Journal of Information Technology, computer science and Electrical Engineering (JITCSE)
Vol. 1 No. S1 (2024): Special Edition
ISSN: 3046-3900, DOI: 10.61306/jitcse.v1i1.17

Implementation of the Asymmetric Fountain Code Algorithm to Secure Aviation Licensing and Regulatory Document Data

1st Daniel D Rumani
Department of Fixed Wing Aviator
Akademi Penerbang Indonesia
Banyuwangi
Banyuwangi, Indonesia
danielrumani@ymail.com

***Abstract*— Several agencies, both government and private, require the security of personal data, especially as this information can only be accessed or transmitted to authorized people or organizations. Likewise with license and regulatory records, these materials are exceedingly confidential. Documents pertaining to permissions and regulations issued by aviation authorities, such as operating permits and aircraft certification, However, the difficulty is that when data is distributed using an internet connection, it can slip into the hands of unscrupulous people. Implementation of the Asymmetric Fountain Code Algorithm to safeguard confidential text data is used to safeguard license and regulatory document data with encryption and description methods using the Fountain Code Algorithm, which has two keys to access plaintext and ciphertext documents using SharpDevelop 5.1 software in the C# language. This research reveals that the outputs of the Fountain Code algorithm can restore plaintext data in its entirety. So it can be inferred that the asymmetric Fountain Code algorithm meets good data integrity and is safe; the complexity of the algorithm is simpler; and the amount of plaintext is exactly related to time.**

Keywords*— *Cryptography, Data Security, Fountain Code Algorithm,

I. INTRODUCTION

Some organizations or companies really need confidential data [1][2]. Data confidentiality is the sustainability of programs and performance, Especially in licensing and regulatory document data, which contains papers pertaining to permits and rules issued by aviation authorities, such as operating permits and aircraft certification[3]. In securing data, one of them is using a cryptographic algorithm consisting of an encryption algorithm and a decryption algorithm [4][5]. Data security problems can be addressed using the Rivest Shamir Algorithm approach Adleman (RSA) is a public key cryptography algorithm [6]. This method uses three steps, namely the process of key formation, encryption and decryption [7]. The RSA algorithm has difficulty to solve because of the exponential process and factoring a number into 2 prime numbers. So it takes a long time to find the factor values [8]. The RSA algorithm will be used as security for text files. This algorithm is difficult to solve because the size of the encryption results is large and the generating factors are two different prime numbers.

Fountain Code is a class of erasure codes, with the potentially sequence-limited property of symbol encoding can result from the set of source symbols such that the original source symbols are ideally 10 can be recovered from any subset of coding symbols with a size equal to or only slightly larger than the number of source symbols. Fountain Code indicates that it does not have fixed tariff codes. Fountain Code is known has efficient coding and decoding algorithms. Fountain Code can be applied flexibly at fixed code rates, or where fixed code rates cannot be determined a priori, and where encoding and decoding efficient use of large amounts of data.

This research uses the Asymmetric RSA Fountain Code algorithm to encrypt the decryption of text data, where in securing the text data there are two different keys, namely (public key) for encrypting the plaintext and (private key) to decrypt the ciphertext.

II. BACGROUN STUDY

Cryptography primarily investigates how a message might be hidden, which relates to issues of information security such as data secrecy, data validity, data integrity, and data authentication.Cryptography is a term for encryption, the conversion ofReadable sentences appear nonsensical or illegible . The creator of the encrypted message exposes the decryption techniques needed to restore the original content only to the intended recipient, preventing undesired people from reading the text message [9]. Modern cryptography works in bit mode rather than character form. Operation in meaningful bit mode: all data and information (both keys, plaintext and ciphertext) are expressed in the form of strings, binary bits (0 and 1) Encryption and description algorithms process all data and information in bit mode. The sequence of bits that represent plaintext is encrypted into ciphertext in the form of a series of bits, and vice versa. Cryptography primarily investigates how a message might be hidden, which relates to issues of information security such as data secrecy, data validity, data integrity, and data authentication. Cryptography is a term for encryption, the conversion ofReadable sentences appear nonsensical or illegible [10]. The creator of the encrypted message exposes the decryption techniques needed to restore the original content only to the intended recipient,

preventing undesired people from reading the text message [11].

Rivest Shamir Adleman (RSA) in the realm of cryptography is an algorithm for public key encryption. RSA was the first method suited for digital signatures as well as encryption and one of the most advanced in the field of public key cryptography. RSA is still frequently used in electronic commerce protocols and is believed to guarantee security with quite long keys [12].

As a public key algorithm, RSA has two keys, namely the public key and the private key. The public key may be known by anyone and can be used for the encryption process. Meanwhile, only certain people can know the private key and can utilize it for the decryption procedure.The security of RSA passwords lies in the difficulty of factoring huge integers. Until today, RSA is still trusted and frequently used on the internet. The RSA password public key algorithm scheme consists of three phases, namely the key formation process, the encryption process, and the decryption process. Previously, we were provided with various mathematical computation ideas used by RSA [13].

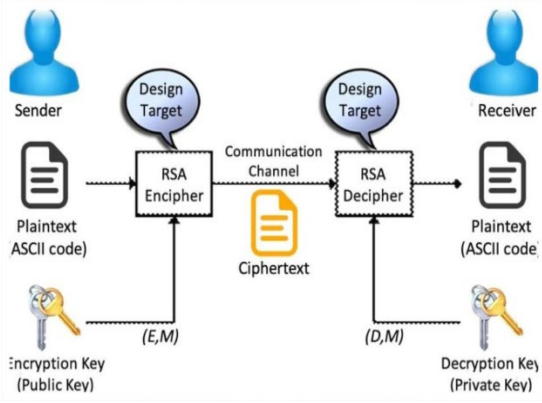

Figure I. RSA Algorithm

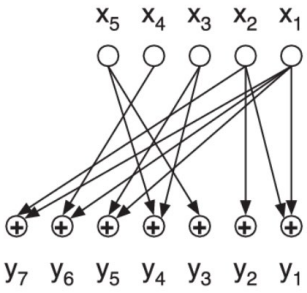

Figure II. Fountain Code Algorithm Scenario for Securing Aviation Licensing Data and Regulatory Documents

Fountain codes, also known as rateless erasure, are a class of erasure codes with the property that a potentially finite sequence of encoding symbols can be generated from a set of source symbols such that the original source symbols can ideally be recovered from any subset of encoding symbols of the same or only slightly larger size. from a number of source symbols [14]. Fountain code generates a theoretically limitless stream of new code symbols on demand and decodes symbols as they become available, offering exceptionally efficient message encoding and recovery [15]. Fountain code can be implemented flexibly with a set code rate (the fraction of relevant data flow) and where efficient coding and decoding of a large amount of data is required. Fountain code is employed in data storage applications because of large savings in the number of storage units for a given level of redundancy and limitations [16]. One of the coding requirements for a data storage system is a systematic form; that is, the original message symbol is part of the coded symbol. In addition, because bandwidth (the capacity or tamping power of an Ethernet cable to allow a certain amount of data packet traffic to pass) and communication load between storage nodes can be a bottleneck, code that allows minimum communication is very useful, especially when nodes fail and system reconstruction is required to achieve the desired level. So the availability of the Fountain code enables an expedient repair process in case of failure. When one or more encoded symbols are lost, it should not take too much communication and calculation between other encoded symbols to resuscitate the lost symbols [17]. Fountain codes can be employed flexibly at fixed code levels, or when fixed code levels cannot be identified a priori and efficient coding and decoding of huge volumes of data are necessary. One example is a data carousel, in which multiple huge files are simultaneously broadcast to a network of receivers. Using fixed-rate erasure codes, a receiver that loses a source symbol (due to a transmission fault) faces a coupon collector problem: the receiver must successfully receive an encoding symbol that it does not already have. This difficulty becomes considerably more evident when using typical short-length erasure codes, as the file must be broken into blocks, each encoded independently. The recipient must now collect the number of missing encoding symbols required for each block [18]. By employing the fountain code, it is enough for the recipient to take each group of coding symbols with a size somewhat greater than the source symbol set. (In fact, broadcasts are usually scheduled for a given time period by the operator depending on network and receiver characteristics and the required delivery reliability, and thus the fountain code is used at a dynamically determined code rate at the time the file is scheduled for broadcast.)

TABLE I. BACKGROUND STUDY CITATION OF PUBLICATION

No.	Research Year	Research methods	Purpose
1.	2010 [19]	Fountain Codes	Data decryption encryption system using Matlab
2.	2017 [20]	Fountain Codes	Symmetric encryption algorithm that has been proven to have high security
3.	2020 [21]	Digital Fountain Codes	The simulation results show that the performance of the algorithm is significantly improved compared with the traditional coding algorithm

4.	2021 [22]	Analog Fountain Codes	The results show that more precoding rates result in better low realization rates over a wide range for short information block lengths
5.	2022 [23]	Secure and Private Fountain Code-based Architecture for Blockchain	two new architectures for blockchain-based systems, which reduce storage and communication costs associated with blockchain historical data and provide simultaneously confidentiality of stored data
6.	2023 [24]	Triangular Code	A new class of fountain codes with near-zero redundancy and the computational complexity of linear coding and decoding of TC, TC reduces the redundancy of LT codes by 68%–99%

III. METHODE

The approach method offered in this research uses the Rivest Shamir Adleman (RSA) algorithm. The following is the research framework carried out in Figure 1.

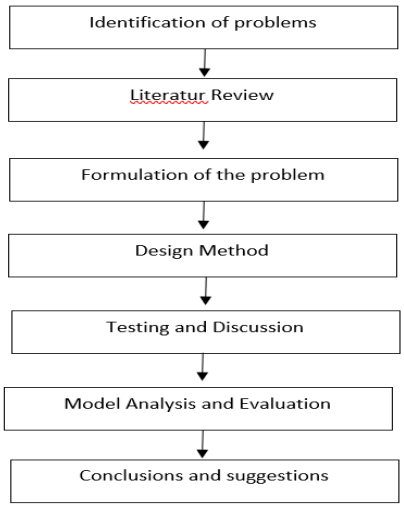

Figure III. Reasearch Methode

This study stage begins with identifying the problem. The challenge that will be solved in this research is how to safeguard aviation licensing data and regulatory papers utilizing the Asymmetric Fountain Code algorithm technique.The literature review is used to help with the resolution of the problems to be investigated. The literature review involves the development of research directions that have been carried out before as a reference for research and research development. Review of studies made on cryptography and the Asymmetric Fountain Code Algorithm Determining the problem formulation so that method design can be carried out. The program design process presented is explained in general by the development architectural model, as seen in the figure IV.

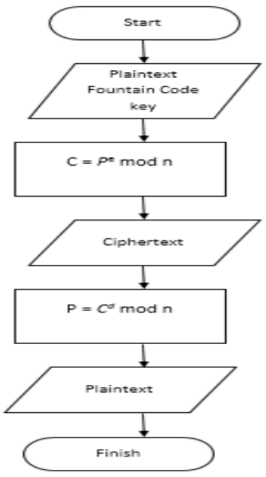

Figure IV. General Architecture Model Development

This researcher uses characters from ASCII as characters text after encryption. The characters used are as follows:

1. Letters :
   - A is the 0th character
   - B 1st character
   - C 2nd character
   - D 3rd character
   - E 4th character
   - F 5th character
   - G 6th character
   - H 7th character
   - I 8th character
   - J 9th character
   - K 10th character
   - L is the 11th character
   - M is the 12th character
   - N 13th character
   - O 14th character
   - P 15th character
   - Q is the 16th character
   - R 17th character
   - S is the 18th character
   - T is the 19th character
   - U is the 20th character
   - V 21st character
   - W is the 22nd character
   - X 23rd character
   - Y 24th character
   - Z 25th character
   - a 26th character
   - b 27th character
   - c 28th character
   - d 29th character
   - e 30th character
   - f 31st character
   - g 32nd character
   - h 33rd character
   - i 34th character
   - j 35th character
   - k 36th character
   - l 37th character
   - m 38th character
   - n 39th character
   - o 40th character
   - p 41st character
   - q 42nd character
   - r 43rd character
   - s the 44th character
   - t the 45th character

- u 46th character
- v 47th character
- w 48th character
- x 49th character
- y 50th character
- z 51th character

2. Numbers :
- 0 to 52 characters
- 1 character to 53
- 2 characters to 54
- 3 characters to 55
- 4 characters to 56
- 5 characters to 57
- 6 characters to 58
- 7 characters to 59
- 8 characters to 60
- 9 characters to 61

3. Symbol :
- . 62nd character
- , 63rd character
- ! 64th character
- ? 65th character
- /66th character
- : 67th character
- = 68th character
- (space) 69th character
- (70th character
-) 71st character
- | Σ | = 72

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

The Fountain Code algorithm cryptography process consists of 3 stages, namely:

1. Key Generation
   a. Determine the values of p and q as two arbitrary prime numbers.
   b. Calculate $n = p\ x\ q$ (preferably p is not the same as q).
   c. Calculate $m = p(p-1)(q-1)$d. Determine the public key, e, that is prime relative to m $(\gcd(e,m)-1)$.
   e. Find d, so that $e * d = 1$ mod (m), or $d = (1+nm)/e$ for numbers big
2. Encryption Process
   a. Retrieve the message recipient's public key, e, and modulus n.
   b. Define plaintext P into blocks P1, P2, ..., so that each block displays values in the interval [0, n – 1].
   c. Each Pi block is encrypted into a Ci block with the formula $Ci = Pi\hat{}mod\ n$
3. Decryption Process Each block of ciphertext ci is decrypted again Pi block with the formula $Pi = Ci^d mod\ n$.

Asymmetric Fountain Code Psedocode Algorithm Program to Secure Aviation Licensing Data and Regulatory Documents.

1. Pseudocode Encryption Fountain Code

```
int encrypt(data, keys, out=None, bsize=BLOCK_SIZE,
pkcs7=True):


    q, r = divmod(len(data), bsize)

    q = q if r == 0 else q + 1


    # check pkcs7 and block


if not pkcs7 and r != 0:

        raise Exception('pkcs7 is disabled, but given unblocked
data')
```

2. Pseudocode Description Fountain Code

```
int decrypt(cliper, keys, out=None, bsize=BLOCK_SIZE,
pkcs7=True):


 if not out:

        out = io.BytesIO()

    q, r = divmod(len(cliper), bsize)


if r != 0:

        raise Exception("cliper is broken")
```

3. Fountain Code Key Pseudocode

```
int key_check(key):

    ksize = len(key) * 4


if isinstance(key, str):

        try:

            int(key, 16)

        except ValueError:

            raise Exception('key not a valid hex str')

        q, r = divmod(ksize, 8)


if q == 0 or q > 32 or r != 0:

            return False

        return True

    return False
```

IV. Result And Discussion

Testing was carried out on the fountain code algorithm to measure the success of the system in carrying out the encryption and decryption operations. In this test, the

parameter calculated is the time the load code algorithm conducts encryption and decryption. This test is carried out with numerous criteria, as follows:

1. Plaintext in the form of sentences consisting of ASCII printable characters with a length of 12, 20, and 60 characters sourced from text that is input in the textbox.
2. The list used is Σ = { A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U ,V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, ., ,, !, ?, /, :, =, -, (,) }

   | Σ | = 72
3. The fountain code key consists of ASCII Printable characters. From the accumulation of character indices and numbers, the length of this key can be equal to the length of the plaintext entered.
4. The decryption encryption process uses the fountain code algorithm with the encryption calculation formula C = Pe mod n and P = Cd mod n.

```
--------------------------------------------------------------------------------
keys:
[(8, '53'),
 (16, '53d4'),
 (32, '53d42ffe'),
 (64, '53d42ffefe5c71be'),
 (128, '53d42ffefe5c71befa625b1e093463db'),
 (256, '53d42ffefe5c71befa625b1e093463db65e4c77e1a67c0b4aeab205d7b17bd69')]
--------------------------------------------------------------------------------
Tester Result:
          key         status     runtime(s)

            8        success       0.014081
           16        success       0.022192
           32        success       0.039784
           64        success       0.074691
          128        success       0.143053
          256        success       0.278241
--------------------------------------------------------------------------------
```

5. Display of the Asymmetric Fountain Code Algorithm Program to Secure Aviation Licensing Data and Regulatory Documents.

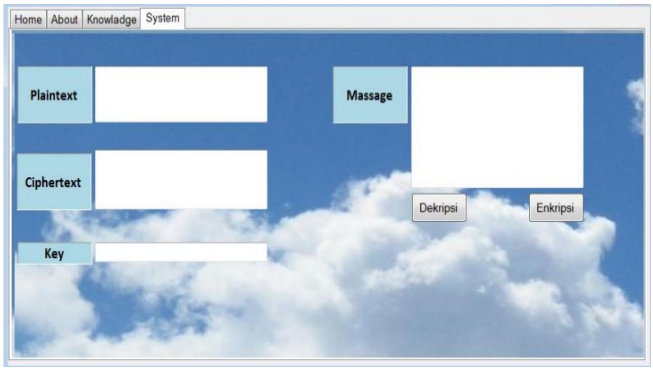

Figure V. Display of the Asymmetric Fountain Code Algorithm Program to Secure Aviation Licensing Data and Regulatory Documents

V. CONCLUSION

Based on the outcomes of constructing a text data security application system by using the asymmetric fountain code method, the author derived the following conclusions:a. In safeguarding text data, the Fountain Code algorithm is ideal to utilize to help keep the text data confidential.b. The process of encrypting and decrypting the Fountain Code algorithm meets data integrity requirements.b. After applying the Fountain Code algorithm to protect text data, it is known that the length of plaintext is directly proportional to time. So the longer the text, the more time it will take.d. Based on changes in the character of each test data, the Fountain Code technique is relatively secure and simple.

REFERENCES

[1] W. W. Widiyanto, D. Iskandar, S. Wulandari, E. Susena, and E. Susanto, "Implementation Security Digital Signature Using Rivest Shamir Adleman (RSA) Algorithm As A Letter Validation And Distribution Validation System," in *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, 2022, pp. 599–605.

[2] N. P. Douglass, J. Langel, W. J. Moore, L. Ng, R. M. Dudukovich, and S. Mal-Sarkar, "Application of Fountain Code to High-rate Delay Tolerant Networks," *IEEE Access*, 2023.

[3] J. Juhari and M. F. Andrean, "On the application of noiseless steganography and elliptic curves cryptography digital signature algorithm methods in securing text messages," *CAUCHY J. Mat. Murni dan Apl.*, vol. 7, no. 3, pp. 483–492, 2022.

[4] A. Hamza and B. Kumar, "A review paper on DES, AES, RSA encryption standards," in *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, 2020, pp. 333–338.

[5] J. Jamaludin and R. Romindo, "Implementation of Combination Vigenere Cipher and RSA in Hybrid Cryptosystem for Text Security," *IJISTECH (International J. Inf. Syst. Technol.*, vol. 4, no. 1, pp. 471–481, 2020.

[6] K. N. Jassim *et al.*, "Hybrid cryptography and steganography method to embed encrypted text message within image," in *Journal of Physics: Conference Series*, 2019, vol. 1339, no. 1, p. 12061.

[7] D. Y. SYLFANIA, F. P. JUNIAWAN, and H. A. PRADANA, "Blowfish–RSA Comparison Analysis of the Encrypt Decrypt Process in Android-Based Email Application," in *Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019)*, 2020, pp. 113–119.

[8] M. Nurullaev and R. D. ALOEV, "Software, algorithms and methods of data encryption based on national standards," *IIUM Eng. J.*, vol. 21, no. 1, pp. 142–166, 2020.

[9] A. Tidrea, A. Korodi, and I. Silea, "Elliptic Curve Cryptography Considerations for Securing Automation and SCADA Systems," *Sensors*, vol. 23, no. 5, p. 2686, 2023.

[10] S. S. Roy and U. Banerjee, "Preventing a Cryptoapocalypse: From mathematics to circuits for postquantum cryptography," *IEEE Solid-State Circuits Mag.*, vol. 15, no. 1, pp. 38–44, 2023.

[11] S. Ullah, J. Zheng, N. Din, M. T. Hussain, F. Ullah, and M. Yousaf, "Elliptic Curve Cryptography; Applications, challenges, recent advances, and future trends: A comprehensive survey," *Comput. Sci. Rev.*, vol. 47, p. 100530, 2023.

[12] Q. Zhang, X. Li, M. Yang, Y. Liu, and H. Zhang, "Adaptive Space-Time Short-Frame Fountain Code With Plasma Sheath Channel," *IEEE Trans. Plasma Sci.*, 2023.

[13] L. Laouamer, M. Al Shaikh, L. T. Nana, and A. C. Pascu, "Informed symmetric encryption algorithm for DICOM medical image based on N-grams," in *2013 Science and Information Conference*, 2013, pp. 353–357.

[14] H. Nie, X. Jiang, W. Tang, S. Zhang, and W. Dou, "Data security over wireless transmission for enterprise multimedia security with fountain codes," *Multimed. Tools Appl.*, vol. 79, pp. 10781–10803, 2020.

[15] Z. Qin, J. Huang, and Z. Fei, "Improved HTLO Algorithm for On-Line Fountain Codes with Limited Feedback," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1–6.

[16] Y. Xue, Z. Sun, Z. Liang, C. Fan, and X. Xu, "Data collection method of space-based internet of things based on layered fountain code," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, 2020, pp. 2475–2479.

[17] Z. Zheng, L. Yuan, and F. Fang, "Performance analysis of fountain coded non-orthogonal multiple access with finite blocklength," *IEEE Wirel. Commun. Lett.*, vol. 10, no. 8, pp. 1752–1756, 2021.

[18] H. Shi, J. Lei, J. Tu, and G. Qiu, "An Optimized Algorithm on the Design of Degree Distribution for Fountain Code based on Physical Layer Security," in *Proceedings of the 2020 4th International*

Conference on Electronic Information Technology and Computer Engineering, 2020, pp. 44–49.

[19] G. Joshi, J. B. Rhim, J. Sun, and D. Wang, "Fountain codes," in *Global telecommunications conference (GLOBECOM 2010)*, 2010, pp. 7–12.

[20] M. Hu, W. Li, F. Yu, and X. Hu, "The Fountain-codes-based Encryption and Decryption Algorithm Research," in *2017 2nd International Conference on Control, Automation and Artificial Intelligence (CAAI 2017)*, 2017, pp. 386–389.

[21] Y. Yang and X. Jiang, "A selective enhancement degree and non-uniform selection encoding algorithm for digital fountain code," in *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, 2020, pp. 389–393.

[22] W. J. Lim, R. Abbas, Y. Li, B. Vucetic, and M. Shirvanimoghaddam, "Analysis and design of analog fountain codes for short packet communications," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 12662–12674, 2021.

[23] J. Singh, A. Banerjee, and H. Sadjadpour, "Secure and private fountain code based architecture for blockchains," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 1521–1526.

[24] J. Qureshi and C. H. Foh, "Triangular code: Near-optimal linear time fountain code," *Digit. Commun. Networks*, vol. 9, no. 4, pp. 869–878, 2023.