Reducing the e-KYC file searching time in the blockchain system using searchable symmetric encryption and turbulence padded chaotic map

Lalu Raynaldi Pratama Putra
Graduate School of Forensic Science and Cyber Security
Telkom University
Bandung, Indonesia
laluraynaldipp@student.telkomuniversity.ac.id

Ari Moesriami Barmawi
Graduate School of Forensic Science and Cyber Security
Telkom University
Bandung, Indonesia
mbarmawi@melsa.net.id

***Abstract*—E-KYC systems often face severe challenges regarding the security and privacy of the related documents stored in the cloud, which becomes a crucial issue. As the volume of data continues to grow, efficient verification becomes increasingly critical. Traditional methods, which require files to be verified individually, are time-consuming and inefficient. The proposed system implements Searchable Symmetric Encryption is used to handle searches from large data sets and maintain the security aspect of seed generation using Turbulence Padded Chaotic Map. Experimental research shows that the time for data searching on large datasets improved significantly while maintaining security.**

***Index Terms*—e-KYC,searchable symmetric encryption,turbulence padded chaotic map,blockchain**

I. Introduction

The implementation of Know Your Customer (KYC) processes are essential for banks and financial institutions to verify the identities of their customers [1]. With technological advancements, many institutions are transitioning from traditional KYC to electronic KYC (e-KYC) systems. However, this shift has introduced new challenges related to efficiency, effectiveness, and customer experience [2]. Customers often repeat the data entry process, and institutions find it difficult to manage and secure the large volumes of sensitive data involved. Furthermore, the use of cloud-based e-KYC systems raises significant privacy concerns. Storing sensitive customer data in the cloud increases the risk of unauthorized access and data breaches [3]. While some banks implement encryption and decryption mechanisms on the cloud side to enhance security, this introduces key distribution and centralized validation issues, particularly concerning key revocation [2]. As the volume of data continues to grow, efficient verification becomes increasingly critical. Traditional methods, which require files to be verified individually, are time-consuming and inefficient. Batch verification could offer a solution by allowing multiple files to be verified simultaneously. However, existing e-KYC systems have no batch verification capabilities, particularly in decentralized environments such as blockchain networks [2]. This study addresses these challenges by proposing the use of Searchable Symmetric Encryption (SSE)—a cryptographic method that allows encrypted data to be searched efficiently [4] [5]—integrated with blockchain technology [6]. This research introduces a novel enhancement to the Pseudorandom Number Generation PRNG within the SSE framework by employing turbulence-padded chaotic map to further enhance the efficiency and maintain the search method's security. Based on the experimental results, it can be proven that the proposed method is more efficient in terms of search time to search a large amount of data compared to the previous one while maintaining the security aspects.

II. Fugkeaw Method

Recent research highlights ongoing challenges in balancing security, privacy, and regulatory compliance within cloud-based e-KYC systems, particularly around issues of data exposure and inefficient key management [2]. The increasing reliance on cloud-based e-KYC systems in financial services has led to critical security and privacy concerns, primarily due to risks associated with unauthorized data exposure and complex key management. Existing solutions have addressed some security aspects but need more essential privacy compliance features, such as client consent enforcement and fine-grained access control, which are necessary to meet regulatory requirements like GDPR. Fugkeaw [2] introduces e-KYC TrustBlock, a novel blockchain-based e-KYC framework that integrates public key encryption with client consent management through smart contracts to address those limitations. This approach leverages attribute-based encryption (CP-ABE) to achieve granular access control for sensitive transactions stored on the blockchain, thereby enhancing data security and privacy while meeting regulatory compliance standards. Additionally, TrustBlock uses the Interplanetary File System (IPFS) for decentralized document storage, ensuring accessibility and data integrity. The core functionalities of e-KYC TrustBlock are supported by a combination of smart contracts and hybrid encryption protocols, including symmetric and public key encryption, to balance security with processing efficiency. Smart contracts automate key processes such as client registration, consent enforcement, and e-KYC verification, streamlining the workflow. In comparison to e-KYC frameworks proposed by Hanbar et al. [1], Kapsoulis et al. [7], Bhaskaran et al. [8], and Shabair et al. [9], Fugkeaw experimental results indicate that TrustBlock performs better in encryption, decryption, and policy update costs, as demonstrated through a series of simulations testing encryption/decryption times and policy update throughput. By utilizing a tree-based CP-ABE structure, TrustBlock reduces the computational complexity and improves the efficiency of access policy management. These improvements demonstrate the system's scalability and practicality for real-world applications in e-KYC processes, enabling financial institutions to perform secure and privacy-compliant identity verification. The overview of Fugkeaw method is shown in Figure 1.

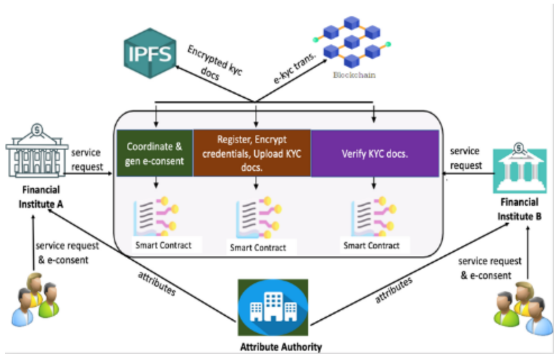

Fig. 1. Fugkeaw's TrustBlock method

Thus, it can be concluded that while e-KYC TrustBlock offers a significant advancement in addressing the privacy and efficiency limitations of traditional e-KYC systems, further work is required to assess its performance with larger datasets and to explore the integration of batch verification and searchable encryption capabilities. These enhancements would enable TrustBlock to accommodate high transaction volumes and provide a more robust solution for e-KYC in decentralized environments, ensuring scalability and enhanced data privacy.

III. Proposed Method

This section consists of four sub-sections, System Design discusses the end-to-end process of the implementation of blockchain-based searchable symmetric encryption; Pseudo-random Number Generation (PRNG) discusses the Turbulence Padded Chaotic Map (TPCM) works; Token Generation discusses how the token is generated on the SSE; Searching Encrypted Data is discussing how the encrypted data can search using the SSE.

A. System Design

This section provides a detailed explanation of each component within the blockchain built upon the Searchable Symmetric Encryption (SSE) algorithm. As shown in Figure 2, TrustBlock includes an additional blockchain layer specifically designed to support SSE functionalities.

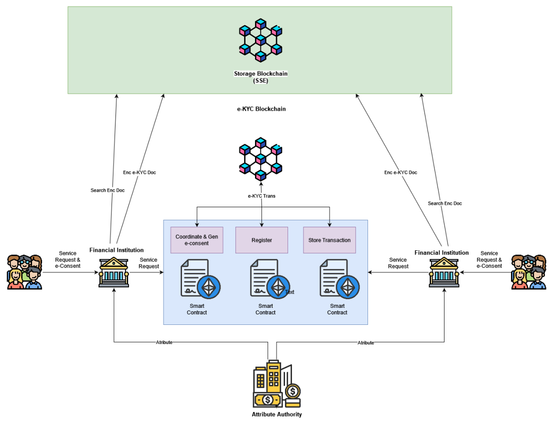

Fig. 2. Blockchain-based SSE on TrusBlock

In general, the system design of TrustBlock remains similar to Fugkeaw's models, but it introduces an additional blockchain layer specifically to facilitate encrypted document search. This enhancement allows for a more efficient retrieval of encrypted data, maintaining the security and integrity essential in e-KYC applications. By integrating this search functionality directly within the blockchain framework, TrustBlock addresses a the significant improvement compared with traditional e-KYC systems, which can not perform seamless, secure searches on encrypted data stored across decentralized networks.

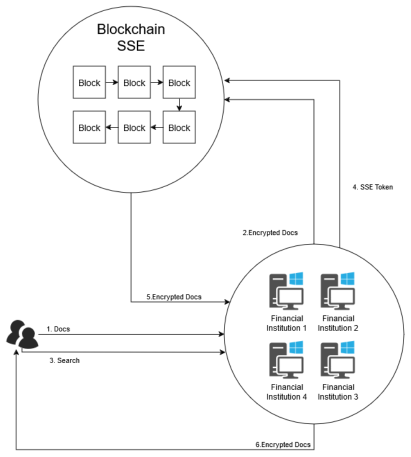

Fig. 3. SSE Blockchain Scenario

The Blockchain-based Searchable Symmetric Encryption (SSE) implemented in TrustBlock is designed to handle search computations while preserving the core decentralized nature of blockchain. Figure 3 shows this native integration of SSE enables TrustBlock to support secure, privacy-preserving searches without compromising the decentralized principles inherent to blockchain technology. With this approach, TrustBlock maintains the security and enhance the efficiency of data handling in e-KYC processes, as well as providing a scalable solution for secure document retrieval in a decentralized environment. This research presents an innovative improvement to the PRNG within the SSE framework by integrating turbulence-padded chaotic map. This approach harnesses the inherent unpredictability of chaotic systems to maintain the cryptographic strength of the SSE, improving search time without compromising the security aspect.

B. Pseudorandom Number Generation (PRNG)

This section explains the pseudorandom number generation (PRNG), the process that underpins the Searchable Symmetric Encryption (SSE) system. As illustrated in Figure 4, the PRNG used in this framework is the Turbulence Padded Chaotic Map, which combines two chaotic map algorithms: Bakers and Arnold. By merging these two chaotic algorithms, the PRNG achieves a high degree of unpredictability and randomness [10], crucial for secure and efficient encryption in SSE. This approach enhances the encryption of each token that been generated, ensuring the token has strong encryption by improving the Seed.

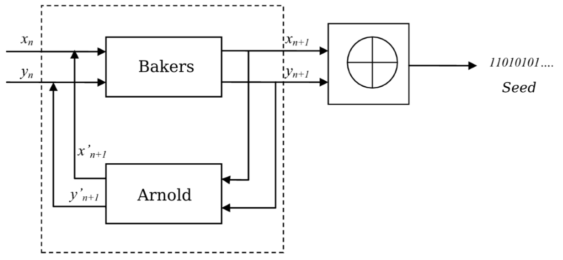

Fig. 4. Turbulence Padded Chaotic Map

The details of the pseudorandom number generation is presented in Algorithm 1. The detail of the Baker is represented

Algorithm 1 TPCM

```
    input :valX,valY,iteration
    output : s
 1: procedure TPCM
 2:     for $i \leftarrow 0$ to $iterations - 1$ do
 3:         $(BX, BY) \leftarrow$ BakerMapTransform$(valX, valY)$
 4:         $(AX, AY) \leftarrow$ ArnoldCatMap$(BX, BY)$
 5:         $valX \leftarrow AX$
 6:         $valY \leftarrow AY$
 7:     end for
 8:     for $i \leftarrow 0$ to length$(BX) - 1$ do
 9:         $bit \leftarrow$ evaluateXOR$(BX[i].x, BY[i].y)$
10:         Append $bit$ to $s$
11:     end for
12:     return $s$
13: end procedure
```

in Algorithm 2, and Arnold chaotic map in Algorithm 3. The input of those two chaotic map algorithms is a float number produced by the results of the preceding process. Algorithms 2 and 3 are repeated based on the setting of output length. In the proposed method the output is set of 16 bits in length, which needs 128 iterations.

Algorithm 2 BakerMapTransform

```
    input : x,y
    output : value
1: procedure BAKERMAPTRANSFORM$(x, y)$
2:     if $x < 0.5$ then
3:         $value \leftarrow 2 \cdot x, y/2$
4:         return value
5:     else
6:         $value \leftarrow 2 \cdot x - 1, y/2 + 0.5$
7:         return value
8:     end if
9: end procedure
```

Algorithm 3 ArnoldCatMap

```
    input : x,y
    output : x,y
1: procedure ARNOLDCATMAP$(x, y)$
2:     $newX \leftarrow$ mod$(x + y, 1)$
3:     $newY \leftarrow$ mod$(x + 2y, 1)$
4:     $x \leftarrow newX$
5:     $y \leftarrow newY$
6:     return $x, y$
7: end procedure
```

C. Token Generation

This section discusses how users perform token generation. Figure 5 shows the overview of token generation, which is part of uploading documents to decentralized storage.

The procedure to perform token generation is as follows:

1) The user encrypts the search word w using an encryption function E_k, producing Encrypted Word (X):
$$X = E_k(w). \tag{1}$$
2) The encrypted value X is split into two parts, the left part L and the right part R. The user generates Word-based key k_{wb} uses a pseudorandom function or cryptographic function of left part L and k, where k is symmetric key:
$$k_{wb} = HMAC(k, L). \tag{2}$$
3) Generating the Seed s using Turbulence Padded chaotic map function.
$$s = TPCM. \tag{3}$$
4) Generating the Fks using the pseudorandom function of s and k_{wb}.
$$Fks = HMAC(s, k_{wb}). \tag{4}$$
5) Generating left side of ciphertext C_1 by XOR-ing s and L
$$C_1 = s \oplus L \tag{5}$$
6) Generating right side of ciphertext C_2 by XOR-ing Fks with R
$$C_2 = Fks \oplus R \tag{6}$$

7) Generating the token (C) for each word by concatenating the C_1 and C_2

$$Token = C_1 || C_2 \tag{7}$$

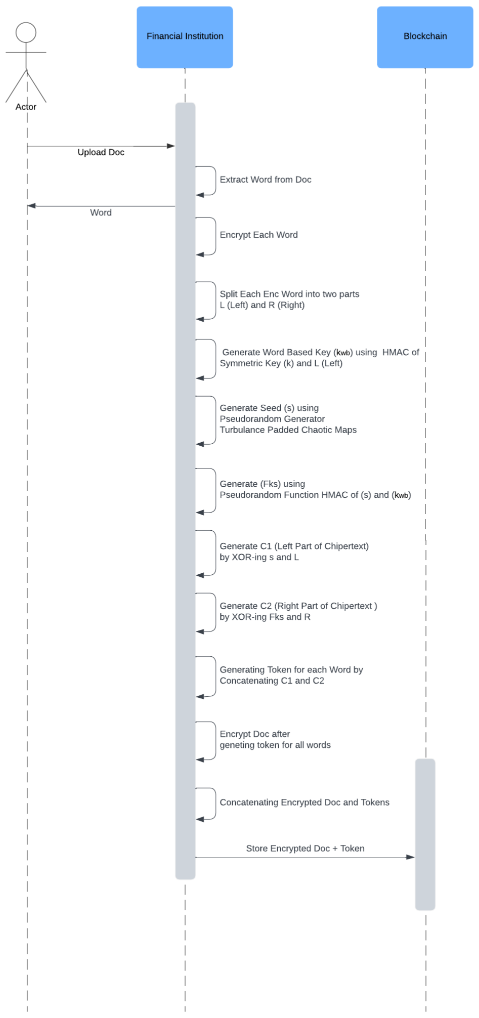
Actor
Financial Institution
Blockchain
Upload Doc
Extract Word from Doc
Word
Encrypt Each Word
Split Each Enc Word into two parts
L (Left) and R (Right)
Generate Word Based Key (kwb) using HMAC of
Symmetric Key (k) and L (Left)
Generate Seed (s) using
Pseudorandom Generator
Turbulance Padded Chaotic Maps
Generate (Fks) using
Pseudorandom Function HMAC of (s) and (kwb)
Generate C1 (Left Part of Chipertext)
by XOR-ing s and L
Generate C2 (Right Part of Chipertext)
by XOR-ing Fks and R
Generating Token for each Word by
Concatenating C1 and C2
Encrypt Doc after
geneting token for all words
Concatenating Encrypted Doc and Tokens
Store Encrypted Doc + Token

Fig. 5. Token Generation

D. Searching Encrypted Data

The data search process in the condition of the encrypted file can be achieved by carrying out the data upload process using SSE method, then continue with the search process, which applies the SSE method as well. As we saw in The following Figure 6 shows the search process is divided into two processes, namely generate searchToken, then process the proof token on each file.

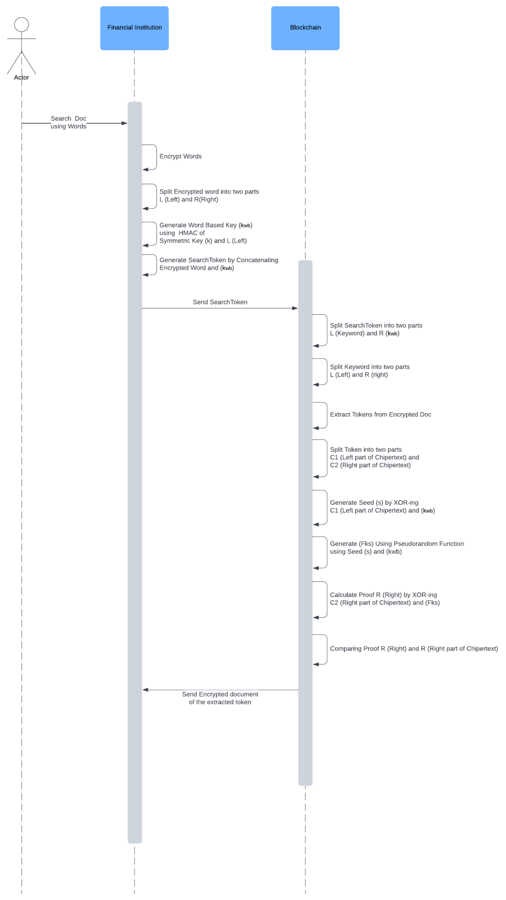
Actor
Financial Institution
Blockchain
Search Doc
using Words
Encrypt Words
Split Encrypted word into two parts
L (Left) and R(Right)
Generate Word Based Key (kwb)
using HMAC of
Symmetric Key (k) and L (Left)
Generate SearchToken by Concatenating
Encrypted Word and (kwb)
Send SearchToken
Split SearchToken into two parts
L (Keyword) and R (kwb)
Split Keyword into two parts
L (Left) and R (right)
Extract Tokens from Encrypted Doc
Split Token into two parts
C1 (Left part of Chipertext) and
C2 (Right part of Chipertext)
Generate Seed (s) by XOR-ing
C1 (Left part of Chipertext) and (kwb)
Generate (Fks) Using Pseudorandom Function
using Seed (s) and (kwb)
Calculate Proof R (Right) by XOR-ing
C2 (Right part of Chipertext) and (Fks)
Comparing Proof R (Right) and R (Right part of Chipertext)
Send Encrypted document
of the extracted token

Fig. 6. Searching Encrypted Data

Generate Search Token

For generating the search token, w given by the user has to be encrypted, which will become a token for searching to produce Encrypted Word (X):

$$X = E_k(w). \tag{8}$$

Furthermore, splitting X into two parts, L and R, uses L to generate again Word-based key k_{wb} using a pseudorandom function of left part L and k, where k is symmetric key:

$$k_{wb} = HMAC(k, L). \tag{9}$$

Then, concatenate the (X+k_{wb}) as $SearchToken$ and send it to decentralized storage.

Compute the Token Proof

The next part will processed on the server side :

1) Splitting $SearchToken$ into two parts:

$$SearchToken \rightarrow [X, k_{wb}],$$

2) Splitting X into two parts:

$$X \rightarrow [L, R],$$

3) Extracting the ciphertext token from the encrypted document and then for each ciphertext C in the encrypted documents
4) Splitting C into two parts:

$$C \rightarrow [C_1, C_2],$$

where C_1 is the left part and C_2 is the right part.
5) Computing the Seed s by XOR-ing $C1$ and k_{wb} :

$$s = C1 \oplus k_{wb}. \tag{10}$$

6) Computing the Fks by XOR-ing s and k_{wb}

$$Fks = s \oplus k_{wb}. \tag{11}$$

7) Computing proof R by XOR-ing $C2$ and Fks

$$ProofR = C_2 \oplus Fks. \tag{12}$$

8) Comparing proof R with R from splitting the X

$$ProofR == R. \tag{13}$$

When the proof is correct, then the files are found, and the server returns the corresponding encrypted document to the user.

IV. Experiment Evaluation

In this experiment, we will analyze the process of searching for data with a large dataset on decentralized storage. To be able to search for data, we first need to upload the data. There are subprocesses in the data search process, namely generating search tokens and proofing token processes. This process is done by performing a lookup in a directory containing encrypted files. Proofing will be carried out from each file on the tokens contained in it. In this experiment, the hardware specifications used are an eight-core CPU, 32 GB RAM, and Linux OS.

A. Performance Analysis

The experiment result of execution time can be seen in the Figure 7 using a data range of 1000 - 5000 files because it adopts batch verification, the return from each search can produce the same file because the same token when the search process is carried out can return several files at the same time.

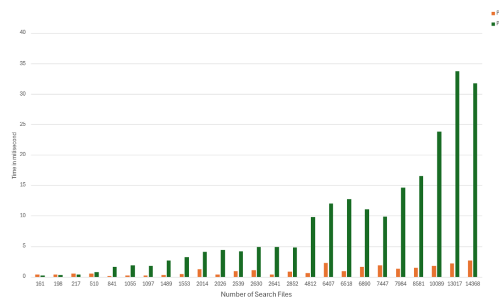

Fig. 7. Search Performancy

When comparing the two systems of the previous method and the proposed method, it is found that when the number of files searched is still in the hundreds range, the performance of the previous method is better than the proposed method. For example, when the number of files is around 161 to 217, the search time with the previous ranges from 0.19 to 0.34 seconds, slightly faster than the proposed method in the range of 0.34 to 0.51 seconds. The proposed method begins to show a much better search time performance advantage in the range of thousands of files, where the proposed method's search time is relatively stable and increases very slowly. At around 2,000 files, the proposed method takes between 0.39 to 1.2 seconds, while the previous method can take 4 to 5 seconds or more. This difference becomes more expansive when the number of files increases to tens of thousands. The proposed method can still keep the search time below three seconds, while the previous method can jump to tens of seconds.

B. Security Analysis

The security testing process of the resulting seed uses the entropy calculation method, assuming the previous method uses time-based-PRNG while the proposed method uses TPCM (Turbulence Padded Chaotic Map).

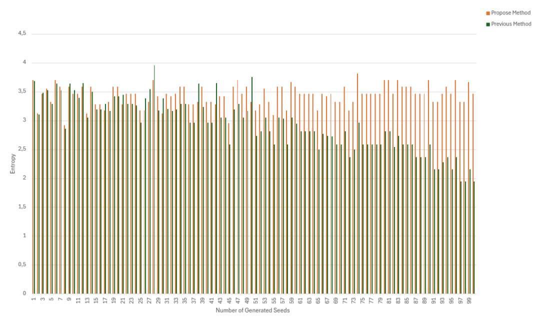

Fig. 8. Entropy comparison

From the data in Figure 8, the proposed method's entropy value tends to be 2.9 to 3.8. Based on Figure 7, the proposed method can produce a relatively stable and consistent level of entropy. Although there are some gaps between the proposed and previous methods' entropy, but they are insignificant. The entropy value of the previous method shows a wider variation. The resulting entropy value of the previous method can reach almost 3.9 under specific conditions but can also drop drastically to a range below 2.0. This extensive range indicates that the previous method has a higher sensitivity to time conditions or external factors that affect the random number generation process. so it does not always guarantee high entropy consistency.

V. Conclusion

This research aims to overcome the problem of Fugkeaw's method, which is a time-consuming and inefficient file-searching process. Searchable Symmetric Encryption can overcome the problem of the existing method, which is to search data in large datasets on a blockchain network while maintaining the security aspect of the seed generation using Turbulence Padded Chaotic Map.

Based on the experiment, the proposed method's searching time is proven to be less than Fugkeaw's method searching time because Fugkeaw's method compared the hash of each file in the file searching process. From the point of view of the security aspect, the proposed method has high entropy. Meanwhile, Fugkeaw's method has less entropy than the proposed method. Thus, the security of the proposed method is better than that of Fugkeaw's method.

REFERENCES

[1] H. Hanbar, V. Shukla, C. Modi, and C. Vyjayanthi, "Optimizing e-kyc process using distributed ledger technology and smart contracts," pp. 132–145, 2020.

[2] S. Fugkeaw, "Enabling trust and privacy-preserving e-kyc system using blockchain," *IEEE Access*, vol. 10, pp. 49028–49039, 2022.

[3] T. A. Mohammed and A. B. Mohammed, "Security architectures for sensitive data in cloud computing," in *Proceedings of the 6th International Conference on Engineering MIS 2020*, pp. 1–6, ACM, 9 2020.

[4] K. Borg, N. Supervisor, and C. Li, "Searchable symmetric encryption and its applications," 2022.

[5] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," pp. 44–55, IEEE Comput. Soc.

[6] C. Xu, L. Yu, L. Zhu, and C. Zhang, "Blockchain-based verifiable dsse with forward security in multi-server environments," pp. 163–171, 2021.

[7] N. Kapsoulis, A. Psychas, G. Palaiokrassas, A. Marinakis, A. Litke, and T. Varvarigou, "Know your customer (kyc) implementation with smart contracts on a privacy-oriented decentralized architecture," *Future Internet*, vol. 12, p. 41, 2 2020.

[8] K. Bhaskaran, P. Ilfrich, D. Liffman, C. Vecchiola, P. Jayachandran, A. Kumar, F. Lim, K. Nandakumar, Z. Qin, V. Ramakrishna, E. G. Teo, and C. H. Suen, "Double-blind consent-driven data sharing on blockchain," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 385–391, IEEE, 4 2018.

[9] W. M. Shbair, M. Steichen, J. Francois, and R. State, "Blockchain orchestration and experimentation framework: A case study of kyc," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–6, IEEE, 4 2018.

[10] S. Krishnamoorthi, P. Jayapaul, R. K. Dhanaraj, V. Rajasekar, B. Balusamy, and S. H. Islam, "Design of pseudo-random number generator from turbulence padded chaotic map," *Nonlinear Dynamics*, vol. 104, pp. 1627–1643, 4 2021.