Received February 18, 2019, accepted March 16, 2019, date of publication April 17, 2019, date of current version May 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2907530

Data Hiding for Ensuring the Quality of the Host Image and the Security of the Message

HUAIBO SUN, HONG LUO, AND YAN SUN
Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Huaibo Sun (sunhuaibo@ bupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772085, Grant 61877005, and Grant 61672109.

ABSTRACT Concealing a message in the APPn markers of a JPEG image not only protects the security of the message but also induces no change to the quality of the image. However, from the existing literature, whether it is a plaintext message or a ciphertext message hidden in APPn, it is easy for an attacker to identify the confidentiality of the hidden message, which is not conducive to the security of the message. Inspired by the natural language processing (NLP) and format-preserving encryption (FPE), this paper proposes a data hiding method, which is focused on the quality assurance of the host image and the concealment of the plain text having complete semantics based on the NLP and FPE. This method first uses the NLP and FPE to identify and encrypt the sensitive words in the plain text and then hides the ciphertext text with the plaintext style in the APPn of a JPEG image after replacing the plaintext words with the ciphertext words that have the plaintext style. The experimental results confirm that the structure, size, and quality of the host image do not show any changes before or after the data hiding and the recovered host image is also identical to its original appearance. In addition, the strategy that the semantic similarity is regulated autonomously by the user also makes it possible to obtain the ideal ciphertext words with very low similarity. Moreover, more than 80% of the ciphertext texts have reasonable semantics. Compared with the existing literature, our algorithm has a better performance.

INDEX TERMS APPn, common text, format-preserving encryption, natural language processing, quality of host image.

I. INTRODUCTION

It is well known that data hiding is one of the effective measures used to protect image data or other messages. An image is one kind of the important carriers for data hiding, according to many algorithms that have been proposed in recent decades; for example, algorithms that hide messages in the spatial domain or transform domain of the host image. However, we find that the quality of the host image can be reduced, and the structure of the image can be changed regardless of the algorithm, and the host image also has difficulty returning to its initial state after the secret information is extracted [1]–[5]. This causes serious damage to the host image, especially a medical carrier image. Therefore, in order to ensure the visual quality and structure of the host image and to protect the restored host image from being damaged, the message has to be hidden in a domain that is outside of the spatial or transform domain of the host image.

Hiding a message in the APPn of a JPEG image can meet the requirements mentioned above, and it is also very easy to find the right JPEG host images. From the existing literature [6], the JPEG standard has defined many markers for the application, such as the markers for SOI, DQT, EOI fields, as well as the markers APP0 (JFIF application data block) and APPn (other application data blocks, $n = 1 - 15$). Where APPn is the reserved marker of application, it can be used to store some data information, and the content of data information is uncertain. For example, APPn can be used to store a copy of an image, or the metadata of an image. In addition, APPn is a non-essential marker when decoding JPEG images. At the same time, JPEG is also an image compression format that is widely used in imaging devices, such as digital cameras and smart phones. Furthermore, these advantages of JPEG images have already sparked interest among researchers [7]–[11].

The associate editor coordinating the review of this manuscript and approving it for publication was Mahmoud Barhamgi.

However, we find that the hidden messages can be obtained easily with professional tools such as JPEGsnoop and MagicEXIF, if we hide a plaintext message in the APPn of a JPEG image. Moreover, if we hide a message encrypted with conventional encryption algorithms (such as DES and AES) in the APPn, the attacker then would infer the confidentiality of the hidden data when he/she sees the garbled characteristics of the hidden data, and then he/she would destroy the data. Obviously, the effective protection of these kinds of methods is debatable. Therefore, we think that if we could display the ciphertext with a plaintext style, and then hide it in the APPn of a JPEG image, it will not destroy the visual quality or structure of the host image, and the ciphertext would have a plaintext style, thus protecting the message better.

FPE can be used to accomplish the requirements mentioned above for encrypting plaintext, because FPE is a symmetric encryption technology, it can present the user with the ciphertext with a plaintext style while retaining the original format of the message. For example, the ciphertext would still be an integer, when the plaintext integer is encrypted by FPE. FPE already has good security effectiveness in the protection of digital character data, such as social security numbers, credit card numbers, IP addresses, and strings with a special format. However, as far as we know, FPE for a common text with full semantics has not been provided. For instance, the method for encrypting ‘‘I will wait for you with a red umbrella at the gate of MIT and give you ***information’’ has not been provided consistently.

NLP is a technology that uses the computers to perform a variety of processing on the written or spoken language. This technology can perform parsing, semantic analysis, automatic summarization, entity recognition, part-of-speech tagging, information extraction, etc. As far as we know, there is no case of introducing NLP technology into FPE so far.

So, inspired by the techniques of NLP, such as semantic analysis, entity recognition, and information extraction, and the encryption methods of FPE for encrypting integers and characters, in this paper, we propose a technique for hiding common text information in the APPn markers of a JPEG image after it is encrypted with FPE, i.e., *a data hiding technology based on NLP and FPE*. This technology first analyses the category of each word in the text message using the NLP technology, and then identifies the key words and sensitive words in these analysed words (or sets the sensitive words according to users’ needs). Next, the technique obtains the numbers of these sensitive words in the vocabulary, and encrypts these numbers with FPE, which encrypts the integers with the help of unbalanced Feistel network. Thereafter, the method obtains the ciphertext words through seeking the corresponding words in the vocabulary according to the numbers of words encrypted; with that, the algorithm determines whether to use the current words as the ciphertext words, according to the semantic similarities between the plaintext words and the ciphertext words (the reference value of similarity can be set by the user). Finally, the ciphertext text is hidden in some APPn markers of the host JPEG image after successful encryption. The experimental results show that there is no influence on the structure, size, and image quality of the host image after hiding the ciphertext text in the APPn, and the restored host image is exactly the same as its original appearance. Furthermore, the hidden ciphertext text is also a text message, which has a plaintext style and reasonable semantics, thus protecting the security of the message better.
The main contributions of this article are as follows:

1) A method that hides the ciphertext text in the APPn markers after being encrypted with FPE is proposed for the first time, thereby ensuring the image quality and better message security.

2) FPE that is based on NLP is proposed for the first time.

3) DES and AES are combined to construct the double-encrypted unbalanced Feistel network.

4) A method is implemented to control the semantic similarity between the plaintext words and ciphertext words by the users.

5) The security of FPE proposed in this paper is demonstrated.

The rest of this paper is organized as follows. Related works are discussed in Section 2. The basic knowledge is given in Section 3. In addition, the framework is proposed in Section 4, followed by FPE based on the double-encryption unbalanced Feistel network in Section 5. The experiment and discussion are in Section 6. Finally, Section 7 finishes this article with the conclusion and the future work.

II. RELATED WORK

In this section, we give the related work from two aspects, namely, data hiding using APPn markers and FPE.

A. HIDING DATA USING APPn MARKERS

The marker section of a JPEG image was used to embed the icons and the metadata related to the icons in [8], so that the recipients could use the thumbnails of the icons to display the metadata associated with each icon when the image was sent to them. A JPEG transmorphing algorithm was provided in [7]. In this algorithm, part of the information about the original image was inserted into the APPn markers of the processed image, which could ensure the recovery of the original image while saving communication bandwith. The JPEG image was encrypted first in [9], and the padding data generated in the encryption process were hidden in the APPn markers after the padding data were encrypted. Next, the secret data were hidden in the encrypted padding data; then, an encrypted JPEG bitstream was generated after the key and image parameters were also hidden in APPn. The method proposed in [9] could preserve the format of the image, and it had little impact on PSNR for some images. In view of the fact that some areas in an image were the region of interest(ROI) and other regions were unprotected areas, the method proposed in [10] first blurred the appearance of the ROIs of the image, and then saved the information of the ROIs in the APPn markers of the blurred image. In this way, the ROIs of the blurred image can be recovered without distortion. A black box approach was used to select the candidate from APP0–APPn to hide the secret information in [11], so that the execution efficiency of the algorithm could be improved without destroying the structure of the image.

B. FORMAT-PRESERVING ENCRYPTION

An identity-based format-preserving encryption was proposed in [12], with the key-derivation function, and this algorithm combined the identity of the device and the master key used by the user to generate the key used in FPE, thereby limiting the damage to the FPE due to key exposure and increasing the lifetime of the key. The data from the first row of the database were extracted to generate the derived key in [13], and the subkey for the first encryption was generated by combining the first derived key with the static key. Furthermore, the sensitive areas in the first line of data were encrypted with FPE in [13]. The authors in [14] proposed to encrypt a Base64 binary plaintext as a ciphertext with the Base64 format using FFX, VAES3, BPS-BC, and Visa FPE, and the Base64 ciphertext could be verified by the Base64 validator. A novel, similarity recoverable and format-preserving string encryption framework was proposed in [15], and this framework first grouped the secret string into a plaintext pair, then encrypted the plaintext pair to obtain a ciphertext pair. This encryption algorithm made the Levenshtein distance between the two strings in the plaintext pair equal to the Levenshtein distance between the two strings in the ciphertext pair, as well as the Q-gram based distance between the two strings in the ciphertext pair and the plaintext pair. Therefore, there was a certain similarity between the ciphertext string and the plaintext string in that the ciphertext was still a string in [15]. Cycle Slicer was introduced in [16] to solve two important problems in FPE, namely, domain targeting and domain completion. In domain targeting, Cycle Slicer was used to construct the ciphertexts in large domains into ciphertexts in small domains, which improved the efficiency of the algorithm. In domain completion, the authors used Cycle Slicer to provide an alternative construction scheme that was more efficient than the Zig-Zag construction in keeping with the existing mappings. The authors proposed a modification method that could guarantee the function of the original encryption algorithm in [17]. This method used a new Cycle walking technology to encrypt the plaintext, and the Cycle walking could generate correspondence among the plaintext set$\longrightarrow$the original ciphertext set$\longrightarrow$the new ciphertext set$\longrightarrow$and the plaintext set, so that the original ciphertext could still be correctly decrypted after the ciphertext space of the encryption algorithm was amended. BPS model was proposed in [18], which was similar to the Cipher-Block Chaining model. The BPS model used a 16-bit counter to XOR the 16-bit highly significant bits of the left and right blocks of the unbalanced Feistel network, and it could encrypt a string with a radix of 2^{48}. Dworkin recommended three FPE modes of FF1, FF2, and FF3 in [19], and guided the round number of the Feistel network. An encryption method suitable for high-speed optical communication was proposed to guarantee the collection and transmission of data in Gigabit Ethernet in [20]. This method used the FPE block cipher algorithm to realize the symmetrical encryption of the 8b/10b data stream at the level of physical coding sublayer. It not only realized the security of the physical layer but also introduced no overhead during encryption. Furthermore, FPE in FFX mode was proposed in [21], and FPE for encrypting characters was proposed in [22]. Moreover, Iyer *et al.* improved the data security of FPE by embedding specific key identifiers for rotating keys in [23].

TABLE 1. The symbols used in this article and their meanings.

Symbols	Meanings
FPE	Format-preserving encryption
DEUF	Double Encryption Unbalanced Feistel Network
NLP	Nature language processing
R_i^e	The i^{th} byte data of R_i^{AES}
L_i^l	Result of L_i^{DES} XOR R_i^e and left shift p bits
L_i^e	The i^{th} byte data of L_i^l
R_i^l	Result of R_i^{AES} XOR L_i^e and left shift p bits

III. PRELIMINARIES

In this section, we provide the basic knowledge used in this article.

A. SYMBOLS AND THEIR MEANINGS

In this section, we give the symbols used in this article and their meanings, as shown in Table 1.

In addition, L_i^{DES} refers to the encryption result of DES for the left part of the Feistel network in the i^{th} round data processing; R_i^{AES} refers to the encryption result of AES for the right part of the Feistel network in the i^{th} round data processing; and common text denotes text which has digits, entity, and words having common meanings, and this text has the complete semantics.

B. FEISTEL NETWORK AND ITS ROUND NUMBER FOR SECURITY

There are three types of Feistel networks used in FPE, namely, the classical Feistel network [24], the unbalanced Feistel network [25], and the interactive Feistel network [26], [27]. We use the unbalanced Feistel network in this paper. Since there are no researchers to use DES and AES to construct the unbalanced Feistel network in the existing cases of FPE using an unbalanced Feistel network, we adopt the double encryption scheme with DES and AES in our Feistel network. Therefore, the Feistel network we use is named the Double Encryption Unbalanced Feistel Network (DEUF).

As far as we know, the document [19] published by NIST pointed out that when AES was used as the scrambling function of the unbalanced Feistel network, it could provide 128-bit security for the FF1 model after 10 rounds of transformation; also, it could provide 128-bit security for the FF3 model after 8 rounds of transformation. However, the authors of [28] pointed out that the FF3 model used

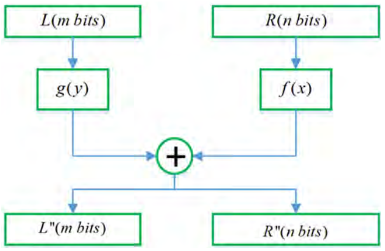


FIGURE 1. The i^{th} round encryption process of DEUF.

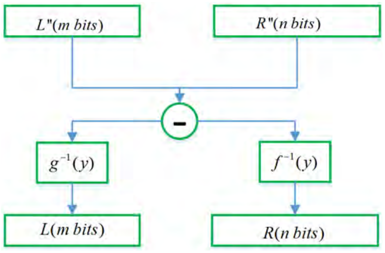


FIGURE 2. The i^{th} round decryption process of DEUF.

in [19] could not provide 128-bit security for a small character set (less than 2^{17}) when the round number is 8; also, the FF1 model could not achieve 128-bit security when the round number is 10, if the size of the character set is less than 2^{11}. Additionally, Durak *et al.* pointed out the same results mentioned above in [29]. To this end, we introduce two encryption methods, namely, DES and AES, in DEUF to construct the round functions. As shown in Fig. 1, DES is used in $g(y)$, and AES is used in $f(x)$. We also improve the confusion of data by shifting left and data exchanging after the data are encrypted by DES and AES. Therefore, we take 8 rounds of transformation in DEUF. Moreover, since the digits encrypted in character format are not always in the radix range [19], we encrypt the injective results of the plaintext words in the vocabulary in the decimal integer format. It should be noted that the left half of the initial input data processed by DEUF is less than or equal to 7 bytes, and the right one is less than or equal to 15 bytes; this is to prevent the results of DES and AES encryption from exceeding the range that the radix can express. According to the knowledge mentioned above, we give the encryption and decryption processes of DEUF below.

1) THE I^{TH} round encryption of DEUF

The encryption process of DEUF is shown in Fig. 1. We give the process in detail as follows.

Step 1: Divide the plaintext into two parts, namely, L_i (m bits) and R_i (n bits, and $m \neq n$). If the input data are less than $m + n$ bits in the beginning, they will be automatically filled into $m + n$ bits.

Step 2: Encrypt L_i with DES into L_i^{DES}, and encrypt R_i with AES into R_i^{AES}.

Step 3: Extract the i^{th} byte R_i^e of R_i^{AES} to XOR the $(7-i)^{th}$ byte of L_i^{DES}, and the L_i^{DES} XORed is shifted left circularly by q bits to generate L_i^l; meanwhile, R_i^e is recorded.

Step 4: Extract the i^{th} byte L_i^e of L_i^l to XOR the $(15-i)^{th}$ byte of R_i^{AES}, and shift the last 8 bytes of the data XORed left circularly by q bits to generate R_i^l; meanwhile, record L_i^e.

Step 5: Remove the i^{th} byte of L_i^l and R_i^l to generate L_i' and R_i', respectively.

Step 6: Exchange the m-bit in L_i' and the last m-bit in R_i' to generate L_i'' and R_i''.

Step 7: Use L_i'' and R_i'' as the inputs of the L half and the R half of the next round of the Feistel network, respectively.

The $\bigoplus$ in the Feistel network above performs the XOR, shift, and exchanging operations to the data generated by the f, g functions. Here, the $m(< n)$ bits data are the position of the plaintext word in the vocabulary, and the n bits data are used as a tweak factor to enhance the security of the position of the plaintext word.

2) THE i^{th} ROUND DECRYPTION OF DEUF

The decryption process of DEUF is shown in Fig. 2. We give the process in detail as follows.

Step 1: Read the ciphertext and divide it into left L_i'' bits and the right R_i'' bits.

Step 2: Exchange the left m bits and the last m bits of the right n bits to generate L_i' and R_i'.

Step 3: Read L_i^e and R_i^e recorded at the time of encryption from the recording slot.

Step 4: Pad R_i^e into the i^{th} byte of R_i', and the original i^{th} byte and the subsequent bytes of R_i' are shifted backward by one byte, thereby generating R_i^l.

Step 5: Shift the last 8 bytes of R_i^l circularly by q bits, and then XOR L_i^e and the $(15-i)^{th}$ bytes of R_i^l shifted to generate R_i^{AES}.

Step 6: Pad L_i^e into the i^{th} byte of L_i', and the original i^{th} byte and the subsequent bytes of L_i' are shifted backward by one byte, thereby generating L_i^l.

Step 7: Shift L_i^l circularly by q bits, and then XOR R_i^e and the i^{th} byte of L_i^l shifted to generate L_i^{DES}.

Step 8: Decrypt L_i^{DES} and R_i^{AES} to obtain L_i and R_i, and then take L_i and R_i as the inputs of the next round of the Feistel network.

The $\ominus$ in Fig. 2 executes the data exchanging, shift, and XOR operations, which is the inverse of the operation in the encryption process. Obviously, the decryption operation here starts with the last round of the encryption operation. Therefore, we set the initial value of i to 7, and the meaning of each symbol here is the same as the one in the encryption process.

IV. PROPOSED FRAMEWORK

The framework proposed in this paper includes three modules, namely, the image input module, the information encryption module, and the image output module. A detailed introduction is given below.

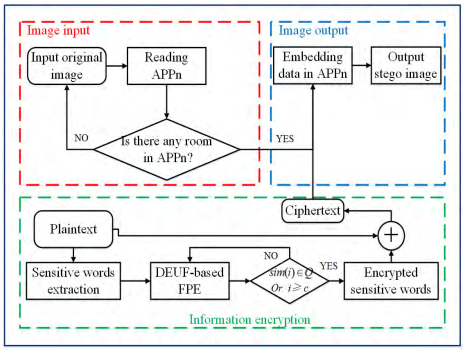


FIGURE 3. The encryption and hiding framework proposed in this paper.

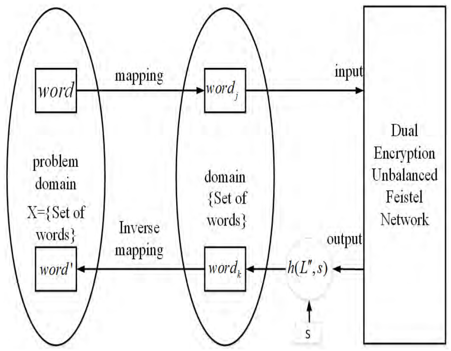


FIGURE 4. The DEUF-based FPE mode proposed in this paper.

A. MODEL

The model of the framework proposed in this article is shown in Fig. 3, in which the red dotted frame is the image input module, the green dotted frame is the information encryption module, and the blue one is the image output module.

1) MODEL ANALYSIS

In this section, we analyse the framework given above in detail.

a: IMAGE INPUT MODULE

The function of this module is to rationally choose the carrier for data hiding. Therefore, in this module, we first select the image which is intended to be used as the carrier; then, we check whether there is enough space in the APPn of the image. If so, the image will be used as a carrier; otherwise, a new image will be selected and checked. Finally, the image that can be used as the carrier is selected, and then we wait for the embedding of the encrypted data.

b: INFORMATION ENCRYPTION MODULE

This module is responsible for implementing DEUF-based FPE encryption on the confidential information. It first uses the Stanford NLP to segment the input plaintext and to mark the property of the words in the text. Then, the sensitive words are extracted according to the requirements of the user, and these words are prepared for encryption with FPE. Next, the DEUF-based FPE encryption is performed on the extracted sensitive words. The model of DEUF-based FPE is shown in Fig. 4.

In the encryption process, if the encryption is performed c times on the same word or the similarity of the words for the i^{th} encryption satisfies the demand of the user, for instance, $sim(i) \in Q$ as shown in Fig. 3, then the encryption is stopped. Next, we replace the sensitive words in the plain text with the ciphertext of the sensitive words (if the encryption is successful), thus obtaining the ciphertext. Here, the semantic similarity between words is calculated using the formula (as shown in Equation 1) in [30] based on WordNet 2.1, c denotes the maximum number of encryptions set by the user, $sim(i)$ is the semantic similarity between the ciphertext words and the plaintext words during the i^{th} encryption, and Q represents the range of $sim(i)$.

$$Simila(word, word') = \frac{\sum max1 + \sum max2}{|sw1| + |sw2|} \tag{1}$$

where $sw1$ and $sw2$ mean all the senses of $word$ and $word'$, respectively; $|swi|$ denotes the number of $swi (i = 1, 2)$; $max1$ refers to the maximum similarity between a sense in $sw1$ and all the senses in $sw2$, as well as $max2$.

c: IMAGE OUTPUT MODULE

The function of this module is to embed the ciphertext in the APPn markers of the JPEG image to generate the stego image. The host image used here is an image that has been prepared in the image input module.

V. DEUF-BASED FPE

The DEUF-based FPE proposed in this paper is shown in Fig. 4. There are three steps for the implementation of DEUF-based FPE as follows.

First, the sensitive word *word* is mapped to its number $word_j$ in the vocabulary. Next, the DEUF operation is performed in r rounds, and there are three steps for the operation of the DEUF as follows.

Step 1, $word_j$ is taken as the left half L of the Feistel network, and the tweak factor is taken as the right half R of the Feistel network, where $|L| \neq |R|$.

Step 2, R and L are processed using the DEUF, and the outputs are R'' and L'', respectively.

Step 3, L'' and R'' are concatenated to obtain $L''||R''$, which is taken as the input of the next round of DEUF.

Second, the function $h(L'', s)$ is run on the left half of the final output of the DEUF to obtain the number $word_k$ of the ciphertext word correspon-ding to the sensitive word. Here, $h(L'', s) = per(L'')+s$, where $per()$ is the permutation on the finite field G_p (p represents the radix of the finite field), and s is a device-independent quantum random number.

Third, the encryption result $word'$ of the sensitive word is obtained by the inverse mapping of $word_k$.

It is worth pointing out that the decryption process of DEUF-based FPE is the reverse of its encryption process.

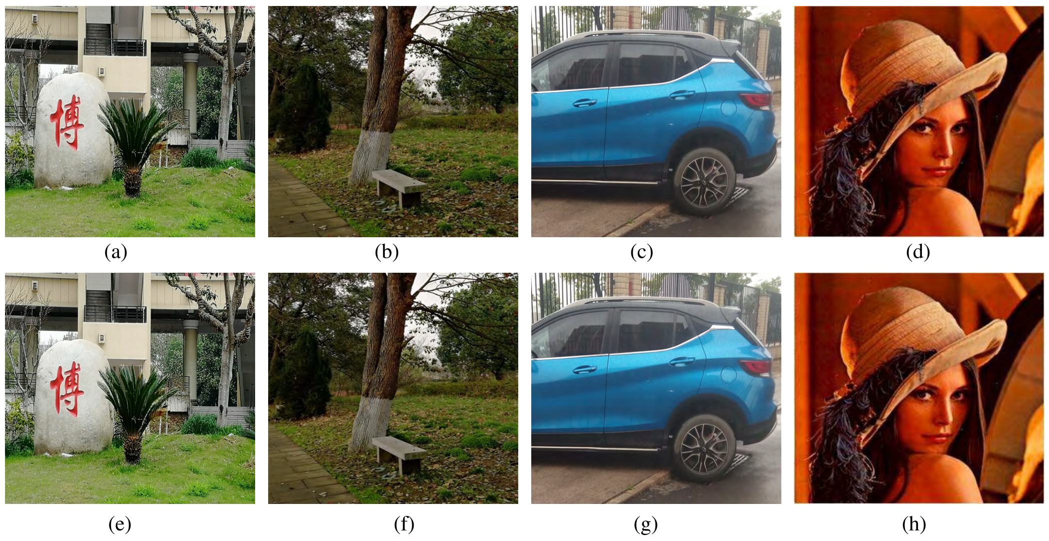


FIGURE 5. **The representatives of the 1000 host images and their corresponding stego images. (a) Host Image 1. (b) Host Image 2. (c) Host Image 3. (d) Host Image 4. (e) Stego Image 1. (f) Stego Image 2. (g) Stego Image 3. (h) Stego Image 4.**

Theorem 1: Encryption for $word_j$ can achieve more than 128 bits AES security under the DEUF-based mode.

Proof: First, it is known from the implementation process of the Feistel network(it includes encryption, XOR, shift, and exchanging) that the encryption with AES in the DEUF can guarantee the AES security of $word_j$. Second, according to the knowledge about the security number of the rounds of the Feistel network discussed in Section 3.2, and about the number of English words being higher than 2^{17} (there are approximately 170000 English words) and the discussion in [19], [28], [29] about the security requirement of the Feistel network, our Feistel can achieve 128-bit AES security. Furthermore, since we use DES and AES to encrypt the data in the Feistel network, the security of our algorithm is higher than that of only encrypting the data of the right (or left) half of the Feistel network. Therefore, the security strength of our Feistel network is higher than that of 128 bits AES. At the same time, the random number s with an uncertain size and the permutation over finite field G_p also enhance the ability for protecting the word number $word_j$. Hence, the DEUF-based FPE mode proposed in this paper can provide more than 128 bits AES security for $word_j$.

VI. EXPERIMENT AND DISCUSSION

The computer used in our experimentation is equipped with Win 10 OS and an Intel(R) Celer-on(R) CPU with a frequency of 2.81 GHz. The algorithm runs on JAVA 7. We collect 1000 images independently taken by 30 students for these experiments, and some of them and their corresponding stego images are shown in Fig. 5. To ensure the diversity of the plain text, we ask the students to embed the original plain text in the APPn of JPEG (we use the user comment of Exif, which is located in APP1 [6]) when they collect the images and ask the length of each plain text to be approximately 15 words.

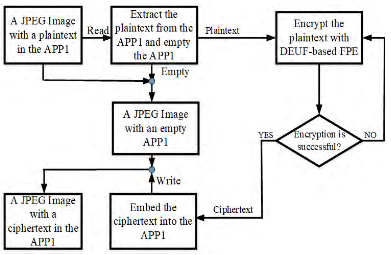

A JPEG Image with a plaintext in the APP1
Read
Extract the plaintext from the APP1 and empty the APP1
Plaintext
Encrypt the plaintext with DEUF-based FPE
Empty
A JPEG Image with an empty APP1
YES
Encryption is successful?
NO
Write
A JPEG Image with a ciphertext in the APP1
Embed the ciphertext into the APP1
Ciphertext

FIGURE 6. **The flowchart that the plaintext is extracted from the APPn and then embedded the ciphertext into the APPn after encryption with our FPE.**

During the experiment, we first read the plaintext text embedded in these images for encryption. After encryption, we embed the ciphertext into the corresponding image again. The detailed processes of extraction and embedding are shown in Fig. 6.

In addition, we mainly encrypt the nouns(i.e., singular nouns) in the text due to the importance of the nouns in understanding the semantics of the text; for example, the semantics of ''red notebook'' will be changed tremendously if we change it to ''red toy''. To this end, more than 3600 nouns are put into the vocabulary to conduct the experiments in this article. In addition, we use 10 random numbers in the experiments, and we take 0 as the first random number. To better observe the performance of our algorithm, we have conducted experiments on 1000 images for each random number; therefore, a total of 10000 experiments is performed in this paper. The experimental results from seven aspects are shown below.

A. NUMBER OF NOUNS IN TEXT

Since there is no specific requirement when collecting the images and the plain text, the plain texts embedded in the

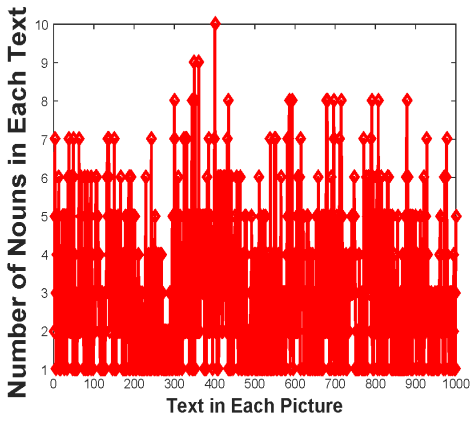


FIGURE 7. **The number of nouns in each text embedded in the host image.**

images all have rich sentence structures. To this end, we first check the number of nouns in each plaintext text, as shown in Fig. 7.

It can be seen from Fig. 7 that at least one noun needs to be encrypted in these 1000 plaintext texts. Moreover, the diversity of the number of nouns would further demonstrate the reliability of the experimental results.

B. WORDS THAT FAILED TO BE MATCHED

In the experiment, we set the range of similarity between the plaintext words and the ciphertext words to be [0.05, 0.20]. Since the current data set is not rich enough, the corresponding ciphertext word for some nouns cannot be found within the given encryption times under the condition of $sim(i) \in [0.05, 0.20]$. For example, the corresponding ciphertext words for *something, anything, everything, anyone, and oneself* all fail to be found in each set of experiments. Furthermore, some words whose matching words could not be found are due to their special natures, for example, the words belonging to a special field such as paper-cutting or calligraphy. Certainly, some words that cannot be found with the matching object are due to the too low similarity (for example, 0), while other words are due to the too high similarity; for example, the lowest value of the similarities between the word *truth* and its matching objects is higher than 0.20. Hence the lowest and highest values of similarity are not in the range of [0.05, 0.20].

C. SEMANTIC SIMILARITY BETWEEN WORDS

Three criteria were used in [15] to measure the similarity between the plaintext string and the ciphertext string, and their similarity values are Cosine 0.033, Jaccard 0.011, and Sorensen Dice 0.013. Although these types of values make the plaintext and the ciphertext almost irrelevant, it is difficult to ensure reasonable semantics for the ciphertext. Therefore, in order to mask the information contained in the plain text and make the ciphertext text have reasonable semantics to enhance the credibility of the ciphertext, we set the range of semantic similarity to be [0.05, 0.20]. That is, only those

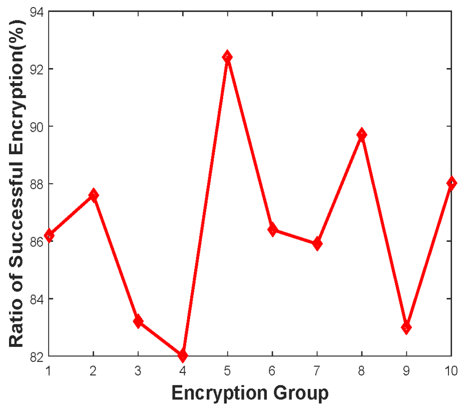


FIGURE 8. **The ratios that successfully find the ciphertext words in each group.**

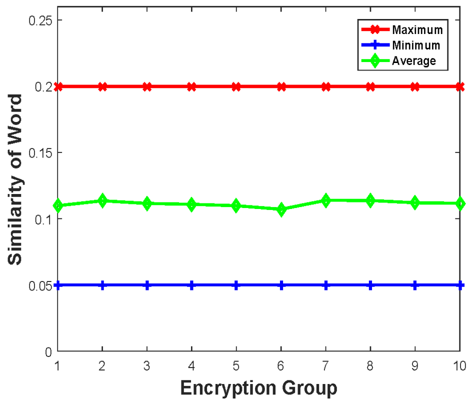


FIGURE 9. **The value of word semantic similarity during the encryption of each group, where Maximum represents the highest value, Minimum means the lowest value, and Average denotes the average value.**

words with a semantic similarity to the plaintext words in [0.05, 0.20] are selected as the ciphertext words when using WordNet 2.1 to find the ciphertext words. We show the ratios of successful encryptions in which the plaintext words can be successfully encrypted in each of the 10 groups in Fig. 8.

As seen from Fig. 8, in the 10 experiment groups, there are at least 82% plaintext words that successfully find the ciphertext words, so our experimental results are very good, and the method that selects the ciphertext words based on the semantic similarity of words is feasible.

In addition, the corresponding semantic similarities of the 10 groups under the condition of words being successfully encrypted are shown in Fig. 9. To obtain a more detailed understanding for the semantic similarity of the selected words, we present the 10 highest and 10 lowest values of word semantic similarities in the first encryption group as shown in Fig. 10. From the experimental results, we can see that both the highest value and the lowest value of word similarities are belong to [0.05, 0.2]. This shows that our method can enable users to control the similarities between the plaintext words and the ciphertext words.

TABLE 2. The values of cosine similarity.

	1	2	3	4	5	6	7	8	9	10
highest value	0.9991	0.9960	0.9968	0.9982	0.9953	0.9929	0.9917	0.9857	0.9978	0.9732
lowest value	0.5594	0.5133	0.7004	0.8024	0.8101	0.9147	0.9193	0.9425	0.9846	0.9732
average value	0.9689	0.9630	0.9617	0.9693	0.9729	0.9656	0.9566	0.9668	0.9912	0.9732

TABLE 3. The values of TFIDF cosine similarity.

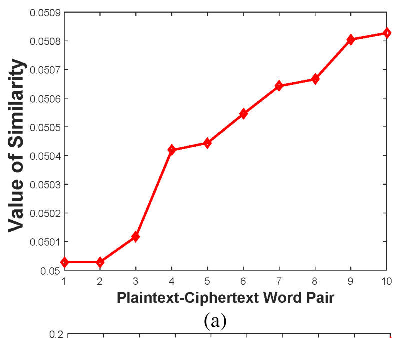
	1	2	3	4	5	6	7	8	9	10
highest value	0.9558	0.9114	0.8971	0.8704	0.8481	0.8283	0.8143	0.8337	0.8574	0.5686
lowest value	0.2586	0.1485	0.1042	0.1485	0.1731	0.2981	0.2586	0.4397	0.6929	0.5686
average value	0.7715	0.6745	0.6409	0.6249	0.5917	0.5501	0.5268	0.5893	0.7752	0.5686

TABLE 4. The values of Jaccard Similarity.

	1	2	3	4	5	6	7	8	9	10
highest value	0.9394	0.9474	0.8750	0.8333	0.7931	0.7241	0.7000	0.6486	0.9286	0.4706
lowest value	0.3333	0.2000	0.1429	0.2000	0.2857	0.4286	0.3333	0.3333	0.6216	0.4706
average value	0.8131	0.7198	0.6810	0.6657	0.6287	0.5963	0.5262	0.4794	0.7751	0.4706

TABLE 5. The values of similarity based on the common substring.

	1	2	3	4	5	6	7	8	9	10
highest value	0.9783	0.9250	0.9189	0.8982	0.8742	0.8311	0.7581	0.7333	0.9669	0.5806
lowest value	0.4286	0.3158	0.3462	0.2667	0.4186	0.5714	0.4493	0.5052	0.6871	0.5806
average value	0.8653	0.7904	0.7606	0.7684	0.7418	0.7092	0.6361	0.6409	0.8270	0.5806



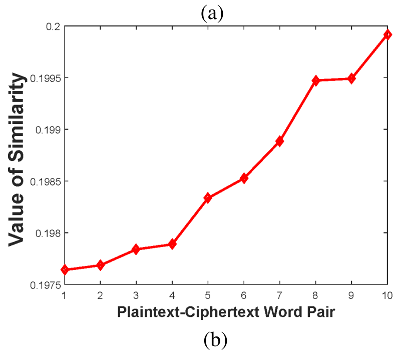


FIGURE 10. The 10 lowest values and the 10 highest values of word semantic similarities in the first encryption group. (a) The 10 lowest values. (b) The 10 highest values.

D. TEXT SIMILARITY BEFORE AND AFTER ENCRYPTION

In this section, we only test four similarities for the texts that successfully match all the singular nouns, namely, cosine similarity based on word frequency (called S1), TFIDF cosine similarity (called S2), Jaccard Similarity (called S3), and the similarity based on the common substring (called S4). When calculating these similarities, we use the common formulas. For example, we use Equation 2 to calculate the cosine similarities. In addition, among these similarities, we all test the highest, lowest, and average values, as shown in Fig. 11.

$$cos(\theta) = \frac{\sum_{i=1}^{n}(Vx_i \times Vy_i)}{\sqrt{\sum_{i=1}^{n}(Vx_i)^2} \times \sqrt{\sum_{i=1}^{n}(Vy_i)^2}} \tag{2}$$

where θ means the angle between the vectors Vx and Vy;

As seen from Fig. 11, the ciphertext text and the plaintext text all have a certain semantic relevance by using the encryption method provided by us, which ensures that the encrypted texts still meet the grammatical rules, thus avoiding the confusion of the ciphertext content. Moreover, from the effect of artificial recognition, more than 80% of the ciphertext texts that are successfully encrypted have reasonable semantics in the case of three students, who have passed CET-6, by reading the ciphertext texts carefully.

For a more detailed understanding of the similarity of the text before and after encryption, we take the first set of data as an example to show the highest, lowest, and average values of similarities, when the numbers of encrypted words are 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10, respectively, as shown in Table 2 to Table 5.

Combined with Fig. 7, Table 2 to Table 5, and the experimental results of the rest of the nine groups, we can find that, by and large, the lower the number of words encrypted in the texts, the higher the similarity between the texts before and

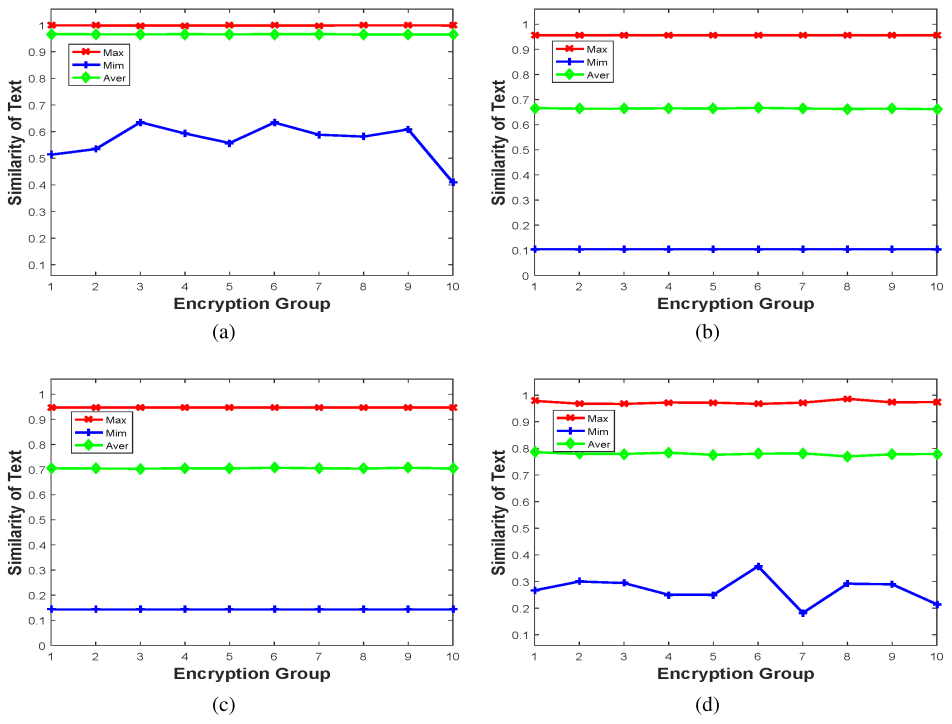


FIGURE 11. The values of four kinds of similarities in 10 groups. (a) S1. (b) S2. (c) S3. (d) S4.

TABLE 6. The differences in bytes between plaintext texts and ciphertext texts.

	1	2	3	4	5	6	7	8	9	10
$Byte_t$	862	876	832	821	874	863	858	872	816	880
$Byte_+$	529	571	542	487	595	527	534	598	514	537
$Byte_-$	250	237	233	251	234	274	259	218	244	287
$Byte_0$	83	68	57	83	45	62	65	56	58	56

after encryption, when the lengths of the sentences are the same.

E. DIFFERENCE IN BYTES BETWEEN PLAINTEXT TEXT AND CIPHERTEXT TEXT

In the experiment of this section, we only consider the case that all the singular nouns in plaintext text are encrypted effectively using DEUF-based FPE. In the experiments of 10 groups, the details of changing bytes are shown in Table 6, where $Byte_t$ means the total number of texts which have been successfully encrypted, $Byte_+$ denotes the number of texts whose number of bytes are increased, $Byte_-$ represents the number of texts whose number of bytes are reduced, and $Byte_0$ is the number of texts whose number of bytes are not changed. The average values of the number of bytes changed are shown in Fig. 12.

In general, it can be seen from Fig. 12 that, regardless of whether the sign is in view, the changes in the number of bytes of the original plaintexts are all extremely small with respect to the amounts of data of the host images after encryption. This shows that our FPE algorithm has a superior performance in keeping with the small difference in size between plaintext and ciphertext.

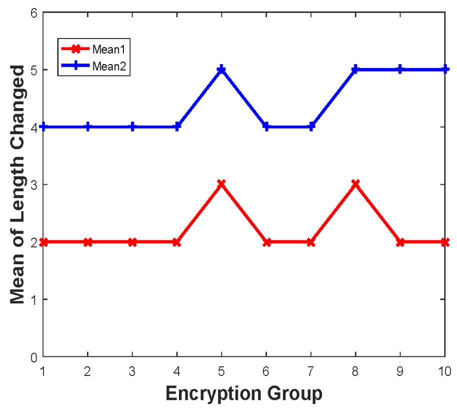


FIGURE 12. The average values of byte lengths changed, where Mean1 refers to the average values of byte lengths changed when the sign is in view, and Mean2 represents the average values of the absolute values of all byte lengths changed.

F. QUALITY OF IMAGE BEFORE AND AFTER ENCRYPTION

In this section, we focus on the structural similarity index Measure (SSIM), image size, PSNR regarding the quality of the image. Moreover, we evaluate the change of image entropy. From the case of successful encryption and hiding,

TABLE 7. The results of comparison with other algorithms.

	SSIM	PSNR	Size	Change of Image Entropy
[9]	1 or less than 0.9999	∞ or less than 53.6dB	unchanged	0 or 0.02 bit/pixel
[10]	0.987	45.08dB	unchanged	0.06 bit/pixel
ours 1	1	∞	unchanged	0
ours 2	1	∞	unchanged	0

the SSIMs, sizes, and entropy of the images in all cases have not been changed, and the values of PSNR are all ∞. The experimental results are shown in Table 7.

G. COMPARISON WITH OTHER ALGORITHMS

We compare our algorithm with [9] and [10], the results are shown as ours 1 in Table 7. To better reflect the advantages of our algorithm, we hide the image data of 2*32*32 pixels in APPn to test the performance of our algorithm with the help of IWT (the original data of the tested image data is blurred after IWT). The experimental results are shown in Table 7, and we label these experimental results as ours 2. It should be noted that the value ∞ of PSNR in [9] means lossless recovery, and 53.6 dB is the highest value of PSNR provided by the authors when their host images were recovered with distortion. And the values 0.02 bit/pixel and 0.06 bit/pixel of the changes of image entropy mean the average values of the image entropies that are reduced after the images being recovered with distortion; and 0 means the lossless recovery. Their SSIM values have the same meanings.

Table 7 shows that regardless of whether the data hidden in the APPn are the data of the image itself or the data out of the image, our method keeps the original values of the SSIM and size of the host image. Moreover, the appearance of the host image also maintains its original shape due to the value ∞ of PSNR and the unchanged entropy values. This result is obviously better than the comparison objects.

VII. CONCLUSION AND FUTURE WORK

Information security is an ever-present theme in the information age. A popular method to achieve information security is by hiding confidential information in images. In this paper, we introduce NLP technology combined with FPE to propose a data hiding technology that is based on NLP and FPE. We achieve the purpose of protecting the security of information from both the appearance and the internal aspects with our technology, because we can encrypt common plaintext text into ciphertext that still has a plaintext style. To be specific, when the ciphertext with the plaintext style is hidden in the APPn of a JPEG image, it not only can make the attacker unaware of the existence of ciphertext, but the plaintext-style text is also only seen when using professional software to analyse the stego image. Furthermore, our algorithm not only can avoid the confusion form of the hidden information due to encryption using the traditional method, such as DES or AES, but can also avoid the harmful consequences due to the degradation of image quality caused by hiding the information in the spatial or transform domain of the image. The experimental results demonstrate the superior performance of our algorithm in maintaining the SSIM, visual quality, and size of the image, and demonstrate the effectiveness of our algorithm in ensuring the format of ciphertext and maintaining the reasonable semantics of the ciphertext text.

In the future, we will make efforts in multi-noun recognition, entity recognition and encryption, the completeness of word set, the delicacy of word classification, encryption for verbs and adjectives, and improving the processing speed of the system, to provide a more perfect encryption system and a higher encryption efficiency. In addition, we will strive to study the technology for encrypting the long text.

REFERENCES

[1] T. C. Lu, J. H. Wu, and C. C. Huang, "Dual-image-based reversible data hiding method using center folding strategy," *Signal Process.*, vol. 115, pp. 195–213, Oct. 2015.

[2] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and reversible data hiding in encrypted images with public-key cryptography," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1622–1631, Sep. 2016.

[3] H. Yao, C. Qin, Z. Tang, and Y. Tian, "Improved dual-image reversible data hiding method using the selection strategy of shiftable pixels' coordinates with minimum distortion," *Signal Process.*, vol. 135, pp. 26–35, Jun. 2017.

[4] W. Hong, X. Zhou, D. C. Lou, T. S. Chen, and Y. Li, "Joint image coding and lossless data hiding in VQ indices using adaptive coding techniques," *Inf. Sci.*, vols. 463–464, pp. 245–260, Oct. 2018.

[5] R. Bhardwaj and A. Aggarwal, "An improved block based joint reversible data hiding in encrypted images by symmetric cryptosystem," *Pattern Recognit. Lett.*, vol. 1, pp. 1–9, Aug. 2018.

[6] FromWikipedia. (2018). *JPEG*. Accessed: Dec. 26, 2018. [Online]. Available: https://en.wikipedia.org/wiki/JPEG

[7] L. Yuan and T. Ebrahimi, "Image transmorphing with JPEG," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2015, pp. 3956–3960.

[8] H. Fox, J. Benjamin, and R. Wulfson, "Method and system for utilizing a JPEG compatible image and icon," U.S. Patent 11 144 704, Jun. 5, 2008.

[9] Z. Qian, H. Zhou, X. Zhang, and W. Zhang, "Separable reversible data hiding in encrypted JPEG bitstreams," *IEEE Trans. Depend. Sec. Comput.*, vol. 15, no. 6, pp. 1055–1067, Dec. 2018.

[10] L. Yuan and T. Ebrahimi, "Image privacy protection with secure JPEG transmorphing," *IET Signal Process.*, vol. 11, no. c, pp. 1031–1038, Dec. 2017.

[11] M. F. Sabir, J. H. Jones, H. Liu, and A. V. Mbaziira, "A non-algorithmic forensic approach for hiding data in image files," in *Proc. Int. Conf. Comput. Data Anal.*, Jul. 2018, pp. 60–64.

[12] M. Bellare and V. T. Hoang, "Identity-based format-preserving encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1–48.

[13] T. W. Arnold, J. C. Dayka, S. R. Hart, G. G. Jackson, E. S. Powers, and J. W. Sweeny, "Secure format-preserving encryption of data fields," U.S. Patent 9 858 436, Jan. 2, 2018.

[14] S. R. Hart, E. S. Powers, and J. W. Sweeny, "Format-preserving encryption of base64 encoded data," U.S. Patent 14 968 006, July.3, 2018.

[15] Y. Li and W. H. Wang, "Similarity recoverable, format-preserving string encryption," in *Proc. Asia-Pacific Web Conf.* Cham, Switzerland: Springer, 2016, pp. 439–443.

[16] S. Miracle and S. Yilek, "Cycle Slicer: An Algorithm for Building Permutations on Special Domains," in *Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Cham, Switzerland: Springer, 2017, pp. 392–416.

[17] P. Grubbs, T. Ristenpart, and Y. Yarom, "Modifying an Enciphering Scheme After Deployment," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, New York, NY, USA: Springer, 2017, pp. 499–527.

[18] E. Brier, T. Peyrin, and J. Stern, "BPS: a format-preserving encryption proposal," in *Proc. NIST*, 2010, pp. 1–11.

[19] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Methods for Formatpreserving Encryption*, vol. 800. NIST Special Publication, 2016, p. 38G.

[20] A. P "erez-Resa, M. Garcia-Bosque, C. Sáńchez-Azqueta, and S. Celma, "Physical layer encryption for industrial ethernet in gigabit optical links," *IEEE Trans. Ind. Electron.*, vol. 66, no. 4, pp. 3287–3295, Apr. 2019.

[21] M. Bellare, P. Rogaway, and T. Spies, *The FFX Mode of Operation for Format-Preserving Encryption*, vol. 20. Gaithersburg, MD, USA: NIST, 2010.

[22] M. Li, Z. Liu, J. Li, and C. Jia, "Format-preserving encryption for character data," *J. Netw.*, vol. 7, no. 8, pp. 1239–1244, 2012.

[23] S. R. Iyer and N. R. Keerthi, "Progressive key rotation for format preserving encryption(FPE)," U.S. Patent 15 276 125, Dec. 18, 2018.

[24] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM J. Comput.*, vol. 17, no. 2, pp. 373–386, 1988.

[25] B. Schneier and J. Kelsey, "Unbalanced feistel networks and block cipher design," in *International Workshop on Fast Software Encryption*. Berlin, Germany: Springer, 1996, pp. 121–144.

[26] R. Anderson and E. Biham, "Two practical and provably secure block ciphers: BEAR and LION," in *Proc. Int. Workshop Fast Softw. Encryption* Berlin, Germany: Springer, 1996, pp. 113–120.

[27] S. Lucks, "Faster luby-rackoff ciphers," in *Proc. Int. Workshop Fast Softw. Encryption*. Berlin, Germany: Springer, 1996, pp. 189–203.

[28] M. Bellare, V. T. Hoang, and S. Tessaro, "Message-recovery attacks on Feistel-based format preserving encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Jun. 2016, pp. 444–455.

[29] F. B. Durak and S. Vaudenay, "Generic round-function-recovery attacks for feistel networks over small domains," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, Cham, Switzerland: Springer, 2018, pp. 440–458.

[30] Y. Wei and X. Endong, "English words similarity calculation based on wordnet," in *Proc. 2nd Nat. Students Symp. Comput. Linguistics*, May 2004, pp. 281–287.

HUAIBO SUN received the M.S. degree in applied mathematics from the PLA Information Engineering University, China, in 2006. He is currently pursuing the Ph.D. degree with the Internet of Things Lab, Beijing University of Posts and Telecommunications, Beijing. His research interests include the Internet of Things, data hiding, information security, image processing, and crowd computing.

HONG LUO received the B.S., M.S., and Ph.D. degrees from the Beijing University of Posts and Telecommunications, Beijing, China, in 1990, 1993, and 2006, respectively. She is currently a Professor with the School of Computer Science, Beijing University of Posts and Telecommunications. She is also a Research Member of the Beijing Key Lab of Intelligent Telecommunication Software and Multimedia. From 2004 to 2005, she held a visiting position with the Department of Computer Science and Engineering, The University of Texas at Arlington. She was a Visiting Professor with the Helsinki University of Technology, in 2008. She is also a recipient of the New Century Excellent talents in the University of China, in 2008. Her research interests include the Internet of things, wireless networking, sensor networks, smart environments, and communication software.

YAN SUN received the B.S. degree from Beijing Jiaotong University, in 1992, and the M.S. and Ph.D. degrees from the Beijing University of Posts and Telecommunications, in 1996 and 2007, respectively. She is currently a Professor with the School of Computer Science, Beijing University of Posts and Telecommunications, China. She is also a Research Member of the Beijing Key Lab of Intelligent Telecommunication Software and Multimedia. Her research interests include the Internet of Things, sensor networks, smart environments, and embedded systems.