# Improving Chinese Storing Text Retrieval Systems' Security Via a Novel Maximal Prefix Coding

D.Y. Long [a,c], W.J. Jia [a], M. Li [b], P.O. Au [a], K. Su [c]

[a] *Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong SAR, PRC; dylong@cs.cityu.edu.hk*
[b] *School of Computing, National University of Singapore*
[c] *Department of Computer Science, Zhongshan University, Guangzhou 510275, PRC; dylong25112002@yahoo.com*

**Abstract:** As we have seen that Huffman coding has been widely used in data, image, and video compression. In this paper novel maximal prefix coding is introduced. Relationship between the Huffman coding and the optimal maximal prefix coding are discussed. We show that all Huffman coding schemes are optimal maximal prefix coding schemes and that conversely the optimal maximal prefix coding schemes need not to be the Huffman coding schemes. Moreover, it is proven that, for any maximal prefix code $C$, there exists an information source $I = (\sum, P)$ such that $C$ is exactly a Huffman code for $I$. Therefore, it is essential to show that the class of Huffman codes is coincident with one of maximal prefix codes. A case study of data compression is also given. Comparing the Huffman coding, the maximal prefix coding is used for not only statistical modeling but also dictionary methods. And it is good at applying to a large information retrieval system and improving its security.

**Keywords:** coding, data compression, Huffman coding, maximal prefix coding

## 1. Introduction

Huffman codes [1, 5-9] have been widely used in data, image, and video compression [3, 5, 7-9, 15, 20-21]. For instance, the Huffman coding is used to compress the result of a quantitative stage in JPEG [15]. Huffman codes belongs into a family of codes with a variable length not a fixed length. That means that individual letter which makes a file is encoded with bit sequences that have distinct length. This characteristic of the code words helps to decrease the amount of redundancy [6, 14, 16, 21] in message data

i.e. it makes data compression possible. Decreasing of redundancy in data by Huffman codes is based on the fact that distinct letters have distinct probabilities of incidence. This fact helps to create such code words, which really contribute to decreasing of redundancy i.e. to data compression. Generally, letters with higher probabilities of incidence are encoded with shorter code words, while letters with lower probabilities of incidence are encoded with longer code words. One of our motivations in conducting this research is to design a secure coding scheme with high efficiency. A novel coding based on maximal prefix codes is introduced. We discuss relationships between Huffman coding and maximal prefix coding schemes. It is shown that all Huffman coding schemes are maximal prefix encoding schemes and have the shortest average code word length among all maximal prefix coding schemes and that conversely optimal maximal prefix coding schemes need not to be Huffman coding schemes. Moreover, we prove that for any maximal prefix code *C* over a finite alphabet $\Sigma$, there exists an information source $I = (\Sigma_1, P)$ so that *C* is exactly a Huffman code for $I = (\Sigma_1, P)$. Therefore, it is essential to show that the class of Huffman codes is coincident with the class of maximal prefix codes.

Some basic concepts and notations of coding theory [2,10,13,17-19] are first given. An *alphabet* $\Sigma$ is a finite set and $\Sigma^*$ is the set of all finite length words formed from the letters of $\Sigma$ (include *empty word* $\lambda$) and $\Sigma^+ = \Sigma^* - \{\lambda\}$. A subset $C \subseteq \Sigma^+$ is called a *code* [2,8] (sometimes called an *uniquely decipherable code* [4,16]) if, for all words $x_{i_1}, x_{i_2}, \ldots, x_{i_n}, x_{j_1}, x_{j_2}, \ldots, x_{j_m} \in C$, the equality $x_{i_1} x_{i_2} \ldots x_{i_n} = x_{j_1} x_{j_2} \ldots x_{j_m}$ implies m = n, $x_{i_1} = x_{j_1}$ and $x_{i_k} = x_{j_k}$, $k = 1,\ldots,n$. And any subset $C \subseteq \Sigma^+$ is called a *language* but not a code [2]. A code $C \subseteq \Sigma^+$ is called a *maximal code*, if for any $x \in \Sigma^* - C, C \cup \{x\}$ is not a code. A *word* (or an *element*) in a code *C* is called a *code word*. For each word $w \in \Sigma^*$, let $l(w)$ denote the word length of *w*. Maximal codes reflect extreme properties of codes when they are considered as communication tools. In some sense, maximal codes make use of the whole capacity of the transmission channel [8]. A code $C \subseteq \Sigma^+$ is called a *prefix* (or *instantaneous*) *code* [2,4,16] if $C \bigcap C\Sigma^+ = \varnothing$, that is if no code word is a prefix of any other code word. A prefix code $C \subseteq \Sigma^+$ is called a *maximal prefix code*, if for any $x \in \Sigma^* - C, C \cup \{x\}$ is not a prefix code.

An *information source* [16] is an ordered pair $I=(\Sigma, P)$, where $\Sigma = \{s_1, s_2, ..., s_q\}$ is a *source alphabet* and $P$ is a *probability* law that assigns to each element $s_i \in \Sigma$ a probability $P(s_i)$. For an information source $I = (\Sigma, P)$, an ordered pair $(C, f)$ is said to be a *coding, if C is a code and f:* $\Sigma \rightarrow C$ *is a one-to-one mapping.* The *average code word length* of $(C, f)$ is $\sum_{i=1}^{q} l(f(s_i))P(s_i)$, where $l(f(s_i))$ denotes the length of the code word $f(s_i) \in C$. A coding is said to be *prefix, if the corresponding code of the coding is a prefix code.* A coding is said to be *maximal prefix, if the corresponding code of the coding is a maximal prefix code.* A *prefix code* is called *optimal* for a source alphabet $\Sigma$ if no other prefix code has a smaller average code word length. A *maximal prefix code* is called *optimal* for a source alphabet $\Sigma$ if no other maximal prefix code has a smaller average code word length. Huffman published a method for constructing highly an efficient coding (or encoding) for finite source alphabets in 1952 [6]. This method is known as the *Huffman coding* (or the *Huffman's algorithm*) and *the corresponding code of a Huffman coding* (or *the code generated by the Huffman's algorithm*) is said to be the *Huffman code* [4,14-15].

## 2. Optimal Maximal Prefix Coding Schemes

First, it is proven that existence of the optimal maximal prefix coding for finite source alphabets and that all Huffman coding schemes are optimal maximal prefix coding schemes.

**Theorem 1** *Let* $I = (\Sigma, P)$ *be a finite information source. Then all Huffman coding schemes for* $\Sigma$ *are maximal prefix coding schemes and have the shortest average code word length among all maximal coding schemes for* $\Sigma$*. Conversely, the maximal prefix coding schemes need not be the Huffman coding schemes.*

In order to show Theorem 1, we first give the following Lemmas.

**Lemma 1** (McMillan's Theorem, [4,16]) *Let* $C = \{c_1, c_2, ..., c_q\}$ *be a code over r letters alphabet and let* $n_i$ *denote the length of code word* $c_i$*. Then the code word lengths* $n_1, n_2, \ldots, n_q$ *must satisfy Kraft's inequality* $\sum_{i=1}^{q} r^{-n_i} \leq 1$.

**Lemma 2** (Kraft's Theorem, [4,16]) *There exists a prefix code* $C = \{c_1, c_2, ..., c_q\}$ *over r letters alphabet, with code word lengths* $n_1$ , $n_2$ ,…, $n_q$, *if and only if these lengths satisfy Kraft's inequality* $\sum_{i=1}^{q} r^{-n_i} \le 1$.

**Lemma 3** *If there exists a code with code word lengths* $n_1$ , $n_2$ ,…, $n_q$, *then prefix codes must also exist with these same code word lengths* $n_1$ , $n_2$ ,…, $n_q$,.

**Lemma 4** *Any r-ary* ($r \ge 2$) *Huffman code for a given finite information source alphabet is exactly corresponding to a complete r-ary tree.*

**Proof:** When $r = 2$, the conclusion is clearly. If $r \ge 3$, we may not have a sufficient number of letters of the source alphabet so that we can combine them $r$ at a time as we have done for $r = 2$. But in such a case, we can add *dummy* letters to the end of the information source alphabet and assign source probability zero to the added letters. The dummy letters are inserted to fill the tree. Since at each stage of the reduction, the number of letters is reduced by $r - 1$, we want the total number of letters to be $1 + k(r - 1)$, where $k$ is the number of levels in the tree. Therefore, we add enough dummy letters so that the total number of information source alphabet is of $r + k(r - 1)$. The details of $r$-ary ($r \ge 3$) Huffman codes can be referred to References [4] ( p.92-94) and [14] (p.67-69). #

**Proof of Theorem 1:** Since the r-ary ($r \ge 2$) Huffman code generated by the Huffman algorithm for any given finite information source alphabet is exactly corresponding to a complete r-ary tree (by Lemma 4). Also, it is known that a complete r-ary tree has to be corresponding to a maximal prefix code over the alphabet with $r$ letters (see [2], p.85-88) Therefore, all Huffman codes are maximal prefix codes, and all Huffman coding schemes are maximal prefix coding schemes. Now, let $I = (\Sigma, P)$ be an information source, and let $(C, f)$ be any maximal prefix coding scheme for $\Sigma$, where $C = \{c_1, c_2, \dots, c_q\}$ is a maximal prefix code with code word lengths $n_1, n_2, \dots, n_q$. By Lemma 3, there must exist a prefix code $C_1 = \{d_1, d_2, \dots, d_q\}$and a prefix coding scheme ($C_1$, g) for $\Sigma$ such that $C_1 = \{ d_1, d_2, \dots, d_q \}$has the same code word lengths $n_1$ , $n_2$ ,…, $n_q$ as the maximal prefix code $C$. Again, let ($C_h$, $h$) be a Huffman coding scheme for $\Sigma$ with code word lengths $m_1$ , $m_2$ ,…, $m_q$. By the Huffman algorithm [4,6,12], we know that

$$\sum_{i=1}^{q} l(h(s_i))P(s_i) = \sum_{i=1}^{q} m_i P(s_i) \le \sum_{i=1}^{q} l(g(s_i))P(s_i) = \sum_{i=1}^{q} n_i P(s_i).$$

Since

$$\sum_{i=1}^{q} l(f(s_i))P(s_i) = \sum_{i=1}^{q} n_i P(s_i) = \sum_{i=1}^{q} l(g(s_i))P(s_i),$$

thus

$$\sum_{i=1}^{q} l(h(s_i))P(s_i) = \sum_{i=1}^{q} m_i P(s_i) \le \sum_{i=1}^{q} l(f(s_i))P(s_i) = \sum_{i=1}^{q} n_i P(s_i).$$

This shows that the average code word length of the maximal prefix coding scheme ($C$, $f$) is equal to or greater than the average code word length of a corresponding Huffman coding scheme ($C_h$, $h$). That is, all Huffman codes are optimal maximal prefix codes.

Conversely, all optimal maximal prefix codes need not to be the Huffman codes. For example, for an given information source $I = (\sum, P)$ where $\sum$ = {A, B, C, D, E} and $P$ ={0.50,0.25,0.14,0.09,0.02}. We easily deduce all the four Huffman codes $C_1$ = {1,01,001,0001,0000}, $C_2$ = {0,11,101,1001,1000}, $C_3$ = {1,00,011,0100,0101}, and $C_4$ = {0,10,111,1100,1101} in Table 1. Clearly, the maximal prefix code $D$ in Table 1 is no Huffman code. #

**Table 1:** A maximal prefix coding different from the Huffman coding

| Source Letter | Probability | Huffman Code $C_1$ | Huffman Code $C_2$ | Huffman Code $C_3$ | Huffman Code $C_4$ | Maximal Prefix Code $D$ |
|---|---|---|---|---|---|---|
| A | 0.50 | 1 | 0 | 1 | 0 | 0 |
| B | 0.25 | 01 | 11 | 00 | 10 | 11 |
| C | 0.14 | 001 | 101 | 011 | 111 | 100 |
| D | 0.09 | 0001 | 1001 | 0100 | 1100 | 1011 |
| E | 0.02 | 0000 | 1000 | 0101 | 1101 | 1010 |

Although Huffman codes are a proper subclass of maximal prefix codes in general, the following Theorem 2 shows nearly relations between the Huffman codes and the maximal prefix codes. In a sense, it illustrates that the class of maximal prefix codes is coincident with the one of Huffman codes.

**Theorem 2** *If $C \subseteq \Sigma^+$ is any maximal prefix code, then there must exist a suitable information source $I = (\Sigma_1, P)$ such that $C$ is a Huffman code for $I = (\Sigma_1, P)$.*

To prove Theorem 2, we first give the following two Lemmas.

**Lemma 5** [2,10,13] *Let $C \subseteq \Sigma^+$ be a finite maximal prefix code. Then for every $a \in \Sigma$ there exists some positive integer m such that $a^m \in C$.*

**Lemma 6** *Let* $C \subseteq \Sigma^+$ *be a finite maximal prefix code,* $\Sigma = \{a_1, a_2, ..., a_k\}$, *and* $C = \{w_1, w_2, ..., w_n\}$ *with* $l(w_1) \geq l(w_2) \geq l(w_3) \geq \ldots \geq l(w_n)$, *where* $l(w)$ *denotes the length of the word* $w \in \Sigma^+$. *Then we have that* $l(w_1) = l(w_2) = l(w_3) = \ldots = l(w_k)$ *and* $w_1 = wa_1, w_2 = wa_2, ..., w_k = wa_k$ *for some* $w \in \Sigma^*$.

**Proof:** By Lemma 5, we know that $n \geq k$. Without loss of generality, suppose that $l(w_1) = l(w_2) = l(w_3) = \ldots = l(w_{j-1})$, $l(w_j) < l(w_1)$ and $w_1 = wa_1, w_2 = wa_2, ..., w_{j-1} = wa_{j-1}$. Now, consider the word $wa_j$. Since $wa_j$ has not the prefix relation with all words in $C$, $C \cup \{wa_j\}$ is a prefix code. This contradicts with $C$ being a maximal prefix code. Therefore, we show that $l(w_1) = l(w_2) = l(w_3) = \ldots = l(w_k)$ and $w_1 = wa_1, w_2 = wa_2, ..., w_k = wa_k$ for some $w \in \Sigma^*$. #

**Proof of Theorem 2:** We only show that Theorem 2 is true for $\Sigma = \{0,1\}$. Similarly, it is easy to prove that Theorem 2 is also true for the number of letters in $\Sigma$ to be greater than 2.

Let $C \subseteq \{0,1\}^+$ be a maximal prefix code. Let $C = \{w_1, w_2, ..., w_n\}$ with $l(w_1) \geq l(w_2) \geq l(w_3) \geq \ldots \geq l(w_n)$. First, we define the $i$th letter of a finite alphabet $\Sigma_1$ as $a_i$, $i$ = 1, 2,…, n, namely $\Sigma_1$ = $\{a_1, a_2, a_3, ..., a_n\}$ and assign each letter $a_i \in \Sigma_1$ to a probability $P(a_i)$, $i$ = 1, 2, …, $n$ such that $P(a_1) \geq P(a_2) \geq ... \geq P(a_n)$. Therefore, we can construct the Huffman coding schemes for a given information source alphabet $\Sigma_1$. Although such as the Huffman codes are not uniquely in general, they are equivalent [1] (with respect to average code word length). That is, all Huffman codes for a given information source $(\Sigma_1, P)$ have the same average code word length. Given a set of different probability distribution $P'(a_i)$ for $i$ = 1, 2,…, $n$, we easily construct different Huffman coding schemes, and consequently different Huffman codes will be also obtained.

Next, we will prove that there is a suitable set of probability distribution $P(a_i)$ for $i$ = 1, 2, …, $n$ such that a Huffman code corresponding to information source $I = (\Sigma_1, P)$ is exactly equal to the maximal prefix code $C \subseteq \{0,1\}^+$. By induction on the number $n$ of the code words in $C$, we will show that the above

statement is true. According to the graphic representation of maximal prefix codes ([2], p.85-88), we know that maximal prefix code $C$ over {0,1} associates with a unique complete binary tree. And the leaves of the tree represent the code words in $C$. Since $C$ is a maximal prefix code over {0,1}, by Lemma 6, $n \geq 2$. When $n = 2$, by Lemma 5, then $C = \{0, 1\}$. Clearly the conclusion is true. Assume that the conclusion is true for $n$-1. That is, if $C' = \{u_1, u_2, ..., u_{n-1}\}$ is a maximal prefix code over {0, 1} then there exists a set of probability distribution $P'(b_i)$ for i = 1, 2,…, $n$-1 such that $C'$ is a Huffman code for information source $(\{b_1, b_2, ..., b_{n-1}\}, P')$. Since $C = \{w_1, w_2, ..., w_n\}$ is a maximal prefix code over {0, 1} and $l(w_1) \geq l(w_2) \geq l(w_3) \geq \ldots \geq l(w_n)$, by Lemma 6, then $l(w_1) \geq l(w_2)$ and $w_1 = x0$, $w_2 = x1$. Now, consider $C'' = \{x, w_3, w_4, ..., w_n\}$. According to the graphic representation [2] of the maximal prefix code $C$, we easily follow that $C''$ is also a maximal prefix code over {0, 1}. By induction hypotheses, there is an information source $I_1 = (\Sigma', P')$ such that $C'' = \{x, w_3, w_4, ..., w_n\}$ is a Huffman code for $I_1$. Extending the alphabet $\Sigma' = \{b_1, b_2, ..., b_{n-1}\}$ and $P'$ to $\Sigma_1 = \{b_1, b_2, ...b_{n-2}, a_{n-1}, a_n\} = \{a_1, a_2, a_3, a_4, ..., a_n\}$ and $P$ respectively, such that $P(a_i) = P(b_i)$, $1 \leq i \leq n - 2$, and $P(a_{n-1}) + P(a_n) = P'(b_{n-1})$, $0 < P(a_n) \leq P(a_{n-1})$, we immediately verify that $C$ is exactly a Huffman code corresponding to the information source $I = (\Sigma_1, P)$. This completes the proof of theorem 2. #

## 3. Case Study

In this section, case studies are given for maximal prefix coding schemes. From the definition of prefix codes, we easily follow that the following set

$$C = \{010, 011, 0^3 10^3, 0^3 1010, 0^3 1^3 0, 0^3 1^2 0^2, 0^3 10^2 1, 0^3 101^2, 0^3 1^4, 0^3 1^2 01, 0^5 10^5, 0^5 1010^3\}$$

is a prefix code over the alphabet {0,1}. By Proposition 2.6 in [17], we immediately construct a maximal prefix code $M = C \cup (\{0,1\}^{11} - C\{0,1\}^*)$. Clearly, the prefix code $C$ contains only 12 code words. However, the maximal prefix code $M$ contains

$2^{11}-(2\times 2^8+8\times 2^4)+12=2048-(512+128)+12=1420$ code words. Similarly, consider a subset *D* of the prefix code $C=\bigcup_{m=1}^{\infty}0^m1\Sigma^m$ over alphabet $\Sigma=\{0,1\}$:

$D=\{010,0^310^3,0^310^21,0^510^5,0^510^41,0^510^31^2,0^510^21^3,\overbrace{0^710^7,0^710^61,...,0^711^7}^{2^3},$

$\overbrace{0^{11}10^{11},...,0^{11}11^{11}}^{2^4}$, $\overbrace{0^{13}10^{13},...,0^{13}11^{13}}^{2^5}$, $\overbrace{0^{17}10^{17},...,0^{17}11^{17}}^{2^6}$,

$\overbrace{0^{19}10^{19},...,0^{19}11^{19}}^{2^7}$, $\overbrace{0^{23}10^{23},...,0^{23}11^{23}}^{2^8}$, $\overbrace{0^{29}10^{29},...,0^{29}11^{29}}^{2^9}$, $0^{37}10^{37}\}$.

*D* contains 1024 code words. Where the length of code words in *D* are 3, 7, 11, 15, 23, 27, 35, 39, 47, 59, and 75, respectively. And the number of code words in *D* is $2^0+2^1+2^3+2^4+2^5+2^6+2^7+2^8+2^9+1=2^{10}=1024$. By $M=D\bigcup(\{0,1\}^{47}-D\{0,1\}^*)$, we easily generate a maximal prefix code *M*. But *M* contains

$$2^{75}-(2^{72}+2\times 2^{68}+2^2\times 2^{64}+2^3\times 2^{60}+2^4\times 2^{52}+$$
$$2^5\times 2^{48}+2^6\times 2^{40}+2^7\times 2^{36}+2^8\times 2^{28}+2^9\times 2^{16})+2^{10}>2^{75}-2^{74}+2^{10}>2^{74}$$

code words. Note that we take the prefix code *D* as only a simple example. We show that it is easy to generate a maximal prefix code with enough code words.

In addition, by the above maximal prefix code *M* containing many code words ($>2^{74}$), we can easily construct many different prefix coding schemes. Suppose that the information source alphabet is the alphabet of 128 ASCII characters, then the number of all different prefix coding schemes will be $C_{2^{74}}^{128}=2^{74}(2^{74}-1)(2^{74}-2)...(2^{74}-127)/128!(>2^{74}(2^{74}-1)...(2^{74}-116)>2^{8510})$.

Furthermore, the prefix code *C* may be a prefix code over several alphabets completely different from the plaintext alphabet $\Sigma$. As we have known that the prefix code *C* has not concerned with crypt-analytic tools which include frequency distribution, diagram study, index of coincidence, searching for repeated patterns, and study of probable letters. Since the prefix code *C* is randomly produced for a given alphabet. Therefore, the file encoded by a prefix coding scheme can prevent the basic attacks based on traditional substitutions and transpositions (or permutations). In order to strengthen *Confusion* and *Diffusion* of the

file, we should change the prefix code $C$ and the coding scheme $f$ and repeat $f$ several times periodically. For example, let $\Sigma = \{a, b, c, d\}$, the following Table 3 gives the two prefix codes $C_1$, $C_2$, and the corresponding coding schemes $f$ and $g$. Suppose that Alice want to send the plaintext message $M$: ***abc*** to Bob. Alice encrypt the message M by the secret keys ($C_1$, $ff$). Then Bob will receive the encoded file $M_1$: ***f*(*f*(*abc*)) = *f*(*abaabbcd*) = *abaabbababaabbaabbcdccdd***. If Alice sends the plaintex file $M$ : ***abcd*** by the coding scheme ($C_2$, $g$), then Bob will receive the same encoded file $M_2$: ***g*(*abcd*) = *abaabbababaabbaabbcdccdd*** as $M_1$. Similarly, Alice can send the encrypted message

***f*(*f*(*f*(…(*abc*)…)))** or ***f*(*g*(*f*(…(*abc*)…)))**

**Table 3:** Two coding schemes

| Letter | Prefix code $C_1$ | Prefix code $C_2$ | $f$ | $g$ |
|---|---|---|---|---|
| a | ab | abaabb | a→ab | a→abaabb |
| b | aabb | ababaa | b→aabb | b→ababaa |
| c | cd | bbaabb | c→cd | c→bbaabb |
| d | ccdd | cdccdd | d→ccdd | d→cdccdd |

to Bob. As we have seen, $M_1$ is the same as $M_2$. That is Bob received the encoded file ***abaabbababaabbaabbcdccdd***, which of two possible meaningful plaintext file ***abc*** and ***abcd***. If a coding scheme encodes two or more different file to the same file, an interceptor cannot determine which of two or more possible meaningful plaintext file is the authentic one. When the number of such multiple-file encoded is high, the coding is also more secure. On the other hand, if an interceptor obtained the encoded ***abaabbababaabbaabbcdccdd*** and did not know the coding schemes ($C_1$, $ff$) and ($C_2$, $g$), then it is impossible to get the original file ***abc*** or ***abcd***. One of the greatest difficult problems is that we have not decided the code $C$ when we divide the encoded ***abaabbababaabbaabbcdccdd*** into all possible words by a brute force attack. For example, if we divide the file ***abaabbababaabbaabbcdccdd*** into the words $c_1$ = ***ab***, $c_2$ = ***aabb***, $c_3$ = ***abab***, and $c_4$ = ***aabbaabbcdccdd***, then it is easy to verify that the language $C = \{c_1, c_2, c_3, c_4\}$ is not a code, since the word ***abababab*** is able to express as $c_1\ c_1\ c_1\ c_1$ and $c_4\ c_4$. Generally, if an interceptor obtained the encoded $M'$ which is encoded by basic substitutions and transpositions, then the brute force attack to $M'$ is common efficient attack. But an interceptor want to break the file $M''$ encoded by the coding ($C$, $f$) in the brute force attack is not successfully. One of the reason is there is no an efficient algorithm to decide whether or not a finite language over finite alphabets [2] is a code and that the set of some words

which divide into the file $M''$ need not to be a code. For example, let the length of the encoded files $M'$ and $M''$ be *N*. Since $M'$ is encoded by a block code, $M'$ can be divided into at most *N* block codes. But $M''$ can divide into at most $2^{N-1}$ variable length codes (the details can be referred to the following Theorem 3). Therefore, when *N* is a greater natural number it is impossible for an interceptor to tries all possible solutions (all possible prefix codes) by a brute force attack.

**Theorem 3** *Let $M''$ be a file encoded by the coding scheme (C, f) and the length of $M''$ be N. Then $M''$ is encoded by at most $2^{N-1}$ prefix codes instead of block codes.*

*Proof:* Let $M'' = a_1a_2a_3...a_N$, $a_i \in \Sigma$, $i$=1, …, *N*. Suppose that we divide $M''$ into $c_1c_2...c_k$ i.e., $M'' = c_1c_2...c_k$, such that $\{c_1, c_2, ..., c_k\}$ is a possible code. Since the length $l(c_j)$ of $c_j$ satisfy $1 \le l(c_j) \le N$, $c_j$ (j=1, …, k) are able to take all the sub-words of the word $M''$ [1, 4]. Let the number of the set $\{c_1, c_2, ..., c_k\}$ satisfying with $M'' = c_1c_2...c_k$ be $D_N$. Then the number of the set $\{c_1, c_2, ..., c_k\}$ such that $a_2a_3...a_N = c_1c_2...c_k$ is $D_{N-1}$. Repeating the above discussion, we have that the number of the set $\{c_1, c_2, ..., c_k\}$ with $a_ia_{i+1}...a_N = c_1c_2...c_k$ is $D_{N-i}$. According to the choice of $c_j$ (j=1, …, k), we easily obtain that $D_N = D_{N-1} + D_{N-2} + \dots + D_{N-(N-1)} + 1$. Therefore, $D_N = D_{N-1} + D_{N-2} + \dots + D_{N-(N-1)} + 1 = D_{N-2} + \dots + D_{N-(N-1)} + 1 + (D_{N-2} + \dots + D_{N-(N-1)} + 1) = 2(D_{N-2} + \dots + D_{N-(N-1)} + 1) = 2^2(D_{N-3} + \dots + D_{N-(N-1)} + 1) = 2^3(D_{N-4} + \dots + D_{N-(N-1)} + 1) = \dots = 2^{N-2}(D_{N-(N-1)} + 1) = 2^{N-2}(D_1 + 1) = 2^{N-1}$. Note that it is easy to verify that $D_1 = 1$ and $D_2 = 2$. #

Fortunately, an efficient algorithm generating a prefix code or maximal prefix code is given. Denote the longest length of code words in the maximal prefix code *C* as max_length, the total number of code words in *C* as total_code, and the number of code checking as test_code.

**Algorithm:** Generating a maximal prefix code *C*

Input: An alphabet $\Sigma$

Output: A maximal prefix code *C*.

Step 1: Set *C* = empty, the number of code words in *C*, *m* = 0, and the code checking times, *t* =0.

Step 2: Repeat following steps 3 through 6 until *m* > total_code or *t* > test_code, go to Step 7.

Step 3: Randomly generate an integer $n$ as the length of word, $1 \le n \le$ max_length; $t = t + 1$.

Step 4: Randomly select a word $w$ of length n from $\Sigma^*$.

Step 5: Compare $w$ with each word in $C$, if there exist prefix relations between $w$ and a word in $C$, then go to step 3. Otherwise continue.

Step 6: $C = C + w$ (add the word $w$ to $C$), $m = m + 1$, $t = t + 1$, go to step 3.

Step 7: Generate the sets $D = \Sigma^{\max\_length}$ and $E = \{xy | (\forall x \in C)(\forall y \in \Sigma^*) l(xy) = \text{max\_length}\}$, and then calculate the complement $D - E$ of $E$ with respect to $D$, i.e., $D - E = \{x | x \in D, x \notin E\}$.

Step 8: $C = C + (D - E)$(Add $D - E$ to $C$).

First, we are able to generate a prefix code efficiently. The fact immediately follows from Theorem 4 below.

**Theorem 4** *Let* $\Sigma$ *be a finite alphabet and n be any positive integer. The above* ***algorithm*** *must generate a prefix code C containing n code words in polynomial time of n (i.e. O($n^3$)).*

**Proof:** Consider a word $w = a_1 a_2 ... a_n$ of length n. Since all proper prefix of w are $w_0 = 1, w_1 = a_1, w_2 = a_1 a_2, ..., w_{n-1} = a_1 a_2 ... a_{n-1}$, they are total n words. At first, let $C = \{w\}$, then we obtain a prefix code $\{w, w_1\}$ from the prefix code $C = \{w\}$ by at most n steps (comparing $w_1$ with w in $C$, we add the word $w_1$ to $C$ if there is no prefix relations between $w_1$ and $w$.). Similarly, we get a prefix code $\{w, w_1, w_2\}$ from the prefix $\{w, w_1\}$ by at most $2n$ steps. Continuing the above discussion, we have a prefix code $\{w, w_1, w_2, \ldots, w_{n-1}\}$from the prefix code $\{w, w_1, w_2, \ldots, w_{n-2}\}$by at most $(n-1)n$ steps. Therefore, we generated a prefix code containing $n$ code words from $C$ by at most $n + 2n + 3n + ... + (n-1)n = (n^3 - n^2)/2$ steps. This shows that the above **algorithm** must generate a prefix code $C$ containing $n$ code words in polynomial time. #

According to the above **algorithm** and Theorem 4, we easily obtain the following corollary 1.

**Corollary 1** *There is an efficient algorithm generating a maximal prefix code.*

Similarly, we have

**Theorem 5** *Let* $\Sigma$ *be an alphabet and L be any finite language over* $\Sigma$. *Then we can efficiently decide whether or not L is a prefix code over* $\Sigma$.

By Theorem 5, we easily get

**Corollary 2** *Let* $\Sigma$ *be an alphabet and L any finite language over* $\Sigma$. *Then we can efficiently decide whether or not L is a maximal prefix code over* $\Sigma$.

Again, from a theoretical and a practical point of view, data compression essentially improves the security effect of coding schemes [12]. As the Huffman coding, we apply the maximal prefix coding to data compression. Different compression ratios between the maximal prefix coding and the Huffman coding are given.

For example, we will encode the following file M:

**STATUS REPORT ON THE FIRST ROUND OF THE DEVELOPMENT OF THE ADVANCED ENCRYPTION STANDARD**

According to Table 4, we easily calculate that the average code word length of the block code $C_1$ is 5 bits/symbol, and that the average code word length of the Huffman code $C_2$ is 342/87 bits/symbol. Therefore, the encoded file by the block code $C_1$ and the Huffman code $C_2$ will take up $87\times5 = 435$ bits and $87\times342/87 = 342$ bits respectively. Therefore, the compression ration is 435/342 = 1.27:1. By Theorem 1, we can give the optimal maximal prefix coding schemes such as $C_3$ in Table 4. But the file encoded by them will also need 342 bits.

Additionally, in a way without statistical methods we will encode the file M by a maximal prefix code different from the Huffman code. First a maximal prefix code is generated as follow:

$$C = \{0,101,0001,1001,00011,10011,01011,11011,00111,10111,01111,11111\}$$

Clearly, the code $C$ is not a Huffman code. By the following Table 3, we will calculate that the encoded file will take up $1\times13+3\times3+4\times2+4\times1+5\times8 = 74$ bits. Therefore, the compression ratio is 435/74 = 5.87:1. It easily follows that there exist a lot of various maximal prefix codes with the completely different compression ratio such as the compression ratio of the maximal prefix code in Table 4 is 435/114 = 3.82:1.

Especially, the maximal prefix coding is good at encoding the Chinese or the Japanese text such that readable text can be produced only by means of the supplied software.

**Table 4:** A Huffman coding scheme for the file M

| Letters | Probability | Block Code $C_1$ | A Huffman Code $C_2$ | No Huffman Code $C_3$ |
|---|---|---|---|---|
| (Space) | 13/87 | 00000 | 010 | 101 |
| T | 10/87 | 00001 | 101 | 010 |
| E | 9/87 | 00010 | 110 | 001 |
| N | 7/87 | 00011 | 0000 | 1111 |
| O | 7/87 | 00100 | 0010 | 1101 |
| D | 6/87 | 00101 | 0111 | 1000 |
| R | 6/87 | 00110 | 0110 | 1001 |
| A | 4/87 | 00111 | 1001 | 0110 |
| S | 4/87 | 01000 | 1110 | 0001 |
| C | 3/87 | 01001 | 00010 | 11101 |
| F | 3/87 | 01010 | 00011 | 11100 |
| H | 3/87 | 01011 | 00111 | 11000 |
| P | 3/87 | 01100 | 00110 | 11001 |
| I | 2/87 | 01101 | 10000 | 01111 |
| U | 2/87 | 01110 | 10001 | 01110 |
| V | 2/87 | 01100 | 11110 | 00001 |
| L | 1/87 | 10001 | 111110 | 000001 |
| M | 1/87 | 10011 | 1111110 | 0000001 |
| Y | 1/87 | 10101 | 1111111 | 0000000 |

**Table 5:** A maximal prefix coding without the Huffman coding

| Words of the File | A Coding |
|---|---|
| (Space) | 0 |
| THE | 101 |
| OF | 1000 |
| DEVELOPMENT | 1001 |
| STANDARD | 11000 |
| ADVANCED | 11001 |
| STATUS | 11010 |
| REPORT | 11011 |
| FIRST | 11100 |
| ROUND | 11101 |
| ENCRYPTION | 11110 |
| ON | 11111 |

**Table 6:** A maximal prefix coding without the Huffman coding

| Words of the File | A Coding |
|---|---|
| DEVELOPMENT | 000 |
| ENCRYPTION | 001 |
| STANDARD | 011 |
| ADVANCED | 100 |
| STATUS | 110 |
| REPORT | 0100 |
| FIRST | 0101 |
| ROUND | 1011 |
| THE | 1111 |
| OF | 1010 |
| ON | 11100 |
| Space | 11101 |

For instance, we will encode the following Chinese text M':

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 文 | 本 | 壓 | 縮 | 將 | 產 | 生 | 好 | 的 | 加 | 密 | 文 | 本 |

By the following Tables 7 and 8, we easily get the encoded files $M_1$ and $M_2$ :

**$M_1$ : 011100010000001111011111100110110101011011100**

**$M_2$ : 10100100010000100000100000010000000100000000010000000000101**

Furthermore, we calculate that the encoded file $M_1$ will take up 7×3+6×4 = 45 bits. Therefore, the compression ratio is 52/45 = 1.15:1. But the encoded file $M_2$ will take up 68 bits. Although $M_2$ will not produce compression than the original file M', it is very difficult to decode the file $M_2$ if we have not known the Table 8 before.

**Table 7:** A maximal prefix coding

| Words of the File | A Coding |
|---|---|
| 文 | 011 |
| 本 | 100 |
| 壓 | 010 |
| 縮 | 000 |
| 將 | 001 |
| 產 | 1110 |
| 生 | 1111 |
| 好 | 1100 |
| 的 | 1101 |
| 加 | 1010 |
| 密 | 1011 |

**Table 8:** A maximal prefix coding

| Words of the File | A Coding |
|---|---|
| 文 | 1 |
| 本 | 01 |
| 壓 | 001 |
| 縮 | 0001 |
| 將 | 00001 |
| 產 | 000001 |
| 生 | 0000001 |
| 好 | 00000001 |
| 的 | 000000001 |
| 加 | 0000000001 |
| 密 | 0000000000 |

## 4. Conclusion

Variable length codes, such that those constructed by the well-known two-pass algorithm due to Huffman [6], are becoming increasingly important for several reasons [21]. Communication costs in distributed systems are beginning to dominate the costs for internal computation and storage. Variable length codes often use fewer bits per file symbol than do fixed length codes (or block codes) such as ASCII, which require $\lceil \log_2 n \rceil$ bits per symbol, where *n* is the number of symbols in the file alphabet and $\lceil\ \rceil$ denotes rounding up to the nearest integer. This can yield tremendous savings in packet-based communication systems. Moreover, the buffering needed to support variable length coding is becoming an inherent part of many systems. One disadvantage of Huffman coding is that it makes two passes over the data: one pass to collect frequency counts of the symbols in the file, followed by the construction of a

Huffman tree and transmission of the tree to the receiver; and a second pass to encode and transmit the symbols themselves, based on the Huffman tree. This cause delay when used for network communication, and in file compression applications the extra disk accesses can slow down the scheme.

Comparing with Huffman coding, the maximal prefix coding may not collect frequency counts of the symbols in the file. By the algorithm given in Section 3, we easily generate a maximal prefix code. This procedure is equivalent to constructing a Huffman code. Secondly, as we see a case study in Section 3, block maximal prefix coding are introduced, i.e., constructing a maximal prefix coding from the words of the file into the code words in a maximal prefix code. Usually, the Huffman coding only consider stream coding, that is, they are the functions from the letters of the file into the code words in a Huffman code. Block coding schemes have advantages that stream encoding schemes lack, such as diffusion-information from the file is diffused into several encoded symbols, and immunity to insertion. Because blocks of symbols are encoded, it is impossible to insert a single symbol into one block. Furthermore, one character from the file does not produce just one encoded character. Therefore, an active interceptor cannot simply cut one encoded letter out of a file and paste a new one in to change an amount, a data, or a name in a file. Thirdly, making use of abundant structure of maximal prefix codes, data compression of maximal prefix coding schemes will greatly improve the security effect of the coding scheme, especially for the one of Chinese storing text retrieval systems.

**ACKNOWLEDGMENTS:** This work was partially sponsored by HK UGC grants CityU 1055/01E, CityU 1076/00E, and CityU Grants 7001189, 7001060, and by the National Natural Science Foundation of China (project No. 60073056) and the Guangdong Provincial Natural Science Foundation (project No. 001174).

## REFERENCES

[1] E. E. Adrin, S. Perkins, *Binary Huffman Equivalent Codes with a Short Synchronizing Code Word*, IEEE Trans. Inform. Theory, **44**(1998), 346-351.

[2] J. Berstel, D. Perrin, *Theory of Codes*, Academic Press, Orlando, 1985.

[3] T. C. Bell, J.G. Cleary and I.H. Witten, *Text Compression*, Prentice Hall, Englewood Cliffs, NJ, 1990.

[4] T. Cover, J. Thomas, *Elements of Information Theory*, New York, Wiley, 1991.

[5] Thomas J. Ferguson, Joshua H. Rabinowitz, *Self-Synchonizing Huffman Codes*, IEEE Trans. Inform. Theory, Vol. **IT-30** (1984) 687-693.

[6] D. A. Huffman, *A Method for the Construction of Minimum Redundancy Codes*, In Proc. IRE **40**(1951), 1098-1101.

[7] D. Hankerson, G.A. Harris, P.D. Johnson, Jr., Introduction to Information Theory and Data Compression, CRC Press LLC, 1997.

[8] L.G. Kraft, *Device for Quantizing, Grouping, and Coding Amplitude Modulated Pulses*, Q.S. Thesis, Electrical Engineering Department, MIT, 1949.

[9] S.M. Lei, M.T. Sun, *An Entropy Coding System for Digital HDTV Applications*, IEEE Trans. Circuit Systems Video Technology, **1**(1991), 147-155.

[10] D.Y. Long, *On Group Codes*, Theoretical Computer Science, **163**(1996) 259-267.

[11] B. McMillan, *Two Inequalities Implied by Unique Decipherability*, IRE Trans. Infom. Theory, **IT-2** (1965), 115-116.

[12] Tibor Nemetz, Pal Papp, *Data-compression Aiding Data-security*, in S. Qing, J.H.P. Eloff (editors): IFIP/SEC2000, International Academic Publishers, 2000, 61-64.

[13] H. Jürgensen, S. Konstantinidis, *Codes*, in: G. Rozenberg, A. Salomaa (editors), *Handbook of Formal Languages*, Vol.1, Springer-Verlag Berlin Heidelberg,1997, 511-607.

[14] M. Nelson, *The Data Compression Book*, M&T Books, New York, 1996. 1997.

[15] W.B. Pennebaker and J.L. Mitchell, *JPEG: Still Image Data Compression Standard*, New York, 1993.

[16] Steven Roman, *Introduction to Coding and Information Theory*, Springer-Verlag New York, 1996.

[17] H.J. Shyr, *Free Monoids and Languages*, Hon Min Book Company, Taichung, Taiwan, 1991.

[18] M.P. Schützenberger, *On the Synchronizing Properties of Certain Prefix Codes*, Information and Control, **7**(1964) 23-36.

[19] M.P. Schützenberger, *On Synchronizing Prefix Codes*, Information and Control, **11**(1967), 396-401.

[20] K.H. Tzou, *High-order Entropy Coding for Images*, IEEE Trans. Circuit Systems Video Technology, **2**(1992) 87-89.

[21] Jefferey Scott Vitter, *Design and Analysis of Dynamic Huffman Codes*, Journal of the Association for Computing Machinery, **34**(1987)4, 825-845