THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 1, January 1983
Printed in U.S.A.

Analog Voice Privacy Systems Using TFSP Scrambling: Full Duplex and Half Duplex

By R. V. COX and J. M. TRIBOLET *

(Manuscript received July 20, 1982)

In this paper we present possible full-duplex and half-duplex analog voice privacy systems that have been simulated over real channels. Previous papers have been concerned primarily with the issues of the strength of a system (i.e., unintelligibility to the casual eavesdropper and relative cryptanalytical strength for the sophisticated eavesdropper) and the amount of delay of a system. Well-known but not addressed have been the problems of decoding the scrambled signal in a real-channel environment. At the heart of the encryption systems proposed here is the sequential time and frequency segment permutation structure proposed by Jayant and Cox. This structure relies on digital processing to divide the signal into sub-bands and then to permute these bands in both time and frequency simultaneously to synthesize the scrambled analog signal. In discussing the decoding we address the issues of compensating for the properties of the channel, re-sampling the analog signal, and establishing and maintaining synchronization between the de-scrambler and the scrambler.

I. INTRODUCTION

In this paper we discuss a possible analog voice encryption scheme. Although the channel signal is analog, all of the signal processing is done digitally. This gives us a flexibility that analog processing cannot give, and thereby allows us to do simultaneous time and frequency permutation. It also allows us to perform synchronization and equalization at the receiver. With the advent of the Digital Signal Processor (DSP),[1] digital processing becomes economically competitive as well as feasible.

* This work was performed while Mr Tribolet was on sabbatical leave from the Instituto Superior Tecnico, Lisbon, Portugal.

The systems that we will discuss in this paper are a full-duplex system and a half-duplex system. We have assumed either system would be used over all standard telephone channels with a bandwidth of 200 to 3500 Hz. Because of the delays involved in the signal processing, echoes could become a problem, so we chose to look at solutions that would avoid the echo problem. Alternatively, echoes might be cancelled using an appropriate algorithm, but this problem was not addressed in this study. One such solution to the echo problem is to choose a half-duplex system. Alternatively, for a full-duplex system we have divided the telephone band into three segments: 200 to 500 Hz is used for signalling and equalization by both users, 500 to 2000 Hz is allocated to the first user, and 2000 to 3500 Hz is allocated to the second user. In this way faulty echo cancellation need not impair the quality of the speech, since the two speakers use disjoint bands. In a half-duplex system only one user could speak at a time and would be allocated the entire channel except for the frequencies used for equalization.

In this study we have relied almost entirely on computer simulation of the algorithms involved. However, the algorithms were designed with future implementation on the Bell Laboratories DSP in mind.[1] The DSP is a powerful, single-chip, programmable processor that is especially suited for performing digital signal processing functions. It has an 800-ns machine cycle that is established by a 5-MHz clock. It contains provision for a 1024- × 16-bit read-only memory (ROM) for storage of the program, tables, and coefficients. A 128- × 20-bit random-access memory (RAM) is available for the storage of dynamic data and state variables. Our experience in working with the DSP has shown that the three constraints of RAM, ROM, and execution speed all come into play in the final design of the algorithm.

In Section II we give an overall description of the system. Subsequent sections deal with the major components of the system and the problems we addressed and solved for these components. Section VIII presents conclusions.

II. THE OVERALL SYSTEM

Figures 1a and 1b are overall block diagrams for the proposed full-duplex system's transmitter and receiver. At the transmitter an analog speech signal $s(t)$ is first filtered and sampled at 8 kHz by a standard analog/digital (A/D) converter, such as the M7062 coder-decoder (CODEC). The resulting digital signal is denoted as $s(n)$. This signal is then fed to a filter bank that splits $s(n)$ into three frequency components: $s_1(n)$ is the 0- to 500-Hz band, $s_2(n)$ is the 500- to 1000-Hz band, and $s_3(n)$ is the 2000- to 2500-Hz band. Each of these signals has its sampling rate reduced from 8 kHz to 1 kHz as part of the filtering

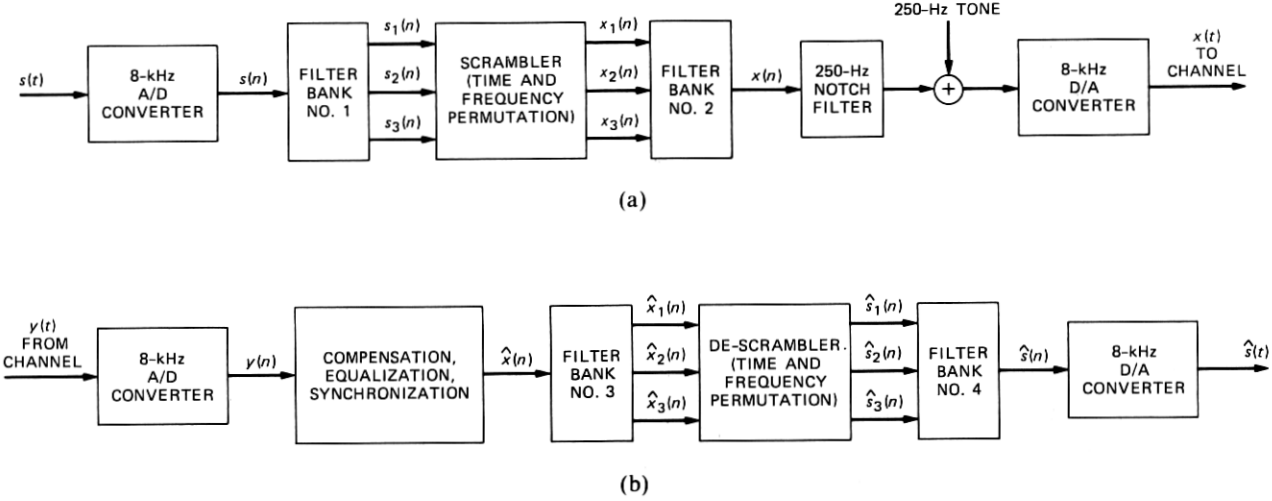

Fig. 1—Block diagram of the proposed full-duplex system. (a) Transmitter. (b) Receiver.

process. For the half-duplex system there are five bands, including these three and also the bands 1000 to 1500 and 1500 to 2000 Hz. The sub-band signals are then fed to a time-and-frequency block scrambler. We have been using a block size of 5 ms. The scrambler uses a pseudo-random key to output three or five blocks from its buffer. These output signals, $x_i(n)$, are fed to a second filter bank. The second filter bank is a synthesis filter bank using quadrature mirror filters. For the full-duplex system it combines the three signals into either a 500-Hz to 2-kHz band or a 2-kHz to 3.5-kHz band, and each user gets one of these bands. In the half-duplex system the five bands are combined into a 500- to 3000-Hz band. As mentioned previously, this division of the available bandwidth can be used to prevent echoes.

At start-up and as required subsequently an impulse is transmitted to equalize the channel at the receiver. A 250-Hz tone is also transmitted continuously to compensate for frequency shift. For this reason $x(n)$ is filtered by a 250-Hz notch filter before adding the 250-Hz tone. Then the synthesized signal is fed to a standard digital/analog (D/A) converter and transmitted as $x(t)$.

The received signal is denoted as $y(t)$ because a number of changes to $x(t)$ may have taken place. The job of the synchronization and equalization subsystem is to produce an output $\hat{x}(n)$ that is as close to $x(n)$ as possible. To produce $\hat{s}(t)$ with as little noise as possible, $\hat{x}(n)$ must have sample-to-sample integrity with $x(n)$. A small slippage of the sampling instants of the receiver relative to the transmitter will result in an error at the ends of the blocks, thus producing a frame-rate-type noise. Because the filters are long the time duration of the noise is much longer than the relative slippage. More details are given about this subsystem in the section on synchronization and equalization. Next, the signal $\hat{x}(n)$ is fed to the receiver analysis filter bank (filter bank #3). This filter bank uses the same quadrature mirror filter (QMF) structure as filter bank #2 (a synthesis filter bank). Because the same QMFs are used, $x_i(n)$ can be almost exactly recovered in the absence of noise and other distortion in the channel. Only QMFs have this property.[2] This is because they are bandwidth preserving. The scrambling process increases the bandwidth of the signal. The individual bands can only be recovered if the increased bandwidth is fully preserved. The de-scrambler delays the time-and-frequency-permuted blocks by an amount complementary to their delay in the transmitter. In this way all blocks experience the same total delay and can be output in the correct order—oldest blocks first. Filter bank #4 is the synthesis counterpart to filter bank #1. Presumably, the $\hat{s}_i$ have the same bandwidths as the s_i and so QMFs are not necessary. However, in order that as much as possible of the 0- to 1000-Hz bandwidth be preserved in the full-duplex system, it is recommended that a QMF be

used to split the 0- to 1000-Hz band and subsequently to reconstruct it. The signal $\hat{s}(n)$ is then fed to a standard D/A converter to form the decrypted output signal $\hat{s}(t)$. For the full-duplex system $\hat{s}(t)$ is not a full-bandwidth signal. Its frequency content is limited to 0 to 1 kHz and 2 to 2.5 kHz. As a result it sounds somewhat muffled. Nevertheless, it seems quite intelligible.

In the following sections details will be presented on the major subsystems: the filter banks, the scrambler and de-scrambler algorithms, and the synchronization-equalization system.

III. THE FILTER BANKS FOR THE FULL-DUPLEX SYSTEM

Considerable time and effort went into the design of the filter banks used here. Our primary constraint was to come up with a set of filter banks such that each filter bank would occupy only one DSP. At the same time the overall structure must have fairly sharp filters to preserve as much bandwidth as possible of the original speech, since this in turn preserves intelligibility. Also, aliasing between speakers had to be prevented.

Figure 2 shows the structure of filter bank #1. The signal $s(n)$ is first split into two bands (0 to 1 kHz and 2 to 3 kHz). A 95-tap finite impulse response (FIR) filter is used for each band. Since both filters use the same input data the memory required can be shared by both. That way only 95 RAM locations are used. The output of each filter is decimated by four since each filter accomplishes a 4:1 bandwidth reduction on the original signal. This also reduces the computation load on the processor. The output of the low-band filter is fed to a QMF pair to split the signal by another factor of two, producing s_1 and s_2. These two filters can also share RAM memory, thus using 16 additional locations. The high-band signal is low-pass filtered using the same QMF to produce s_3. This requires 16 more RAM locations in the DSP. In all we have used 127 DSP RAM locations. One way of

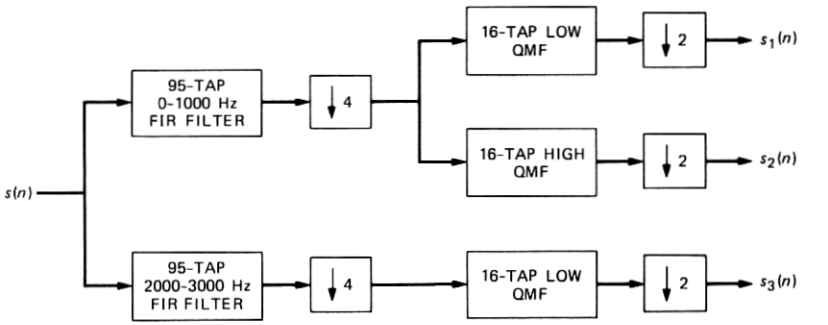

Fig. 2—Structure of filter bank #1 for the full-duplex system.

viewing this structure is that for every eight input samples there will be three output samples. Thus, there is ample computation time for the DSP.

Filter bank #4 uses the same filters. Its structure is shown in Fig. 3. Each QMF requires only eight RAM locations since the input signals are being interpolated 2:1. This accounts for 24 RAM locations. The 95 tap filters each require 24 RAM locations since they are interpolating 4:1. Thus, only 72 RAM locations are used. Larger filters might have been used but they do not add appreciably to the quality. The number of multiply-accumulates per output point is also 72, since each RAM location must be accessed once per output sample. This is also ample time for the DSP.

Figure 4 shows the structure of filter bank #2. The three outputs from the scrambler are first interpolated by 16-tap QMFs. This requires eight RAM locations for each filter, or 24 total. The two outputs from x_1 and x_2 are combined using a QMF pair before the next stage. The two outputs from the first stage are combined using a 32-tap QMF pair in the second stage. This produces a 4-kHz sampled signal with a bandwidth of 500 to 2000 Hz. Depending on whether the user is assigned to the upper or lower band, the next QMF is either high or low. The second stage required 32 RAM locations as does the third stage. This gives a total RAM requirement of 88 and also means 88 multiply accumulates are performed per output point. The 64-point QMF is a fairly sharp filter with a large rejection band. It is quite adequate to keep separate the encrypted speech of the two talkers. All of the QMFs were designed by Johnston.[3]

Filter bank #3 uses the same QMFs for analysis and its structure is shown in Fig. 5. The encrypted speech is first filtered by a 64-tap QMF. This is a high- or low-pass filter, depending on the band assigned. The output is decimated 2:1 to put the signal on the 0- to 2000-Hz band. A 32-tap QMF pair splits this band and then 16-tap QMF filters are used to do the final splitting. Stage 1 requires 64 RAM locations and stages

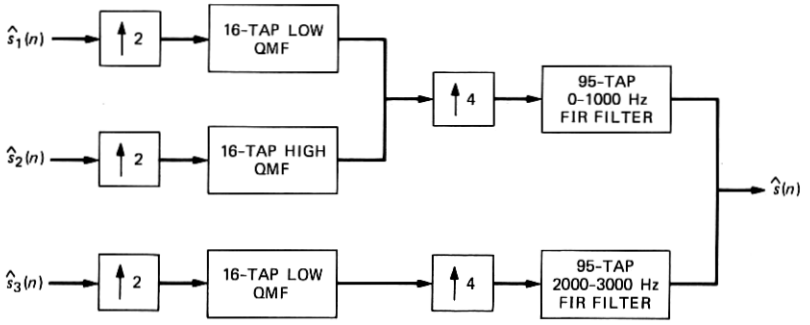

Fig. 3—Structure of filter bank #4 for the full-duplex system.

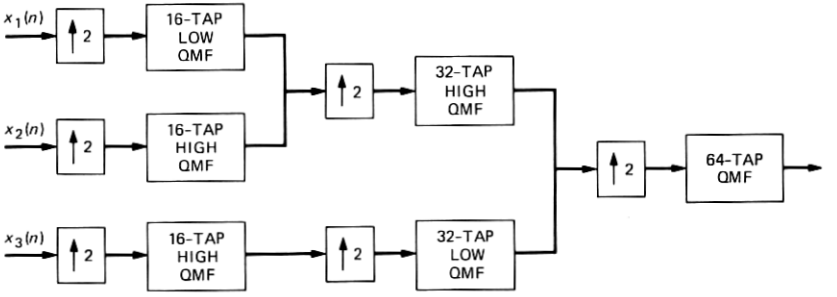

Fig. 4—Structure of filter bank #2 for the full-duplex system.

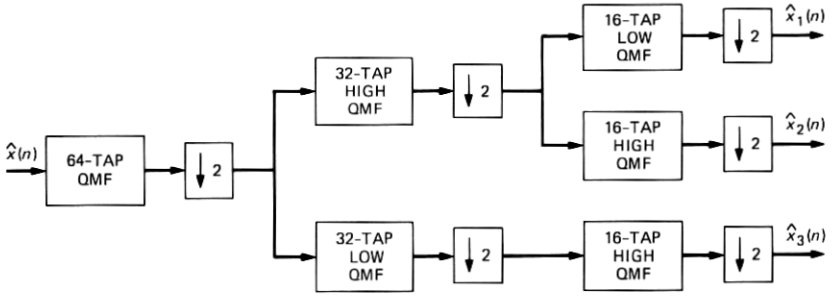

Fig. 5—Structure of filter bank #3 for the full-duplex system.

2 and 3 require 32 each, thus using all 128 RAM locations in a DSP. As in the other analysis filter, eight input samples produce three output samples, thus allowing adequate time for the computations.

The reason for using quadrature mirror filters in filter banks 2 and 3 is worth reiterating. Since we do segment permutation both within and between bands, the bandwidth of the signals x_1, x_2, and x_3 will be the full 500 Hz. If we chose a filter that did not preserve the full bandwidth, these signals would be altered. The result would sound like frame rate noise or "burble." By using the QMF structure the bands will be preserved to within 30 dB with no loss of bandwidth. Conversely, if we chose a full bandwidth filter other than a QMF, then when the bands were combined they would alias with each other. This also happens with a QMF, but when they are separated by the same QMF in filter bank #3 the aliasing is cancelled. This is not the case with an ordinary full-bandwidth filter. This is why the QMF structure is essential.

IV. THE FILTER BANKS FOR THE HALF-DUPLEX SYSTEM

For the half-duplex system quadrature mirror filters were used in all the filter banks. The five bands of the input speech that are used are

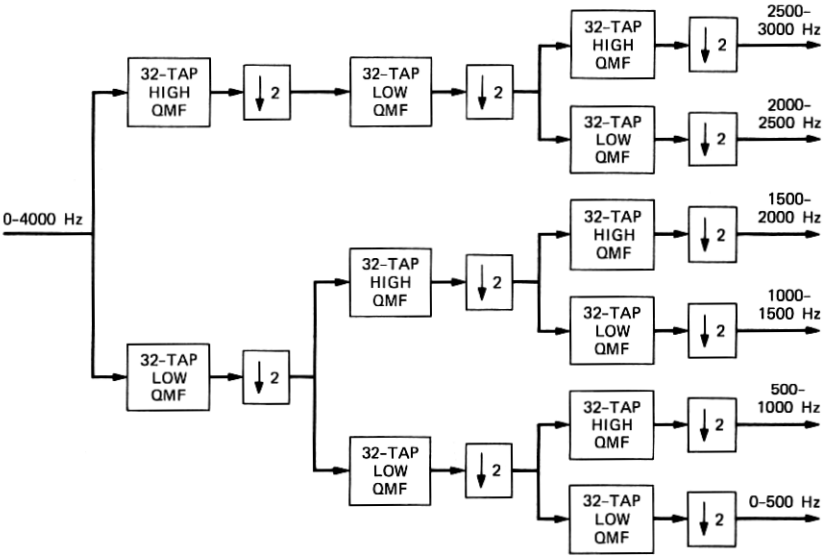

Fig. 6—Analysis filter bank for the half-duplex system.

0 to 500, 500 to 1000, 1000 to 1500, 1500 to 2000, and 2000 to 2500 Hz. A three-stage filter bank structure was devised as shown in Fig. 6. In the first split a 32-tap filter divides the 0- to 2000- and 2000- to 4000-Hz bands. In the next stage the same 32-tap filter was used to divide these two bands into four bands: 0 to 1000, 1000 to 2000, 2000 to 3000, and 3000 to 4000. The highest band was discarded. In the third stage each of these three bands is again split using the 32-tap QMF. After each stage of filtering 2:1 decimation takes place. Thus, each band is sampled at 1000 Hz. For the transmitter the 2500- to 3000-Hz band is discarded. However, for the receiver it is the 0- to 500-Hz band that is discarded.

The synthesis filter bank counterpart is shown in Fig. 7. For the transmitter nothing is put in the 0- to 500-Hz band, since it is used later for the phase-roll compensation. For the receiver nothing is put in the 2500- to 3000-Hz band.

These filter banks require more RAM memory than the full-duplex banks did. The analysis filter bank requires 192 memory locations while the synthesis filter bank requires 160 memory locations. No structure could be found that used less than 128 locations and also gave good performance.

V. THE SCRAMBLER AND DE-SCRAMBLER

The scrambler and de-scrambler are based on Jayant's idea of a rolling code scrambler[4] with one new twist. Figure 8 shows a memory

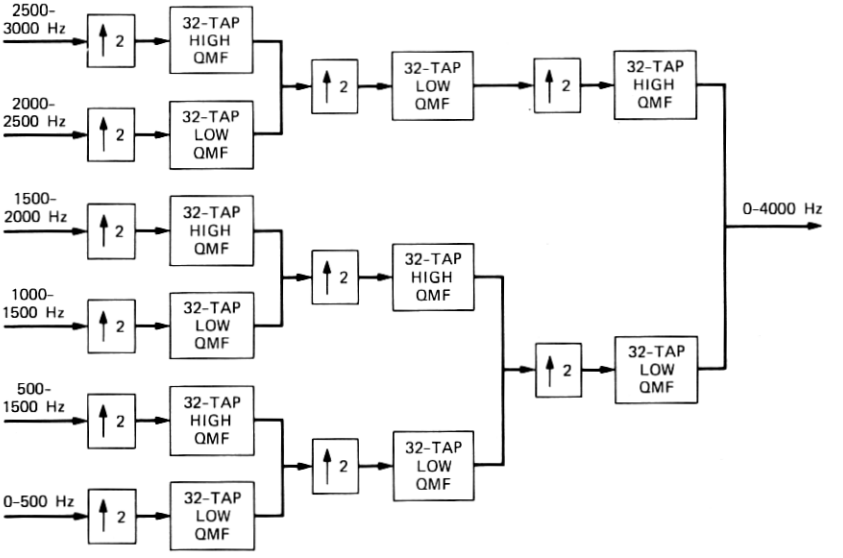

Fig. 7—Synthesis filter bank for the half-duplex system.

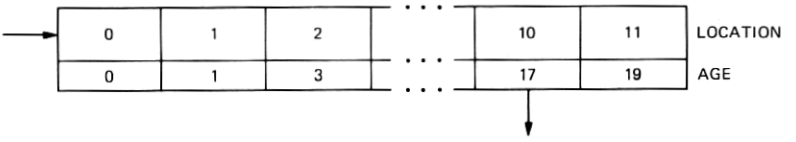

Fig. 8—Scrambler buffer.

buffer. The locations actually correspond to blocks of memory. The age is the age of the block in that location. At any instant data is always being entered into the leftmost block; hence its age is always zero. One of the blocks is also being emptied from the buffer. In this example it is block 10. The algorithm for choosing the block to output is derived as follows. First, the rightmost block (block 11, here) is checked to see if it has reached its maximum age. It if has, then it must be transmitted. Otherwise a random number generator is used to pick a random integer from 0 to 11, and that number determines the block chosen. Once a block is emptied, all blocks to its left are shifted right one location, thus freeing up block 0 for a new input.

The de-scrambler buffer shown in Fig. 9 is similar to the scrambler buffer. Here we see block 10 from Fig. 8 being entered into the buffer. Block 23 is always the oldest block in the buffer and so it is chosen for emptying. Other blocks in this buffer are kept in order of increasing age. So we see that the structure for the two buffers is identical. The method chosen for selecting the output block from the scrambler

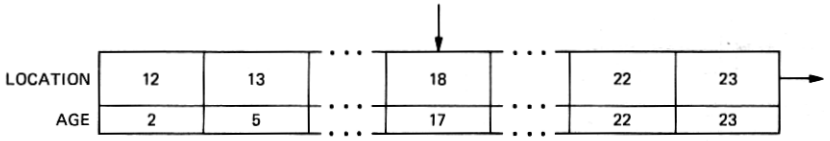

Fig. 9—De-scrambler buffer.

buffer is equivalent to determining where to place the input block in the de-scrambling buffer. Note that the total delay for all blocks will be the same. All of this is exactly the same as Jayant's ideas.

In any real system provision must be made for resynchronizing the de-scrambler if an error occurs at the receiver. This is the basis for modifying Jayant's ideas. The new idea presented here is to periodically reset the transmitter or scrambler buffer to its initial state. In its initial state the age of each block corresponds to its location, i.e., block 11 is only age 11, etc. Suppose we want the period between re-initialization to be N and we have a buffer with M blocks in it. At times 1 through $N - M$ we use Jayant's ideas as just described to select the output block. At time $N - M + 1$ we eliminate block 0 as a candidate for transmission (we still have $M - 1$ blocks to choose from). At time $N - M + 2$ we eliminate both blocks 0 and 1. We continue in this manner, restricting ourselves to a smaller set of blocks each time. At time $N - 2$ we can only choose block $M - 2$ or block $M - 1$ and at time $N - 1$ we can only choose block $M - 1$. At that point the buffer will have been re-initialized, i.e., at time N the locations and ages of the blocks will be in correspondence, just as they were at time 1. At time N we substitute a synchronization pulse for block $M - 1$. Note that the eavesdropper without the key will still be unable to tell what block is coming out at any given time, even though one can tell when the key ends by detecting the sync pulse.

In our proposed system each block represents a frequency band as well as a time. At any given instant three or five blocks are being entered rather than one. This easily can be accounted for without modifying the algorithm very much. At any time we have three or five output pointers instead of one. We just read the key once for each band. The 3- or 5-band structure can be used to our advantage in doing the re-initialization. Suppose we let N and M be multiples of three for the full-duplex case or five for the half-duplex case. The N-th entry in the key will correspond to the output for the highest band. At time N we are supposed to transmit the known sync signal. Because of the way we constructed the key, the omitted data in block $M - 1$ will also be from the highest band and therefore its loss will cause less degradation to the speech than if it were from a lower band. At the receiver the reception of this known signal indicates that the scrambler buffer

has been re-initialized and so the de-scrambler buffer should also be re-initialized. While this procedure will allow a potential eavesdropper to know the period of the key, it will not give away the key or any more information than before. At the receiver when the known sync signal is received, zeros are substituted so that the sync signal is not heard as part of the decrypted signal. The momentary loss of bandwidth in the decrypted signal does not affect intelligiblity while the signaling "chirps" that we send are quite noticeable in the encrypted signal.

We note that for the buffer sizes shown in Figs. 8 and 9, here, namely 12 blocks with five samples per block, a single DSP is more than adequate for use as the scrambler buffer. A DSP would be sufficient for a buffer twice this size.

VI. SYNCHRONIZATION AND EQUALIZATION

Our system uses the same signals for both synchronization and equalization and therefore we discuss them together. First we discuss what we mean by these terms. Then we address how the problems are solved.

As mentioned previously sample-to-sample integrity is essential if a good quality $\hat{s}(t)$ is to be recovered from $y(t)$. This means both that the input samples $\hat{x}(n)$ should be as close as possible to $x(n)$ and that the state of the de-scrambler should exactly match that of the scrambler. Thus, there are two types of synchronization to perform. We must synchronize the de-scrambler state with the scrambler state and match $\hat{x}(n)$ with $x(n)$.

The signaling chirp indicating the re-initialization of the transmitter is used to synchronize the state of the receiver with the transmitter. This chirp can be detected to one of two ways. One way is to use a matched filter to detect the chirp. The advantage of this method is that it can be used on the full-band, 8000-Hz sampled signal. The alternate method is to use the filter bank structure and apply a match filter to $\hat{x}_3$ or $\hat{x}_5$. This method will work provided that the decimation cycles of the filter bank structure are in the proper phase. In general the odds of randomly selecting the right phase are 7:1 against. For this reason we use the first method for start-up. After sync is established the second method can be used to check sync.

To match $\hat{x}(n)$ with $x(n)$ we need to do channel equalization so that the two sets of samples are as close as possible. In turn this means looking at all the possible channel degradations and being prepared to handle each one. Figure 10 shows some of the things that can happen to the signal $x(n)$. First the signal is passed through an anti-aliasing filter. Next it is modulated up by a frequency f and passed through the channel. We can view the passage through the channel as more filtering

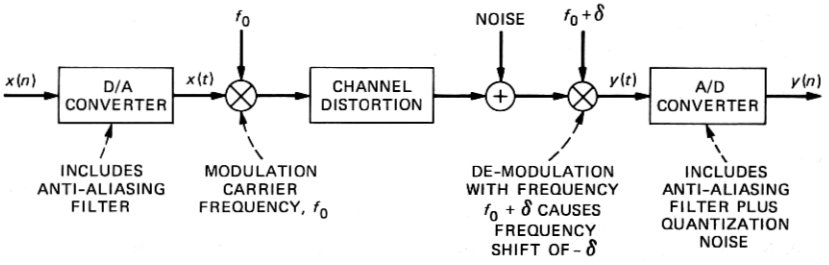

Fig. 10—Channel model.

followed by additive noise. The signal is then demodulated by a frequency $f + \delta$, which causes a frequency shift, or "phase roll," of $-\delta$. Finally, in the A/D process it is filtered once more and quantization noise is added. Fortunately, all of these degradations are independent and can be attacked separately.

We first argue that the noise will have the same effect as white noise of the same power, since the speech bands are constantly being permuted. Other than that the noise is essentially left unchanged, since we have no way to remove it.

We next attack the frequency shift and remove it. This frequency shift is also referred to as "phase roll" in the literature. A small shift in frequency of normal speech would probably not be detectable. However, for our work it is catastrophic. This is because the QMF filters cannot tolerate any frequency shift if they are to properly cancel out aliasing.

To compensate for the phase roll we propose that a 250-Hz sidetone be transmitted. This tone will be received with the same frequency shift as the rest of the signal. Figure 11 shows a novel phase-roll compensation scheme we have devised that works quite well. In Figure 11 the input signal branches immediately. The top branch goes first through a sharp bandpass filter (BPF #1), which passes only 240 to 260 Hz. The purpose is to immediately isolate the 250-Hz tone (which may have been slightly shifted). Next the tone is frequency inverted, making it approximately 3750 Hz. Then it is interpolated 2:1 using a second bandpass filter as the interpolation filter (BPF #2) with a center frequency of 3750 Hz. This signal is now modulated by a 250-Hz tone and then bandpass-filtered again (BPF #3). This produces a modulation signal of about 4000 Hz. If the original signal was shifted up δ Hz, the modulation signal will be 4000-δ Hz. In the second branch the remainder of the signal is interpolated 2:1 and then modulated up by 4000-δ. It is then decimated and frequency inverted. In this way the received signal $s(n)$ is modulated by exactly the right amount to compensate for the phase roll. One might even say the phase roll is

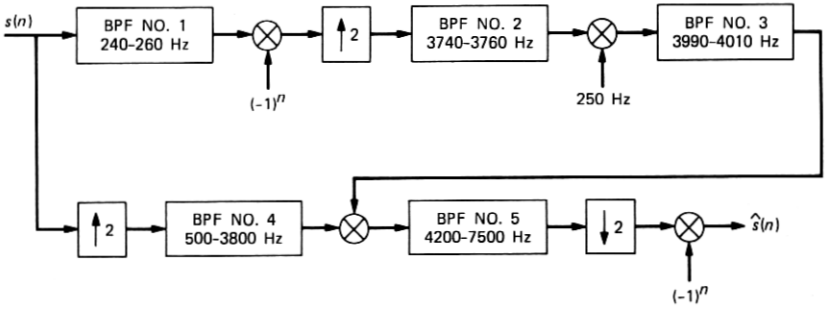

Fig. 11—Phase-roll compensation network.

made to compensate for itself. The interpolation filter in the second branch is actually a bandpass filter that removes the 250-Hz sinewave. Thus, the final output signal does not contain the 250-Hz sinewave.

The phase-roll compensation scheme was tested using computer simulations. The received signal was passed through a computer program that introduced an unknown amount of phase roll. The output from this program was then passed through the phase-roll compensation system. The compensation system completely cancelled the phase roll and its effects, but introduced a small amount of background noise. The noise appeared to be independent of the amount of phase roll in the signal. It was such a low-level noise that a listener needed to be in a quiet room to hear it. On this basis the compensation scheme was judged to work well.

Next, we can measure $h(n)$ by transmitting an impulse and measuring the response at the receiver. If noise is a problem, the measurement can be done several times. Given $h(n)$, $n = 0, 1, \cdots, N$ we form $g(n) = h(N - n)$, $n = 0, 1, \cdots, N$. If we convolve $h(n)$ with $g(n)$, the result will be symmetric sequence $f(n)$, $n = 0, 1, \cdots, 2N$. Because $f(n)$ is symmetric it is linear phase. This suggests that if we use $g(n)$ to filter all the data, $y(n)$, the output will be an integer-delayed version of the signal $x(n)$ with some amplitude modification in the different frequency bands. If we feed $f(n)$ into filter bank #3, we can find the gain to apply to the signals in each band to correct for the amplitude modification. In general the frequency response of $h(n)$ is fairly flat in amplitude for the frequency bands of interest. We have found that an impulse response length of 64 samples centered around the maximum value of the impulse response works well.

VII. SIMULATION RESULTS TO DATE

Simulation results to date have been extremely encouraging. In the transmitter program some header information is first generated. This

consists of a sine wave that has one of two frequencies, 1000 Hz or 3000 Hz, and thereby determines whether the receiver will use the high channel or low channel. This sine wave is 32 ms long. After a 16-ms wait it transmits an impulse for measuring the channel's impulse response. After 16 ms more it transmits the sine wave again for 32 ms. It then begins the encryption process by sending the "chirp" sync pulse in x_3 (or x_5) to indicate that the buffer is in initialized status. Superimposed on the signal throughout is a 250-Hz sine wave for the phase-roll compensation. The output of this program is the digital file $x(n)$.

The first channel we used for transmission was just to output $x(n)$ to the computer's D/A and loop right back through the computer's A/D. This "channel" is noise free and has no phase roll but is not perfectly flat in frequency response. It was a good test of our synchronization and equalization algorithms. Our receiver program is able to automatically synchronize itself and do the adaptive equalization needed for this channel. We found that using 5-ms block lengths gave the best results. For block lengths this short, any frame-rate noise sounds more like quantization noise. For larger block lengths, such as 16 ms, the frame-rate noise sounds like burble. Since some frame-rate noise always creeps in, we chose the 5-ms block length as subjectively better.

The next channel we tried was the telephone system at Murray Hill. Using two data sets connected to the D/A and A/D of the computer we were able to loop through the Murray Hill switch. Again the results were good. We then tried introducing a phase roll to the received signal and using our phase-roll compensation system in the same way as described in the previous section. As reported above, this also worked well.

Thus far we have not encountered a noisy channel and so we were forced to simulate one, as we did for the phase roll. The result confirmed our prediction that the noise would sound the same as white noise added to clear speech.

Another advantage we have enjoyed thus far is that the A/D and D/A have the same clock. Therefore, there is no relative drift between them. Such a drift for real hardware would cause a problem if it were at all large. In practice, if the system is regularly resynchronized, the receiver clock can be tied to the resynchronization to compensate for any drift. When we resynchronize the buffers, the chirp could also be used to resynchronize the receiver clock with the transmitter clock.

VIII. CONCLUSIONS

We have proposed a novel time-and-frequency scrambling system. To avoid the echo cancellation problem we have resorted to one of

two solutions. For a full-duplex system we used a narrow bandwidth, which reduced the quality of the speech but permitted simultaneous talking by both speakers. The narrow bandwidth may have also caused some small loss in intelligibility. For a half-duplex system this is not the case. We used a wider bandwidth system. One can even conceive of an automatic system in which each user receives half bandwidth when both speak but when one is silent the other receives the full bandwidth.

What makes our novel system feasible are the flexibility of digital signal processing and the equalization and synchronization schemes that were devised using digital processing. The digital processing also permits a full time-and-frequency permutation.

REFERENCES

1. "Digital Signal Processor—A Programmable Integrated Circuit for Signal Processing," B.S.T.J., *60*, No. 7, Part 2 (September 1981).
2. D. Esteban and C. Galand, "Application of Quadrature Mirror Filters to Split Band Voice Coding Schemes," Proc. of 1977 ICASSP (May 1977), pp. 191–5.
3. J. D. Johnston, "A Filter Family Designed For Use in Quadrature Mirror Filter Banks," Proc. of 1980 ICASSP (April 1980), pp. 291–4.
4. N. S. Jayant, R. V. Cox, B. J. McDermott, and A. M. Quinn, "Analog Scramblers for Speech based on Sequential Permutations in Time and Frequency," B.S.T.J., this issue.