
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2025 SCPE. Volume 26, Issues 3, pp. 1092–1103, DOI 10.12694/scpe.v26i3.4108

ENHANCING SECURITY OF CLOUD DATA USING CRYPTOGRAPHIC ALGORITHM BASED ON PFECCRS

AMRUTA GADAD* AND DEVI A†

Abstract. A web-based cloud computing application is basically used to save data with a view of accessing it from anywhere at any time. After analyzing the literature review, it is known that the work for cloud data security is either maintaining the security level or increasing the transmission speed of plain text of cloud, but failed to prove both security level as well as data transmission speed of cloud from one end to another end. Hence, to strengthen the data security of cloud and also to improve the data transmission speed, an integration of encoding, compression and cryptographic algorithms is important. An encoding technique of Prime Factorization (PF) for changing the original plain text into an intermediate plain text as encoded plain text followed by compression technique of Run Length Encoding (RLE) to reduce the file size so that the transmission speed of encoded message will be increased as well as the compression ratio will be higher and finally the Dynamic RSA algorithm is pertained to intensify the security by converting the compressed message into cipher text wherein Integrated Compressed Cryptosystem (ICC) and hence Prime Factorization Encoded Compressed Cryptosystems (PFECCRS) is proposed. The comparative analysis proved that the proposed methodology has increased the security level to 99.25%.

Key words: Prime Factorization, Encoding, Compression, Run Length Encoding, Encryption, Dynamic RSA.

1. Introduction. Cloud computing, a carriage of all computing assistance such as a carrier of software, servers majorly the databases, where each and every human try to save their data on this carriage. The security of this service carrier should be of prime concern to protect it from unauthorised users who may try to alter, destroy or misuse the data. The protection of all forms of cloud data can be done with the help of different concepts of cryptography, combination of compression and cryptography, or combining any mathematical encoding, compression and cryptography. The care must be taken that the data must be secured from several types of attacks such as phishing, replay attacks, cycle attack, fraudulent transactions, data stealing and many more [1]. A data is secured by converting the plain text into an unintelligible form of coded message and this process is called encryption. Transforming the cipher text back to original text is called decryption. The integrated process of encryption and decryption is known as cryptography. There are many cryptographic algorithms being used which are classified based on the type of key used. The usage of both public key and private key is known as asymmetric cryptography and only a single private key is said as symmetric cryptography. Symmetric-key encryption is the process where the plain text is converted into the non-readable text by using anyone of the symmetric-key encryption algorithm [2]. The converted non readable text is again decrypted back to the original plain text using the identical symmetric-key. Similarly asymmetric encryption is the process where the plain text is converted into the cipher text by using two separated keys basically known as public key and private key. The public key is used to convert the original plain text into the cipher text during the transfer from sender to receiver and private key is used during decryption i.e., converting the coded text message back to the original plain text [3].

Many different approaches are analysed to convert the plain text into encoded message using both public key and private key techniques, similarly there are different methodologies for compression, this compression helps in reducing the file size which furthers reduces the transmission speed and required storage space for file. There are mainly two approaches of compression lossless and lossy compression techniques. The lossless compression technique is best approached for text data and lossy works for image and other types of data. [4] Researchers have also showed how the different compression algorithms have also worked efficiently for cloud data[5].

*School of Computer Science and Applications, REVA University, Bangalore, India
†School of Computer Science and Applications, REVA University, Bangalore, India

Continuing further the document is structured into following sections. Background and related work are explained in section 2. Section 3 explains the relevant mathematical work used in this methodology. The proposed technique of Prime Factorization Encoded Compressed Cryptosystems (PFECCRS) is explained in section 4 and continued its illustration with an example in section 5. The experimental results of the same are discussed in section 6. Finally, section 7 ends up with conclusion.

2. Background and Related Work. The most widely used symmetric-key algorithms for data security are the stream cipher and block cipher algorithms. A stream cipher typically works on smaller units of plaintext, usually bits or bytes, whereas a block cipher symmetric-key algorithm converts a fixed length block of plaintext data into a block of ciphertext data of the same length The authors proposed the encoded compressed cryptosystems to improve the security level along with encoding using the Lucas and Fibonacci number systems and proved the security level to be 94.4% for 16MB files after Huffman compression and dynamic RSA [6]. The most commonly and strongly used symmetric algorithms are Advanced Encryption Standards (AES), Data Encryption Standards (DES) and many more [7,8]. Similarly, the asymmetric algorithms that are frequently used are Rivest-Shamir-Adleman (RSA), Elliptic Curve Cryptography (ECC) and some more [9]. The major functions used to analyse the strength of all the cryptographic algorithms are confidentiality, data integrity, security, authentication and non-repudiation.

As cryptography plays a vital role in data security similarly compression algorithms are also used to increase the transmission speed from one end to other end and then the encoding methods are used to convert the plain text into intermediate plaintext, to protect data from the hackers. The different compression algorithms that are widely used for data security are Huffman coding, RLE, Arithmetic encoding, Burrows wheeler transform (BWT) and many other algorithms resulting in good compression ratio by reducing the storage space and increasing the transmission speed [3]. These algorithms are classified as lossless and lossy data compression algorithms. The intermediate plain text can also be formed by different encoding algorithms such as Binary Number Systems, Fibonacci Series Lucas Series, two dimensional matrices and many others which helped to increase the security level of all the designed methodology [6,7,10].

Wid Akeel Awadh, Ali Salah Alasady, Mohammed S Hashim [13] proposed a multilayer data security model where the authors concentrated on merging the cryptographic and compression algorithm and further added a steganographic approach to enhance the security. AES-256 using RSA for encryption followed by Brotli compression and finally the LSB steganography technique ensured to achieve confidentiality, privacy, and integrity of the data. Sunday Adeola Ajagbe, Oluwashola David Adeniji, Adedayo Amos Olayiwola, Seun Femi Abiona [14] here the authors focused on AES based text encryption for NFC using Huffman Compression algorithm. AES was implemented in both ECB and CBC cipher-modes to compare performance, focusing on the time required for encryption. They mainly concentrated on implementing intrusion mitigation system to prevent interference in communication levels and integration with other security measures like multi-factor authentication for fortification. Shiladitya Bhattacharjee, Himanshi Sharma, Tanupriya Choudhury, Ahmed M. Abdelmoniem [15] the authors proposed a combined approach to enhancing encryption and compression algorithms for large data transfer. The chaotic S box encryption and adaptive Huffman compression algorithm proved to achieve superior time and space efficiency with enhanced privacy and integrity for any generic data in terms of entropy, bits per code, information loss percentage, and throughput.

N. Sugirtham, R. Sherine Jenny, B. Thiyaneswaran, S. Kumarganesh, C. Venkatesan, K. Martin Sagayam, Lam Dang, Linh Dinh, Hien Dang, [16] explained using a modified Playfair algorithm, partitioning the plaintext, adding filler characters, inserting filler information, compressing using LZMA, and utilizing a variety of encoding schemes are all part of the methodology. The suggested approach removes fillers for authentic retrieval and fortifies the Playfair cipher. With only minor key changes, the avalanche effect ranges from 65% to 93.7%. For compressed, secure text, the encrypted document is further encrypted using LZMA. The complete study of all such different cryptographic and compression algorithms used for data security are as explained, which says how each cryptographic algorithm merged or unmerged with compression algorithms are how efficient in satisfying any of the parameters like efficiency, security level, integrity and many more [9].

3. Mathematical Background.

Prime Factorization. Let $a_m(n)$ be the sum of the m^{th} powers of the primes in the prime factorization of n. For example, $a1(2^3.5.11^3)$ = 2+2+2+5+11+11+11, $a2(2^3.5.11^3)$ =22+22+22+52+112+112+112, $a5(2^3.5.11^3)$

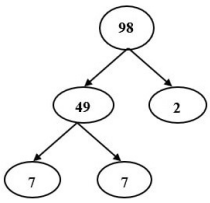

Fig. 3.1: Example for the process of prime factorization

= 25+25+25+55+115+115+115. Let $b_m(n)$ be the mth power of the maximum prime factor in the prime factorization of n. For example, $b1(2^3.5.11^3) = 11$, $b2(2^3.5.11^3) = 112$, $b5(2^3.5.11^3) = 11^5$ [9].
The prime factors of any non-prime integer n can be found among a set.

$$(P1, P2, ..Pk) \; where \;\; Pi \leq \sqrt{n}, 1 \leq i \leq n,$$

where P1, P2,..., Pk are the prime factors that the trivial division method finds for the given number n. This trial division method, divides n by smaller prime numbers (beginning with 2, 3, 5, 7 and so on) in a blind manner and is the simplest way to factor n. If the remainder of division is zero, a prime number is chosen as a factor. This process is continued until all prime numbers that are less than or equal to n are identified and hence is used to factor small integers formed by some digits, but it is not suitable for large numbers due to its enormous time complexity [12].

For example, calculating the prime factors for the ASCII value of letter A which is 98 and hence the value of n = 98. i.e., 98 = 49 x 2, as shown in the Fig. 3.1. The factors found for the number 98 by using trivial division method where the number 98 is divided by the smallest possible prime number 2, in the second step the value 49 is processed through trivial division and hence dividing it by 7 times resulting as 7 x 7. The final obtained prime factors for the number 98 is $2(7^2)$.

4. Proposed Methodology. The proposed methodology is illustrated in Fig. 4.1 which is basically designed to strengthen the data security and increase the transmission speed of plain text during the transfer of data from one end to another end. The plain text PT of cloud server is initially converted into intermediate Encoded Plain Text (EPT) before it is encrypted. The EPT is generated by applying prime factorization for all ASCII values of the plain text, these prime factors are converted into binary digits which intern forms the first level of data security. The EPT is not encrypted directly instead it is processed through the RLE Compression algorithm first forming the next intermediate message known as Intermediate Compressed Message (ICM), this ICM helps in strengthening the rate of data transfer from the user to cloud server. The ICM is finally used in forming the cipher text applying Dynamic RSA where the intermediate message ICM is given as input message for RSA algorithm whose block size is less than n (formed from two distinct large prime numbers). This cipher text on the sender side is converted back to the plain text by reversing the process i.e., the encrypted cipher text is subjected to decryption of Dynamic RSA were finding back the message ICM which is again processed through decompression of RLE obtaining back the Intermediate Decompressed Message IDCM. The IDCM is decoded back to the original PT by using reverse process of prime factorization.

4.1. Prime Factorization. To encode the PT of cloud into intermediate EPT, the prime factors are found for the ASCII values of the plain text. The resultant prime factors are further substituted into equivalent binary numbers. The binary values formed for each ASCII value is considered and the process is repeated for all the letters of the given PT. After conversion of all ASCII values into prime factors followed by substitution into binary numbers, all these binary numbers are merged and found the intermediate EPT. The PT processing through multiple steps to find the intermediate EPT is as shown below in Algorithm 1.

4.2. Compression with Run Length Encoding. Compression techniques are basically classified as Lossy and Lossless compression methods. RLE is also one of the Lossless data compression techniques which is applied when data is the sequence of characters in which some particular characters are repeated consecutively

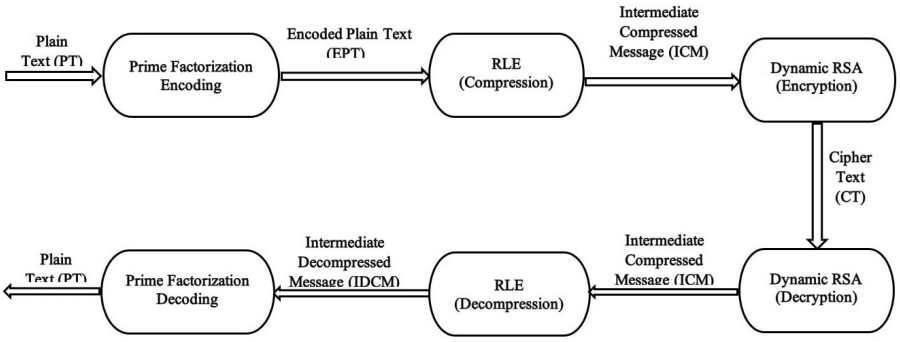

Fig. 4.1: Proposed PFECCRS scheme

Algorithm 1 Prime factorization

BPF Prime- Factorization (N)
// N is an integer for which prime factorization is to be found
// Pn is a prime number n=0,1,2,3,
// BPF is Binary Prime Factorization for N, LBPF is the length of BPF.
// RCW is the Right most Code Word; || is concatenation Input N

Output BPF

1. Read N
2. BPF ← ϕ
3. Find P(i) ← Pn, where i is the position of nearest value of n and Pn is i, i-1, i -2
4. **While**(N=0) do begin
   If P(i) ≤ n **then** P(i) ←1 **else** P(i) ← 0
   End if
   BPF ← BPF || P(i)
   Find the reminder N ←N-P(i) i ← i-1
   End While N
5. **While** (i≠ 0) do begin
   P(i) ← 0
   BPF ← BPF || P(i)
   End while i
6. RCW ← 1
7. BPF ← BPF || RCW
   return BPF

many times. The consecutive repeated characters are compressed by representing it with a number which tells how many times the character has been repeated consecutively.

If the sequence is of the form WWWWWWWWEEEETTTTTTTTTTTTDDDDDDSSSSSSS then this sequence can be compressed using RLE as 7W4E11T5D7S so that instead of occupying 34 bytes of memory it can be reduced to only 11 bytes of memory. This process of RLE is majorly used in image compression and binary sequence compression. As the EPT obtained after PF is a binary sequence of characters, the use of RLE could be justified and the same is elaborated in the algorithm 2 for the BPF to obtain the ICM.

4.3. Dynamic RSA for Encryption and Decryption. The ICM obtained after RLE is taken as the input for converting the ICM into the cipher text which is carried out using Dynamic RSA algorithm. The conventional RSA algorithm usually uses the public key of size 1024 bits or 2048 bits but in this proposed methodology some changes are made, such as limiting the ICM block size to n, which is the product of two powerful prime numbers, p and q, implying that n = p x q. By using the concept of dynamic RSA, the resulted ICM is encrypted as shown in the algorithm 3.

Algorithm 2 RLE Compression

RBPFm is Compressed RLE Code for BPFm
Where m = mi, i=1,2,3, l(m)
Input: BPF(m)
Output: RBPFm

1. RBPFm ← ϕ
2. For each DBPFmi, mi∈m, i=1,2,3…..,n
   If LEN(DBPFmi > 0)
      LC =1
      v = DBPFmi[0]
      v1= DBPFmi[i]
   If (v1 = = v)
      LC=LC+1
   Else
      RBPFm = RBPFm +LC +v
      LC =1
      v= v1
      RBPFm = RBPFm + v1

return RBPFm

Algorithm 3 Dynamic RSA for encryption and decryption

Determine the block size b

3. Generate two large distinct prime p and q, both are of same size.
4. Compute n=pq; $\phi(n)$ = **(p-1) x (q-1)**
5. Convert n into binary nb ← n_2
6. Find b ← Len(nb)
7. Select a random integer e, 1 < e < $\phi(n)$ such that gcd (e, $\phi(n)$)
8. Use the Extended Euclidean Algorithm to compute the unique Integer d
   Such that ed≡1(mod) $\phi(n)$, 1< e < $\phi(n)$
9. A's Public Key is (n,e); A's Private Key is (n,d)
   // RSA Encryption and Decryption Based on Compressed Prime factorization

1. Encryption

1. Obtain A's authenticate Public Key (n,e)
2. c ← ϕ
3. **Repeat**
   i. Read the first b bits from RBPFm
   ii. Convert the bits into binary
      Let it be ICM
      compute CT= $(ICM)^e$ mod n
   iii. c ← c||CT
   iv. Read the next b bits from RBPFm
4. Send the ciphertext C to A

The encrypted message is sent to the sender which is further decrypted back to the DM (Decrypted Message) as shown in the Decryption algorithm.

2. Decryption

1. To recover the CT from C, A do the following
2. Use the private key d to recover
3. ICM = $(CT)^d$ mod n
   Once ICM is obtained the whole process is reversed.

The ICM is decompressed back to IDCM and finally decoded back to the PT by reversing the process of PF.

Table 5.1: Prime factorization encoding to obtain encoded plain text

PT	ASCII Value	Prime Factors for ASCII	Binary Values for Prime Factors																																																
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
b	98	2x7x7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
a	97	1x97	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	98	2x7x7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
a	97	1x97	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Binary Values for Prime Factors																																																	
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			

5. Proposed Methodology – Example. The proposed methodology is explained in brief for the plain text ‘baba’ further used as PT. Here the PT is first converted to its equivalent ASCII values which is further processed through all the different levels of security to find the cipher text. Firstly, the EPT is obtained by finding the prime factors for each letter, such as for the letter ‘b’ the equivalent ASCII value is 98 and its equivalent prime factors are 2 x 7 x 7. After calculating the prime factors, a binary sequence of characters are found for this prime factor by giving the magnitude as 1 for all the values of the obtained prime factors and 0 for all the other numbers i.e. for all non-prime factors, and hence the same process is repeated for all the characters of the plain text ‘baba’ as shown in the Table 5.1.

This sequence of characters is moved to process through RLE Compression Algorithm which results in the following output. RLE o p: ICM: 14802195010148021950l. Its Binary equivalent is 11001101011010111111101011 111001001010111011000100101001101101101 which is called as ICM. The ICM is finally converted into cipher text by using the Dynamic RSA for which the value of two large prime numbers p = 57548534591 and the value of q = 57548534663, then calculating the value of n = 3311833837715018027833, result of multiplication of two prime numbers n = p × q whose binary equivalent is 1011001110001000111011010110000010001101110110100 1010101000100110011100l.

Further $\Phi(n)$ = 57548534590 × 57548534662 = 3311833837599920958580.

Let e = 331185747557334567, using the Extended Euclidean algorithm d is calculated as 20173444791987593 48237. Length of number of bits ‘n’ is 72 and hence the block size should be less than 72 wherein here the block size as 64 bits in intermediate message ICM which is the result of RLE Compression Algorithm. Now using the value of ‘e’ and ‘d’ the PT is encrypted as CT, calculated as CT = $(ICM)^e$ mod n = 14802195021480219501331185747557334567 mod 3311833837715018027833, resulting in CT= 238379815787132 0965090. To further decrypt the encrypted PT, ICM = CT^d mod n = 2383798157871320965090201734447919875 9348237 mod 3311833837715018027833, resulting in ICM = 14802195021480219501. The final PT is received by the receiver by completely reversing the process, i.e., the CT message is decrypted and decompressed into IDCM.

The obtained IDCM is a sequence of binary numbers, for example for the letter ‘b’ the sequence of binary numbers is 0100000000000000000000000000000000000000000000001 as shown in Table 5.2. Decoding this sequence of bits is done by replacing the binary numbers with respective magnitude values wherever the bit value is ‘1’ which results into the respective prime factors obtained for the ASCII value of b, and finally converting the respective ASCII values back to PT.

6. Experimental Results. As explained in the proposed methodology the data security is enhanced by encoding the PT using mathematical concept of PF, followed by compression, the data is compressed using RLE and finally the result of compression is encrypted using Dynamic RSA algorithm. This methodology is implemented in VC++ using Core i5 processor for different text files of different sizes. One text file is created which is containing the information of i2k2 Website and the same is used to generate the text files of different sizes such as 1024 KB, 2048 KB , 4096 KB and so on. The security level is calculated for all these files of

Table 5.2: IDCM for prime factorization decoding

Binary Values for Prime Factors																																																	Prime Factors for ASCII Values	ASCII Value	PT
50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98			
																																																	2x7x7	98	b
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		1x97	97	a
																																																	2x7x7	98	b
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		1x97	97	a

Binary Values for Prime Factors																																																
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6.1: Encryption and decryption time for Conventional RSA and Dymamic RSA before compression

Conventional RSA before Compression										
METHOD	**Encryption Time File Size in KB**					**Decryption Time File Size in KB**				
	1024	**2048**	**4096**	**8192**	**16384**	**1024**	**2048**	**4096**	**8192**	**16384**
FRSA	3806	7448	15015	29871	59649	3918	7570	14900	29940	59538
LRSA	3884	7717	15315	30623	61253	3835	7839	15305	30707	61203
PRSA	4094	8141	16214	32435	64877	4230	8282	16334	32565	64927
Dynamic RSA before Compression										
METHOD	**Encryption Time File Size in KB**					**Decryption Time File Size in KB**				
	1024	**2048**	**4096**	**8192**	**16384**	**1024**	**2048**	**4096**	**8192**	**16384**
FDRSA	3738	7454	14881	29863	59630	3768	7545	14978	29839	59534
LDRSA	3927	7686	15471	30735	61227	3880	7693	15460	30707	61194
PDRSA	4157	8218	16277	32458	64811	4229	8120	16294	32446	64892

different sizes using IBM CAT, which provides a graphical interface for searching, displaying, and analysing data extracted from various cryptographic components. A comparison study is also conducted between the existing method and the proposed method, as well as with conventional and Dynamic RSA with and without the use of a compression algorithm. The different parameters such as encryption time, decryption time, security level and compression ratio are calculated for all file of various sizes which is as shown in the following tables. All these different parameters are calculated using for different file sizes of the cloud data, wherein here the i2k2 cloud desktop as a service is used and the Common Gateway Interface CGI is built for the same.

Table 6.1 and their corresponding graphical representations are shown in Fig. **??** which shows the difference in the encryption and decryption time for text of cloud of different file sizes for all the existing and proposed methodologies using both Prime Factorization Rivest Shamir Adleman PRSA and Prime Factorization Dynamic Rivest Shamir Adleman PDRSA before applying RLE.

The encryption and decryption time taken for the proposed methods PRSA and PDRSA is more than that of the existing methods Fibonacci Rivest Shamir Adleman FRSA and Fibonacci Dynamic Rivest Shamir Adleman FDRSA and Lucas Rivest Shamir Adleman LRSA and Lucas Dynamic Rivest Shamir Adleman LDRSA. The encoding process using PF takes multiple steps like converting the letter to their ASCII values and then finding the prime factors for obtained ASCII value and lastly to encode to their subsequent binary sequence and hence the process of encoding using PF takes more time for encryption and decryption.

Table 6.2 is depicted in Fig. 6.2 which shows the encryption and decryption time for the proposed Prime Factorization Rivest Shamir Adleman Run Length Encoding PRSAR and Prime Factorization Dynamic Rivest Shamir Adleman Run Length Encoding PDRSAR along with existing methods Fibonacci Rivest Shamir Adleman Run Length Encoding FRSAR, Lucas Rivest Shamir Adleman Run Length Encoding LRSAR and Fi-

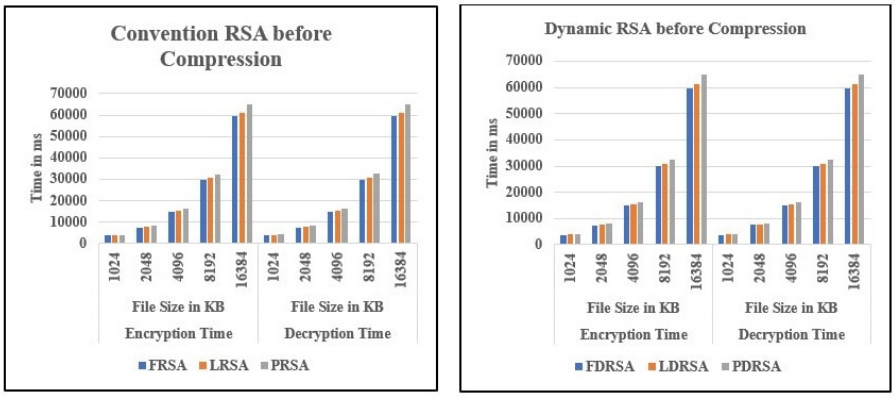

Fig. 6.1: (a) Encryption and decryption time for Conventional RSA before compression; (b) Encryption and decryption time for Dynamic RSA before compression

Table 6.2: Encryption and decryption time for Conventional RSA and Dymamic RSA after compression

Conventional RSA after Compression										
METHOD	**Encryption Time File Size in KB**					**Decryption Time File Size in KB**				
	1024	**2048**	**4096**	**8192**	**16384**	**1024**	**2048**	**4096**	**8192**	**16384**
FRSAR	3899	7637	15000	29802	59558	3869	7446	15069	29805	59658
LRSAR	4013	7916	15489	31155	62081	4010	7897	15613	31021	61984
PRSAR	4101	8115	16344	32403	64961	4066	8189	16323	32534	64841
Dynamic RSA after Compression										
METHOD	**Encryption Time File Size in KB**					**Decryption Time File Size in KB**				
	1024	**2048**	**4096**	**8192**	**16384**	**1024**	**2048**	**4096**	**8192**	**16384**
FDRSAR	3778	7450	14954	29809	59589	3784	7592	14926	29836	59640
LDRSAR	3862	7780	15408	30723	61212	3830	7703	15350	30627	61213
PDRSAR	4193	8279	16242	32579	64895	4097	8102	16349	32575	64837

bonacci Dynamic Rivest Shamir Adleman Run Length Encoding FDRSAR, Lucas Dynamic Rivest Shamir Adleman Run Length Encoding LDRSAR after applying RLE Compression algorithm.

The proposed methodology exhibits lesser encryption and decryption time as there is an increase in file size after using RLE compression algorithm in both conventional and dynamic RSA. The encryption time for 16MB file for FRSA is 59649 ms and for FDRSA is 59630 ms as shown in Table 6.1 but the results of Table 6.2 analyse that there is decrease in encryption and decryption time after the addition of the RLE algorithm.

Fig. 6.3 represents the contents of Table 6.3. in which the compression ratio is calculated for proposed methods and PDRSAR and further the comparison is made with existing methods of FRSAR, LRSAR and FDRSAR and LDRSAR. The compression ratio for the above is calculated using the formula as

$$Compression\ ratio = \frac{Uncompressed\ file\ size}{Compressed\ file\ size} \tag{6.1}$$

As mentioned in the equation 1 the compression ratio is calculated for different file size and for all existing and the proposed methods and the same is shown in the Table 6.3. The compression ratio for the file size of 1MB for FRSAR is 1024/747 = 1.371, for LRSAR is 1024/731 = 1.401 and that of for the proposed PRSAR method is 1024/727 = 1.408. Similarly, the compression ratio for FDRSAR is 1024/758=1.35, for LDRSAR is 1024/733=1.397 and finally the compression ratio for the proposed methodology PDRSAR is 1024/717=1.428.

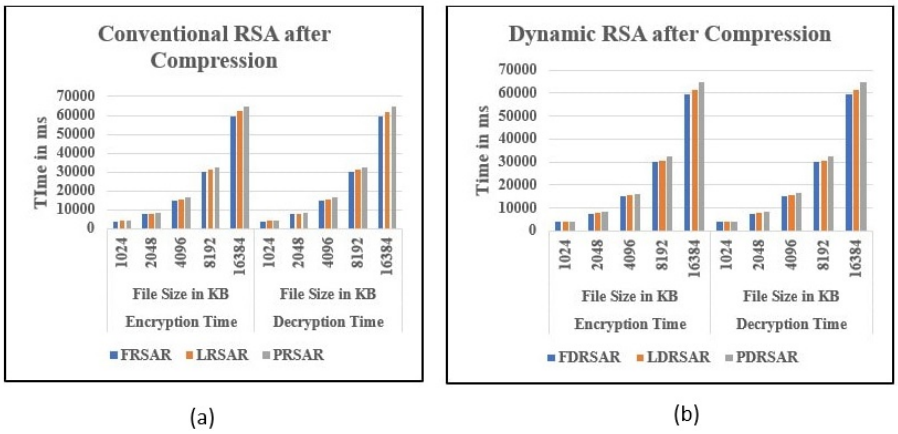

Fig. 6.2: (a) Encryption and decryption time for Conventional RSA after compression; (b) Encryption and decryption time for Dynamic RSA after compression

Table 6.3: Compression rate for Conventional RSA and Dynamic RSA after compression

METHOD	Compression Ratio for Conventional RSA File Size in KB				
	1024	2048	4096	8192	16384
FRSAR	1.371	1.356	1.374	1.375	1.373
LRSAR	1.401	1.382	1.383	1.406	1.391
PRSAR	1.408	1.413	1.422	1.42	1.417
METHOD	Compression Ratio for Dynamic RSA File Size in KB				
	1024	2048	4096	8192	16384
FDRSAR	1.35	1.353	1.361	1.35	1.374
LDRSAR	1.397	1.395	1.394	1.406	1.389
PDRSAR	1.428	1.428	1.428	1.428	1.428

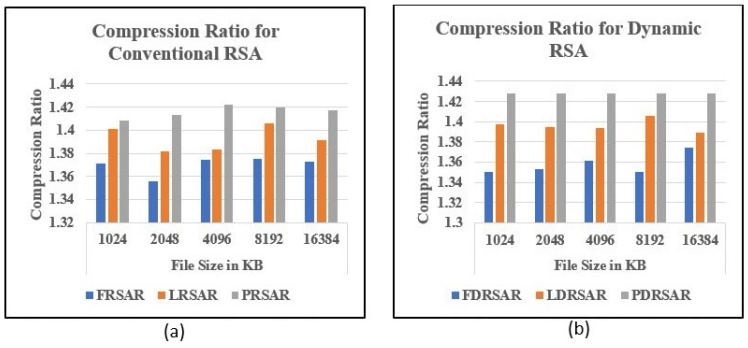

Fig. 6.3: (a) Compression for Conventional RSA after compression; (b) Compression ratio for Dynamic RSA after compression

This analysis clears that the compression ratio is comparatively better for the proposed methods PRSAR and PDRSAR than that of the existing methods.

The security level is analysed for the proposed and existing methods and the comparison analysis is made

Table 6.4: Security level for Conventional RSA and Dynamic RSA before compression

METHOD	Security Level (%) for Conventional RSA File Size in KB				
	1024	2048	4096	8192	16384
FRSA	93.59	92.17	91.49	90.51	89.09
LRSA	95.6	94.755	93.48	92.675	91.98
PRSA	97.89	96.92	95.59	94.96	93.11
METHOD	Security Level (%) for Dynamic RSA File Size in KB				
	1024	2048	4096	8192	16384
FDRSA	94.25	93.65	92.66	91.65	91.13
LDRSA	96.69	95.4	94.49	93.79	92.93
PDRSA	98.47	98.47	98.47	98.47	98.47

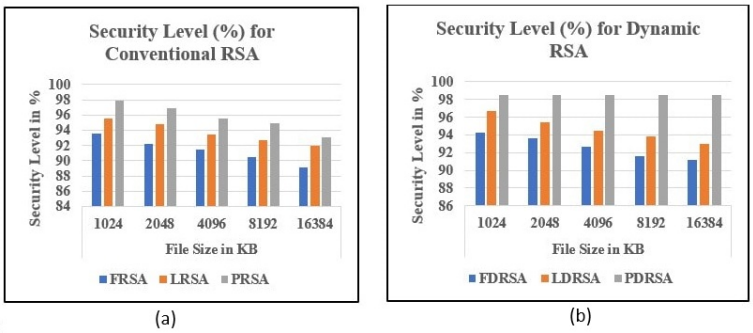

Fig. 6.4: (a) Security level (%) for Conventional RSA before compression; (b) Security level (%) for Dynamic RSA before compression

Table 6.5: Security level for Conventional RSA and Dynamic RSA after compression

METHOD	Security Level (%) for Conventional RSA File Size in KB				
	1024	2048	4096	8192	16384
FRSAR	94.15	92.79	92.13	90.81	90.36
LRSAR	95.45	94.475	93.22	92.865	91.92
PRSAR	98.97	97.72	97.45	96.41	94.94
METHOD	Security Level (%) for Dynamic RSA File Size in KB				
	1024	2048	4096	8192	16384
FDRSAR	94.67	93.93	93.01	92.79	91.7
LDRSAR	96.6	96.24	95.83	94.32	93.82
PDRSAR	99.25	98.67	97.67	96.32	95.54

before using compression and after using compression algorithm.

The results of Table 6.4 shows that the security level is improved in both cases of conventional and dynamic RSA for the proposed methods of PRSA and PDRSA compared to that of existing methods of FRSA, FDRSA and LRSA, LDRSA and the same is graphically represented in Fig. 6.4. The results analysis also depicts that there is a decrease in security level as the size of the file increases for the conventional RSA but that of the proposed methodology of PDRSA the security level remains same for the varying file sizes.

Similarly, the results of Table 6.5. shows the security level which is again compared for the proposed and all the existing methods after applying the RLE compression algorithm which intern is graphically represented in

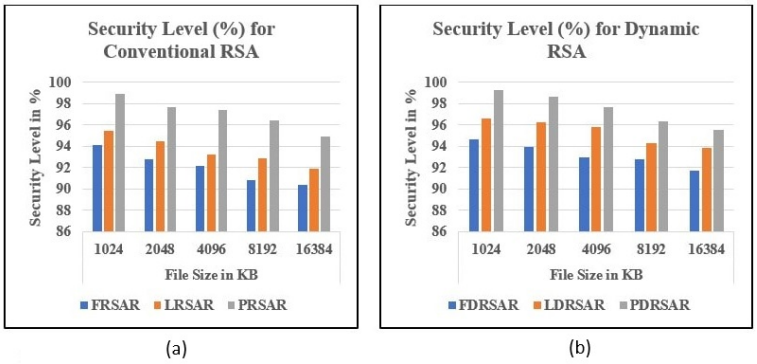

Fig. 6.5: (a) Security level (%) for Conventional RSA after compression; (b) Security level (%) for Dynamic RSA after compression

Fig. 6.5. The security level is proved to be improved for the proposed PRSAR and PDRSAR compared to that of the existing LRSAR, LDRSAR and FRSAR, FDRSAR. The encryption and decryption time is compromised for the proposed methodology as the major aim was to achieve the security level which intern is proved by achieving the results of 99.25%for 1MB file obtained finding multiple intermediate messages.

The encryption time and decryption time is always more when process the plain text through the compressed cryptosystems along with encoding i.e., as in here the plain text is first converted to M1, then moved for compression and lastly it is encrypted. But during this entire process the security level is increased compared to the conventional RSA algorithm applied without any intermediate message like M1 and M2.

Even though the encryption time and decryption time are more for PF compared to Fibonacci and Lucas, the results showed a good progress in the security level and also in the compression ratio. The security level of the plain text is more for the proposed Dynamic RSA Prime factorization compared to the existing Dynamic RSA Fibonacci and Dynamic RSA Lucas which is as shown in the Table 6.5. Along with increase in the security level the concentration is also given for the compression ratio which found to be more efficient for the proposed methodology.

7. Conclusion. The proposed method could be applied for the text data of any cloud-based application and the implementation of the prime factorization to enhance the security level of the cloud data proved to be 95.54% for 16 MB files for the proposed PDRSAR which is improved than LDRSAR that resulted with security level of 93.82% and that for FDRSAR achieving the results of 91.7%. The compression ratio for the proposed method proved to be 1.428 which is improved than that of the existing methods of LDRSAR with 1.389 and that for FDRSAR with 1.374. The compression algorithm helped to improve the compression ratio and also helped to increase the security level by developing the intermediate messages at each stage. The complete work is implemented for the text of the cloud and hence in future it is assured to work the process for the images and then with that of the combined approach wherein in all the three approaches the concentration will be on enhancing the security for the cloud data irrespective of the type of data.

Acknowledgement. I extend my sincere gratitude to REVA University for their invaluable support and guidance throughout the course of this research, which greatly facilitated its successful completion.

REFERENCES

[1] Mohammed Aamir Ali, Muhammad Ajmal Azad, Mario Parreno Centeno, Feng Hao, Aad van Moorsel, “Consumer-facing technology fraud: Economics, attack methods and potential solutions”, Future Generation Computer Systems,Volume 100,2019, Pages 408-427,ISSN 0167-739X.

[2] V. Pavani, P. S. Krishna, A. P. Gopi, and V. L. Narayana, “Secure data storage and accessing in cloud computing using enhanced group based cryptography mechanism,” in Materials Today: Proceedings, Dec. 2020, pp. 1-5, doi: 10.1016/j.matpr.2020.10.262.

[3] A. Devi and K. Mani, "CSEIT1831152 | Enhancing Security in RSA Cryptosystem Using Burrows-Wheeler Transformation and Run Length Encoding," 2018. [Online]. Available: www.ijsrcseit.com.
[4] Luluk Anjar Fitriya, Tito Waluyo Purboyo, Anggunmeka Luhur Prasasti, "A Review of Data Compression Techniques", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 8956-8963.
[5] I. Sandhya Rani and Bondu Venkateswarlu, "A Systematic Review of Different Data Compression Technique of Cloud Big Sensing Data" ICCNCT 2019, LNDECT 44, pp. 222–228, 2020. doi.org/10.1007/978-3-030-37051-0-25
[6] K. Mani and A. Devi, "Enhancing security in cryptographic algorithm based on LECCRS," 2017. [Online]. Available: http://en.wikipedia.org/wiki/
[7] M. Tajammul and R. Parveen, "Auto encryption algorithm for uploading data on cloud storage," International Journal of Information Technology (Singapore), vol. 12, no. 3, pp. 831–837, Sep. 2020, doi: 10.1007/s41870-020-00441-9.
[8] P. William, A. Choubey, G. S. Chhabra, R. Bhattacharya, K. Vengatesan, and S. Choubey, "Assessment of Hybrid Cryptographic Algorithm for Secure Sharing of Textual and Pictorial Content," in Proceedings of the International Conference on Electronics and Renewable Systems, ICEARS 2022, 2022, pp. 918–922. doi: 10.1109/ICEARS53579.2022.9751932.
[9] K. Mani and A. Devi, "Modified DES using Different Keystreams Based On Primitive Pythagorean Triples," International Journal of Mathematical Sciences and Computing, vol. 3, no. 1, pp. 38–48, Jan. 2017, doi: 10.5815/ijmsc.2017.01.04.
[10] Amruta Gadad, Devi Anbusezhiyan, "Cloud security: literature survey", International Journal of Electrical and Computer Engineering (IJECE), Vol. 13, No. 4, August 2023, pp. 4734 4742, ISSN: 2088-8708, DOI: 10.11591/ijece.v13i4.pp4734-4742
[11] J. Zalaket and J. Hajj-Boutros, "Prime factorization using square root approximation," Computers and Mathematics with Applications, vol. 61, no. 9, pp. 2463–2467, May 2011, doi: 10.1016/j.camwa.2011.02.027.
[12] R. Jakimczuk, "Sums of Prime Factors in the Prime Factorization of Smooth Numbers Diophantine equations View project Prime Numbers View project Sums of Prime Factors in the Prime Factorization of Smooth Numbers".
[13] Wid Akeel Awadh, Ali Salah Alasady, Mohammed S. Hashim, "A multilayer model to enhance data security in cloud computing", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 32, No. 2, November 2023, pp. 1105 1114, ISSN: 2502-4752, DOI: 10.11591/ijeecs.v32.i2.pp1105-1114.
[14] Sunday Adeola Ajagbe, Oluwashola David Adeniji, Adedayo Amos Olayiwola, Seun Femi Abiona, "Advanced Encryption Standard (AES)-Based Text Encryption for Near Field Communication (NFC) Using Hufman Compression", SN Computer Science (2024) 5:156, https://doi.org/10.1007/s42979-023-02486-6.
[15] Shiladitya Bhattacharjee, Himanshi Sharma, Tanupriya Choudhury, Ahmed M. Abdelmoniem, "Leveraging chaos for enhancing encryption and compression in large cloud data transfers", The Journal of Supercomputing.
[16] N. Sugirtham, R. Sherine Jenny, B. Thiyaneswaran, S. Kumarganesh, C. Venkatesan, K. Martin Sagayam, Lam Dang, Linh Dinh, Hien Dang, "Modifed Playfair for Text File Encryption and Meticulous Decryption with Arbitrary Fillers by Septenary Quadrate Pattern", International Journal of Networked and Distributed Computing, https://doi.org/10.1007/s44227-023-00019-4.

Edited by: Dhilip Kumar V
Special issue on: Unleashing the power of Edge AI for Scalable Image and Video Processing
Received: Jun 8, 2024
Accepted: Aug 29, 2024