TamilNadu, India, Affiliated to Manonmaniam Sundaranar University
\textsuperscript{b}Associate Professor and Head, Department of Computer Science, Kamaraj College, Thoothukudi – 628 003, TamilNadu, India, Affiliated to Manonmaniam Sundaranar University

\textsuperscript{a}21felistaa@gmail.com \textsuperscript{b}v.jose08@gmail.com

**Article History:** Received: 10 November 2020; Revised 12 January 2021 Accepted: 27 January 2021; Published online: 5 April 2021

**Abstract:** Cryptography is used for secretly sending information. The information or given data is protected by cryptographic technique. The technique is used in Text and images. The technique is supported by a lot of algorithms. RSA is a better encryption technique for smart cards. In this paper, an image in the Aadhaar card is encrypted using the RK-RSA algorithm for better protection and confidentiality. The proposed RK-RSA algorithm is very secure for smart cards and Aadhaar cards. The better performance of the RK-RSA is evaluated based on the Avalanche Effect, Speed, Throughput, and Power Consumption. The improved performance of the RK-RSA algorithm’s experimental results is reported. The mathematical justification supporting the RK-RSA algorithm is also detailed.

**Keywords:** Cryptography, Decryption, Encryption, RSA, Security

1. Introduction

In the cryptosystem, two distinctive kinds of keys are used. One is the public-key and the other is a private key. Private Key is stored secretly and the public-key is recognized to all. This process is referred to as an uneven system. The data encrypted utilizing the usage of public-key can solely be decrypted through the private key. In a public-key cryptosystem, the data are saved in secret, with no want to share the data, and no want to share the facts between two parties so that the facts are very tightly closed with less danger to be stolen.

One essential element of the encryption process is that it nearly constantly includes both an algorithm and a key. A key is just some other piece of facts nearly constantly a wide variety that specifies how the algorithm is applied to the plaintext to encrypt it. Even if you be aware of the method by way of which some message is encrypted, it’s hard or impossible to decrypt it except this key.

2. Literature Survey

Cryptosystems are frequently thinking to refer solely to mathematical procedures and computer programs; however, they additionally encompass the rules of human behavior such as selecting challenging bet passwords, logging unused systems, and now not discussing touchy strategies with outsiders.

Using cryptographic techniques, protection execs can:

- Keep the contents of information confidential
- Authenticate the identification of a message’s sender and receiver
- Ensure the integrity of the data, showing that it hasn’t been altered
- Demonstrate that the supposed sender sent this message, a precept recognized as non-repudiation

There are two sorts of cryptography strategies symmetric-key cryptography and Asymmetric-key cryptography. Symmetric key cryptography is a conventional system. To perform operations equal keys are used. The symmetric encryption adjusts plaintext into cipher-text using a secret key and an encryption algorithm. To acquire plaintext form, cipher-text, and the decryption algorithm, the equal keys should be applied to the cipher-text.

In uneven key cryptography, two keys are utilized to scramble and decode a message with the aim that it arrives safely at the receiver. Hence it is also regarded as Public Key Cryptography. In this technique, the key utilized for encryption of a message is not pretty identical as to the key used to decode the message, and each uses two keys, public key and non-public key for encryption and decryption respectively. When a sender desires to talk
with the receiver, the receiver’s public key is utilized to encode the message, and then by way of the use of the personal key the receiver decoded it.

3 Existing RSA Algorithm

RSA stands for Rivest, Adi Shamir, and Leonard Adleman discovered in 1977. RSA is the first successful public key cryptographic algorithm. It is also regarded as an asymmetric cryptographic algorithm because two one-of-a-kind keys are used for encryption and decryption.

RSA is based on the factoring product of two giant prime numbers.

Key generation

1) First, select two prime numbers p & q.
2) Now, calculate n = p x q.
3) Calculate φ(n) = φ(p x q)
   = φ(p) x φ(q)
   = (p-1) x (q-1)
4) Choose an integer e such that 1 < e < φ(n) and also 'e' should be co-prime to φ(n).
5) Now we will determine the value of d. The value of d can be calculated from the formula given below:
   d = e^(-1)(mod φ(n)), d is the multiplicative inverse of e
6) e = 1(mod φ(n))

Encryption

C ≡ m^e (mod n)

Decryption

M ≡ c^d (mod n)

3.1 Existing RK Method

Runge-Kutta methods (there is not simply one) are methods for the numerical answer of regular differential equations. So they are based on applications in actual existence to ordinary differential equations. The RK-RSA algorithm is unique below. Numerical methods like these are normally compared on two simple criteria.

The first main consideration is efficiency. Every algorithm strolling on a computer requires time to run—often counted in phrases of floating-point operations, though for basic evaluation functions it is adequate to consider how many instances the function f(z) has to be evaluated in getting from z_n to z_{n+1}.

The second consideration is accuracy, where precisely Euler’s method falls over and dies.

A numerical approach is only honestly a method if it converges to the authentic vector subject in the limit as the time step Δt → 0.

The fundamental benefits of Runge-Kutta techniques are that they are effortless to implement, every stable, and “self-starting” (i.e., unlike in multi-step methods, we do now not have to deal with the first few steps taken by way of a single step integration technique as a one of a kind case).

4 Proposed RK-RSA Algorithm in Image

The block diagram of the proposed RK-RSA algorithm which is attained by RSA and RK technique is shown in Figure 4.1. The RK-RSA algorithm is detailed below.

The second order RK methods are found using two slopes in the RK methods

\[ y_{i+1} = y_i + \frac{1}{2}(k_1 + k_2) \]

where \( k_1 = hf(x_i, y_i) \) and \( k_2 = hf(x_i + h, y_i + k_1) \)

The original F function is given by

\[ dy/dx = 0.5(y * (1 - (y/100))) \]

Change the above function to produce the proposed algorithm.
5 Experimental Results

A Laptop with Intel(R) Celeron(R) CPU3865U@1.80GHz 1.80GHZ is used in which the performance of the records is added. The experimentation is completed with the input file dimension changing from 226 bytes to 289 bytes. Each file dimension is intended for the average of the ten values (ten times). The overall performance metrics are the encryption time, decryption time, execution time, encryption throughput, decryption throughput, and the avalanche effect. The RK-RSA algorithm has utilized the use of MATLAB.

The experimental effects of numerous performance metrics for the RK-RSA algorithm are detailed below.

5.1 Encryption Time

Following Figure 5.1 suggests the average encryption time for exclusive input sizes. In the bar chart, the average encryption time for the RKRSA_TextImage algorithm compared to that for RSA_Image, RSA_TextImage, and RKRSA_Image algorithm takes the tiniest time. The consequences are exact in Table 5.1.
Table 5.1. Comparative Encryption Times (in Secs)

| Input Size in Bytes | RSA_ Image (ET) | RKRS A_ Image (ET) | RSA_ TextImage (ET) | RKRSA _TextImage (ET) |
|---------------------|----------------|-------------------|---------------------|------------------------|
| 226                 | 3.15475        | 4.92598           | 3.87737             | 3.57537                |
| 252                 | 6.00005        | 9.28442           | 6.95148             | 6.16685                |
| 253                 | 9.50709        | 1.67349           | 10.1304             | 9.41995                |
| 263                 | 16.3055        | 26.209            | 17.5115             | 16.4959                |
| 268                 | 20.3675        | 31.6411           | 21.0661             | 19.5293                |
| 270                 | 16.9957        | 26.1716           | 17.8869             | 16.3652                |
| 279                 | 19.5046        | 31.2599           | 19.7057             | 18.8158                |
| 280                 | 60.6158        | 93.7272           | 57.2517             | 54.9921                |
| 282                 | 32.0438        | 52.3567           | 33.6283             | 31.8259                |
| 289                 | 16.0778        | 23.5446           | 16.1686             | 14.8197                |
| Avg. Time (Secs)    | 20.0575        | 30.0793           | 20.4178             | 19.2005                |

(ET) – Encryption Time

5.2 Decryption Time

Following Figure 5.2 indicates the average decryption time for extraordinary enter sizes. The quantity of decryption time taken by way of the RKRSA_ Image algorithm is the least compared to that for RKRSA_TextImage, RSA_TextImage, and RSA_ Image algorithm. The outcomes are designated in Table 5.2.

![Figure 5.2. Comparison of Average Decryption Time](image-url)
Table 5.2. Comparative Decryption Times (in Secs)

| Input Size in Bytes | RSA_Image (DT) | RK_RSA_Image (DT) | RSA_TextImage (DT) | RK_RSA_TextImage (DT) |
|---------------------|----------------|-------------------|--------------------|-----------------------|
| 228                 | 36.0017        | 8.44781           | 51.89313           | 5.294227             |
| 233                 | 57.0387        | 9.80971           | 56.91759           | 9.6864               |
| 235                 | 93.6248        | 15.2184           | 92.95697           | 14.9447              |
| 263                 | 172.535        | 26.7892           | 164.0542           | 26.41979             |
| 268                 | 198.525        | 32.2379           | 200.123            | 33.05877             |
| 270                 | 167.983        | 26.7514           | 182.8438           | 26.25871             |
| 279                 | 159.767        | 31.8599           | 200.4629           | 31.4311              |
| 281                 | 652.288        | 94.5967           | 603.1465           | 93.09529             |
| 283                 | 326.854        | 53.0836           | 33.1480            | 33.70886             |
| 289                 | 148.837        | 2.11519           | 151.0599           | 24.50115             |
| Avg. Time (Secs)    | 202.744        | 29.7811           | 169.9685           | 31.8459              |

(DT) – Decryption Time

5.3 Execution Time

Following Figure 5.3 suggests the average execution time for exceptionalenter sizes. It is clear from the bar chart, the execution time for the RK_RSA_Image algorithm is the smallest in contrast to that for RK_RSA_TextImage, RSA_TextImage, and RSA_Image algorithm. The results are unique in Table 5.3.

![Figure 5.3. Comparison of Execution Time](image)
Table 5.3. Comparative Execution Times (in Secs)

| Input Size in Bytes | RSA_ Image (EXT) | RKRSA_ Image (EXT) | RSA_ TextImage (EXT) | RKRSA_ TextImage (EXT) |
|---------------------|------------------|--------------------|----------------------|------------------------|
| 226                 | 30.5478          | 5.44795            | 32.3544              | 5.84519                |
| 252                 | 57.6579          | 9.80979            | 60.4838              | 10.2038                |
| 253                 | 94.2575          | 15.2185            | 93.5205              | 15.4719                |
| 263                 | 173.413          | 26.7893            | 164.754              | 26.9816                |
| 268                 | 199.199          | 32.2379            | 200.776              | 33.6510                |
| 270                 | 168.626          | 26.7515            | 163.709              | 27.0318                |
| 279                 | 200.427          | 31.8510            | 201.318              | 32.0263                |
| 280                 | 633.385          | 94.5068            | 604.335              | 93.8661                |
| 282                 | 327.657          | 53.0837            | 335.254              | 54.4028                |
| 289                 | 149.458          | 24.1158            | 151.726              | 25.1346                |
| Avg. Time (Secs)    | 203.464          | 31.98118           | 200.823              | 32.4619                |

(EXT) – Execution Time

5.4 Encryption Throughput

Following Figure 5.4 indicates the assessment of Encryption Throughput of RSA_ Image, RKRSA_ Image, RSA_ TextImage, and RKRSA_ TextImage algorithms with one-of-a-kind enter files. It is viewed from the bar chart, RKRSA_ TextImage algorithm has the best possible encryption Throughput in contrast to RSA_ Image, RSA_ TextImage, RKRSA_ Image algorithms. The results are designated in Table 5.4.

The throughput of the encryption scheme is calculated with the aid of dividing the complete plaintext in megabytes encrypted on the whole encryption time in seconds for every algorithm.

![Figure 5.4. Comparison of Encryption Throughput](image-url)
Table 5.4. Comparison of Encryption Throughput (in Secs)

| Input Size in Bytes | RSA_Image | RKRSA_Image | RSA_TextImage | RKRSA_TextImage |
|---------------------|-----------|-------------|---------------|-----------------|
| 226                 | 3.1547    | 4.92598     | 3.877376      | 3.575375        |
| 252                 | 6.0005    | 9.28442     | 6.951448      | 6.166853        |
| 253                 | 9.5073    | 1.67349     | 10.13034      | 9.419905        |
| 263                 | 16.305    | 26.2090     | 17.51151      | 16.49589        |
| 268                 | 20.367    | 31.6411     | 21.06611      | 19.52931        |
| 270                 | 16.995    | 26.1716     | 17.88693      | 16.3652         |
| 279                 | 19.504    | 31.2599     | 19.70577      | 18.81538        |
| 280                 | 60.615    | 93.7272     | 57.25177      | 54.99201        |
| 282                 | 32.043    | 52.3567     | 33.62813      | 31.82579        |
| 289                 | 16.077    | 23.5446     | 16.16861      | 14.81974        |
| Average             |           |             |               |                 |
| Throughput (KB/Secs)| 1.3272    | 0.88499     | 1.30376436    | 1.3864189       |

5.5 Decryption Throughput

Following Figure 5.5 shows the evaluation of Decryption Throughput of RSA_Image, RKRSA_Image, RSA_TextImage, and RKRSA_TextImage algorithms with distinctive input data files. The bar chart truly shows that the RKRSA_Image algorithm has the best possible decryption Throughput compared to the RKRSA_TextImage, RSA_TextImage, RSA_Image algorithms. The results are targeted in Table 5.5.

The total plaintext in Megabytes divided with the aid of the Decryption time in seconds offers the Decryption Throughput.

![Figure 5.5. Comparison of Decryption Throughput](image)
Table 5.5. Comparison of Decryption Throughput (in Secs)

| Input Size in Bytes | RSA_Img | RKRSA_Img | RSA_TextImage | RKRSA_TextImage |
|---------------------|---------|-----------|---------------|-----------------|
| 226                 | 30.0017 | 5.4478    | 51.80133      | 5.334237        |
| 252                 | 57.0387 | 9.8097    | 59.91959      | 9.6864          |
| 253                 | 93.6248 | 15.2184   | 92.95667      | 14.9447         |
| 263                 | 172.535 | 26.7894   | 164.0542      | 26.41979        |
| 268                 | 198.525 | 32.2378   | 200.1233      | 33.05877        |
| 270                 | 167.983 | 26.7514   | 162.8438      | 26.25871        |
| 279                 | 199.767 | 31.8509   | 200.4629      | 31.4511         |
| 280                 | 632.288 | 94.5067   | 603.1465      | 93.09523        |
| 282                 | 326.854 | 53.0836   | 33.31480      | 53.70888        |
| 289                 | 148.837 | 2.11519   | 151.0599      | 24.50115        |
| Avg. Throughput (KB/Secs) | 0.13129 | 0.89385 | 0.15661725 | 0.835900 |

5.6 Execution Throughput

Following Figure 5.6 suggests the contrast of Execution Throughput of RSA_Image, RKRSA_Image, RSA_TextImage, and RKRSA_TextImage algorithms with exceptional input data files. The RKRSA_Image algorithm has the best execution Throughput compared to the RKRSA_TextImage, RSA_TextImage, RSA_Image algorithms. The outcomes are special in Table 5.6.

Throughput = Total Original text in Megabytes/Execution Time.
Table 5.6. Comparison of Execution Throughput (in Secs)

| Input Size in Bytes | RSA_Image | RKRSA_Image | RSA_TextImage | RKRSA_TextImage |
|--------------------|-----------|-------------|---------------|-----------------|
| 226                | 30.5478   | 5.447905    | 32.35440      | 5.845192        |
| 252                | 57.6579   | 9.809799    | 60.48381      | 10.20382        |
| 253                | 94.2575   | 15.21851    | 93.52045      | 15.47198        |
| 263                | 173.410   | 26.78933    | 164.7524      | 26.98163        |
| 268                | 199.191   | 32.23794    | 200.7762      | 33.65102        |
| 270                | 168.621   | 26.7515     | 163.7093      | 27.03178        |
| 279                | 200.427   | 31.85103    | 201.3180      | 32.02631        |
| 280                | 633.382   | 94.50684    | 604.3352      | 93.86611        |
| 282                | 327.657   | 53.08371    | 335.2540      | 54.40258        |
| 289                | 149.458   | 24.11528    | 151.7262      | 25.13446        |
| Avg.               | 203.460   | 31.98118    | 200.8230      | 32.46149        |
| Throughput (KB/Secs) | 0.13083 | 0.83236    | 0.1325545     | 0.820048        |

5.7 Power Consumption

From the above findings, it is truly proved that the power consumption will be the least for the RKRSA_TextImage algorithm which has the perfect Execution Throughput when in contrast to the RKRSA_TextImage, RSA_TextImage, and RSA_Image algorithms.

5.8 Avalanche effect

Following Figure 5.7 suggests the contrast of Avalanche effect of RSA_Image, RKRSA_Image, RSA_TextImage, and RKRSA_TextImage algorithms with specific enter facts files. The bar chart virtually suggests that the RKRSA_Image algorithm has the lowest Avalanche impact compared to the RSA_Image, RSA_TextImage, RKRSA_TextImage algorithms. The outcomes are specified in Table 5.7.

![Figure 5.7. Comparison of Avalanche effect](image-url)
Table 5.7. Comparison of Avalanche Effect

| Encryption Technique | Avalanche Effect |
|-----------------------|------------------|
| RSA_Image             | 51               |
| RKRSA_Image           | 54.3             |
| RSA_TextImage         | 51.7             |
| RKRSA_TextImage       | 53               |

6 Conclusion

There is the implementation of the RKRSA_Imaga algorithm for higher protection of images. In this work, an image is and chosen RK-RSA algorithm is applied to it. RK-RSA is used in the Aadhaar card with the application of the RK-RSA algorithm for better security. Then an encrypted image is got which is very challenging to decrypt utilizing any other person. So, the conclusion is that the image is more secure.

References

R. K. Sharma, Neeraj Kishore and Parijat Das, “Secure and efficient application of MANET using Identity Based cryptography combined with Visual cryptography technique”, International Journal of Engineering and Computer Science ISSN: 2319-7242, Volume 3, Issue 2, February 2014.

Kumaravel and Ramalatha Marimuthu, VLSI “Implementation of High-Performance RSA Algorithm Using Vedic Mathematics”, International Conference on Computational Intelligence and Multimedia Applications 2007.

Mohsen Bafandekhar and Ramlan Mahmood “A literature review on scalar recoding algorithms in elliptic curve cryptography”, Vol 5, No 4 (2015) https://doi.org/10.15866/irecap.v5i4.5673.

U.S. National Bureau of Standards, “Data encryption standard”, U.S. Fed. Inform. Processing Standards Pub., FIPS PUB 46, January 1977, pp. 2-27.

Dilbag Singh and Ajit Singh, “A Secure Private Key Encryption Technique for Data Security in Model Cryptosystem”, BJIT Journal, ISSN 0973-6568, Vol. 2, BJIT 2010, pp. 251-254, 270.

Nehha Mishra, Shahid Siddiqui, and Jitwst P. Tripathi, “A Compendium over Cloud Computing Cryptographic Algorithms and Security Issues”, BJIT Journal, ISSN 0973-6568, Vol. 7, BJIT 2015, pp.810-814.

A. Nadeem, “A performance comparison of data encryption algorithms”, IEEE information and communication technologies, pp.84-89, 2006.Bn.

Atul Kahte, “Cryptography and Network Security”, Tata McGraw Hill, 2007.

Chen, Z., “The encryption algorithm and security of RSA”, J.Hengyang Normal Univ., 2012, 12:69-69

Wei, X., Z Li and Y. Zhu “On the RSA algorithm and application”, J. Honghe Univ., 2011, 4:31-32

Shi, Z., “Computer Network Security Tutorial”, Tsinghua University Press, 2007, Beijing, China

Cai, C. and Y. Lu, “Asymmetric encryption JAVA and VC Computer Knowledge”, Technol., 2011, 18:4306-4307

Chen, C. and Z. Zhu, “Application of RSA algorithm and implementation details”, Computer Eng. Sci., 2006, 9:13-14

William Stallings, “Cryptography and Network Security”, Fifth Edition, Pearson Education, 2011, pp. 119-120.

M.K.Jain, S.R.K. Iyengar, and R.K.Jain, “Numerical Methods for Scientific and Engineering Computation”, Fifth Edition, New Age International Publishers, 2007, pp. 438-445.

L.F. Shampine and H.A. Watts, “Comparing Error Estimators for Runge-Kutta Methods”, Mathematics of Computation, Vol. 25, Number 115, July 1971, pp.445-455.

S.S.Sastry, “Introductory Methods of Numerical Analysis”, Fourth Edition, 2009, pp. 304-306.

E. Balagurusamy, “Numerical Methods”, Tata McGraw-Hill Education Private Limited, pp. 436-437.

S.R.K.Iyengar and R.K.Jain, “Numerical Methods”, First Edition, New Age International Publishers, 2009, pp. 200-203.

Ashok Kumar and T. E.Unny, “Application of Runge-Kutta method for the solution of non-linear partial differential equations”, Applied Mathematical Modelling, Elsevier, Vol.1, Issue4, March 1977, pp. 199-204.

J.C. Butcher, “A History of Runge-Kutta Methods”, Elsevier, Applied Numerical Mathematics, Vol. 20, 1996, pp. 247-260.

V. Josephraj and B. Shamia Ross, “Enhancement of Blowfish Encryption in Terms of Security Using Mixed Strategy Technique”, IIOAB Journal, ISSN 0976-3104, Vol.7, Special Issue-Emerging Technology in Networking and Security 2016, pp. 69-76.
V. Josephraj and B. Shamina Ross, “A Hybrid Blowfish Encryption Algorithm Using Nash Equilibrium with Cautious Attackers”, International Journal of Control Theory and Applications, ISSN 0974-5572, 2016, pp.4761-4769.
V. Josephraj and B. Shamina Ross, “Security Evaluation of Blowfish and Its Modified Version Using GT’s One-Shot Category of Nash Equilibrium”, International Journal of Control Theory and Applications, ISSN 0974-5572, 2016, pp.4771-4777