Text security segmentation algorithm based on key distribution

Quan Zhang¹, Xiaoming Ju¹, Haojin Qi², Yong Wang²

¹East China Normal University, School Computer Science and Software Engineering, Shanghai, 200062
²State Grid Zhejiang Electric Power Company Information and Communication Branch, Hangzhou, 310007

Author: Quan Zhang; email: 51174500157@stu.ecnu.edu.cn; phone: 18852951520
Author: Xiaoming Ju; email: xmju@sei.ecnu.edu.cn; phone: 13774272718
Author: Haojin Qi; email: 453644954@qq.com; phone: 18868615092
Author: Yong Wang; email: 3614149@qq.com; phone: 15906504082

Abstract: The vector space projection theory of key decomposition problem is proposed. The key decomposition algorithm is given, and the theory and algorithm are introduced into the research of text segmentation problem. A text security recovery for practical secrecy is given. The algorithm is compared with the traditional text segmentation algorithm. It realizes that the text is first divided into \( N \) paragraphs, and then the paragraph is mapped to \( n \) sub-texts by using the principle of grouping key. If any \( m \) sub-texts are owned, the original text can be restored, which provides security for the storage and transmission of important texts.

1. Introduction
Key distribution is an important topic discussed in modern cryptography. In 1979, Shamir¹ and Blakley² independently proposed the concept of key decentralized management. The mechanism for implementing this idea is called the \((t,n)\)-threshold scheme. The scheme divides a key (called a system key) into \( n \) parts (called \( n \) sub-keys) and hands them to \( n \) individuals for storage, so that the determined integers satisfy: (1) In this person, any individual collaboration can recover the system key; (2) Any \( r \) person collaboration does not help restore the system key. This idea of key decentralized management makes key management more secure and flexible. Document security recovery refers to dividing a text into \( N \) paragraphs, and then the paragraph is mapped to \( n \) sub-texts by dividing the text into \( n \) sub-texts by using the principle of grouping key. If any \( m \) sub-texts are owned, the original text can be restored, which provides security for the storage and transmission of important texts.

2. Key Sharing Problem

2.1 Key Sharing Theory
Given a cryptosystem \(< M, C, K, E, D >\), where \(M\) is a plaintext space, \(C\) is a ciphertext space, \(K\) is a key space, \(E\) is an encrypted space, and \(D\) is a decryption space. This is a traditional cryptosystem where \(K\) acts as both an encryption key and a decryption key space\(^5\). The following discussion discusses how to assign a key to \(n\) people, allowing any \(m\) person to recover the key, and less than \(m\) people can not.

If \(K\) is regarded as a vector space, it is known from the above discussion that any of the key vectors \(\alpha = (k_1, k_2, \ldots, k_{C(n,m-1)})\), \(k_1 \in A\) and \(A = (a_1, a_2, \ldots, a_K)\) whose dimension is \(C(n, m-1)\), \(K\) is the key character set. Project \(\alpha\) into \(n\) sub-key vectors \(\beta_1, \beta_2, \ldots, \beta_n\),

\[
\beta_i = (k_{i1}, k_{i2}, \ldots, k_{iC(n,m-1)})
\]

where there are \(C(n-1, m-1)\) non-zero elements in the components of \(k_{ij} \in \{k_1, k_2, \ldots, k_{C(n,m-1)}\}\) or \(k_{ij} = 0\). There are \(C(n-1, m-1)\) non-zero elements in the component of \(\beta_i\), that is, the length of the subkey \(\beta_i\).

The vector space \(K\) is projected to its \(m\) \(n\)-dimensional subspaces to obtain \(K_1, K_2, \ldots, K_m\). According to the requirements of key decomposition, they satisfy \(K_1 \cup K_2 \cup \ldots \cup K_m = K\), that is, the sum of \(m\) subkey vector spaces is the original key vector space\(^6\). At the same time, there must be dimension \((K_{i1} \cup K_{i2} \ldots \cup K_{i(n-1)}) < \text{dimension}(K) = C(n, m-1)\), in other words, \(K_{i1} \cup K_{i2} \cup \ldots \cup K_{i(n-1)} < K\). That is, the sum of any \(m-1\) subkey vector spaces must be the true subspace of the key space \(K\), or original key cannot be recovered when the number of subkey vectors is less than \(m\).

In order to explain the above problems better, let’s assume that there are five people, respectively \((P_1, P_2, P_3, P_4, P_5)\), which is known from above. If at least three people are required to open the key, there should be a \(C_5^3 = 10\) key, set to \((K_1, K_2, \ldots, K_{10})\), then no two of the five people can open the key. This allows people to be grouped and then mapped to the key. That is, a combination of any two of the five people: \(P_1 P_2, P_1 P_3, P_1 P_4, P_1 P_5, P_2 P_3, P_2 P_4, P_2 P_5, P_3 P_4, P_3 P_5, P_4 P_5\). Then corresponding to \((K_1, K_2, \ldots, K_{10})\), for example, \(P_1 P_2\) and \(K_1\) correspond to each other, indicating that \(K_1\) is \(P_1 \) and \(P_2\) do not have these two personalities, so it is easy to get a key allocation scheme (Table 1).

| Partner | Missing key | Mastered key |
|---------|-------------|--------------|
| \(P_1\) | \(K_1, K_2, K_3, K_4\) | \(K_5, K_6, K_7, K_8, K_9, K_{10}\) |
| \(P_2\) | \(K_1, K_5, K_6, K_7\) | \(K_2, K_3, K_4, K_9, K_{10}\) |
| \(P_3\) | \(K_2, K_5, K_6, K_9\) | \(K_1, K_3, K_4, K_7, K_{10}\) |
| \(P_4\) | \(K_3, K_6, K_9, K_{10}\) | \(K_1, K_2, K_4, K_5, K_7, K_9\) |
| \(P_5\) | \(K_4, K_7, K_9, K_{10}\) | \(K_1, K_3, K_2, K_6, K_5, K_8\) |

\(2.2\) **Key Sharing Algorithm**

As described above, the key vector \(\alpha = (k_1, k_2, \ldots, k_{C(n,m-1)})\) has \(n\) partners \((A_1, A_2, \ldots, A_n)\), and \(n\) individuals take \(m-1\) combinations with the following \(C(n, m-1)\) methods:

\[
A_1 A_2 \ldots A_{m-2} A_{m-1},
\]

\[
A_1 A_2 \ldots A_{m-2} A_m,
\]

\[
A_1 A_2 \ldots A_{m-2} A_{n},
\]

\[
A_1 A_2 \ldots A_{m-1} A_{m},
\]

\[
A_{n-m-2} A_{n-m-1} \ldots A_{n-1} A_n
\]

Let this combination correspond to \(k_1, k_2, \ldots, k_{C(n,m-1)}\), for example, \(k_1\) and \(A_1 A_2 \ldots A_{m-2} A_{m-1}\) correspond to \(A_1 A_2 \ldots A_{m-2} A_{m-1}\). The \(m-1\) partners do not have a \(k_1\) key. such a correspondence table is as follows:

\[
\alpha = (k_1, k_2, \ldots, k_{C(n,m-1)}), I = (k_1, k_2, \ldots, k_{C(n,m-1)})
\]

\[
k_1 \leftrightarrow A_1 A_2 \ldots A_{m-2} A_{m-1}
\]

\[
k_2 \leftrightarrow A_1 A_2 \ldots A_{m-2} A_m
\]
\begin{align*}
  k_{n-m+2} & \leftrightarrow A_1 A_2 \ldots A_{m-2} A_n \\
  k_{n-m+3} & \leftrightarrow A_1 A_2 \ldots A_{m-1} A_m \\
  k_{C(n,m-1)} & \leftrightarrow A_{n-m-2} A_{n-m-1} \ldots A_{n-1} A_n
\end{align*}

Where \( \alpha \) denotes a key vector and \( I \) denotes a set of each component of the extracted key vector \( \alpha \). For each \( A_j, j = 1,2, \ldots, n \), search for the string corresponding to the vector \( k_1, k_2, \ldots, k_{C(n,m-1)} \). If there is an \( A_j \), we put this component into the set \( \text{No}_{\text{key}}(A_j) \). \( \text{No}_{\text{key}}(A_j) \) represents a set of components that partner \( A_j \) does not have, that is, the subkey vector held by \( A_j \) does not contain all the elements in set \( \text{No}_{\text{key}}(A_j) \).

For example, for \( A_1 \), after searching from \( k_1, k_2, \ldots, k_{C(n,m-1)} \), it is found that the string corresponding to \( k_1, k_2, \ldots, k_{C(n,m-1)-C(n-1,m-1)} \) contains \( A_1 \), and these components can be sequentially classified into set \( \text{No}_{\text{key}}(A_j) \) to obtain a set:

\[ \text{No}_{\text{key}}(A_1) = \{ k_1, k_2, \ldots, k_{C(n,m-1)-C(n-1,m-1)} \} \]

This indicates that \( k_1, k_2, \ldots, k_{C(n,m-1)-C(n-1,m-1)} \) is not included in the key vector held by \( A_1 \). Then, the key vector \( \alpha = (k_1, k_2, \ldots, k_{C(n,m-1)}) \) is projected onto the set \( B(j) = I - \text{No}_{\text{key}}(A_j) \). \( (I - \text{No}_{\text{key}}(A_j)) \) represents the set \( I \) and the set \( \text{No}_{\text{key}}(A_j) \) difference set) to obtain the sub-key vector \( \beta_j \). As mentioned in the previous section, then

\[ \beta_j = (0,0, \ldots, 0, k_{C(n,m-1)-C(n-1,m-1)+1}, \ldots, k_{C(n,m-1)}, \alpha') \]

other subkeys are equally available.

It can be seen that the above sub-key space is obtained\[\text{[7]}\] by projecting from the key space of the \( C(n,m-1) \) dimension to the \( C(n,m-1) \) dimensional space, and the above-mentioned construction process guarantees that one component per \( m-1 \) person must be missing.

Let \( A_{i1} A_{i2} \ldots A_{i,m-2} A_{i,m-1} \leftrightarrow k_j \), and add another \( A_k A_k \neq A_{i1}, A_k \neq A_{i2}, \ldots, A_k \neq A_{i,m-2}, A_k \neq A_{i,m-1} \), then

\[
\begin{align*}
  A_{i1} A_{i2} & \ldots A_{i,m-2} A_{i,m-1} \leftrightarrow k_{i1}, \\
  A_{i1} A_k & \ldots A_{i,m-2} A_{k,m-1} \leftrightarrow k_{i2}, \\
  & \ldots \\
  A_{i1} A_{i2} & \ldots k A_{i,m-1} \leftrightarrow k_{i,m-2}, \\
  & \ldots \\
  A_{i1} A_{i2} & \ldots A_{i,m-2} A_k \leftrightarrow k_{i,m-1},
\end{align*}
\]

according to the relationship between the above components and the string one-to-one, obviously \( k_{i1}, k_{i2}, \ldots, k_{i,m-2}, k_{i,m-1} \neq k_j \). It can be seen that the original key vector can be obtained by adding a person’s subkey to the subkey held by any \( m-1 \) person (Table 2). Thus, it is ensured that any \( m \) individual can recover the entire key component \( k_1, k_2, \ldots, k_{C(n,m-1)} \).

| Key space vector | Collaborators who master the corresponding key vector |
|------------------|--------------------------------------------------|
| \( k_1 \)        | \( A_{1} A_{2} \ldots A_{m-2} A_{m-1} \)         |
| \( k_2 \)        | \( A_{1} A_{2} \ldots A_{m-2} A_{m} \)           |
| \( \vdots \)     | \( \vdots \)                                      |
| \( k_{n-m+2} \)  | \( A_{1} A_{2} \ldots A_{m-2} A_{n} \)           |
| \( \vdots \)     | \( \vdots \)                                      |
| \( k_{C(n,m-1)} \)| \( A_{n-m-2} A_{n-m-1} \ldots A_{n-1} A_{n} \) |

Table 2. SogouC1 data set Key sharing scheme
3. Traditional text segmentation
The traditional purpose of text segmentation is to evenly divide a large-sized text into several sub-texts of the same smaller size. It is convenient for file transfer by a file transfer system of network software such as e-mail or by using a removable storage medium such as a floppy disk for document transfer, since such applications generally have a limitation on the size of the text. This kind of segmentation is easy to implement, both from the purpose and the idea, and its algorithm is described as follows:

Step 1: Enter the split text and split parameters \( n \) (representing the number of copies of the text)
Step 2: Calculate the text size and record it as \( SIZE \) subsection
Step 3: Calculate the size of each sub-section \( size = SIZE / n \). If it is not divisible, let the extra part be merged into the \( n \)-th part.
Step 4: Point the pointer to the document header, read the size subsection to write the subdocument, and move the pointer down. After the loop is executed \( n - 1 \) times, the remaining subsections are written into the \( m \)-th subdocument.

The above is the whole process of traditional document segmentation. Its main idea is to divide the document evenly according to the specified number of copies. Usually the length of the first \( n - 1 \) subdocuments is size, and the remaining part is the last subdocument, and its length is \( SIZE - size(n - 1) \). The traditional document segmentation algorithm is shown in Figure 1.

4. Traditional security segmentation

4.1 Text Security Segmentation Thought
In addition to the traditional text segmentation function, the text segmentation algorithm to be discussed in this paper is mainly used in such a secret occasion: the text is divided into \( n \) sub-texts, which stipulates that any \( m \) sub-texts must be fully restored to restore the original text, and any less than \( m \) sub-texts cannot restore the original text. The advantage of this is that the original text can still be restored when any of the \( n - m \) sub-texts are damaged, and any less than \( m \) sub-texts are not enough to expose the original text. We take \( \text{int}(n/2) + 1 < m < n \) to ensure that the original text can only be restored if most of the subtexts appear.

4.2 Text security segmentation and key sharing
In the case where the text is divided into \( N \) paragraphs and the variables \( n \) and \( m \) are not known, \((n, m)\) satisfying the key sharing requirement is calculated by Formula:

\[ N \geq \text{MAX}(C(n, m - 1)), (n, m \in N^*) \]

If the original text is regarded as a key in the \( C(n, m - 1) \)-dimensional key space, the remaining \( N - \text{MAX}(C(n, m - 1)) \) paragraphs must not apply the key-sharing theory. In order to make the text segmentation and key distribution perfectly combined, the remaining paragraphs are added to the \( n \) subkeys. Therefore, using the idea of key distribution, this key is mapped to \( n \) sub-spaces of subspaces to obtain an \( n \)-subkey. That is, it gets its \( n \) sub-texts from the original text.
4.3 Text Security Segmentation Algorithm

According to the formula, the algorithm takes the document to be segmented as input, and after the text is segmented, \( N \) paragraphs are obtained.

\[
\begin{align*}
N & \geq \text{MAX}(C(n, m - 1)), (n, m \in N^+) \\
\text{int}(n/2) + 1 & < m < n
\end{align*}
\]

(1)

After calculating \( n \) and \( m \), after the key is divided, \( n \) sub-texts satisfying the requirements of text security segmentation are obtained, that is, the output of the algorithm. The algorithm is described as follows:

Step 1 Enter the document to be split.

Step 2 Calculate the parameters \( n \) and \( m \) according to the formula (1) (the meanings of \( n \) and \( m \) are the same as described above), and calculate \( key\_length = C(n, m - 1), \ subkey\_length = C(n - 1, m - 1) \).

Step 3 generates an \( key\_length \) long key array \( m[i] = i \); and allocates an \( n \times subkey\_length \) long subkey array by using a key sharing algorithm.

Step 4 Create \( n \) empty sub-texts, fill in the parameters \( n \), \( m \) and the corresponding sub-keys into the text header of the \( n \) sub-texts respectively (as shown in Figure 2), and fill in the corresponding paragraphs into the sub-text according to the sub-key information. And then go through all the paragraphs.

At this point, all the sub-texts are generated, and the original text segmentation process ends. The key sharing algorithm used in step 3 can be seen in the foregoing.

4.4 Text security recovery algorithm

As the inverse of text security segmentation, text security recovery is an important part of text security segmentation. With this algorithm, text security segmentation is complete. The text recovery algorithm takes \( m \) sub-texts as input, and after obtaining the parameters, creating a sub-document access table, and filling the original text, the original text is obtained. This is the output of the text security recovery algorithm. The algorithm is described as follows:

Step 1 Extract the parameters \( n \), \( m \) and key information according to the given subtext.

Step 2 Create a subtext access table based on parameters and keys.

Step 3 Create a blank text and fill in the corresponding text of the subtext according to the subtext access table.

The recovery work to this text is now complete. The subtext access table mentioned in step 2 is a custom data structure in the form of:

```c
struct position{
    char filepos;
    int detailpos;
};
```

filepos represents the sub-text number to which a particular paragraph in each group of the original text belongs, \( filepos \in \{1, 2, ..., m\} \), and filepos represents the specific position of this particular paragraph in the sub-text group. \( detailpos \in \{1, 2, ..., C(n - 1, m - 1)\} \). The text access table is used to restore the reference of the original text from the subtext. Since a paragraph of the original text can be placed in different positions of different subtexts, the text access table can ensure that each paragraph of the original text is only read from the subtext. It can be recovered in one operation,
avoiding repeated recovery from all its locations, which saves time for reading operations. The greater the redundancy information between sub-texts, the more efficient the text access table saves time.

The first group of the original document is divided as shown in Figure 3, and the opposite direction is the document recovery process:

![Figure 3. Document segmentation and recovery](image)

5. Comparison of algorithms
From the foregoing, the traditional text segmentation algorithm can be compared with the text segmentation algorithm proposed in this paper as follows:

1. In terms of purpose and function, the traditional text segmentation algorithm plays a role of “bigger and smaller, easy to transmit”. The text security segmentation is designed to solve the problem of secure distribution and storage of text, and is also compatible with the traditional text segmentation function, but this feature has relegated to a secondary position.

2. From the perspective of implementation, the traditional text segmentation algorithm is simply divided evenly and is easy to implement. The text security segmentation algorithm is implemented by steps such as key distribution, text grouping, and padding subtext, which is more complicated to implement.

3. From the perspective of efficiency, the time complexity of the traditional text segmentation algorithm is \(O(n)\), while the time complexity of the text segmentation algorithm is \(O(n^2)\). However, the text security segmentation algorithm sacrifices time and processing complexity to ensure the safe distribution of text storage. This is the price that all encryption algorithms must pay. When the number of divisions \(m = n\), that is, only when all the sub-texts work together to restore the original text, the text security segmentation algorithm is equivalent to the traditional text segmentation algorithm, and its time complexity is also reduced to \(O(n)\), which is the degradation phenomenon of the algorithm.

6. Algorithm application
The file security segmentation and recovery algorithm proposed in this paper can be used not only in carrier encryption, but also in the process of communication encryption after a slight improvement, so as to realize the confidentiality of all types of information distribution, not just in text form. For communication encryption, a specific communication model can be combined. A key resolver is designed to implement key distribution from a hardware perspective to improve the efficiency of secret communication.

A typical application of this algorithm is in the code storage of the software development process. The traditional approach is to set up a highly reliable code server to provide services such as code storage and sharing. If the code server is attacked (whether physical or logical), the security of the code will not be guaranteed. Meanwhile, the valuable results of software development will suffer huge losses. At this time, it is possible to adopt the idea of text security segmentation, and organize the code into a file and then use the idea of text security segmentation. If the code is organized into a file and
distributed to a reliable machine, ensure that the complete code can be recovered from any of the m machines, and less than m machines can not be recovered. This reduces the risk by A and the reliability by A times.

7. Conclusion
In this paper, the theory of vector space projection is used to explain the key sharing problem. From this perspective, the mathematical essence of key sharing and the mathematical model of it are established. This facilitates further discussion and research on this issue in theory, making it play a greater role in information security technology. In reality, any security system must encrypt the information. The key distribution algorithm and text security segmentation and recovery algorithm mentioned in this paper also have implicit encryption, but there is no formal encryption algorithm. In the actual allocation security system design, you can consider the use of specific encryption algorithms.

Acknowledgments
This work was financially supported by the State Grid Corporation of China Technology Project (5211XT17000F)

References
[1] Johannes Blömer. How to Share a Secret[J]. Communications of the Acm, 2011, 22(22):612-613.
[2] Blakley G R . Safeguarding cryptographic keys[C]// Afips. IEEE Computer Society, 1979.
[3] Kaijun T . Dynamic Secret Sharing Scheme Based On One Way Function[J]. Journal of China Institute of Communications, 1999.
[4] Jiuhui Z , Mingxin Z . Study on Threshold Cryptosystem Based on RSA[J]. Journal of Henan Science & Technology, 2015.
[5] Delfs H . Introduction to Cryptography, Principles and Applications[J]. Information Security and Cryptography Texts and Monographs, 2001.
[6] Wesolowski T A , Wang Y A . Recent progress in orbital-free density functional theory /[M]/ Recent progress in orbital-free density functional theory. World Scientific, 2013.
[7] Haifeng L . The Research and Realization of Algorithms of Decomposing Data Key[J]. Computer Engineering & Applications, 2004, 40(19):78-81.
[8] Fan Y L, Jiao Z Y. Algorithm about safe file-splitting based on theory of key assignment[J]. Computer Engineering & Design, 2008.
[9] Ying Y U , School C B , Radioamp D , et al. Study on Safe Storage Technology of Cloud Computing Service Date[J]. Agricultural Science & Technology and Equipment, 2016.