JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 18, No. 3, September 2024, pp. 144-151

A Robust Decryption Technique Using Letter Frequency Analysis for Short Monoalphabetic Substitution Ciphers

Dayeong Kang and Jiyeon Lee*
School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea
haru34@knu.ac.kr, jiyeon@knu.ac.kr

Abstract
Substitution ciphers—where characters are systematically replaced with different ones—have represented a vital tool in safeguarding sensitive information for centuries, eventually laying the foundation for modern encryption techniques. Despite their large key space, substitution ciphers are susceptible to frequency analysis that leverages common English letter patterns. While existing research has suggested certain methods that can improve decryption accuracy using n-gram frequencies, these methods face difficulties when used with short ciphertexts due to incomplete letter distribution representation. The present study examines the limitations of current frequency analysis in decrypting short ciphertexts, with the results revealing that deterministic bigram approaches can reduce accuracy in certain cases. To address this shortcoming, we introduce a novel algorithm that uses randomized index selection based on letter distribution to generate multiple candidate keys. We also present a word-level key guessing method using these candidates that maps prominent English words to uncover a secret key. The results of tests with 200 ciphertexts of varying lengths showed an average decryption accuracy of 84.1% for 200-character ciphertexts, an improvement of 147.1% over existing methods. In experiments without dictionary-based decryption, an accuracy of 77.6% was achieved with a decryption time of approximately 0.27 seconds, which is a reasonable completion time. Altogether, these results highlight the efficiency and practicality of our approach for decrypting short ciphertexts.

Category: Ubiquitous computing

Keywords: Cryptography; Cryptoanalysis; Monoalphabetic cipher; Letter frequency analysis

I. INTRODUCTION

Classical ciphers are a foundational aspect of cryptography, and they serve as the bases for various encryption methods that have been used for centuries to secure sensitive information. These ciphers, which are rooted in ancient cryptographic techniques, have paved the way for the development of modern cryptographic algorithms and protocols. The allure of classical ciphers lies not only in their historical significance but also in their inherent simplicity. They are commonly categorized into two groups based on their operational methods: transposition ciphers and substitution ciphers.

In transposition ciphers, the letters of the plaintext are rearranged, while in substitution ciphers, the plaintext letters are replaced with other letters, numbers, or symbols [1]. The security of substitution ciphers relies on the size of the key space, which, for a simple substitution

Open Access http://dx.doi.org/10.5626/JCSE.2024.18.3.144 http://jcse.kiise.org

Received 25 July 2024; **Accepted** 6 September 2024
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

cipher, corresponds to the number of possible permutations of the alphabet. Despite the use of a large key space, substitution ciphers are particularly vulnerable to letter frequency analysis [2]. This cryptanalytic technique uses the frequency of letters in the ciphertext to deduce the original plaintext. For example, certain letters appear more frequently than others in certain languages, such as the letter 'E' in English. This method can be used to effectively break substitution ciphers by matching the frequency patterns of the ciphertext with those of the plaintext.

Recognizing the effectiveness of this technique, many studies have proposed various algorithms that can be used to strengthen such ciphers against this technique [3-9]. Several studies [7-9] have proposed a decryption technique that considers the frequency of letter pairs (bigrams) to overcome the limitations of single-letter frequency analysis, which can be inaccurate due to factors such as the writing style of the ciphertext. While these techniques lead to significant performance improvements, they still face limitations in decryption accuracy when used with short ciphertexts due to the insufficient representation of letter distributions.

The rest of this paper aims to explore the limitations of existing letter frequency analyses in decrypting short ciphertexts. Through a comprehensive investigation, we have confirmed that the algorithm applying bigram distribution that has been used in previous studies does not achieve high decryption accuracy when used with short ciphertexts. This paper then examines deterministic bigram combinations, which are shown to lead to a decrease in decryption accuracy in some ciphertexts, even after numerous iterations. To address this issue, we propose a novel algorithm that selects random indices based on letter distribution to extract multiple candidate keys. We also suggest a word-level key guessing approach based on these candidate keys. Our algorithm traverses the ciphertext in word token units, mapping the highest-ranking English words to ultimately uncover the secret key.

To evaluate the effectiveness of our proposed method, we implemented a prototype using a dictionary API and measured its decryption accuracy and performance overhead. To this end, we tested 200 ciphertexts of lengths varying from 100 to 400 characters. The results showed that the average decryption accuracy achieved for a 200-character ciphertext was 84.1%, representing a 147.1% improvement when compared to an existing decryption method [7]. The execution overhead for decrypting a 200-character ciphertext took 3.9 minutes longer, which can be primarily attributed to the dictionary search. Omitting the dictionary lookup resulted in only a minor loss in accuracy (approximately 6.53%) while allowing for decryption within a very reasonable execution time (0.27 seconds). Altogether, these results demonstrate that our proposed method is highly effective and practical for decrypting short ciphertexts.

II. BACKGROUND

A. Classical Ciphers

Classical categories are based on their methods of operation: transposition ciphers and substitution ciphers. Transposition ciphers operate by altering the arrangement of characters within a plaintext message to obscure its original meaning. Meanwhile, substitution ciphers involve replacing each character in the plaintext with another character or symbol according to a predetermined table or key. One of the earliest known substitution ciphers is the Caesar cipher [10], which shifts each character in the plaintext by a fixed number of positions along the alphabet. While Caesar ciphers are straightforward to implement, they are less secure as they are susceptible to brute force attacks, which involve trying every possible key, especially since there are only 25 potential shifts for alphabetic characters. A monoalphabetic substitution cipher entails replacing each letter in the plaintext with another letter according to a predetermined mapping [11]. Fig. 1 illustrates this encryption process using a key derived from a randomly shuffled alphabet. In this instance, characters "A" and "B" are substituted with "S" and "A," respectively, resulting in the formation of the ciphertext. Utilizing this approach, the key space for alphabetic characters yields 26 factorial (over 400 sextillion), meaning it is resistant to brute force attacks within a polynomial time. The monoalphabetic substitution method also offers the additional advantages of simplicity and efficiency in both encryption and decryption procedures.

	ABCDEFGHIJKLMNOPQRSTUVWXYZ
KEY:	SAHVPBJWUNRXTDMYKEOZIFQLGC
Plaintext:	Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversations?"
Ciphertext:	Sxuhp qso apjuddudj zm jpz fpeg zuepv mb ouzzudj ag wpe ouozpe md zwp asdr, sdv mb wsfudj dmzwudj zm vm: mdhp me zquhp owp wsv yppypv udzm zwp ammr wpe ouozpe qso epsvudj, aiz uz wsv dm yuhziepo me hmdfpeoszumdo ud uz, "sdv qwsz uo zwp iop mb s ammr," zwmijwz Sxuhp "quzwmiz yuhziepo me hmdfpeoszumdo?"

Fig. 1. Example of a monoalphabetic substitution cipher.

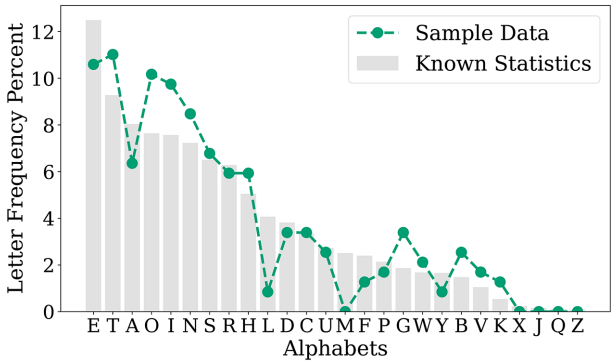

Fig. 2. Letter frequency comparison between known statistics and sample data.

B. Letter Frequency Analysis

Despite the vastness of the key space, monoalphabetic substitution ciphers are susceptible to letter frequency analysis [2]. This technique relies on the observation that certain letters occur more frequently than others in natural language texts, with certain notable examples occurring in English. Within the context of decryption, letter frequency analysis involves counting the occurrence of each letter in the ciphertext and comparing it to the expected frequency distribution of letters in the plaintext language. The most commonly used letter in English is the letter "E," followed in order by "T" and "A." The grey bar in Fig. 2 presents the frequency of each letter in the entire alphabet. By examining the frequency distribution of letters in the ciphertext, cryptanalysts can infer plausible correspondences between letters in the ciphertext and those in the plaintext.

While letter frequency analysis may seem robust, it encounters a significant limitation in practice. Although certain letters like "E," "T," "A," and "O" are indeed more common in English text, their frequency distribution can vary based on factors such as the text's genre and writing style. Moreover, some characters may not appear in some short texts, resulting in a less predictable frequency distribution. The green plot in Fig. 2 demonstrates that the letter frequencies between established statistics and those observed in the example text shown in Fig. 1 do not match perfectly. This discrepancy poses challenges in accurately deducing the substitution key based solely on letter frequencies.

III. RELATED WORK

Early studies have highlighted that distributions of *n*-grams (sequences of *n* letters) are effective tools for decrypting monoalphabetic ciphers [3-9, 11, 12]. For example, Jakobsen [7] demonstrated that leveraging the distribution of bigrams (two-letter sequences) enhances decryption accuracy for monoalphabetic ciphers. Uddin and Youssef [8] also utilized bigram distributions along with particle swarm optimization. This approach adjusts potential solutions based on individual and collective experiences, while using unigram and bigram statistics in its cost function. However, it exhibits low decryption performance on ciphertexts that are shorter than 300 characters. Vobbilisetty et al. [9] employed hidden Markov models (HMMs) for cryptanalysis, modeling the sequences of observed ciphertexts and determining transition probabilities between hidden states based on bigram statistics derived from training data. This approach achieved nearly 70% decryption accuracy on 200-character texts; however, it required about 100,000 HMM training restarts, resulting in significant overhead.

In summary, while these methods aid in quickly uncovering secret keys, their effectiveness in decrypting short ciphertexts remains a subject of ongoing investigation. Our present study distinguishes itself from prior research by addressing the inherent difficulties involved in analyzing short encrypted messages and enhancing the precision of decryption outcomes.

IV. ANALYSIS OF BIGRAM STRATEGIES

In this section, we introduce an algorithm that utilizes bigram statistics to decrypt substitution ciphers and explore its limitations. Jakobsen [7] initially proposed a frequency analysis method based on two-letter combinations in an attempt to overcome the challenges associated with single-letter frequency analysis in substitution cipher decryption. This method begins by taking ciphertext as input, then counting the frequency of individual characters, and lastly creating an initial key using the established frequency of English letters, as illustrated in Fig. 2. It then determines whether to update the key using the following equation for the initial key:

$$f(t) = \sum_{i,j} |D_{ij}(t) - E_{ij}| \quad (1)$$

where $D(t)$ and E are 26 × 26 matrices representing the bigram frequencies of a given text t and the common bigram frequencies of English letters, respectively. The key updating procedure described in [7] iteratively compares these frequencies by traversing the matrix indices and updating the key if $f(t)$ decreases. In other words, if the bigram frequency of a tested combination is found to be closer to the established frequency (thus resulting in a lower $f(t)$), the key is swapped. This process is repeated a specified number of times (e.g., 3000 times) until $f(t)$ converges.

Since $f(t)$ captures bigram frequency in a given ciphertext, it is generally inversely correlated with decryption accuracy. However, our experiment indicates that short ciphertexts still struggle to provide an adequate

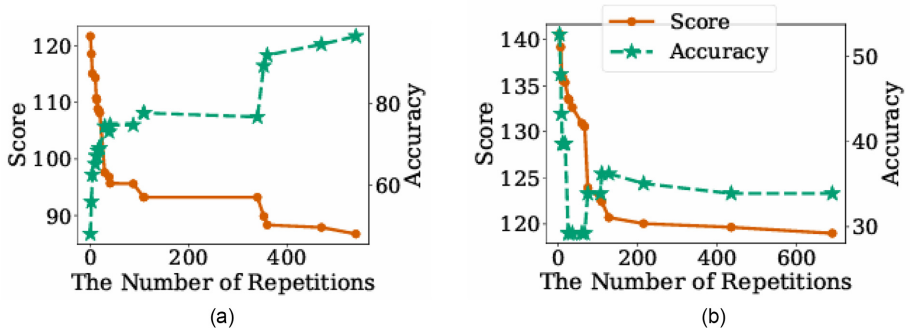

Fig. 3. *f*(*t*) score and accuracy percentage for ciphertexts of different lengths: (a) 302-character ciphertext *t* and (b) 171-character ciphertext *t*.

distribution of English letter frequencies, making it difficult for *f*(*t*) to capture them accurately. Fig. 3 shows the relationship between changes in *f*(*t*) and decryption accuracy with increasing iterations for ciphertexts comprised of 171 and 302 characters, respectively. For the 302-character ciphertext (refer to Fig. 3(a)), the result shows the expected symmetrical representation of *f*(*t*) and decryption accuracy. By contrast, for the 171-character ciphertext (refer to Fig. 3(b)), *f*(*t*) decreases with more iterations, while there is no increase in decryption accuracy. This trend is often observed in texts that are shorter than 250 characters, thus emphasizing that a low *f*(*t*) does not always correlate with high decryption accuracy for short ciphertexts. This ultimately suggests the need for a new letter frequency analysis technique for short ciphertexts.

V. OUR APPROACH

Inspired by the trends discussed in Section IV, our research aims to develop a new method that ensures high decryption accuracy for short ciphertexts. To achieve this, we assume the presence of a known cipher attacker who has obtained arbitrary ciphertext and is attempting to discover the secret key without having any time limitations. We also assume that the secret key is updated infrequently, making the proposed method applicable to multiple ciphertexts. Focusing on improving decryption accuracy for short ciphertexts presents distinct challenges that necessitate the use of alternative strategies to ensure effective decryption. Our objective is to enhance the robustness and reliability of decryption techniques under these specific conditions.

The core of the proposed method lies in generating multiple candidate keys to decrypt the cipher with reference to English words. Fig. 4 depicts the entire process of this approach. First, the frequency of individual letters in the input ciphertext is analyzed to establish an initial key, based on which the proposed bigram cryptanalysis algorithm generates 10 candidate keys (V-A). Next, each word token in the ciphertext is decrypted with the candidate keys to select the key combination that most closely resembles English words (V-B). The following sections provide detailed explanations of each phase.

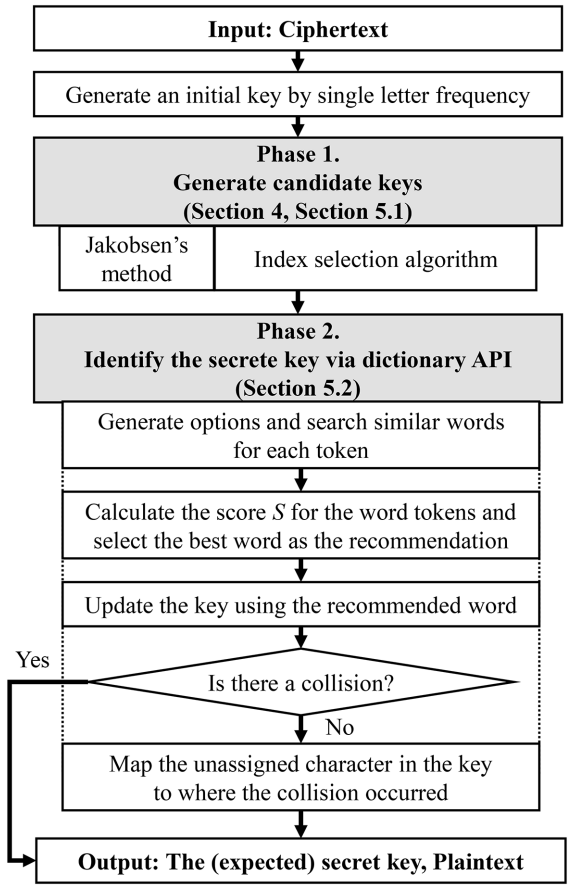

Fig. 4. Overview of the proposed decryption method.

A. Phase 1: Generating Candidate Keys

In this section, we present a method for generating

candidate keys through bigram analysis. As discussed in Section IV, traditional bigram analysis methods often have certain limitations in optimization, as they only examine fixed character combinations, even when iterated indefinitely. To overcome this drawback, we propose a new index selection algorithm for updating an initial key. This algorithm randomly selects indices from a list that is weighted by single-letter frequencies under the assumption that characters with higher frequencies in single-letter analysis will also be prevalent in bigrams. The algorithm operates as follows:

1. First, an initial key is generated using single-letter frequency analysis. Each character is listed multiple times based on its occurrence frequency and then assigned an index in ascending order.
2. Two indices are randomly chosen from a discrete uniform distribution to form a bigram combination.
3. If the $f(t)$ from Eq. (1) improves with this new character combination, the initial key is updated.

Through multiple experiments, we have observed that iterating the above process more than around 3,000 times tends to converge $f(t)$. Therefore, we repeated the procedure 3,000 times, ultimately resulting in the creation of one candidate key.

We note that the proposed algorithm produces non-deterministic outcomes due to the randomness in index selection, which results in varied candidate keys. Fig. 5 shows the average decryption accuracy of n candidate keys (n = 10, 20, 30, 40, and 50) generated by both [7] and our algorithm for a given ciphertext of length 171. Compared to Jakobsen's method, which consistently shows the same decryption accuracy, our algorithm's results exhibit a larger standard deviation. Notably, the average decryption accuracy does not significantly change with an increasing number of keys. Therefore, we generate ten candidate keys, including the one from [7], as it demonstrates stable performance on longer texts.

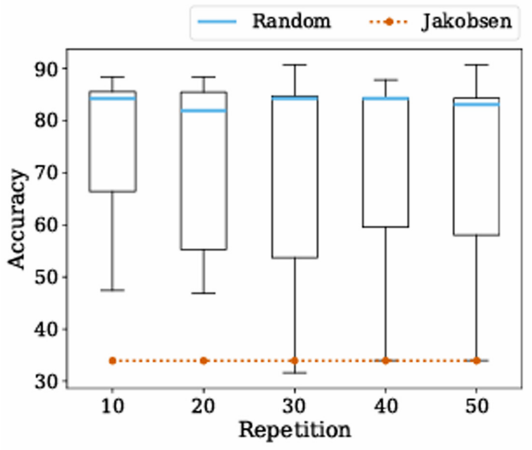

Fig. 5. Accuracy percentage comparison between Jakobsen's method and the proposed index selection algorithm.

B. Phase 2: Finding Secret Key via Dictionary API

Once the candidate keys are generated, we use them to identify the secret key that the attacker is seeking. First, we replace all non-alphabetic characters (e.g., quotation marks) in the ciphertext with spaces (‘ ’), then we split the ciphertext using spaces to create word tokens (T_c). Next, we decrypt each token using the candidate keys to produce options, which are later then ranked collectively to determine the secret key. The number of options may be fewer than 10, since some candidate keys might produce the same result for a given word token. We use the dictionary API [13] to map paronyms (words that are pronounced or written in a similar way) to each option and identify the best word among the recommended words.

To achieve this, we referenced the ranking method from information retrieval theory [14] to devise a formula. Fig. 6 illustrates the process that we used to select the optimal word for a given token. The value S, which represents the recommendation score for each word, is calculated as follows:

$$S = count \times (1 + \log_{10}(reputation)) \times length. \quad (2)$$

Here, *count* represents the number of times the word is found; if a particular word appears multiple times across several options, it is given more weight. The reputation

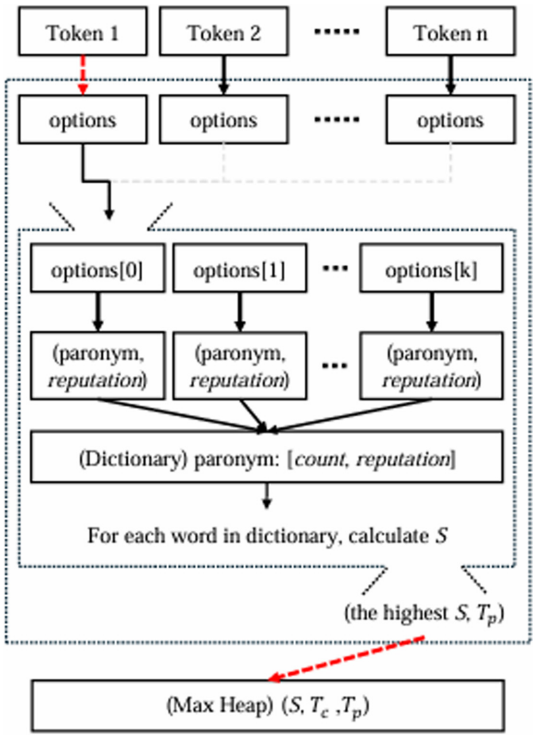

Fig. 6. Process of selecting the optimal word T_p based on options for a given token T_c.

indicates the popularity of the word as provided by the dictionary API. The length of the word is also considered, as longer words tend to provide more specific results, thus giving them higher weight [15]. Finally, the word with the highest S is designated as T_p.

The S values are compared with those of other tokens to prioritize key updates. To manage this process efficiently, we store tuples of (S, T_c, and T_p) in a max heap. We also create an empty list to store the resulting key and update it by sequentially extracting tokens from the max heap. If the length of the token being examined is less than 4, we initially postpone updating the key by adding it to the end of the queue, as immediate updates might lead to collisions in subsequent operations. For other cases, we convert T_c to T_p and update the key if there is no conflict with the key that has been thus far determined. If a key conflict occurs, we leave the conflicting characters as unknown values and attempt additional word searches for the token. Finally, the characters are converted into a new recommended word composed of non-conflicting characters.

There may still be unresolved characters after finishing this procedure. In such cases, we complete the procedure by arbitrarily assigning unused letters in the key, thus ensuring that there are no conflicts with the key. For example, if there are no corresponding keys for A, B, and C in the ciphertext and the unused letters in the key are x, y, and z, we sequentially map A to x, B to y, and C to z. Finally, the completed secret key and the plaintext decrypted with this key are printed, and the program is terminated.

VI. EVALUATION

To assess the effectiveness of the proposed cryptanalysis technique, we employ two evaluation metrics: decryption accuracy and performance overhead. We implemented both Jakobsen's method and our approach using Python, NLTK [16], and NumPy [17], and we conducted a comparative analysis of performance. For the experiments, we randomly selected 50 sentences of varying lengths (100, 200, 300, and 400 characters) from the book, *Alice's Adventures in Wonderland* [18]. We generated secret keys using a random function to encrypt these sentences, ultimately resulting in a total of 200 ciphertexts used in our experiments. We decrypted these ciphertexts using keys that were obtained from both techniques and evaluated the similarity between the deciphered text and the plaintext by calculating accuracy, which was defined here as the number of characters matching the plaintext after decryption divided by the total length of the string. We also measured the execution time of both techniques to assess the performance overhead of the decryption process. All experiments were conducted on a machine equipped with a 3.20 GHz CPU (Intel Core i9-14900K) and 32 GB RAM.

A. Decryption Accuracy

Fig. 7 shows the decryption accuracy of both methods across ciphertext lengths ranging from 100 to 400 characters. Each plot represents the average accuracy calculated from 50 ciphertexts. As shown in Fig. 7, our approach (labeled as OURS) achieves decryption accuracies of 54.6% and 84.1% for ciphertext lengths of 100 and 200 characters, respectively. This surpasses the performance of Jakobsen's method (labeled as Jakobsen) by up to 26.9%. We note that, in Phase 2 (as described in Section V-B), it is possible that human intervention can improve decryption accuracy and speed, depending on the decryptor's language proficiency. However, due to variability in human language skills, our evaluation focuses on the automated version, which demonstrates sufficiently high accuracy for plaintext interpretation.

For a more comprehensive understanding, we assessed the decryption accuracy at each phase of our method. Fig. 7 presents OURS-Phase1-only, thus showing the decryption accuracy achieved using the key with the lowest $f(x)$ after Phase 1. Remarkably, executing Phase 1 alone results in a significant performance improvement when compared to Jakobsen's method. This highlights that examining a broader range of bigram combinations can effectively boost decryption accuracy. Further, applying Phase 2 (OURS) yielded an increase of accuracy up to 6.5% when compared to OURS-Phase1-only for ciphertexts of length 200. By contrast, incorporating Phase 2 in Jakobsen's method (labeled as Jakobsen-with-Phase2) led to a marginal improvement (up to a 1.6% increase) when compared to Jakobsen. These results suggest that leveraging multiple candidate keys in Phase 2 significantly influences decryption performance.

B. Performance Overhead

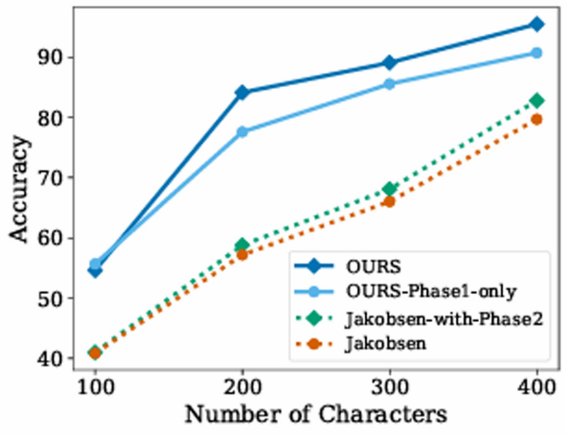

Fig. 7. Comparison of accuracy percentages between our proposed method and others.

Table 1. Performance overhead described in seconds

Method	Number of characters			
	100	200	300	400
Jakobsen	0.03	0.03	0.03	0.03
OURS-Phase1-only	0.27	0.27	0.29	0.28
OURS	147.10	235.63	291.60	306.72

We now discuss the performance overhead of the decryption process. Table 1 presents the comparison of execution times between the two techniques. As can be seen in the results, the most significant overhead occurs during Phase 2. Decrypting a ciphertext of length 100 took approximately 2.45 minutes, while the time needed to decrypt a ciphertext of length 400 increased to a maximum of 5.11 minutes. This is primarily attributable to the time spent querying the dictionary API; decrypting more tokens or frequently re-queuing tokens results in longer processing times. The results of execution time for Phase 1 (referred to as OURS-Phase1-only) show that our index selection algorithm takes about 10 times longer than Jakobsen's method due to the time taken in generating multiple candidate keys. Nevertheless, it maintains a reasonable execution time of approximately 0.28 seconds for the longest ciphertext length. Therefore, attackers who are constrained by decryption time can selectively choose from different attack procedures to mitigate performance overhead.

VII. CONCLUSION

In this paper, we focused on substitution ciphers and proposed an effective decryption method that is tailored to short sentences, thus addressing the limitations of traditional letter frequency analysis methods. Through empirical experiments, we found that existing approaches diminish decryption performance by considering deterministic letter combinations when analyzing bigram frequencies. To overcome this challenge, we have introduced a new index selection algorithm and proposed a novel dictionary-based decryption technique. We generated multiple candidate keys through bigram analysis and selected the most prominent word from these candidates to decrypt the text. Our proposed method ultimately achieved a decryption accuracy of 84.1% for short ciphertexts of length 200, representing a 147.1% improvement over existing methods. The decrypted text obtained from our method is deemed to be sufficiently interpretable by human cognition, thus validating its efficiency for use with short ciphertexts.

CONFLICT OF INTEREST

The authors have declared that no competing interests exist.

ACKNOWLEDGMENTS

This research was supported by Kyungpook National University Research Fund, 2023.

REFERENCES

1. C. P. Pfleeger, S. L. Pfleeger, and J. Margulies, *Security in Computing*, 5th ed. Upper Saddle River, NJ: Prentice-Hall, 2015.
2. I. A. Al-Kadit, "Origins of cryptology: the Arab contributions," *Cryptologia*, vol. 16, no. 2, pp. 97-126, 1992. https://doi.org/10.1080/0161-119291866801
3. B. Hauer, R. Hayward, and G. Kondrak, "Solving substitution ciphers with combined language models," in *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING)*, Dublin, Ireland, 2014, pp. 2314-2325.
4. B. Hauer and G. Kondrak, "Decoding anagrammed texts written in an unknown language and script," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 75-86, 2016. https://doi.org/10.1162/tacl_a_00084
5. M. H. Ahmed, A. K. Shibeeb, and A. H. Mohammed, "Solve polyalphabetic cipher based on intelligent system," in *Proceedings of 2021 International Conference on Communication & Information Technology (ICICT)*, Basrah, Iraq, 2021, pp. 290-296. https://doi.org/10.1109/ICICT52195. 2021.9568455
6. A. Dhavare, R. M. Low, and M. Stamp, "Efficient cryptanalysis of homophonic substitution ciphers," *Cryptologia*, vol. 37, no. 3, pp. 250-281, 2013. https://doi.org/10.1080/01611194.2013.797041
7. T. Jakobsen, "A fast method for cryptanalysis of substitution ciphers," *Cryptologia*, vol. 19, no. 3, pp. 265-274, 1995. https://doi.org/10.1080/0161-119591883944
8. M. F. Uddin and A. M. Youssef, "Cryptanalysis of simple substitution ciphers using particle swarm optimization," in *Proceedings of 2006 IEEE International Conference on Evolutionary Computation*, Vancouver, Canada, 2006, pp. 677-680. https://doi.org/10.1109/CEC.2006.1688376
9. R. Vobbilisetty, F. Di Troia, R. M. Low, C. A. Visaggio, and M. Stamp, "Classic cryptanalysis using hidden Markov models," *Cryptologia*, vol. 41, no. 1, pp. 1-28, 2017. https://doi.org/10.1080/01611194.2015.1126660
10. B. B. Mohammed, "Automatic key generation of Caesar Cipher," *International Journal of Engineering Trends and Technology*, vol. 6, no. 6, pp. 2231-5381, 2013.
11. R. Hilton, "Automated cryptanalysis of monoalphabetic substitution ciphers using stochastic optimization algorithms," Ph.D. dissertation, Department of Computer Science and Engineering, University of Colorado, Colorado Springs, CO, USA, 2012.

12. E. Olson, "Robust dictionary attack of short simple substitution ciphers," *Cryptologia*, vol. 31, no. 4, pp. 332-342, 2007. https://doi.org/10.1080/01611190701272369
13. Datamuse API: a word-finding query engine for developers Online]. Available: https://www.datamuse.com/api/.
14. C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008.
15. P. Norvig, "English letter frequency counts," 2012 [Online]. Available: http://norvig.com/mayzner.html.
16. NLTK Project, "NLTK: natural language toolkit," c2024 [Online]. Available: https://www.nltk.org/.
17. NumPy team, "NumPy: the fundamental package for scientific computing with python," c2024 [Online]. Available: https://numpy.org/.
18. L. Carroll, "Alice's adventures in wonderland," 2008 [Online]. Available: https://www.gutenberg.org/ebooks/11.

Dayeong Kang https://orcid.org/0009-0003-8127-8345

Dayeong Kang received Bachelor's degree in Computer Science and Engineering at Kyungpook National University, Daegu, South Korea, in 2024. Her research interests focus on security, cloud environment, and quantum computing.

Jiyeon Lee https://orcid.org/0000-0003-1005-0637

Jiyeon Lee received her Ph.D. in School of Computing from KAIST, Daejeon, South Korea, in 2021. From 2021 to 2023, she was a postdoctoral researcher in School of AI Convergence at Soongsil University, Seoul, South Korea. Since 2023, she has been a faculty member in the Department of Computer Science and Engineering at Kyungpook National University, Daegu, South Korea. Her research interests focus on information security and side-channel attacks in Internet and web environments.