UDC 621.391.7:621.396
Indexing Terms: Cryptographic systems, Speech scrambling, Time element speech scrambling

A classification of time element speech scramblers

C. J. MITCHELL, BSc, PhD*

and

Professor F. C. PIPER, PhD†

SUMMARY

This paper contrasts four methods of time element speech scrambling (t.e.s.), which remains an extremely important cryptographic technique for narrow band channels, not least because of its robustness in poor transmission conditions such as that experienced on h.f. One method, hopping window t.e.s., although currently widely used, is increasingly being replaced by sliding window t.e.s. systems such as the other three methods described here. The importance of synchronization is emphasized, and three of the four systems described allow continuous synchronization, an overwhelming advantage except in the case when most transmissions are very brief and missing synchronization is not such a disadvantage.

* *Formerly with Racal Comsec Ltd, Milford Industrial Estate, Tollgate Road, Salisbury, Wiltshire SP1 2JG; now at Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, Bristol, BS12 6QZ*
† *Royal Holloway and Bedford New College, University of London, Egham Hill, Egham, Surrey TW20 0EX*

1 Introduction

The spoken word is man's most commonly used form of communication. But for many of the numerous communication channels, such as radio, telephone and satellite, the interception of conversations is both simple and widespread. Consequently if the communicators are discussing confidential information they use some form of scrambling device to make the transmitted signal unintelligible to an unauthorized interceptor. If the transmitter is using a particular scrambler then the authorized receiver must know precisely how it is being used and must have a descrambler to enable him to obtain the genuine message.

Speech scramblers have been used for many years and there are many different types available. For a broad classification they can be divided into two fundamentally different types: 'analogue' and 'digital'. The basic difference between these two types is the form in which the scrambled speech is transmitted. For an analogue scrambler, the output is an analogue signal which is a transform of the original speech signal where this transform normally takes place in the time and/or frequency domains. The output from a digital scrambler, on the other hand, is simply a sequence of binary digits. It is perhaps worth emphasizing that the distinction is in the form of the transmitted signal. For a number of analogue scramblers the speech is 'digitized' prior to being scrambled and the scrambled signal is then converted back to analogue form for transmission. For a general discussion of the various scrambling techniques and their applications we refer the reader to Refs 3 or 4.

In this paper we shall discuss a number of different types of analogue scramblers where the transform is performed in the time domain. In this type of scrambler, often called a time element scrambler, the speech signal is usually divided into time 'segments' (typically of 30–50 ms duration) which are rearranged before transmission. In particular we shall look at some of the mathematical problems associated with the design of these devices and the implementation problems which arise when using them.

One of the main problems associated with the use of time element scramblers (and in fact with any time domain scramblers) is that they necessarily impose a significant delay on the communications channel. This delay, which might be of the order of one second, may pose a serious problem and it is often necessary to take steps to reduce it. This is one of the problems which we shall discuss. The other problem is that of ensuring that the scrambler offers a high level of security. To do this it is necessary to ensure that the scrambler is able to use a large number of different rearrangement patterns for the segments. We shall discuss a number of the strategies currently used to ensure a large number of possible rearrangements while, at the same time, minimizing the time delay.

Although we shall not discuss them in this paper, we must point out that there are a number of other important techniques for speech scrambling, e.g. frequency domain scrambling and reversed time segmentation. The security level of devices using these techniques vary considerably but each one can be combined with a time element scrambler to increase the cryptographic strength.

2 A Classification of Time Element Scramblers

Time domain scrambling techniques can be divided into three broad categories:

(a) Reversed time segmentation
(b) Time element scrambling
(c) Time sample scrambling.

The most commonly used of these is undoubtedly the time element scrambling (or t.e.s.) and this is the method which we discuss. In other publications t.e.s. is also frequently called time division multiplexing (t.d.m.) or time segment permutation (t.s.p.).

In a time element scrambler, the analogue speech signal is digitized and stored prior to being processed. The digitized speech is then divided into short time 'segments' which are rearranged within the device and converted back to analogue form before being output. There are three major practical implementation problems associated with

this type of scrambler. They are:

(i) Time delays
(ii) Choice of the rearrangement patterns
(iii) Synchronization.

We consider each of them in turn.

2.1 Time Delays

A time delay is an inevitable consequence of using any type of time domain scrambler. The reason is clear. The scrambler needs time to accept some of the speech segments and to rearrange them before transmission. Similarly the descrambler needs time to rearrange the segments back into their original order. The precise length of this delay will vary according to the design of the scrambler and the various parameters chosen for a particular application. We shall be looking at this problem in considerable detail. However, we must point out that this problem cannot be considered in isolation. The parameters which affect the time delay are also likely to have a significant effect on the security level offered by the device.

2.2 Choice of the Rearrangement Patterns

Although the choice of the type of time element scrambler to be used affects the range of rearrangement patterns available, they all require some form of selection process to choose a set of 'good' patterns. For any type of time element scrambler there are a large number of rearrangement patterns which, if used, result in a transmitted signal that is not sufficiently different to the original speech signal. If one of these is used then an interceptor can, simply by listening very carefully to the scrambled signal, understand (or at least guess fairly accurately) some of the original message. This is a consequence of a phenomenon known as 'residual intelligibility'. It is sometimes very difficult to predict the level of residual intelligibility of a particular rearrangement. In fact it is often necessary to scramble a message using the given arrangement and to subject the scrambled signal to 'listener tests' in order to see how much of the message is intelligible.

There are many different methods for generating usable permutations. Devising and implementing a method for a given system is almost always a major part of the development program for a time element scrambler.

2.3 Synchronization

Apart from a few very simple, and usually minimally secure, scramblers (such as, for instance, bandsplitters), most analogue scramblers are affected by synchronization problems. There are, in general, two separate but related problems: one concerns initial synchronization and the other involves late entry synchronization.

As its name suggests the initial synchronization problem is concerned with the beginning of a transmission. Before they can conduct a secure conversation the communicators must synchronize their scrambling and descrambling devices. One way of achieving this is to include a special signal at the beginning of each transmission. So, for example, on a half-duplex system, this signal will be transmitted after every 'over'. While initial synchronization is needed for all systems, late entry synchronization is necessary if, for instance, it is desirable to enable a third party to join an existing conversation, if one of the original communicators has lost synchronization or, possibly most importantly, if the initial synchronization has been 'missed'. It is necessary for this synchronization signal to be present more or less continuously so that the receiver may commence descrambling at any stage of the message. Late entry synchronization can pose real problems for the designer but offers significant advantages to the user.

In Section 3 we discuss Hopping Window scramblers. They are probably the most simple type of time element scrambler but have the disadvantage of causing an unavoidably long time delay. To try to overcome this problem various types of Sliding Window scramblers have been introduced. Sections 4, 5 and 6 are devoted to three different sliding window systems. First, in Section 4, we look at the well-known frameless sliding window systems which have been described in a number of recent papers, see for example Refs 9, 10 or 11. These systems can offer a higher security level than most time element scramblers but, in practice, present a number of serious implementation problems. In an attempt to overcome these difficulties the overlapping frame sliding window scramblers of Section 5 and the disjoint sliding window systems of Section 6 were introduced.[1] The overlapping frame systems offer a very large number of 'good' rearrangements for a relatively short time delay. However, the only current known practical implementations of this type of scrambler involve repeated use of the same rearrangement pattern and this limits the level of security available. It was this fault which led to the introduction of the disjoint frame scramblers discussed in Section 6. They are, in some sense, a cross between hopping window scramblers and frameless sliding window ones. They can be implemented as easily as hopping window scramblers but have the advantage that they cause less delay.

3 Hopping Window Scramblers

There are numerous descriptions of these systems in the literature; see, for example Refs. 3, 4, 5, 8, 11, 12 and 13. We will give only a brief description of how they work and refer the reader to Ref. 4 for more details.

The analogue signal is first divided into equal time periods called frames. Each frame is then further subdivided into a fixed number n of smaller equal time periods called segments, where the length of a segment would typically be of the order of 30–50 ms. The scrambler then permutes the segments within each frame and they are transmitted in the new, permuted order. For each frame, the receiver knows the permutation used by the scrambler and is able to recover the original speech by applying the inverse permutation.

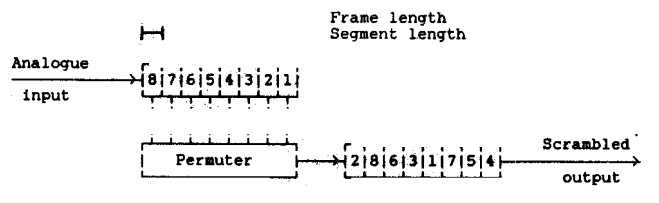

Fig. 1. Hopping window t.e.s.

A typical system with $n = 8$ is illustrated in Fig. 1.

There are many decisions to be made when designing and/or using a hopping window scrambler. Amongst them are the choice of n, the length of a segment and the selection of permutations to be used in the scrambler. Each of these decisions is likely to affect the others and they all have serious consequences for the entire system. One important fact, for instance, is that if the segment length is T seconds, then the total time delay for the system is $2nT$ s. So, for example, if $n = 8$ and the segment length is 50 ms the system delay will be 0·8 s. This is certainly long enough to be noticeable and for any larger choices for n or T the delay is likely to become unacceptably long. Yet,

clearly, the total number of permutations available to the scrambler is $n!$, and so if we make n too small we will seriously restrict the choice of permutations and, as a consequence, severely limit the security offered. Trying to choose n large enough to give a wide choice of permutations and small enough to keep the system delay acceptably short is a delicate problem! Despite this, hopping window scramblers have the great advantage that they are easy to implement. Furthermore, it is not difficult to arrange for a hopping window scrambler to have a continuous synchronization system allowing late entry at any time.

A typical system might involve using frames with 8 segments. Prior to its use, a number of 'good' permutations would be selected and stored. A pseudo-random number generator would then be used to choose the particular permutation to be used for the current frame. Every frame time (typically, say, 250–500 ms) the sequence generator would be reloaded with two different portions of key material. One part of this material, the base key, would be chosen by the user and would be fixed for a set period of time. The other, the message key, would be machine generated and be transmitted at the same time as the scrambled speech. Since the receiving unit obtains a full set of message key material during any frame time, it is able to start descrambling with the second complete frame of scrambled speech which it actually receives. This means that the facility of late entry causes a time lapse of the order of one second between switching on and being able to obtain the message. In most circumstances this is totally acceptable.

4 Frameless Sliding Window Scramblers

As we have already observed, although the delay for late entry is short, the major problem with hopping window scramblers is their inherent overall time delay. This is twice the frame length and is necessary because (i) the transmitting device needs to receive an entire frame before it can begin to scramble it and (ii) the receiver needs to receive an entire frame before it can commence descrambling it.

We now describe a system in which the requirements (i) and (ii) are removed and, as a consequence, the delay time can be shortened. We call these systems frameless sliding window scramblers and one of their characteristics is that the notion of a frame is dispensed with altogether. There are many possibilities for implementing a frameless system and we will describe one of them. We first choose a small positive integer k which, as we shall see, determines the system's time delay. At any given time the transmitter will then store the most recent k segments that it has received. At the start of each segment a pseudo-random number generator is used to select one of the k stored elements for transmission. This selection process is subject to the following two constraints:

(a) if the 'oldest' segment in the store, i.e. the segment received k segment times earlier, has not been sent, then it must be transmitted now
(b) no segment is ever sent twice.

It should be clear that (b) is necessary and that (a) has the effect of limiting the possibilities for the delay. In fact the inherent time delay in this type of implementation is $(k+1)T$ seconds, where the length of a segment is T s.

By using a frameless sliding window scrambler we can obtain a system which, for a time element scrambler, obtains an impressively high security level. There are a number of techniques for improving this level. One is to impose extra conditions on the usable permutations and only use those which ensure that the scrambled order of the segments is not too close to the original one. This can be achieved without putting too severe a restriction on the choice of rearrangement patterns and considerably reduces the level of residual intelligibility.

As we saw in Section 3, the time delay for a hopping window system is $2nT$ s, where n is the number of segments per frame. The value k for a frameless system can be chosen to be of the same order as this n to obtain a system which offers roughly the same security level as a hopping window system and at least as large a range of possibilities for the rearrangement patterns. Thus we have a system which offers the same level of security as a hopping window system, but with roughly half the time delay. However, these systems have a number of practical implementation problems. One major problem is synchronization; in particular, late entry synchronization.

If, in a system like the one we have just described, one wants to synchronize a receiver midway through a transmission, then it is necessary to send not only the key data which relate to the pseudo-random number generator (i.e. the message key), but also sufficient information to determine precisely which segments have already been transmitted at that particular time. This is a great deal of information and, in order to appreciate the difficulties of sending it, it is necessary to understand how data are sent simultaneously with the speech signal.

One common method of sending extra data simultaneously with the speech is to employ suitable filters to remove a specific small part of the frequency band. This small gap, which is often called the 'notch', is then used to transmit f.s.k. data at a low bit rate, typically 50 bits/s. (The precise bit rate is, of course, limited by the bandwidth of the notch.) Clearly the creation of this notch in the frequency band degrades the speech signal and, in order to minimize this degradation, it is not possible to make it very large. Thus this method only permits very limited data rates.

This method of transmitting data with the speech is not, of course, restricted to frameless sliding window systems. It is, in fact, stretched to its limit when used merely to transmit sufficient message key information for a hopping window system. So it is hardly surprising that it is difficult to meet the extra requirements of the frameless system. It might be possible to implement a frameless sliding window system with continuous synchronization where a synchronization update could be available, say, once every few seconds. However, the implementation would be very difficult and the result would be a highly complex system. As a consequence a number of variants of the sliding window scrambler, which incorporate some kind of 'frame' but reduce the synchronization problem to one of the same order as the hopping window system, have been devised. We will discuss two such possibilities in the next two Sections.

5 Overlapping Frame Sliding Window Scramblers

The system described in this Section uses the basic idea of a sliding window scrambler but, in a slightly different context, reintroduces the notion of a 'frame' consisting of a number of speech segments. Thus, as in a hopping window system, the speech is divided into frames of n segments, where each segment is T seconds long. However, our choice of permutations is now severely restricted and we use the permutations in a slightly different way.

Before designing the scrambler we choose an integer k which must be less than n. As we shall see, the total system delay will be equal to $(k+1)T$ s and this is a relevant factor in the choice of k. Thus if $k=16$ and $T=30$ ms, then the system delay will be 0·51 s. Note that the choice of n does

not affect this delay. Having chosen k we then choose our permutations in such a way that each segment is transmitted within kT seconds of entering the scrambler. This is achieved by limiting the scrambling permutations to those permutations t satisfying:

$$\overline{t(i)} \in \{\overline{i-1}, \overline{i-2}, \ldots, \overline{i-k}\} \text{ for each } i \text{ with } 1 \leqslant i \leqslant n,$$

where $\bar{i}$ denotes the residue class of i modulo n.

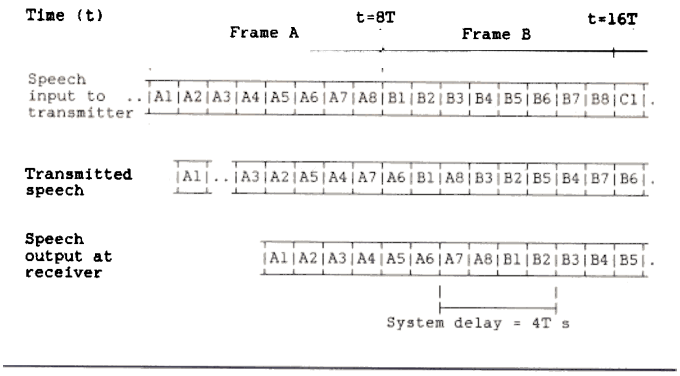

Fig. 2. Overlapping frame sliding window t.e.s.

In Fig. 2 we illustrate a system with $n=8$, $k=3$ and $t=[61832547]$. (There are many different ways for writing down permutations. In the one adopted here the order represents the order of the segments after scrambling. Thus, for our example, $t(1)=6$, $t(2)=1$ etc.) In the Figure we have used different letters to distinguish between frames. So, for instance, $A1$ is the first segment of the first frame while $B1$ is the first segment of the second.

The condition imposed in the permutation means that we know that each segment will be transmitted at most $3T$ seconds after it has been received. Thus the receiver can begin to scramble $4T$ seconds after the beginning of the message and the overall delay is $4T$ seconds.

By using frames this system allows the use of finite permutations on a fixed number of elements. However, we have lost the restriction that each complete frame must be transmitted before the scrambling of the next can begin, and it is this which enables us to decrease the time delay. Unfortunately, the system has a number of practical limitations. Most implementations are restricted to systems which reuse the same (albeit base key dependent) permutation of the segments and this limits the level of security provided. We will now describe one possible way in which such a system might operate.

First, we must choose values for k, n and T and, of course, when making these choices we must consider the effect of each decision on the total system. Once these choices are made we enter a base key, consisting of a set of bits, which is then used to select one of the permitted permutations. This permutation is then used to encrypt the speech in the way we have just described and decryption is achieved by using the inverse permutation.

Apart from the need to indicate to a receiving unit the beginning of a frame of speech data, this implementation removes the need to send any synchronization information. However, the repeated use of the same permutation imposes obvious security limitations. If the permutation is ever discovered by an interceptor, then he will be able to descramble the rest of the message as easily as the legitimate receiver. Although it is not easy to discover the permutation, especially if n and k are sufficiently large, this threat means we should regard our implementation as providing a fairly high level of privacy rather than full security.

Obviously, it would be nice to remove the restriction of having to use the same permutation, but there are two main reasons why this is not easy. The first arises from the fact that it is necessary to start transmitting one frame before transmission of the previous one has been completed. This means that, if we try to use different permutations for successive frames, the second permutation must be chosen so that it 'interlocks' with the first, i.e. it must not send a segment of the second frame into a position needed for one from the earlier frame. This poses extremely difficult implementation problems. The second reason concerns synchronization. In our little example, the repeated use of the same permutation made late entry easy. As soon as we try to use different permutations we need to send extra synchronization information. The 'interlocking' problem causes the volume of this information to be comparable to that needed for the frameless sliding window system of Section 4.

In the next Section we look at systems which are similar to the one we have just described except that they prevent the 'overlap' of the frames. Thus we will be looking at systems which have the ease of implementation associated with hopping window systems but have the shorter time delay of the sliding window systems.

6 Disjoint Frame Sliding Window Scramblers

For this type of scrambler we first determine our segment length, T seconds, and then choose a positive integer h. (The value of this integer will, as we shall see, affect the system delay.) We then choose a second positive integer n, which must be at least as large as h, and this determines the size of our permutations. We then use only those permutations t on $\{1, 2, \ldots, n\}$ with the property that $|i-t(i)| < h$ for all i; we call the set of all such permutations $C(n, h)$. The speech is divided into frames of n segments and, for each frame, one of the permitted permutations is then used to determine the order in which the segments of that particular frame are transmitted. An important characteristic of this system is that all the segments of one frame are transmitted before any segment of the subsequent frame is sent. One possible implementation is the following.

Suppose that a frame of speech commences at time $t=0$ and ends at $t=nT$. Suppose, also, that an appropriate permutation p has been chosen to re-order the segments of this frame. For convenience we will label the segments $1, 2, \ldots, n$ so that segment i lasts from time $(i-1)T$ to iT. The segments of the frame are transmitted between times hT and $(n+h)T$ in such a way that, for any i between 1 and n, the segment transmitted between times $(h+i-1)T$ and $(h+i)T$ is $p(i)$. It should be clear that, for any i, the receiver will have received segment i by the time $(2h+i)T$, and so the receiver will be able to start outputting the speech (in its original order) at time $t=2hT$ and that the frame will be totally output by the time $t=(2h+n)T$. Thus the total delay is $2hT$ seconds.

As an illustration of this type of system we consider an example with $n=8$ and $h=2$. Suppose that $p=[21354768]$ is used to permute the segments of the first frame, and $q=[13254687]$ is used for the second. Just as in Fig. 2, let $A1, A2, \ldots, A8$ be the segments of the first frame and $B1, B2, \ldots, B8$ be the segments of the second. Then Fig. 3 shows how the system works.

Note that if we use this system without changing the permutation, then we merely have a special type of overlapping system with $k=2h-1$. On the other hand we can also regard this system as a special type of hopping window system, with the same value of n, where the added restriction involving h means that the transmission of the

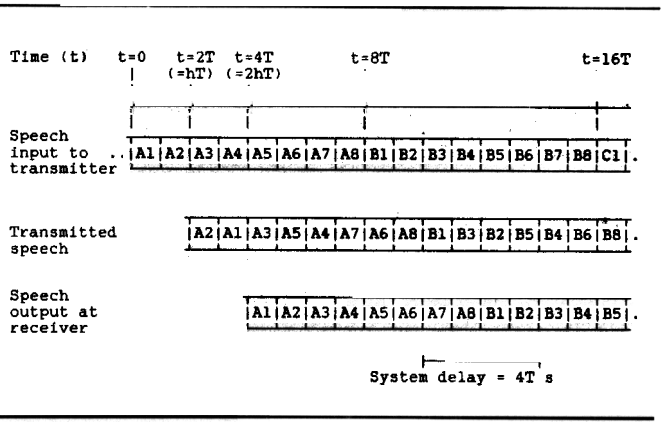

Fig. 3. Disjoint frame sliding window t.e.s.

segments of a frame can commence before the entire frame has been spoken. The implementation problems involved in this system are essentially the same as those of the hopping window scramblers.

One possible method of implementation might be to store a preselected set of permissible permutations within the device and then allow a pseudo-random number generator to select which one is to be used for the next frame. This is, of course, the method suggested for hopping window systems and we can also use the synchronization method which we suggested for them. Thus the synchronization signal involves the transmission of message key material only. The advantage of these systems over hopping window scramblers is that, for the same choice of n, h can be chosen to decrease the delay or, alternatively, for the same delay n can be increased so as to increase the number of permutations and, as a consequence, offer more security.

7 Permutation Selection and Enumeration

One problem which is common to all the scramblers discussed here is the choice of usable permutations. Once one has decided the limitation to be placed upon them, then it is necessary to enumerate the available permutations.

For hopping window systems, any permutation can be used and there are, therefore, $n!$ possible permutations. Of course, they are not all equally effective and some of them, for instance the identity, leave too much residual intelligibility. This last comment applies to all the enumerations and it must not be forgotten that after we have found all the available permutations they must be subjected to listener tests to determine the level of security they provide.

For frameless sliding window scramblers the rearrangement patterns are, essentially, infinite. Thus the number of available permutations cannot be evaluated in the same way. However, some assessments of the variety of rearrangement patterns over a specified period of time are given in Ref. 9.

For overlapping frame sliding window scramblers we must only use permutations t on $\{1, 2, \ldots, n\}$ such that, for each i, $\overline{t(i)} \in \{\overline{i-1}\ \overline{i-2}, \ldots, \overline{i-k}\}$. This set is denoted by $A(n, k)$ and, if we denote the number of permutations in $A(n, k)$ by $a(n, k)$, then it is very hard to determine $a(n, k)$ for arbitrary values of n and k. In fact, $a(n, k)$ is equal to the permanent of the cyclic $(0, 1)$ n by n matrix having $(11\ldots100\ldots0)$ as its first row, where the number of 1's in this vector is k. It is well known, see for example Ref. 7, that evaluating permanents is a difficult problem. The specific problem of evaluating $a(n, k)$ is discussed in Ref. 2.

For disjoint frame sliding window scramblers the enumeration problem is once again equivalent to the problem of evaluating the permanent of an n by n $(0, 1)$ matrix. This matrix has 1 in its (i, j) position if and only if $|i-j| < h$. The evaluation of this permanent is again a difficult problem and is the subject of current research.

As we have already observed, these enumerations determine the number of permutations which can be used in theory. In practice the number of them which scramble the speech effectively is considerably smaller. This problem is discussed in considerable detail in Ref. 4. Practical experiments suggest, see for example Ref. 6, that the most important extra constraint to put on the permutations is that no two segments which were consecutive in the original speech should remain consecutive after scrambling. Obviously the imposition of this extra restraint will decrease the number of usable permutations. We now consider how the number decreases.

For hopping window scramblers the permutations excluded by this new condition are precisely those permutations t for which, for some i, $t(i)+1 = t(i+1)$. (Note that this claim is not quite accurate. If we were to use two permutations p and q on consecutive frames where $p(n) = n$ and $q(1) = 1$, then the last element of the first frame and the first segment of the second would remain consecutive in the scrambled signal. However, having observed this possibility we will ignore it!) The number of usable permutations is now equal to $D(n)+D(n-1)$, where $D(n)$ is the number of 'derangements' on n letters, i.e. the number of permutations of n objects having no fixed elements. This assertion is well known and easily established (see, for example, exercise 20 on page 160 of Ref. 14). Since $D(n)$ is easily evaluated, one simple recurrence being $D(n) = nD(n-1)+(-1)^n$, our enumeration is simple.

For a discussion of the variety of rearrangements for frameless sliding window scramblers, the reader is referred to Ref. 9.

For overlapping frame sliding window scramblers, in order to enumerate the number of permutations which satisfy our extra condition, we introduce a new set of permutations $B(n, k)$. This set $B(n, k)$ consists of all those permutations t from $A(n, k)$ such that, for all $i < n$, $\overline{t(i+1)} \neq \overline{t(i)+1}$ and $\overline{t(n)+1} \neq \overline{t(1)}$ where, as before, the bars denote residue classes modulo n. If, as in the implementation described in Section 5, the same permutation t is used repeatedly, then using permutations from $B(n, k)$ ensures that no segments which were consecutive in the original speech, even at the frame boundaries, will be transmitted consecutively. So, in this case, we have solved the 'boundary' problem which we ignored for hopping window scramblers. For general n and k, the size of $B(n, k)$ is unknown but, for small values of k, some recurrences are given in Ref. 2.

Finally, for the disjoint frame sliding window time element scramblers, the number of permutations which satisfy the adjacency restriction is equal to the size of the set $D(n, h)$, where $D(n, h)$ is the set of permutations t in $C(n, h)$ such that $t(i)+1 \neq t(i+1)$. (The set $C(n, h)$ was defined in the last Section.) Note that, since most implementations of these systems involve using a different segment with each frame,' even if we restrict our permutations to $D(n, h)$, we still cannot guarantee that segments which are consecutive but lie in different frames will not be transmitted consecutively. The evaluation of $|D(n, h)|$ is a problem which needs to be investigated.

Clearly the enumeration problems raised in this paper have an important relation to the security level of various types of scrambler which we have discussed. Thus these combinatorial problems are of sufficient interest and application to merit continued research.

Table 1. Comparison of four types of time element speech scrambling

Type of t.e.s.	Level of security available	Continuous sync possible?	Typical system delay
Hopping window	potentially good	yes	relatively long—typically 1 s
Frameless sliding window	potentially good	no	relatively short—typically 0·5 s
Overlapping frame sliding window	limited—fixed permutation only	yes	relatively short—typically 0·5 s
Disjoint frame sliding window	potentially good	yes	relatively short—typically 0·5 s

8 Comparison

We summarize the comparison of the four types of t.e.s. in Table 1. Note that in this context the term 'good' means good for a time element scrambler. We make no attempt to compare t.e.s. with other cryptographic techniques.

9 References

1 Beker, H. J., 'Analogue speech security systems', 'Cryptography'. (Proceedings Burg Feuerstein, March/April 1982) *Springer-Verlag Lecture Notes in Computer Science*, **149**, pp. 130–46, 1983.

2 Beker, H. J. and Mitchell, C. J., 'Permutations with restricted displacement', (Under consideration for publication in *SIAM J. on Algebraic and Discrete Methods.*)

3 Becker, H. J. and Piper, F. C., 'Cipher Systems' (van Nostrand, UK, 1982).

4 Beker, H. J. and Piper, F. C., 'Secure Speech Communications' (Academic Press, New York, 1985).

5 Beth, T., Hess, P. and Wirl, K., 'Kryptographie' (Leitfaden der ang. Informatik, Teubner, Stuttgart, 1983).

6 Bromfield, A. J. and Mitchell, C. J., 'Permutation selector for a sliding window time element scrambler', (Under consideration for publication in *J. Instn Electronic & Radio Engineers.*)

7 Garey, M. R. and Johnson, D. S., 'Computers and Intractability: A Guide to the Theory of NP-Completeness' (Freeman, Oxford, 1979).

8 Hess, P. and Wirl, K. 'A voice scrambling system for testing and demonstration', 'Cryptography'. (Proceedings Burg Feuerstein, March/April 1982) *Springer-Verlag Lecture Notes in Computer Science*, **149**, pp. 147–56, 1983.

9 Hong, S. T. and Kuebler, W., 'An analysis of time segment permutation methods in analog voice privacy systems', Proc. 1981 Carnahan Conf. on Crime Countermeasures, University of Kentucky, 1981, pp. 167–71.

10 Jayant, N. S., 'Analogue scramblers for speech privacy', *Computers and Security*, **1**, pp. 275–89, 1982.

11 Jayant, N. S., Cox, R. V., McDermott, B. J. and Quinn, A. M., 'Analogue scramblers for speech based on sequential permutations in time and frequency', *Bell Syst. Tech. J.*, **62**, pp. 25–46, 1983.

12 Jayant, N. S., McDermott, B. J., Christensen, S. W. and Quinn, A. M., 'A comparison of four methods for analog speech privacy', *IEEE Trans on Communications*, **COM-29**, pp. 18–23, 1981.

13 MacKinnon, N. R. F., 'The development of speech encipherment', *The Radio and Electronic Engineer*, **50**, pp. 147–55, 1980.

14 Tucker, A., 'Applied Combinatorics' (Wiley, New York, 1980).

Manuscript first received by the Institution on 21st March 1985 and in final form on 2nd May 1985
Paper No. 2211/COMM401