Voice Privacy in Wireless-Phone, Mobile, Communication by Pseudo Random Number Generator and Tompkin-Paig Algorithm

Shahram Etemadi Borujeni, Mohammad Reza Reshadi Nezhad
Computer Engineering Department, University of Isfahan, Iran
etemadi@eng.ui.ac.ir, reshadi@eng.ui.ac.ir

Abstract -The art of security of voice communication in what is known as encryption or scrambling. The main attraction of this method arises from the fact that it can be used with the existing satellite and mobile communication systems without the use of a modem, provided the scrambled signal occupies the same bandwidth as the original signal. The scrambling algorithm is based on the permutation of the samples and provides highly secured scrambled signal by permuting a large number of those samples. The algorithm for generation the permutation matrices is explained. Important items to be considered in designing the system are discussed such as choice and construction of permutation matrices, and configuration of the practical scrambling system. The results of simulation and tests shows that proposed scrambling achieve extremely high-level security.

The method of choice and generation of permutation matrices, Tompkin-Paig algorithm and maximum length shift register are concerned. Simulations of different parts of the system, include scrambler, descrambler and generation of permutation matrices programs are provided. Miscellaneous methods of objective tests are described. Hardware implementation with ADSP-2100 series processor and assembly language programming of the system is described.

Index terms---Voice privacy, Mobile Cryptography, Block Clipper, Information Security.

I. Introduction

Fig. 1 shows a typical encryption system. The key 'K' determines the transformation from the set of all possible samples to the set of all possible scrambled samples.

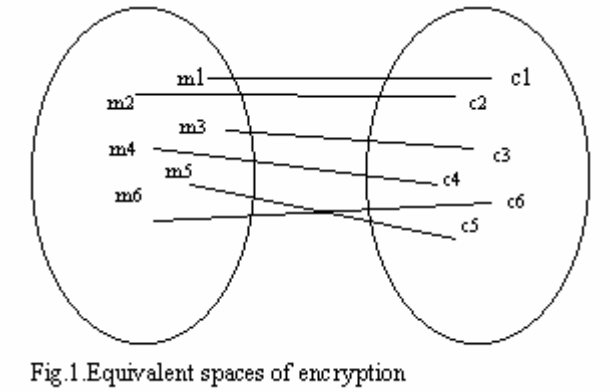

Fig.1.Equivalent spaces of encryption

A cipher system is a finite set 'T' of transformations from a finite sample space 'M' onto a scrambled sample space 'C'. Each sample has associated with it a probability which reflects the chance of its being sent and, similarly, for each transformation there is a probability which reflects the likelihood of it being used. The knowledge of the scrambled sample and the transformation should enable the recipient to determine the sample uniquely. It means that each of the transformation in 'T' must be reversible, so that if a sample 'm' is transformed into the scrambled sample 'c' by transformation 't', if c=t(m), then we must have $m=t^{-1}(c)$, where 't^{-1}' is the inverse of 't'. The scrambled sample 'c' and only the a priori probabilities of the various t's will be known to the interceptor. For a good encryption system we would hope that this information does not help him to work out, or even to guess, the sample 'm'. Indeed, ideally we would not like that it enables him to eliminate any possibilities for 'm'.

This is the basic objective behind a concept called perfect security that is a highly desirable property. For perfect security we need the number of key's, i.e. transformations, to be at least as large as the total number of possible sample. Shannon [6] showed that if the encryption system has the same number of samples, scrambled samples and keys then it has perfect secrecy if, and only if

(1) For any given sample 'm' and any given scrambled sample 'c' there is exactly one key transforming 'm' into 'c', i.e. there is a unique transformation 't' with c=t(m) , and

(2) All keys are equally likely

If this is the case, then we say that the system is unbreakable. In radio communication, including satellite and mobile communication, it is almost impossible to prevent unauthorized people from eavesdropping. When a message is broadcasted from a satellite, it is usually receivable over a very large area. Mobile radio networks are also relatively open to interception. Therefore, the problem of providing some form of privacy with a high level of security is becoming increasingly important. In order to generate permutation matrices in real time, special high speed microprocessor is required. The latest one with enhanced mathematical capabilities are especially suitable for our purpose such as ADSP-2100 family of digital signal processors.

II. Practical Cryptography and Properties of Permutation

However in any practical system we want to transmit a reasonable amount of information. This, of course, requires a large sample space that in turn implies a large number of keys. The distribution of a large number of key material is liable to cause horrendous management problems. In a practical system, a cryptanalyst will have to worry about time and facilities. Often, the time taken to solve a scrambled sample will be of utmost importance to him. It is quite likely that communicators will only need their samples to be secret for a limited time period, called their required cover time. Thus it is certainly possible for a theoretically insecure system to provide adequate practical security[3].

If we can set the cryptanalyst a task requiring too much storage or sufficiently large number of operations, then we may regard our system as practically secure. Therefore, we must estimate the number of operations or storage elements needed to break a secure system and then decide if the cover time is sufficient for our purposes. The only safe assumption that the cryptographer can make is that any would be cryptanalyst has as much knowledge and intelligence information as possible. When assuming the security level the cryptographer should assume the following three worst case conditions:

(1) Cryptanalyst has a complete knowledge of the encryption system.

(2) He has obtained a considerable amount of scrambled samples.

(3) He knows the descrambled sample equivalent of a certain amount of the scrambled samples.

A permutation key is put in at the transmitter to select a permutation for the sample. At the receiver, an inverse permutation key is put in, to select an inverse permutation for those permuted components of the received sample. If "L" samples are permuted, the number of possible permutation is L!. It is clear however, that all of these permutations can not be used. A subset of permutation has to be selected out of the L! permutations for use in the scrambling system. It is determined that this set satisfies the following requirement [1]:

(1) Any permutation in this set must not produce an intelligible scrambled sample.

(2) Any wrong permutation within this set must not produce an intelligible descrambled sample.

III. Pseudo-Random Binary Sequence (PRBS) Generator and Tompkin-Paig Algorithm

A shift register with feedback from specific stages can generate a continuous, though repetitive, random sequence of 1's and 0's. A schematic of this shift register is indicated in fig. 2.

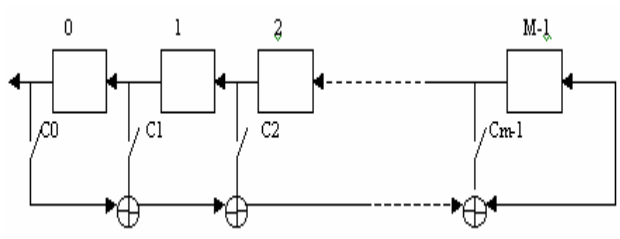

Fig.2.Pesudo random binary sequence generator

It can be shown that the maximum length before repetition is given by Nmax=2^M-1, where "M" is the number of stages in the shift register. The sequence itself is determined by the position of feedback taps on the shift register. Only certain feedback taps result in a maximum length sequence and these are given in table 1 up to "M=25" stages maximum length shift register [4].

All possible state vectors except all zeros, 00..0, can be an initial state and it occurs once among the first "2^M-1" states of the shift register during the generation of a sequence.

As mentioned earlier, since typically, we might wish to have a choice of about a million permutation matrices, the number of stages in the shift register "M" is chosen to be "20". For generating the corresponding maximal length sequence, stages "0" and "17" should be connected to the modulo-2 adder (XOR) as a feedback.

Table 1.Shift-register connections for Generating maximum-length sequence

m	Stage connected Tc modulo-2 adder	m	Stage connected Tc moulo-2 adder
10	0,7	18	0,11
11	0,9	19	0,14,17,18
12	0,6,8,9	20	0,17
13	0,9,10,11	21	0,19
14	0,4,8,12	22	0,21
15	0,14	23	0,18
16	0,4,13,15	24	0,17,22,23
17	0,14	25	0,22

The Tompkin-Paig algorithm [2] is an algorithm to generate a one-to-one mapping between the integers and permutations. In this algorithm, the target permutation is generated as a product of "L-1" simple permutations of order "i+1" and degree g_i where "L" is the number of components to be permuted and $1 \le g \le$ L-1

i=1,2,.....,L-1

The simple permutation [5] of order "L-m_1+1" and degree "m_2-m_1" is generally defined as indicated in the following

$$\begin{bmatrix} 1\ 2\ .\ .\ m_1-1\ m_1\ m_1+1\ .\ .\ m_2-1\ m_2\ m_2+1\ .\ .\ L \\ 1\ 2\ .\ .\ m_1-1\ m_2\ m_2+1\ .\ .\ L\ m_1\ m_1+1\ .\ .\ m_2+1 \end{bmatrix}$$

For example, when L=5, the simple permutation of order four and degree two is

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 5 & 2 & 3 \end{bmatrix}$$

and, the target permutation for given set of four g_i's is obtained as a product of four simple permutation of order i+1 and degree g_i where i=1,2,3,4 and

$1 \le g_i \le 4$

In the encryption system, the number of samples to be permuted, "L", are assumed 100 and therefore the target permutation is obtained from "99"(L-1) simple permutations of order i+1 and degree g_i's where i=1,2,3,...,99 and $1 \le g_i \le 99$

IV. Generation of the Permutation Matrices

The number of possible permutation is "L!=100!". Not all of those permutations can be used. The permutation matrices should have as little closeness as possible to the identity permutation matrix. Permutation matrices that are close to any circularly shifted versions of the identity permutation matrix produce scrambled sample of very high closeness to original sample.

In order to protect the system against someone, who has similar encryption equipment, the number of possible

permutations must be sufficiently large. Assuming that the required number of possible permutations is one million, a 20-stage PRBS generator is used as a first stage of the permutation matrices generation process, because the maximum length of a 20-stage shift register sequence is 1048575, i.e. 2^{20}-1.

The key is entered as a five digit hexadecimal number, which is converted to 20 binary bits and used as an initial state of the 20-stage PRBS generator. Let b_i (i=1,2,3...) be the i'th output bit of the PRBS generator according to the given initial state. "99" random integers (g_i's) are generated from these b_i's by using the formulas [5]:

$g_1 = 1$

$g_2 = [(2b_0 + b_1)/(2^2 - 1)] + 1$

$g_3 = [(2b^2 + b_3)/(2^2 - 1)] + 1$

.

.

$g_i = [(2^{j-1} b_k + 2^{j-2} b_{k+1} + \ldots + b_{k+j-1})(i-1)/(2^j - 1)] + 1$

where $j = [\log_2 I] + 1$ and $k = \sum_{s=2}^{i-1} \{[\log_2 s] + 1\}$ and "[x]" ,here, is the maximum integer less than or equal to "x". Then a permutation matrix is generated from the set of "L-1=99" integers, g_1 g_2 g_3 g_4 . . . g_{98} and g_{99}, by applying the Tompkin-paig algorithm which gives a one-to-one correspondence between the integer and permutation. So, the target permutation matrix is generated as the product of "99" simple permutation matrices of order "i+1" and degree "g_i" where i=1,2...99.

There are four steps in this algorithm, which are explained as follows:

First of all, a key is entered as a five hexadecimal digit number and the program is able to provide a 20-bit binary number equivalent to the given key. This number is used as the initial state of a 20-bit maximum length PRBS generator with suitable feedback. For the calculation of g_i's, 481 b_i's are necessary. Therefore, a string of "481" bits (b[1] to b[481]) is generated by module-2 addition of first and 18th bits(i.e. stages '0' and '17') of shift register. Then, the integer random numbers (g[1] to g[99]), are generated by using those equations. Finally, the Tompkin-Paig algorithm is implemented on a sequential array from "oa[1]=1" to "oa[100]=100" with order i+1 and degree g_i, where "i" starts from "99" and finishes at "1". The permutation array is generated as a na[i] and is copied onto the oa[i] array. The results are then stored in a permutation file and graph of normal and permutation configuration are plotted.

V. ADSP-2100 Family of Processors

ADSP-2100 is a programmable single-chip microprocessor optimized for digital signal processing (DSP) and other high-speed numeric processing applications.The ADSP-2100 chip contains three independent computational units; arithmetic/logic unit (ALU), multiplier/accumulator (MAC) and barrel shifter that operate on 16-bit fixed-point data. There are two data address generators and a program sequencer; data and program memories are external. The ADSP-2101 is a programmable single-chip microcomputer based on the ADSP-2100. Like the ADSP-2100, the ADSP-2101 contains computational units, as well as a program sequencer and dual address generators. Additionally, there are 1K words of data memory and 2K words of program memory on chip, two serial ports, a timer, boot circuitry (for loading on-chip program memory at reset) and enhanced interrupt capabilities. Because the ADSP-2101 is code-compatible with the ADS-2100, the programs can be executed on these chips as well.A modified low cost version of ADSP-2101 has come out recently. The ADSP-2105 is same as the ADSP-2101 with half the on-chip memory (512 words of data memory and 1K words of program memory) and one serial port instead of two. It is pin and code compatible with the ADSP-2101. The ADSP-2105 is ideally suited to high speed low cost DSP applications. It is preferred to use ADSP-2105 instead of ADSP-2101 in the system and take care of one missing serial port and reduced internal memory by suitable modifications in the software and hardware design. Block diagram of ADSP-2100 system is indicated in fig.6.1.

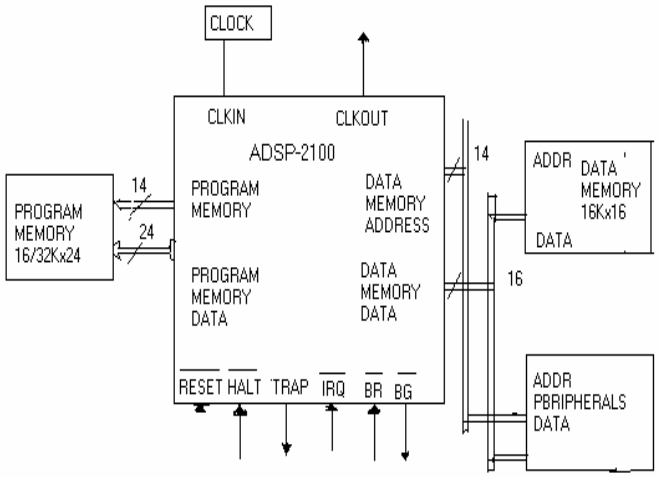

Fig.3.ADSP-2100 System

In ADSP-2100 family of processors each computational unit contains a set of dedicated input and output registers which are indicated in table 6.1. Computational operations generally take their operands from input register and load the result into an output register. The registers act as a stopover point for data between the external memory and the computational circuitry, effectively introducing one pipeline level on input and one level on output. The computational units are arranged side by side rather than in cascade. To avoid excessive pipeline delays when a series of different operations are performed, the internal result (R) bus allows any of the output registers to be used directly (without delay) as the input to another computation. For a wide variety of calculation, it is desirable to fetch two operands at the same time, one from data memory and one from

program memory. Fetching data from program memory, however, makes it impossible to fetch the next instruction from program memory on the same cycle; an additional cycle would be required. To avoid this overhead, the ADSP-2100 incorporates an instruction cache which holds sixteen words. The benefit of the cache architecture is most apparent when executing a program loop that can be totally contained in the cache memory. In this situation, the ADSP-2100 works like a three-bus system with an instruction fetch and two operands fetches taking place at the same time. Many algorithms are readily coded in loops of sixteen instructions or less because of the parallelism and high-level syntax of the ADSP-2100 assembly language.

VI. ADSP-2100 Based Addon Card

The addon-card have been put directly on the PC and works under the supervision of the host processor which in this case is the PC itself. The hardware consists of two codecs (coder, decoder) which sample the speech, encode it, compand it, and transmit it serially to the ADSP-2101/2105 serial ports. The ADSP-2101/2105 processes the received signal and transmit it back via the ADSP codec link. The basic hardware structure of the system for real time testing is indicated in Fig.4.

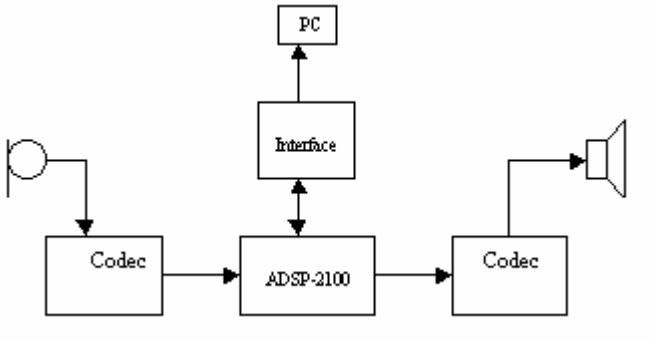

Fig.4.Block diagram of Hardware implementation

The block diagram of the stand alone hardware unit is depicted in Fig.5. For compact implementation and flexible operation, all the signal processing is performed in one DSP processor. The unit is roughly divide into five section:

1) Analog circuitry
2) Scrambler,
3) Descrambler,
4) Permutation matrices generator and
5) Control section.

VII. ADSP Assembly Language Overview

The ADSP-2100 family's assembly language uses an algebraic syntax for ease of coding and readability. The sources and destinations of computations and data movements are written explicitly in each assembly statement. Each assembly statement, however, corresponds to a single 24-bit instruction, executable in one cycle. Register mnemonics are listed in table 2.

The ADSP-2101 instruction set is an upward-compatible superset of the ADSP-2100 instruction set; thus, programs written for the ADSP-2100 can be executed on the ADSP-2101 with practically no changes.

Table.2.Computational I/O Registers

	Source for X input	Source for Y input	Destination for O/P
ALU	AX0,AX1,AR MR0,MR1,MR2 SR0,SR1	AY0,AY1 AF	AR AF
MAC	Source for X input	Source for Y input	Destination for O/P
	NX0,MX1,AR MR0,NR1,MR2 SR0,SR1	MY0,MY1 Mf	MR(MR2,MR1,MR0) MF
Shift-Cr	Source for shift input	Destination for shifter O/P	
	S1,SR0,SR1 AR MR0,MR1,MR2	SR(SR1,SR0)	

The ADSP assembly language coding for generation of permutation matrices has been written by using information in section IV. This assembly language program is modular and consists of two sub-routines which are called by a main program as a divider and a multiplier. The explanation of different parts of the assembly language program is as follows;

1) Accepting 20-bit external key as a input
2) Initializing maximum length shift register with this key values.
3) Generating of 481 bits, which are called b[i]'s,and storing them in data memory.
4) Calculation of 86 g[i] values from b[i]'s with using
5) Applying Tompkin-Paig algorithm to normal configuration of coefficients by using g[i]'s value and generating a permutation matrix which

VIII. Objective Tests and Conclusion

As discussed, the generated permutations should have as little closeness to the identity permutation as possible. Permutations, which are close to any circularly shifted versions of the identity permutation, produce scrambled sample of very high residual intelligibility. The closeness between two permutations will be measured by means of auto-correlation and rank correlation. The auto-correlation of each permutation order should be impulsive in nature. Actual formula for calculation of auto-correlation is given here:

$$R(j)= \frac{1}{N-j}\sum_{i=1}^{N-1} x(i)x(i+j)$$

where N=L-1=99 and x(i)=oa(i)

The auto-correlation of most keys have been calculated and plotted. All of them seem to have the desired properties mentioned above.

The same objective results are seen by evaluating rank correlation. One of the most frequently used rank correlation samples for comparing two permutation "A", identity permutation, and "B", the objective permutation of "L" objects is the Spearman's rank correlation [3], α, which is defined as :

$\alpha = 1 - \frac{6S(d^2)}{L^3 - L}$ where $S(d^2)$ is sum of squares of rank wise difference between the permutations. The range of this samples, is from "-1" to "1", with magnitudes close to "0" showing small correlation and magnitude near to "1" representing large correlation.

In the designed permutation algorithm, the histogram of rank correlation was calculated and plotted by running a 'C' language program. It shows that the distribution of permutations generated using this algorithm is approximately a normal distribution spread over the interval -0.4<a<0.55 which means that the possibility of a permutation to be close to the identity permutation or a shifted version of it is extremely small.

The algorithm to generate the permutation matrices, permutation orders, enables generation of "20!" permutation matrices out of "100!" possible permutation matrices, by choosing five-digit hexadecimal numbers. The distribution of the permutations generated utilizing this algorithm is approximately a normal distribution.

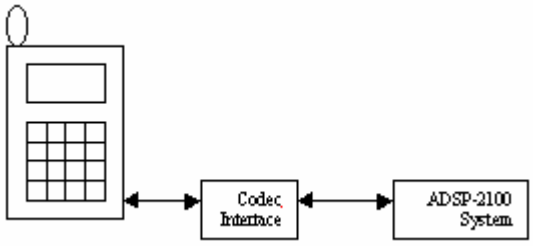

Fig.5.Hardware unit,Transmitter side/Receiver side

It was concluded that the possibility of a generated permutation using this technique lying close to the identity permutation is extremely small and therefore no more screening of permutations is necessary. A cryptanalysis of the system requires original and scrambled samples.

The disadvantage of this attack is that a very large matrix has to be inverted.

The attack could be avoided by changing the permutation more frequently and employing multi-frame permutation method. This attack also required the cryptanalyst to obtain some amount of original samples. In most cases this could be impractical because of the extra number of keys. This method could be utilize for information security in many fields such as database, network, signal and so on.

References

[1] Beker H.J. and Piper F.C., "Secure Speech Communications", Academic Press Inc., London, 1985.

[2] Beckenback E.F., "Applied Combinatorial Mathematics", 1964.

[3] Kak S.C., "Overview of analog signal encryption", IEE Proceedings, Vol.130, No.5, pp.399-404, Aug. 1983.

[4] Proakis J.G., "Digital Communications", McGraw-Hill 1997.

[5] Sakurai K., Koga K., and Muratani T., "A Speech Scrambler", IEEE Journal on selected areas in communication, Vol.SAC-2, No.3, pp.434-442, May 1984.

[6] Shannon C.E., "Communication theory of secrecy systems", Bell Syst. Tech. J.28, pp 656-715, 1994.

[7] Ash R., "Information Theory", Interscience Publishers, New York, 1990.

[8] L'Ecuyer P., "Random Number Generators", Proc. Of Winter Simulation Conference, Vol. I, 1998, pp.97-104.

[9] Kerouedan S., Adde P., "Block Turbo Codes", Proc. Of 8'th IEEE ICECS2001 Conference, Vol. II, 2001, pp.1219-1222.

[10] Etemadi S., "Speech Encryption based on Fast Fourier Transform Permutation", IEEE-ICECS2000 Conference,pp.290-294, Dec. 2000.