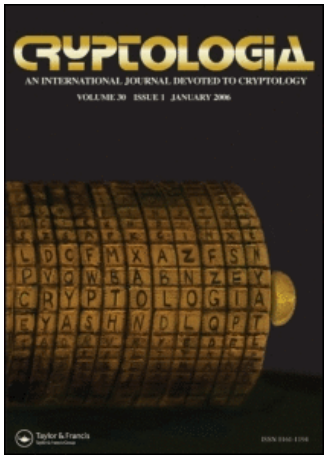

Cryptologia
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t725304178

AN ALGORITHMIC SOLUTION OF SEQUENTIAL HOMOPHONIC CIPHERS

John C. King [a]; Dennis R. Bahler [b]
[a] Hewlett-Packard Co., 3000 Hanover St., MS. 20DY, Palo Alto CA 94304; Email: kingj@corp.hp.com.
[b] Dept. of Computer Science, North Carolina State University, Box 8206, Raleigh, NC 27695-8206; Email: bahler@ncsu.edu..

Online Publication Date: 01 April 1993
To cite this Article: King, John C. and Bahler, Dennis R. (1993) 'AN ALGORITHMIC SOLUTION OF SEQUENTIAL HOMOPHONIC CIPHERS', Cryptologia, 17:2, 148 - 165
To link to this article: DOI: 10.1080/0161-119391867827
URL: http://dx.doi.org/10.1080/0161-119391867827

CRYPTOLOGIA April 1993 Volume XVII Number 2

AN ALGORITHMIC SOLUTION OF SEQUENTIAL HOMOPHONIC CIPHERS

John C. King[1] and Dennis R. Bahler[2]

ADDRESS: (1) Hewlett-Packard Co., 3000 Hanover St., MS. 20DY, Palo Alto CA 94304; Email: kingj@corp.hp.com and (2) Dept. of Computer Science, North Carolina State University, Box 8206, Raleigh, NC 27695-8206; Email: bahler@ncsu.edu.

ABSTRACT: REMOVE_HOMOPHONES is a new cryptanalytic algorithm for the reduction of a sequential homophonic cipher without word divisions into a simple substitution cipher [8]. Sets of homophones, defined in the cipher alphabet, are detected algorithmically, without the use of either frequency analysis or trial-and-error backtracking, in a ciphertext-only attack. Given the output of REMOVE_HOMOPHONES, a simple substitution cipher, probabilistic relaxation [9,13] can complete the algorithmic solution of sequential homophonic ciphers without word divisions.

KEYWORDS: Automated cryptanalysis, Beale Ciphers, homophonic ciphers, sequential homophonic ciphers, Zodiac Murder Ciphers.

SEQUENTIAL HOMOPHONIC CIPHERS

A homophonic cipher [1,2,5,12] is a substitution cipher where each plaintext alphabet symbol a_i $(a_1 \ldots a_t)$ has a corresponding set of unique ciphertext substitutes (homophones), S_i. Each set S_i is disjoint with every other set. Let L_i denote the number of ciphertext symbols in S_i $(L_i = |S_i|)$. L denotes the total number of ciphertext alphabet symbols $(L = L_1 + L_2 + \ldots + L_t)$.

During encryption of a plaintext symbol a_i, any one of the homophones ofS_i may be used as the ciphertext symbol. Random homophonic ciphers select from S_i randomly, while sequential homophonic ciphers use the homophones in sequential order, going back to the first homophone after the last one has been used.

Kahn [7] notes that it was not until the mid-1500's that homophonic ciphers began to include homophones for both vowels and consonants. The authors have found such a cipher dated April 13, 1519 that also happens to be a sequential homophonic cipher. The cipher, a letter written by the Prince of Nassau to Margaret of Austria, Ruler of Holland, is described by Lange and Soudart [11].

MULTIPLICITY OF HOMOPHONIC CIPHERS

Homophonic ciphers can be easily described by two parameters, namely, ciphertext length N and multiplicity M [10]. The multiplicity of a homophonic cipher is defined as follows. Since some available homophones may not be used during an encryption, let L', where $L' \leq L$, denote the number of ciphertext alphabet symbols actually used in a cipher. The multiplicity M is the value of L'/N. M is thus a real number $0.0 \leq M \leq 1.0$.

A desired multiplicity M can be incorporated in a formula for the encryption of homophonic ciphers. The first step in encryption is the creation of a cipher alphabet. The following formula will create L_i proportional to the expected frequency of the plaintext symbol a_i in the language.

FOR α:='A' to 'Z' DO
  L_α = MAX 1, ROUND(prob(α)*N*M)

```
'A' = { 357, 758, 784 }          'N' = { 997, 634, 795 }
'B' = {  55 }                    'O' = { 839, 459, 152 }
'C' = { 761 }                    'P' = { 399 }
'D' = { 149 }                    'Q' = { 602 }
'E' = {  34, 623, 243, 764 }     'R' = { 106, 858 }
'F' = { 792 }                    'S' = { 523 }
'G' = { 938 }                    'T' = { 107, 422, 925 }
'H' = { 727 }                    'U' = { 777 }
'I' = { 963, 607 }               'V' = { 153 }
'J' = { 484 }                    'W' = { 141 }
'K' = { 383 }                    'X' = { 774 }
'L' = { 353 }                    'Y' = {  20 }
'M' = { 273 }                    'Z' = { 785 }
```

Table 1. Homophonic cipher alphabet ($N = 200$, $M = 0.15$).
Key used to encode the input cipher of Table 2.

```
792 839 777 106 523 761 459 858  34 357 997 149 523 623 153 243 634  20 764 758
106 523 784 938 152 839 777 858 792 357 107 727  34 106 523  55 858 459 777 938
727 422 792 152 106 925 727 839 795 107 727 963 523 761 459 997 422 607 634 623
795 925 758 997 243 141 634 784 107 963 152 795 761 839 997 761 764 607 153  34
149 963 634 353 607  55 623 858 422  20 357 795 149 149 243 149 963 761 758 925
764 149 107 459 422 727  34 399 106 152 399 839 523 607 925 963 459 997 107 727
784 422 357 353 353 273 623 634 758 858 243 761 106 764 784 925  34 149 623 602
777 357 353 795 152 141 141 243 758 858 764  34 997 938 784 938 623 149 607 634
357 938 106 243 758 107 761 963 153 607 353 141 784 858 422 764 523 925 963 795
938 141 727  34 107 727 623 106 422 727 357 925 997 758 107 607 839 634 459 858
```

Table 2. INPUT: Sequential homophonic cipher ($N = 200$, $M = 0.15$)

EXAMPLE INPUT AND OUTPUT OF REMOVE_HOMOPHONES

This example shows the input and output of REMOVE_HOMO- PHONES. A plaintext of length 200 was encrypted as a sequential homophonic cipher using the above formula with multiplicity $M = 0.15$. The resulting cipher alphabet is shown in Table 1. The resulting sequential homophonic cipher, which is the only input to REMOVE_HOMOPHONES, is shown in Table 2. Table 3 shows the sets of homophones found by REMOVE_HOMOPHONES. These sets correspond exactly to the homophonic sets in the cipher alphabet of Table 1. Note that although REMOVE_HOMOPHONES does not associate each homophonic set with any particular plaintext symbol, the homophonic sets defined in the cipher alphabet are detected. The first homophone in each set is then used as a substitute for the remaining homophones in the set, thus removing them from the output cipher. The output, a simple substitution cipher is shown in Table 4.

```
{  34, 623, 243, 764 }
{ 107, 422, 925 }
{ 357, 758, 784 }
{ 997, 634, 795 }
{ 839, 459, 152 }
{ 106, 858 }
{ 963, 607 }
```

Table 3. Homophonic sets found by REMOVE_HOMOPHONES.

```
792 839 777 106 523 761 839 106  34 357 997 149 523  34 153  34 997  20  34 357
106 523 357 938 839 839 777 106 792 357 107 727  34 106 523  55 106 839 777938
727 107 792 839 106 107 727 839 997 107 727 963 523 761 839 997 107 963 997 34
997 107 357 997  34 141 997 357 107 963 839 997 761 839 997 761  34 963 153 34
149 963 997 353 963  55  34 106 107  20 357 997 149 149  34 149 963 761 357107
 34 149 107 839 107 727  34 399 106 839 399 839 523 963 107 963 839 997 107 727
357 107 357 353 353 273  34 997 357 106  34 761 106  34 357 107  34 149  34 602
777 357 353 997 839 141 141  34 357 106  34  34 997 938 357 938  34 149 963 997
357 938 106  34 357 107 761 963 153 963 353 141 357 106 107  34 523 107 963 997
938 141 727  34 107 727  34 106 107 727 357 107 997 357 107 963 839 997 839 106
```

Table 4. OUTPUT: a simple substitution cipher

ALGORITHM OVERVIEW AND OUTLINE

REMOVE_HOMOPHONES is based on the following observations. If there is a homophonic set S_i used sequentially, then there is a homophone from the set that is used first. If that first homophone is known, then strings of cipher symbols from the cipher text, each beginning with the homophone and ending with the symbol right before the next occurrence of the homophone, contain the

set of homophones in sequence. These strings can be analyzed to determine the homophonic set.

To continue the example, the cipher alphabet of Table 1 defines the set of homophones for 'O' as {839, 459, 152}. The following is an example of how the homophones appear in the input cipher. From the input cipher (refer back to Table 2), the following strings in Table 5 are the first three strings of cipher symbols beginning with the first homophone for 'O', which is '839'. The start of each string is marked with an asterisk '*'. The homophones in the set are highlighted so that their sequential use can be seen.

```
792 *839 777 106 523  761 459  858  34 357 997 149 523  623  153 243 634  20 764 758
106  523 784 938 152 *839 777  858 792 357 107 727  34  106  523  55 858 459 777 938
727  422 792 152 106  925 727 *839 795 107 727 963 523  761  459 997 422 607 634 623
795  925 758 997 243  141 634  784 107 963 152 795 761 *839 ...
```

Table 5. Homophones for 'O' highlighted in input cipher.

```
BEGIN (* MAIN PROGRAM *)

 (* SEGMENT I -- INITIALIZATION *)
 INPUT_CIPHER;
 TRANSLATE_CIPHER;
 num_seq:=0;

 (* SEGMENT II -- FIND POSSIBLE HOMOPHONIC SETS *)
 (* For each distinct cipher symbol do *)
 FOR symbol:=1 to num_dist DO
   BEGIN
     num_strings:=0;
     FIND_STRINGS(symbol,num_strings);
     IF (num_strings>1)
       THEN ANALYZE_STRINGS(num_strings);
  END;

 (* SEGMENT III -- SELECT BEST HOMOPHONIC SETS *)
 SELECT_SEQUENCES(num_seq);

 (* SEGMENT IV -- OUTPUT A SIMPLE SUBSTITUTION CIPHER *)
 UN_TRANSLATE_SETS;
 REDUCE_CIPHER_TO_SSC;
 OUTPUT_SSC;

END.  (* MAIN PROGRAM *)
```

Table 6. Main program of REMOVE_HOMOPHONES

REMOVE_HOMOPHONES finds all strings for each distinct cipher symbol (each symbol in the cipher alphabet). The strings for each symbol are processed and possible homophonic sets are generated. A count is kept of how many times each possible homophonic set is used sequentially in the ciphertext. A heuristic value is assigned to each possible homophonic set, based on its cardinality and the count. A selection algorithm selects disjoint sets with the highest heuristic values. These sets correspond to the sets defined in the cipher alphabet. These homophonic sets are then used to reduce the sequential homophonic cipher into a simple substitution cipher. Table 6 shows the main program of REMOVE_HOMOPHONES. The entire program contains about thirty procedures and functions and has about 900 lines of code, including documentation. The outline just described can be followed in the Pascal code of Table 6. The processing performed by each program segment is described in this paper.

SEGMENT I – INITIALIZATION

A numerical cipher is the only input. TRANSLATE_CIPHER is a procedure that is used to make the required processing tractable. The ciphertext is translated such that each cipher symbol is replaced by the ordinal number of that symbol. The ordinal number of a cipher symbol represents the order that the distinct cipher symbol was first used in the ciphertext. To continue the example, the translated input ciphertext is shown in Table 7. Refer back to Table 2 for the input cipher.

1	2	3	4	5	6	7	8	9	10	11	12	5	13	14	15	16	17	18	19
4	5	20	21	22	2	3	8	1	10	23	24	9	4	5	25	8	7	3	21
24	26	1	22	4	27	24	2	28	23	24	29	5	6	7	11	26	30	16	13
28	27	19	11	15	31	16	20	23	29	22	28	6	2	11	6	18	30	14	9
12	29	16	32	30	25	13	8	26	17	10	28	12	12	15	12	29	6	19	27
18	12	23	7	26	24	9	33	4	22	33	2	5	30	27	29	7	11	23	24
20	26	10	32	32	34	13	16	19	8	15	6	4	18	20	27	9	12	13	35
3	10	32	28	22	31	31	15	19	8	18	9	11	21	20	21	13	12	30	16
10	21	4	15	19	23	6	29	14	30	32	31	20	8	26	18	5	27	29	28
21	31	24	9	23	24	13	4	26	24	10	27	11	19	23	30	2	16	7	8

Table 7. Translated input cipher. This cipher is identical to the input cipher of Table 2 except for cipher symbol identities.

It will later be seen in "Reducing String Length" how this translation actually makes the REMOVE_HOMOPHONES algorithm computationally feasible. Finally, num_seq is initialized to 0. The variable num_seq represents the number of possible homophonic sets that will be found, evaluated, and, from which, the best will be selected.

SEGMENT II – FIND POSSIBLE HOMOPHONIC SETS

Finding Strings

For each distinct cipher symbol, strings of ciphertext symbols, each beginning with the symbol, are pulled from the ciphertext. Refer back to Table 5 to see example strings as they appear in the input cipher. The last string found is discarded if it is not a complete string, that is, if its terminal symbol is not followed by another occurrence of the symbol in the ciphertext. The strings found for the first ten distinct cipher symbols of the cipher in Table 7 are shown in Table 8. Strings are shown only for the first ten symbols so that a sufficiently rich, but not overly long, example can be presented.

Reducing String Length

There are three reductions of string length for all the strings found for a particular distinct cipher symbol. The first reduction takes advantage of the ciphertext translation performed in Segment I of the algorithm. If a homophone is used as the first homophone in a sequence, then the remaining homophones in the sequence will be greater than the first. Therefore, all cipher symbols in a string that are less than the first symbol can be removed from the string. For an example refer to Table 8 for the strings beginning with cipher symbol '4'. In those strings the symbols '1'-'3' can be removed.

The second reduction is to remove all occurrences of any cipher symbol that occurs more than once in the string. If it occurs more than once, we know it has not been used sequentially with the first symbol of the string. For an example refer to Table 8 for the strings beginning with the cipher symbol '1'. For string #1, the symbols '2'-'5' and '8' can be removed. For string #2, only '24' can be removed.

The third reduction is to remove all occurrences of any cipher symbol that does not occur in each string found for a particular cipher symbol. Since any sequence will be found in all strings, any symbol that does not occur in all strings has not been used in a sequence beginning with the first symbol of the string. For an example refer back to Table 8 for the strings beginning with cipher symbol '1'. The symbols '6', '11'-'19', '20', and '22' occur in string #1 but not in string #2, so they can be removed. Symbols '2'-'5' and '8' have already been removed by the second reduction. The symbols '3'-'5', '8', '23', and '25'-'26' occur in String #2 but not in String #1, so they can be removed.

Table 9 shows the results of the reductions of the strings in Table 8. Notice that no strings exist for cipher symbols '3', '5', and '6' after the reductions, indicating that these symbols are not the initial homophones of homophonic

```
 SYMBOL '1':
STRING   #1: 1  2  3  4  5  6  7  8  9 10 11 12  5 13 14 15 16 17 18 19  4  5
            20 21 22  2  3  8
STRING # 2:  1 10 23 24  9  4  5 25  8  7  3 21 24 26
 SYMBOL  '2':
STRING # 1:  2  3  4  5  6  7  8  9 10 11 12  5 13 14 15 16 17 18 19  4  5 20
            21 22
STRING # 2:  2  3  8  1 10 23 24  9  4  5 25  8  7  3 21 24 26  1 22  4 27 24
STRING # 3:  2 28 23 24 29  5  6  7 11 26 30 16 13 28 27 19 11 15 31 16 20 23
            29 22 28  6
STRING # 4:  2 11  6 18 30 14  9 12 29 16 32 30 25 13  8 26 17 10 28 12 12 15
            12 29  6 19 27 18 12 23  7 26 24  9 33  4 22 33
STRING # 5:  2  5 30 27 29 7  11 23 24 20 26 10 32 32 34 13 16 19  8 15  6  4
            18 20 27  9 12 13 35  3 10 32 28 22 31 31 15 19  8 18  9 11 21 20
            21 13 12 30 16 10 21  4 15 19 23  6 29 14 30 32 31 20  8 26 18  5
            27 29 28 21 31 24  9 23 24 13  4 26 24 10 27 11 19 23 30
 SYMBOL  '3':
STRING # 1:  3  4  5  6  7  8  9 10 11 12  5 13 14 15 16 17 18 19  4  5 20 21
            22  2
STRING # 2:  3  8  1 10 23 24  9  4  5 25  8  7
STRING # 3:  3 21 24 26  1 22  4 27 24  2 28 23 24 29  5  6  7 11 26 30 16 13
            28 27 19 11 15 31 16 20 23 29 22 28  6  2 11  6 18 30 14  9 12 29
            16 32 30 25 13  8 26 17 10 28 12 12 15 12 29  6 19 27 18 12 23  7
            26 24  9 33  4 22 33  2  5 30 27 29  7 11 23 24 20 26 10 32 32 34
            13 16 19  8 15  6  4 18 20 27  9 12 13 35
 SYMBOL   '4':
STRING # 1:  4  5  6  7  8  9 10 11 12  5 13 14 15 16 17 18 19
STRING # 2:  4  5 20 21 22  2  3  8  1 10 23 24  9
STRING # 3:  4  5 25  8  7  3 21 24 26  1 22
STRING # 4:  4 27 24  2 28 23 24 29  5  6  7 11 26 30 16 13 28 27 19 11 15 31
            16 20 23 29 22 28  6  2 11  6 18 30 14  9 12 29 16 32 30 25 13  8
            26 17 10 28 12 12 15 12 29  6 19 27 18 12 23  7 26 24  9 33
STRING # 5:  4 22 33  2  5 30 27 29  7 11 23 24 20 26 10 32 32 34 13 16 19  8
            15  6
STRING # 6:  4 18 20 27  9 12 13 35  3 10 32 28 22 31 31 15 19  8 18  9 11 21
            20 21 13 12 30 16 10 21
STRING # 7:  4 15 19 23  6 29 14 30 32 31 20  8 26 18  5 27 29 28 21 31 24  9
            23 24 13
 SYMBOL   '5':
STRING # 1:  5  6  7  8  9 10 11 12
STRING # 2:  5 13 14 15 16 17 18 19  4
STRING # 3:  5 20 21 22  2  3  8  1 10 23 24  9  4
STRING # 4:  5 25  8  7  3 21 24 26  1 22  4 27 24  2 28 23 24 29
STRING # 5:  5  6  7 11 26 30 16 13 28 27 19 11 15 31 16 20 23 29 22 28  6  2
            11  6 18 30 14  9 12 29 16 32 30 25 13  8 26 17 10 28 12 12 15 12
            29  6 19 27 18 12 23  7 26 24  9 33  4 22 33  2
STRING # 6:  5 30 27 29  7 11 23 24 20 26 10 32 32 34 13 16 19  8 15  6  4 18
            20 27  9 12 13 35  3 10 32 28 22 31 31 15 19  8 18  9 11 21 20 21
            13 12 30 16 10 21  4 15 19 23  6 29 14 30 32 31 20  8 26 18
```

Table 8. Strings for first ten distinct cipher symbols.

```
 SYMBOL  '6':
STRING #1:   6  7  8  9 10 11 12  5 13 14 15 16 17 18 19  4  5 20 21 22  2  3
             8  1 10 23 24  9  4  5 25  8  7  3 21 24 26  1 22  4 27  2 28
            23 24 29  5
STRING #2:   6  7 11 26 30 16 13 28 27 19 11 15 31 16 20 23 29 22 28
STRING #3:   6  2 11
STRING #4:   6 18 30 14  9 12 29 16 32 30 25 13  8 26 17 10 28 12 12 15 12 29
STRING #5:   6 19 27 18 12 23  7 26 24  9 33  4 22 33  2  5 30 27 29  7 11 23
            24 20 26 10 32 32 34 13 16 19  8 15
STRING #6:   6  4 18 20 27  9 12 13 35  3 10 32 28 22 31 31 15 19  8 18  9 11
            21 20 21 13 12 30 16 10 21  4 15 19 23
 SYMBOL  '7':
STRING #1:   7  8  9 10 11 12  5 13 14 15 16 17 18 19  4  5 20 21 22  2  3  8
             1 10 23 24  9  4  5 25  8
STRING #2:   7  3 21 24 26  1 22  4 27 24  2 28 23 24 29  5 6
STRING #3:   7 11 26 30 16 13 28 27 19 11 15 31 16 20 23 29 22 28  6  2 11  6
            18 30 14  9 12 29 16 32 30 25 13  8 26 17 10 28 12 12 15 12 29  6
            19 27 18 12 23
STRING #4:   7 26 24  9 33  4 22 33  2  5 30 27 29
STRING #5:   7 11 23 24 20 26 10 32 32 34 13 16 19  8 15  6  4 18 20 27  9 12
            13 35  3 10 32 28 22 31 31 15 19  8 18  9 11 21 20 21 13 12 30 16
            10 21  4 15 19 23  6 29 14 30 32 31 20  8 26 18  5 27 29 28 21 31
            24  9 23 24 13  4 26 24 10 27 11 19 23 30  2 16
 SYMBOL  '8' :
STRING #1:   8  9 10 11 12  5 13 14 15 16 17 18 19  4  5 20 21 22  2  3
STRING #2:   8  1 10 23 24  9  4  5 25
STRING #3:   8  7  3 21 24 26  1 22  4 27 24  2 28 23 24 29  5  6  7 11 26 30
            16 13 28 27 19 11 15 31 16 20 23 29 22 28  6  2 11  6 18 30 14  9
            12 29 16 32 30 25 13
STRING #4:   8 26 17 10 28 12 12 15 12 29  6 19 27 18 12 23  7 26 24  9 33  4
            22 33  2  5 30 27 29  7 11 23 24 20 26 10 32 32 34 13 16 19
STRING #5:   8 15  6  4 18 20 27  9 12 13 35  3 10 32 28 22 31 31 15 19
STRING #6:   8 18  9 11 21 20 21 13 12 30 16 10 21  4 15 19 23  6 29 14 30 32
            31 20
STRING #7:   8 26 18  5 27 29 28 21 31 24  9 23 24 13  4 26 24 10 27 11 19 23
            30  2 16  7
 SYMBOL  '9':
STRING #1:   9 10 11 12  5 13 14 15 16 17 18 19  4  5 20 21 22  2  3  8  1 10
            23 24
STRING #2:   9  4  5 25  8  7  3 21 24 26  1 22  4 27 24  2 28 23 24 29  5  6
             7 11 26 30 16 13 28 27 19 11 15 31 16 20 23 29 22 28  6  2 11  6
            18 30 14
STRING #3:   9 12 29 16 32 30 25 13  8 26 17 10 28 12 12 15 12 29  6 19 27 18
            12 23  7 26 24
STRING #4:   9 33  4 22 33  2  5 30 27 29  7 11 23 24 20 26 10 32 32 34 13 16
            19  8 15  6  4 18 20 27
STRING #5:   9 12 13 35  3 10 32 28 22 31 31 15 19  8 18
STRING #6:   9 11 21 20 21 13 12 30 16 10 21  4 15 19 23  6 29 14 30 32 31 20
             8 26 18  5 27 29 28 21 31 24
```

Table 8. Strings for first ten distinct cipher symbols (continued).

```
 SYMBOL  '10':
STRING #1:   10 11 12  5 13 14 15 16 17 18 19  4  5 20 21 22  2  3  8  1
STRING #2:   10 23 24  9  4  5 25  8  7  3 21 24 26 1  22  4 27 24  2 28 23 24
             29 5 6  7 11 26 30 16 13 28 27 19 11 15 31 16 20 23 29 22 28   6
              2 11 6 18 30 14  9 12 29 16 32 30 25 13  8 26 17
STRING #3:   10 28 12 12 15 12 29 6 19 27 18 12 23  7 26 24  9 33  4 22 33 2
              5 30 27 29  7 11 23 24 20 26
STRING #4:   10 32 32 34 13 16 19  8 15  6  4 18 20 27  9 12 13 35  3
STRING #5:   10 32 28 22 31 31 15 19  8 18  9 11 21 20 21 13 12 30 16
STRING #6:   10 21  4 15 19 23  6 29 14 30 32 31 20  8 26 18 5 27 29 28 21 31
             24  9 23 24 13  4 26 24
```

Table 8. Strings for first ten distinct cipher symbols (continued).

```
SYMBOL '1':
 STRING #1:   1   7   9  10  21
 STRING #2:   1  10   9   7  21

SYMBOL '2':
 STRING #1:   2   7  22
 STRING #2:   2   7  22
 STRING #3:   2   7  22
 STRING #4:   2   7  22
 STRING #5:   2   7  22

SYMBOL '4':
 STRING #1:   4   8
 STRING #2:   4   8
 STRING #3:   4   8
 STRING #4:   4   8
 STRING #5:   4   8
 STRING #6:   4   8
 STRING #7:   4   8

SYMBOL '7':
 STRING #1:   7  22
 STRING #2:   7  22
 STRING #3:   7  22
 STRING #4:   7  22
 STRING #5:   7  22
```

```
SYMBOL '8':
 STRING #1:   8   9
 STRING #2:   8   9
 STRING #3:   8   9
 STRING #4:   8   9
 STRING #5:   8   9
 STRING #6:   8   9
 STRING #7:   8   9

SYMBOL '9':
 STRING #1:   9  13  15  18  19
 STRING #2:   9  13  19  15  18
 STRING #3:   9  13  15  19  18
 STRING #4:   9  13  19  15  18
 STRING #5:   9  13  15  19  18
 STRING #6:   9  13  15  19  18

SYMBOL '10':
 STRING #1:  10  15  18  19  20
 STRING #2:  10  19  15  20  18
 STRING #3:  10  15  19  18  20
 STRING #4:  10  19  15  18  20
 STRING #5:  10  15  19  18  20
 STRING #6:  10  15  19  20  18
```

Table 9. Reduced strings for first ten cipher symbols.
Strings from Table 8 after string reductions.

sets used sequentially. After these reductions of string length, each string for a particular ciphertext symbol has the same symbols, but possibly in different permutations.

Finding Valid Subsets From the Strings

If a homophonic sequence exists in one string, then it exists in all strings for a particular cipher symbol. The way to find a possible sequence is to generate all possible combinations of cipher symbols of the first string and test to see if each subsequence exists in all strings. If the subsequence does exist in all strings, it is saved as a possible homophonic set. If the subsequence does not exist in all strings, it is discarded.

As an example, the strings for the tenth cipher symbol will be analyzed to find possible homophonic sets. Refer to Table 9 for the strings beginning with cipher symbol '10'. Combinations of the first string are generated such that each combination begins with the first symbol of the string and has more than one element. Therefore, given a string of length n, there will be $2^{(n-1)} - 1$ combinations to test. Table 10 shows, for each combination, if it is a possible homophonic set, i.e. the sequence occurs in all strings. Table 10 shows that there are seven possible homophonic sets out of the 15 combinations.

```
SYMBOL '10':
      STRING #1:  10  15  18  19  20
      STRING #2:  10  19  15  20  18
      STRING #3:  10  15  19  18  20
      STRING #4:  10  19  15  18  20
      STRING #5:  10  15  19  18  20
      STRING #6:  10  15  19  20  18
Combinations of STRING #1 symbols      Classification
      Combination:  10  20               Possible homophonic set
      Combination:  10  19               Possible homophonic set
      Combination:  10  19  20           Possible homophonic set
      Combination:  10  18               Possible homophonic set
      Combination:  10  18  20           Discard
      Combination:  10  18  19           Discard
      Combination:  10  18  19  20       Discard
      Combination:  10  15               Possible homophonic set
      Combination:  10  15  20           Possible homophonic set
      Combination:  10  15  19           Discard
      Combination:  10  15  19  20       Discard
      Combination:  10  15  18           Possible homophonic set
      Combination:  10  15  18  20       Discard
      Combination:  10  15  18  19       Discard
      Combination:  10  15  18  19  20   Discard
```

Table 10. String analysis for the cipher symbol "10"

F	COUNT	HOMOPHONIC SEQUENCE				
4	2	1	21			
4	2	1	10			
6	2	1	10	21		
4	2	1	9			
6	2	1	9	21		
4	2	1	7			
6	2	1	7	21		
10	5	2	22			
10	5	2	7			
15	5	2	7	22		
14	7	4	8			
10	5	7	22			
14	7	8	9			
12	6	9	19			
12	6	9	18			
12	6	9	15			
18	6	9	15	18		
12	6	9	13			
18	6	9	13	19		
18	6	9	13	18		
18	6	9	13	15		
24	6	9	13	15	18	-This set has the highest F value.
12	6	10	20			-This set is from Table 10.
12	6	10	19			" "
18	6	10	19	20		" "
12	6	10	18			" "
12	6	10	15			" "
18	6	10	15	20		" "
18	6	10	15	18		" "
12	6	11	28			
12	6	11	16			
18	6	11	16	28		
12	6	13	20			
12	6	13	19			
18	6	13	19	20		
12	6	13	18			
12	6	13	15			
18	6	13	15	20		
18	6	13	15	18		

Table 11. Evaluated possible homophonic sets. The heuristic function value F equals the cardinality of the homophonic set times the count (number of strings in which it was found).

F	COUNT	HOMOPHONIC SEQUENCE			
4	2	14	25		
4	2	14	17		
10	5	15	20		
10	5	15	18		
12	6	16	28		
12	6	19	20		
14	7	23	27		
14	7	23	26		
21	7	23	26	27	-This set has the second highest F value
12	6	26	27		and is disjoint with the set with the
12	6	29	30		highest F value.

Table 11. (continued) Evaluated possible homophonic sets. The heuristic function value F equals the cardinality of the homophonic set times the count (number of strings in which it was found).

When each possible homophonic set is found, its cardinality is associated with it, along with a count (the number of strings in which it was found) and a heuristic function value F. The heuristic function value is equal to the count times the cardinality. Table 11 shows all the possible homophonic sets for all cipher symbols with their corresponding heuristic function values and counts. Note that the possible homophonic sets of Table 10 for cipher symbol '10' are included in Table 11. All possible homophonic sets are then evaluated in SEGMENT III.

SEGMENT III – SELECT BEST HOMOPHONIC SETS

After the possible homophonic sets are found, a selection algorithm iteratively selects all disjoint sets, starting with set with the highest heuristic value. In case of ties, the set with the highest cardinality is selected. In this example, seven sets were chosen from the sets in Table 11. One of these is the set {9 13 15 18}, with the largest heuristic value of F=24. These sets are shown in Table 12.

```
{  9, 13, 15, 18 }
{ 23, 26, 27 }
{ 10, 19, 20 }
{ 11, 16, 28 }
{  2,  7, 22 }
{  4,  8 }
{ 29, 30 }
```

Table 12. Untranslated homophonic sets found by REMOVE_HOMOPHONES.

SEGMENT IV – OUTPUT A SIMPLE SUBSTITUTION CIPHER

Since the cipher was translated in SEGMENT I the cipher symbols in the selected homophonic sets must be translated back to their original identities. Table 3 shows the sets of Table 12 once this has been done. The sets of Table 3 correspond to the homophonic sets as defined in the cipher alphabet in Table 1. A simple substitution cipher is generated by using the first homophone in each set as a substitute for all occurrences of the remaining homophones in the set.

Known Limitations

At $M > 0.50$, there will not be at least two strings, each containing the entire homophonic set defined in the cipher alphabet. Without at least two strings for a particular cipher symbol, nothing can be compared. This is a limitation of the algorithm that cannot be remedied. For a description of the process of finding strings, see the section "Finding Strings".

Another limitation concerns the code that finds valid subsets from the strings (See the section "Finding Valid Subsets From the Strings"). The number of combinations to be tested for a string of length n is $2^{(n-1)} - 1$. The maximum n for which the current code can generate combinations is $n = 20$, which results in 524,287 combinations that must be analyzed. This limitation affects only the processing of ciphers with length N and multiplicity M of sufficient size. This limitation of the algorithm could be remedied by adding code to generate combinations for strings of greater length, but with the trade-off in increased processing time and memory requirements.

Testing and Results

The performance of REMOVE_HOMOPHONES was measured for sequential homophonic ciphers of varying length N and multiplicity M. Four different plaintext selections were used, each encrypted as fifty different ciphers. If all homophonic sets were found accurately the result was recorded as a success. Any other result, no matter how close to success, was recorded as a failure.

The effective range for REMOVE_HOMOPHONES is illustrated in Figure 1. Figure 1 also shows the value of L' for an example plaintext. L' is the number of distinct cipher symbols used in the ciphertext. REMOVE_HOMOPHONES performs successfully over a wide range of sequential homophonic substitution ciphers of varying length and multiplicity. Given a simple substitution cipher, REMOVE_HOMOPHONES correctly finds no homophonic sets. Given a sequential homophonic cipher, REMOVE_HOMOPHONES is successful for multiplicity

$0.0 \leq M \leq 0.10$. The algorithm is also generally successful for sequential homophonic ciphers of length $N \leq 600$ at multiplicity $0.10 < M \leq 0.15$. Testing was not done at this level for ciphers of length $N > 600$ because of the excessive length of the reduced strings. Reduced string lengths at this level ranged from 31 to 57, however, the code only handles a maximum of 20 (see the section "Known Limitations").

A single example has been used throughout this paper to illustrate the algorithm. Table 13 below shows the homophonic sets correctly determined for a more complicated test case.

Processing times averaged twenty seconds of CPU time except for a few cases of high multiplicity that took anywhere from two to thirty minutes of CPU time. The test cases were run on a Hewlett-Packard PA-RISC machine, an HP3000 Series 960 with a floating point coprocessor. The code was optimized during compilation.

```
{ 273, 997, 634, 795, 839, 459, 152, 399, 602, 106, 858, 523, 107, 422 }
{ 162, 467, 333, 944, 218, 745, 808,  41, 545, 517, 702 }
{ 357, 758, 784,  55, 761, 149,  34, 623, 243 }
{ 656, 555, 108,  84,  39, 849, 203,  99 }
{  17, 652, 892, 551, 418, 878,  31 }
{ 271, 374, 828, 847, 673, 453, 952 }
{  20, 785, 248,  77, 362, 302, 977 }
{ 756, 161, 249, 812, 982,  28 }
{ 963, 607, 484, 383, 353 }
{ 644,  16, 136, 786 }
{ 792, 938, 727 }
{ 153, 141, 774 }
{ 179, 822, 872 }
{ 699, 976, 934 }
{ 752, 578 }
{ 749, 988 }
{ 925, 777 }
```

Table 13. Sets found by REMOVE_HOMOPHONES for ($N = 1000$, $M = 0.10$). The number of distinct cipher symbols used to encrypt the corresponding cipher was $L' = 104$. The following are the homophonic sets of cardinality greater than one in the cipher alphabet.

APPLICATION TO HISTORICAL CIPHERS

Table 14 gives the multiplicity of five well known homophonic ciphers, the Zodiac Murder Ciphers [3] and the Beale Ciphers [4,5,6,14]. Z1 and B2 have known solutions. No generally accepted solutions for the others have been published.

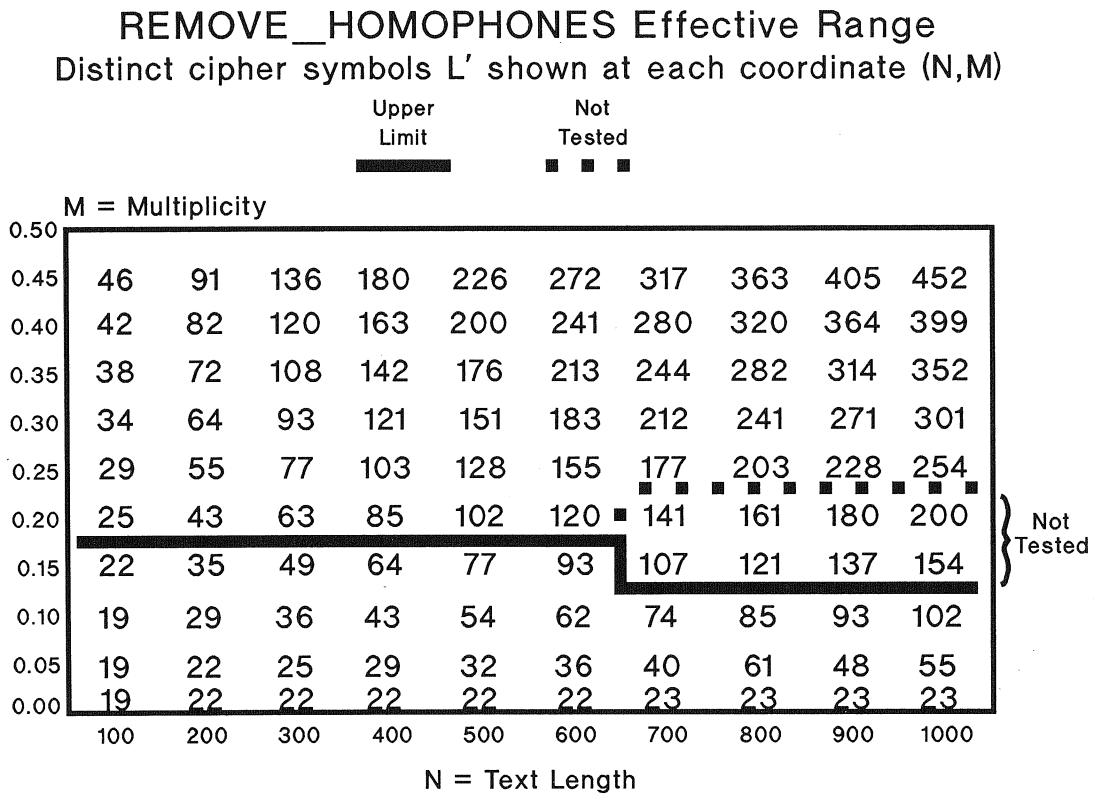

Figure 1. REMOVE_HOMOPHONES Effective Range.

Zodiac Murder Ciphers Z1 and Z2.
Beale Ciphers B1, B2, and B3.

Cipher	L'	N	M
Z1	54	408	0.13
Z2	64	340	0.19
B1	298	520	0.57
B2	182	763	0.24
B3	263	618	0.43

Table 14. Multiplicity of historical homophonic ciphers [10].

REMOVE_HOMOPHONES is not applicable to the Beale Ciphers because their multiplicity is too high. In fact, B1 and B3 are probably not sequential homophonic ciphers because B2 is not [4]. Z1 incorporates the sequential use of homophones, but only partway through the cipher. Z1 also has a polyphone [2] and several encoding mistakes. However, enough of the cipher was encoded as a sequential homophonic cipher so that REMOVE_HOMOPHONES was able to find the homophonic sets shown in Table 15. These three sets are accurate, but are not complete, and not all sets were found. They can still be used to reduce Z1 from 54 distinct cipher symbols down to 48. An analysis of Z2 gave no indication that it is a sequential homophonic cipher.

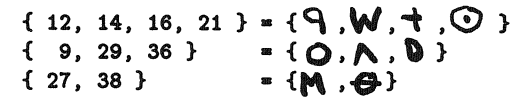

Table 15. Homophonic Sets found for Z1. The geometric cipher-text symbols of Z1 were translated to ordinal numeric symbols. The original geometric cipher symbols are also shown.

SUMMARY

REMOVE_HOMOPHONES is a new cryptanalytic algorithm for the reduction of a sequential homophonic cipher without word divisions into a simple substitution cipher [8]. Sets of homophones, defined in the cipher alphabet, are detected algorithmically, without the use of either frequency analysis or trial-and-error backtracking, in a cipher-text only attack. Given the output of REMOVE_HOMOPHONES, a simple substitution cipher, probabilistic relaxation [9,13] can complete the algorithmic solution of sequential homophonic ciphers without word divisions.

Although the test cases were based on a plaintext alphabet A where $|A| = 26$, REMOVE_HOMOPHONES is applicable for languages where $|A| <> 26$.

REFERENCES

1. Denning, D. 1982. *Cryptography and Data Security*. Reading MA: Addison-Wesley.

2. Gaines, H. 1956. *Cryptanalysis; A Study of Ciphers and Their Solutions*. New York NY: Dover. Originally published as *Elementary Cryptanalysis*. 1939.

3. Graysmith, R. 1986. *Zodiac*. New York NY: St. Martin's.

4. Hammer, C. 1979. How Did TJB Encode B2?. *Cryptologia*. 3(1): 9-15.

5. Hammer, C. 1988. Second Order Homophonic Ciphers. *Cryptologia*. 12(1): 11-20.

6. Hammer, C. 1971. Signature Simulation and Certain Cryptographic Codes. *Communications of the ACM*. 14(1): 3-14.

7. Kahn, D. 1967. *The Codebreakers; The Story of Secret Writing*. New York, NY: Macmillan.

8. King, J. 1991. An Algorithmic Solution of Sequential Homophonic Ciphers. Unpublished Master's thesis, Dept. of Computer Science, North Carolina State University, Raleigh NC 27695-8206.

9. King, J. and D. Bahler. 1992. An Implementation of Probabilistic Relaxation in the Cryptanalysis of Simple Substitution Ciphers. *Cryptologia*. 16(3): 215-225.

10. King, J. and D. Bahler. 1993. A Framework for the Study of Homophonic Ciphers in Classical Encryption and Genetic Systems. *Cryptologia*. 17(1): 45-54.

11. Lange, A. and E. Soudart. 1981. *Treatise on Cryptography. An English Translation of the Original "Traite De Cryptographie"*. Laguna Hills CA: Agean Park Press.

12. Meyer, C. and S. Matyas. 1982. *Cryptography: A New Dimension in Computer Data Security*. New York NY: John Wiley & Sons.

13. Peleg, S. and A. Rosenfeld. 1979. Breaking Substitution Ciphers Using a Relaxation Algorithm. *Communications of the ACM*. 22(11): 589-605.

14. Viemeister, P. 1987. *The Beale Treasure: A History of a Mystery*. Bedford VA: Hamilton's.

ACKNOWLEDGEMENTS

Financial support by Tektronix, Inc. through the GEM graduate fellowship program and by Hewlett-Packard Co. through the Employee Educational Assistance Program is acknowledged.

Appreciation goes to Dr. Carl Hammer for his thorough review of the thesis manuscript and to Dr. Stephen Matyas for several helpful discussions.

BIOGRAPHICAL SKETCHES

John C. King received his BS in Computer Science from the University of Oklahoma in 1986 and his MS in Computer Science from North Carolina State University in 1991. John has been with Hewlett-Packard Co. since 1988 as a knowledge engineer developing an expert system for computer system configuration. Research interests include artificial intelligence techniques in automated cryptanalysis, homophonic ciphers in classical encryption and genetic systems, and computational algorithms in molecular biology. John is a member of AAAI.

Dr. Dennis Bahler received the PhD in Computer Science from the University of Virginia in 1987. Since then he has been an Assistant Professor of Computer Science at North Carolina State University. His research interests lie in artificial intelligence, especially constraint processing, programming languages for AI, intelligent robotics, and machine learning. His research has been funded by NASA, DARPA, NIH, IBM, and NSF. Dr. Bahler is a member of ACM, AAAI, and the IEEE Computer Society.