The Automated Cryptanalysis of Analog Speech Scramblers

B Goldburg**, E Dawson*, S Sridharan**

*School of Mathematics
and
Information Security Research Centre
Queensland University of Technology
GPO Box 2434
Brisbane Qld 4001
Australia

**School of Electrical and Electronic Systems Engineering
Queensland University of Technology
GPO Box 2434
Brisbane Qld 4001
Australia

Abstract

An automated method of attacking commonly used speech scramblers is presented. The cryptanalysis relies on the availability of the scrambled speech only and makes use of the characteristics of speech. It is shown that some of the currently available time and frequency domain scramblers, based on a fixed permutation, can be cryptanalysed. For systems where the permutation is changed with time, methods for partial recovery of the encrypted speech for several existing systems are given. In the case of the frequency domain scramblers a novel method of attack using a codebook is presented.

1. Introduction

Most commercially available analog speech scrambling systems are based on permutation of components in either the time domain or the frequency domain. This paper describes a systematic computer attack on several analog speech scramblers. The scramblers attacked consist of the hopping window scrambler as described in [1], bandsplitter as described in [1] and Discrete Fourier Transform (DFT) scrambler as described in [10]. A description will be given of the method used to attack each cipher.

In Carroll [2], [3] and [4] there are descriptions of methods for systematic computer cryptanalysis of simple ciphers, using permutation and substitution, for encryption of written text. These attacks referenced a large amount of data concerning the nature of the language. For example letter frequency in terms of single characters, digraphs and trigraphs were used. Speech has more redundancy than written text. However it is inherently difficult to use this redundancy when conducting a computer cryptanalysis of scrambled speech since much of the redundancy depends on the perception of the listener.

In assessing the security of an encryption algorithm it is assumed that an attacker has complete knowledge

D.W. Davies (Ed.): Advances in Cryptology - EUROCRYPT '91, LNCS 547, pp. 422-430, 1991.

of the encryption system. Under this assumption the entire security of the cipher resides in the key. This assumption will be made in all the attacks discussed in this paper. For many analog scrambling systems this is a reasonable assumption to make since some designers present details of the system as part of the marketing exercise. It may also be possible for an attacker to purchase the scrambler to determine what type of system is used. In this paper ciphertext only attacks will be described. This attack assumes that a cryptanalyst has access to only an encrypted version of a conversation. A cipher which can be compromised under such an attack is very insecure. To the understanding of the authors this is the first time that the results of such attacks have been reported in the literature.

2. Cryptanalysis of Hopping Window Scrambler

A hopping window type time domain speech scrambler achieves security by first dividing a continuous speech signal into equal length frames, typically half a second in duration. These frames are further subdivided into a fixed number of time segments of equal length. The scrambling algorithm reorders the time segments within a given frame according to the current permutation. This scrambled frame is then sent in place of the original frame. Figure 1 indicates the process used for a six segment scrambler.

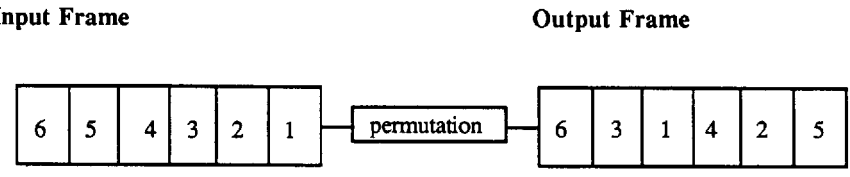

Figure 1

Hopping Window Scrambler

Suppose that a hopping window scrambler is used with a fixed permutation. Knowledge of the speech characteristics can be applied to give the cryptanalyst information about the permutation being used. The frequency content of the speech signal either side of a segment boundary will not change significantly. It should be therefore possible to match the end of one segment with the beginning of its succeeding segment as suggested in [1]. A spectral distance measure is calculated between the end of each segment and the start of all other segments in the frame. Two segments having minimum distance constitute a match. Each segment is used in turn as a starting segment. The second segment is found using the technique described above. The second segment can in turn be used to locate the third and so on until the end of the frame is reached.

There are several different spectral distances that one can apply to match segments. The spectral distances which were used were the Log Spectral Distance (LSD) as described in [6], the Frequency Weighted Log Spectral Distance (FWLSD) as described in [9], the Frequency Variant Spectral Distance (FVSD) as described in [9] and the Segmental Spectral Signal to Noise Ratio (SSNR) as described in [8].

In practice there will be a finite probability of error when choosing a successor to any segment. To reduce

this error the most probable matches can be stored. When investigating the correct segment sequence it becomes necessary to employ dynamic programming techniques in the form of a path search algorithm. After choosing a starting segment, a path is traced until it violates the conditions for an allowable path. Paths which encounter the same segment twice represent a violation. If an illegal segment is found the routine is required to backtrack and try an alternative route until the sequence is completed successfully. The path which yields the minimum total distance between segments is selected as the best approximation to the permutation for the current frame. In general this algorithm is unable to determine the correct permutation using a single frame. It is necessary therefore to collect evidence until a definite decision can be made. Information concerning the succeeding segment, for each segment in a frame, is stored together with the most common succeeding segments from previous frames in a matrix. A permutation approximation is constructed from this information using the path search algorithm described above.

The attack process is outlined in the following using m time segments over N speech frames.

Step 1: Input ciphertext frame $i = 1, 2, ..., N$.

Step 2: For each frame i calculate the spectral distance measure for the end of each segment and the start of all other segments.

Step 3: For $j = 1, ..., m$ using a path search algorithm, calculate the most likely permutation with segment j as a starter.

Step 4: For the m permutations choose the path with the minimum total distance between segments.

Step 5: Update the table containing the most common succeeding segments for each segment in the frame.

Step 6: After N frames use above matrix to generate the most likely permutation.

Attacks were conducted, using the algorithm described above, on hopping window scramblers using eight, and sixteen segments. In each case the correct permutation was recovered using less than twenty seconds of encrypted speech. The frame size was kept constant at 0.512 seconds, which is equivalent to 4096 samples at an 8 KHz sampling rate. The average number of half second frames required to make a successful attack as the segment size is varied is shown in Table 1. These values are given for each of the four spectral distance measures namely LSD, FWLSD, FVSD and SNR. The eight and sixteen segment scramblers on average required 3 and 12 $\frac{\text{sec}}{\text{frame}}$ respectively to make an attack. These results indicate that the FWLSD measure offers a reliable and efficient method to use in attacking the hopping window scrambler. These results demonstrate the poor security offered by the fixed permutation hopping window type scrambler. There is an increase in the average number of frames required for a successful attack as the number of segments is increased. This result is demonstrated by the increase in key entrophy when more segments are added. It should be noted that the number of frames required to attack is very sensitive to the position of the segment boundaries.

Eight Segment Hopping Window Scrambler	
Average Number Measure	**Distance of Frames Required**
LSD	5
FWLSD	4
FVSD	4
SSNR	6

Sixteen Segment Hopping Window Scrambler	
Average Number Measure	**Distance of Frames Required**
LSD	9
FWLSD	7
FVSD	11
SSNR	18

Table 1
Results of Fixed Permutation
Attack on Hopping Window Scrambler

Now consider a hopping window device which chooses a different permutation with which to encrypt each frame. The attack proposed for the fixed permutation system described previously is directly applicable to the varying permutation case. It is not possible, however, to use information extracted from previous frames to aid the recovery of permutations. The permutation must be determined from a single frame. The results above from Table 1 indicate that a single frame is not sufficient to recover the correct permutation. Experimentation proved this to be true. Very little intelligibility could be recovered from the ciphertext.

3. Cryptanalysis of Bandsplitter

The bandsplitter scrambles speech by partitioning the original speech spectrum into a fixed number of frequency bands, usually 16 or less. The current permutation then defines how these bands will be rearranged within the frequency space. Figure 2 illustrates the technique for five bands. This scrambling algorithm need not be frame based and can be done on a continuous basis. The receiver must be made aware of the point at which the permutation is changed however.

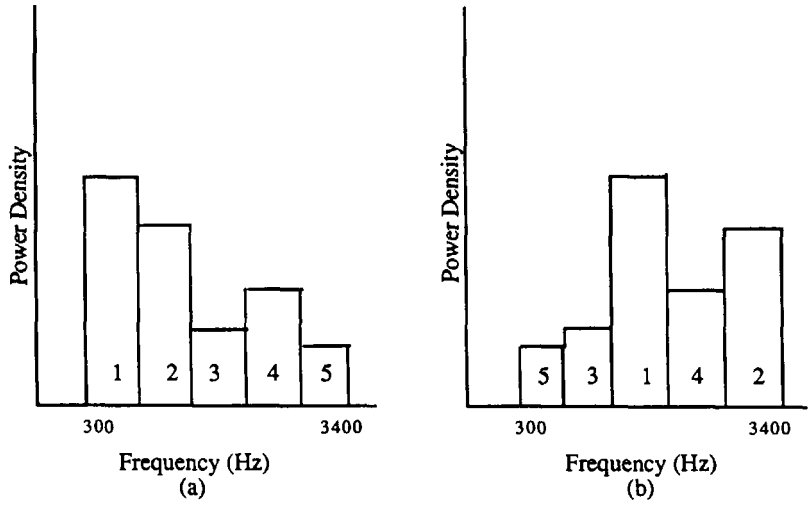

Figure 2

Bandsplitter (a) Original (b) Scrambled

A ciphertext only attack on the bandsplitter is possible, in which a pattern matching technique is used to return the scrambled spectral envelope to its original state. The attack requires a spectral template representative of the speech waveform. This template is used for comparison when trying to recover the original spectral envelope from the scrambled spectrum. For each frequency subband in the template the attack aims to locate the subband in the scrambled speech (ciphertext) which gives the best match. A mean square error (MSE) criterion is used to make this choice. The ciphertext subbands can thus be rearranged so that the scrambled spectrum conforms to the shape of the spectral template.

Recent work in speech coding has shown that the speech waveform can be represented by a finite number of speech parameter vectors or a codebook [5]. Consider a codebook in which the vectors contain spectral components obtained from a large set of training data. A codebook vector can be used as the required template for the above mentioned attack. A ciphertext only attack is proposed in which each of these codebook vectors is used to decrypt the scrambled spectrum under attack. The resulting decrypted spectrum giving the best match with its corresponding codebook vector following the attack is assumed to be the recovered speech spectrum. It is beneficial to use information from previous frames in order to recover the true permutation for a fixed permutation system. Following an attack on a given frame the position of each band is used to update a table. The table contains the frequency with which each band was assigned to a certain position. The correct permutation should be able to be constructed by selecting the bands with maximum frequency. The Hungarian assignment algorithm [6] was used to obtain an optimum solution from the table.

The attack process is outlined in the following using n frequency bands over N speech frames where k

codebook vectors are used.

Step 1: Transform ciphertext frame i to frequency domain.

Step 2: Partition into n bands.

Step 3: For codebook vector j = 1, ..., k do a pattern match and calculate mean square error.

Step 4: Record the permutation used to match the ciphertext vector to the codebook vector giving the minimum mean square error.

Step 5: Update possible permutation matrix.

Step 6: Use matrix to derive most probable permutation.

With a codebook size as small as 180 vectors it is possible to completely cryptanalyse eight and sixteen band frequency scramblers using the above ciphertext alone attack. The average number of half second frames required to make a successful attack were 16 and 26 for the eight and sixteen bands frequency scramblers respectively. The average time to attack in sec/frame was 4.6 and 10.6 seconds for the eight and sixteen band frequency scrambler respectively. These results demonstrate the poor security offered by the fixed permutation bandsplitter.

A bandsplitter encryption scheme, using a continuously varying permutation, was attacked in the same manner as the fixed permutation system. Using the codebook template matching technique, the best match was used as the decrypted spectrum for each frame in turn. Previous frames can no longer be used to provide information about the current permutation since the permutation is different for each frame. It is well known that if the most significant bands in each frame can be located correctly the resulting speech will be intelligible. The decrypted speech resulting from an attack on the eight and sixteen band systems was clearly recognisable. Table 2 shows the probability with which a given band was located correctly. The table shows that the first band in each case was positioned correctly with the highest probability. This band contains a large proportion of the voice speech energy and hence a high proportion of the speech information. It is clear from the table that as the number of bands was increased it became more difficult to correctly position the vital low frequency bands. The results of these attacks demonstrate the poor security offered by the bandsplitter even in the case where a different permutation is used for each frame.

Number of Bands : k = 8		Number of bands : k = 16	
Band	Probability	Band	Probability
1	1.00	1	0.83
2	0.53	2	0.67
3	0.46	3	0.48
4	0.29	4	0.32
5	0.49	5	0.29
6	0.43	6	0.29
7	0.44	7	0.28
8	0.67	8	0.21
		9	0.33
		10	0.32
		11	0.32
		12	0.27
		13	0.33
		14	0.32
		15	0.24
		16	0.46

Table 2
Results of Varying Permutation Attack on Bandsplitter

4. Cryptanalysis of DFT Scrambler

A DFT scrambler typically operates on speech frames containing 256 speech samples, giving a frame length of 32ms. Application of the DFT to these speech samples produces 256 spectral components. The spectral components with in the band of the channel undergo a permutation before the inverse DFT is used to return the frame to the time domain. In the scrambler simulated for the attacks described in this section 88 spectral components were permuted. The scrambling process is indicated in block diagram fashion in Figure 3. Permutation of spectral components results in the destruction of formant and pitch structures, leaving little information with which an attack can be performed. The scrambling process is frame based since the discrete transform operation is performed on a fixed number of time samples, N.

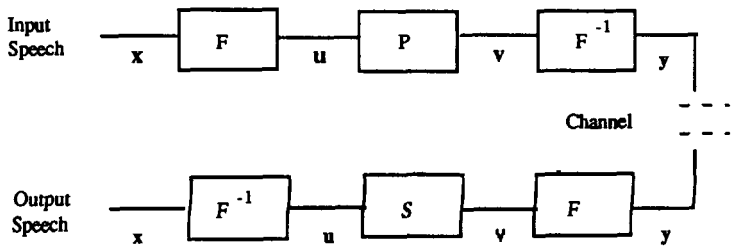

Figure 3

DFT Scrambler

The attack described for use with bandsplitter encryption devices is also applicable to the DFT scrambler. The only modification required is in the method for obtaining the plaintext/ciphertext mapping. In the case of the bandsplitter a frequency subband in the plaintext frame is mapped to a subband in the ciphertext frame. For the DFT scrambler however, it is necessary to map a single transform component in the plaintext frame to a single component in the ciphertext frame. This is achieved by locating the largest component in the plaintext frame and assigning it to the largest component in the ciphertext frame. This mapping is recorded and the next largest in the corresponding frames are assigned. This process continues until all components have been mapped. Again a MSE criterion is used to choose the codebook vector giving the best spectral match. The mapping used for this vector becomes the permutation estimate for the frame under attack.

The attack process is outlined in the following where 88 DFT coefficients are permutated, each codebook vector has dimension 88 and N speech frames are used.

Step 1: Obtain the 88 transform coefficients from time frame i. i = 1, ..., N.
Step 2: For codebook vector j = 1, ..., k pattern match the scrambled frame i to the codebook vector j and calculate M.S.E.
Step 3: Record the permutation giving the minimum M.S.E.
Step 4: Update possible permutation matrix.
Step 5: Use matrix to derive most probable permutation.

A DFT scrambler permuting 88 coefficients, with a frame size N = 256, was cryptanalysed using the procedure described above. The attack was performed on a system using a fixed permutation, as well as one using a constantly varying permutation. The attack was able to correctly position only a small number of coefficients, however in each case this was sufficient to recover intelligible speech. As expected, the fixed permutation attack gave more intelligible speech. The improvement in the intelligibility of the recovered speech in each case with respect to the original is shown in Table 3 which gives the % of components which are placed within two positions of their correct location. In the case of a fixed permutation the large number of components (23%) which are "close" to the correct position demonstrates the improvement in the intelligibility of the recovered speech.

Speech Segments Compared	% of Components
Original and Scrambled	3.7%
Original and Recovered (Using Varying Permutation)	8.9%
Original and Recovered (Using Fixed Permutation)	23%

Table 3
Results of DFT Scrambler Cryptanalysis

5. Conclusion

It has been shown that it is possible to conduct a computer cryptanalysis of several analog scramblers based on ciphertext alone. These attacks have demonstrated the poor security offered by many of the commonly used analog speech scramblers. The results from this paper can be used as a guideline for attacking speech ciphers as well as improving the security of existing scramblers. It should be noted that the actual attack on a cipher is dependent on the quality of the intercepted signal and the voice characteristics of the individual speaker.

References

1. H. Beker, and F. Piper, **Secure Speech Communications**, Academic Press, 1985.
2. J.M. Carroll, and S. Martin, "The Automated Cryptanalysis of Substitution Ciphers", **Cryptologia**, Vol. 10, 1986, pp 193 - 209.
3. J.M. Carroll, and L.E. Robbins, "The Automated Cryptanalysis of Polyalphabetic Ciphers", **Cryptologia**, Vol. 11, 1987, pp 193 - 205.
4. J.M. Carroll, and L.E. Robbins, **Computer Cryptanalysis**, Technical Report No. 223, Dept. of Computer Science, The University of Western Ontario, Dec. 1988.
5. R.M. Gray, "Vector quantization", **IEEE ASSP Mag.**, April 1984, pp 4 - 29.
6. N. Kitawaki, H. Nagabuch and Itok, "Objective quality evaluation for low bit rate speech coding systems", **IEEE Journal on Selected Areas in Communication**, Vol. 6, No. 2, February 1988, pp 282 - 248.
7. R.W. Llewellyn, **Linear Programming**, Hold, Rinehart and Windston, 1964.
8. A. Matsunaga, K. Koga, and M. Ohkawa, "An analog speech scrambling system using the FFT technique with high level security" **IEEE Journal in Selected Areas in Communications**, Vol. 7, May 1989, pp. 540 - 547.
9. S.R. Quackenbush, T.P. Barnwell, and M.A. Clements, **Objective Measures of Speech Quality**, Prentice Hall, 1988.
10. S. Sridharan, E. Dawson, B. Goldburg, "A Fast Fourier Transform Based Speech Encryption System", to appear in **IEE Proceedings Part I Communications**, Speech and Vision.

ACKNOWLEDGEMENT

This work was supported by a QUT Research and Development grant.