CSCC'99 Proceedings, pp. 1811-1817

DSP Applications in Secure Voice and Data Communications Systems

GEORGE TROULLINOS
INTRACOM S.A.
19.5 Km. Peania - Markopoulo Ave.
P.O. Box 68
19002 Peania - Attika - GREECE

1 Introduction

Secure voice and data systems have been of much interest to commercial and military users since the advent of modern communications. The basic drawbacks of early implementations, however, were their large size, difficulty of use, and excessive cost. This was a direct result of the limitations imposed by hardware based architectures performing the variety of complex processing functions of a secure communications system.

The current standard architecture of a digital secure voice and data system is shown on Figure 1. The major system functions, i.e., voice coding, encryption and decryption, cryptographic key management, and system control, are all implemented using distinct hardware modules. As shown on Figure 1, the analog voice signal of the speaker $S_i(t)$ is converted to its digital form $S_i(nT_s)$ which is then compressed by employing some voice coding technique (VOCODER). The most effective voice compression techniques are based on Linear Predictive Coding (LPC) that converts the incoming voice samples from a bit rate of 64 kbps to 2.4 kbps, 4.8 kbps or 9.6 kbps. The compressed voice samples $P_o(nT_b)$ are transferred to the encryption module and the encrypted samples $E_o(nT_b)$ are transmitted to the communications medium.

Similarly, secure data operation is achieved by encrypting the computer data $D_i(nT_b)$. Finally, the Key Management Module of Figure 1 is required to ensure that the encryption and decryption algorithms operate with the same cryptographic key, thus making the secure communication feasible.

Recent developments in the performance of Digital Signal Processing (DSP) microprocessors allow new system architectures to be deployed. These architectures could benefit from the high computational power of DSP microprocessors to offer low cost, integrated, flexible solutions to an old problem. (Early implementations could occupy a whole room full of hardware).

2 Voice Coding

The voice technology backbone used in secure voice communications systems is based on Linear Predictive Coding (LPC) techniques [1]. This is so since LPC based voice coding achieves low bit rates and high voice intelligibility. High voice compression (2.4 kbps, 4.8 kbps, 9.6 kbps) is achieved by collecting voice samples, creating speech frames, and then processing these frames as shown in Figure 2.

The analysis procedure is initiated by sampling that converts the speaker's voice to a digital signal. This signal is conditioned by the Preemphasis Filter so that energy is allocated equally in the low and high frequency components of the voice signal. At the output of this filter, the voice coder of Figure 2 generates a 10^{th} order digital filter for every speech frame (usually 22.5 msec in duration). This filter consists of ten (10) coefficients, called Reflection Coefficients.

The most critical phase in speech analysis is Pitch estimation, and a variety of Pitch trackers have been proposed, such as the Gold-Rabiner Pitch Tracker, the Autocorrelation Pitch Tracker, the AMDF, and others [2]. Since the Pitch information is found at the low frequencies of the voice signal, the digitised speech is low-pass filtered (see Figure 2). Also, the energy level at the output of this low-pass filter is used to characterise each speech frame as voiced or unvoiced.

All information derived for every speech frame is converted to a predetermined format by the Coding Block of Figure 2. The output of this Coding Block is the digital compressed voice ready for encryption.
On the receiving side, all parameters of every speech frame, i.e., Pitch, Energy, Reflection Coefficients, Voice/Unvoiced indicator, are used for the digital synthesis of the voice signal.

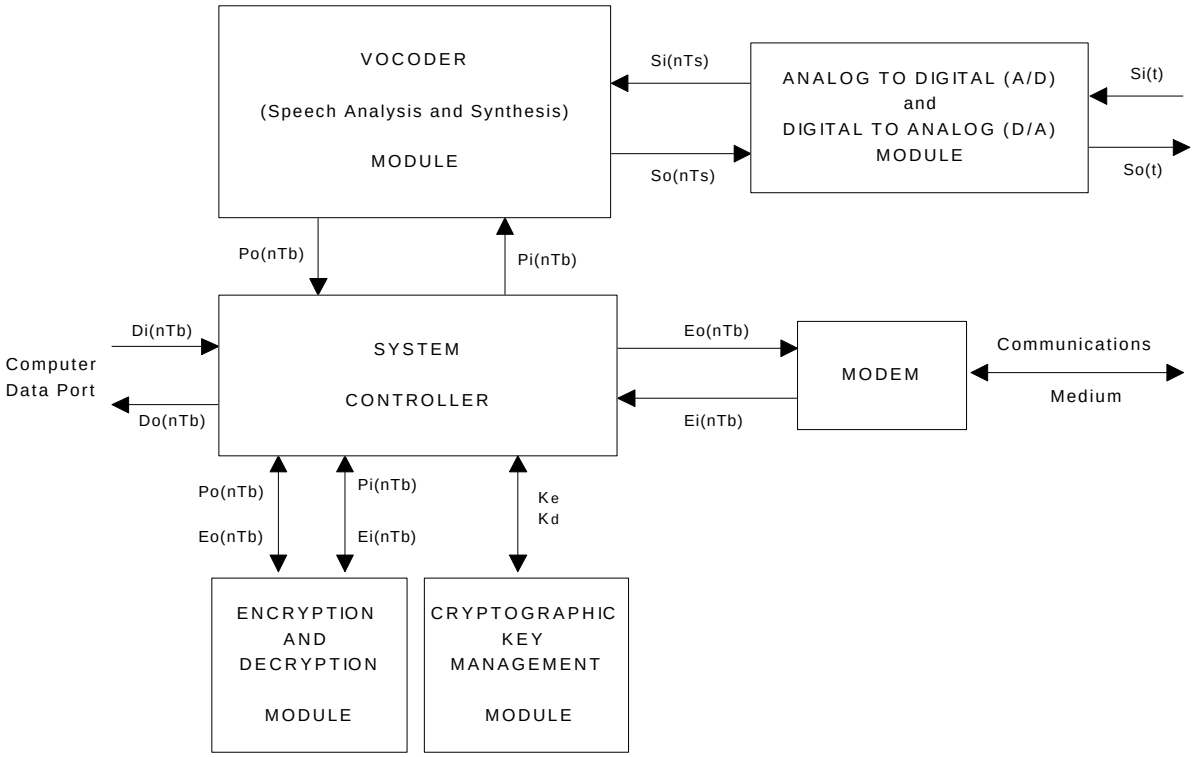

Figure 1: Secure Voice and Data System Architecture

3 Data Encryption

Encryption refers to hiding information during transmission, thus making this information unintelligible to all but its intended recipient. The messages to be encrypted, known as plaintext, are modified by the Encryption/Decryption Module in a unique way determined by a cryptographic key shared between the parties that wish to communicate in privacy. The output of the encryption process, known as the ciphertext, is then transmitted to the intended recipient. The basic idea behind any encryption technique is that an intruder does not know the cryptographic key, and thus, cannot decrypt the ciphertext and reconstruct the original information.

In general, there are two types of encryption algorithms, Stream Cipher and Block Cipher [3]. Stream Cipher algorithms operate on the plaintext a bit at a time. Block Cipher algorithms process plaintext a block at a time. A typical block Cipher encryption algorithm is the Data Encryption Standard (DES). The DES has been specified by the National Bureau of Standards (Now NIST) and has been adopted as an ISO standard [4].

In this cryptosystem, plaintext information is divided into blocks which are then operated upon independently to generate a sequence of ciphertext blocks. The basic idea behind DES is to build a strong system out of simple, individually weak, components. The DES cryptosystem is based on a system of transpositions and permutations. The permutation box, or P-box, is used to transpose, or map a sequence of input values to another sequence of values of the same length. Substitutions are performed by what is called S-boxes.

A combination of the S-boxes and P-boxes can be viewed as a decoder/coder operation, where the output is simply a linear mapping of the input values. Each combination of the S-box and P-box comprises a single weak component of the algorithm. By including a sufficiently large number of stages in the product cipher, the output can be made to be a non-linear function of the input.

Plaintext is encrypted in blocks of 64 bits, yielding 64 bits of ciphertext. The DES algorithm, uses a 56 bit key, and has 19 distinct stages. The first stage is a key independent permutation on the 64 bits of the plaintext. The last stage is an exact inverse of this permutation. The stage prior to the last one exchanges the left most 32 bits with the right most 32 bits. The remaining 16 stages are functionally identical but are transformed by different functions of the key. This algorithm has been designed to allow decryption to use the same flow as encryption. That is, for decryption the steps are simply run in the reverse order of encryption.

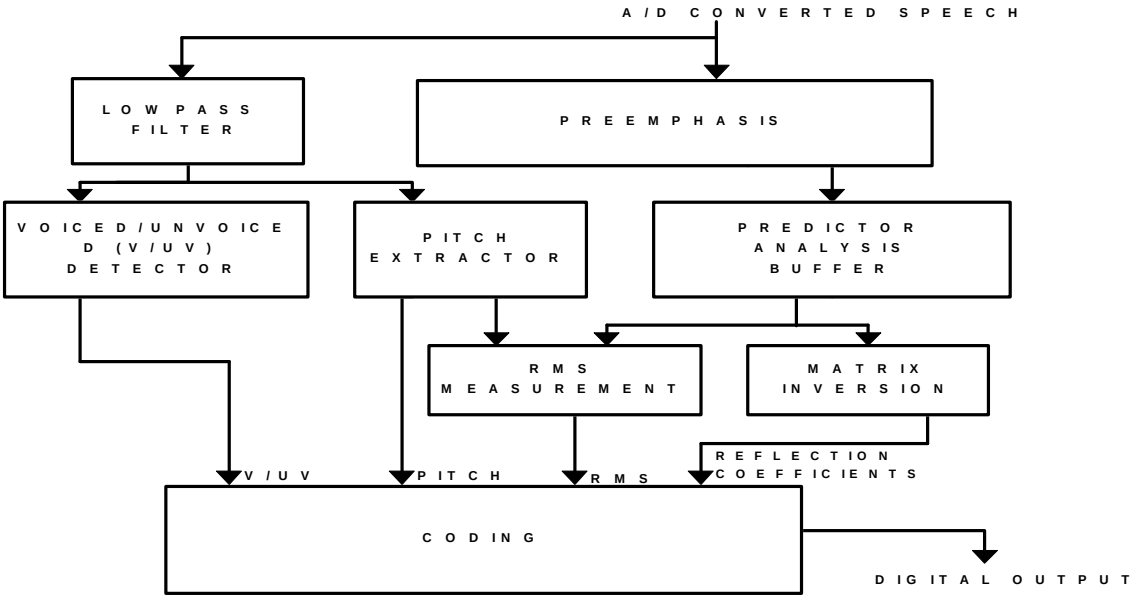

Figure 2: Speech Analysis Signal Processing

The operation of one of these intermediate stages is as follows. At each stage, the 32 bit input is transformed with a function of the encryption key and produces two 32 bit outputs. The left output is simply a copy of the right input. The right output is the bitwise Exclusive OR of the left input and a function of the key at that stage, K_i. All the complexity of the DES algorithm lies in this function.

It is interesting to note that in each of the intermediate 16 stages, a different key is used. Before the algorithm starts, a 56 bit transposition is applied to the key. Just before each stage the key is split into two 28 bit sections, each of which is rotated to the left by a number of bits which depends on the stage number. K_i is derived from this through another 56 bit transposition to it.

4 Cryptographic Key Management and Public Key Cryptography

In a single key cryptosystem, the key used for encryption must be used for decryption as well. Therefore, the parties that wish to establish a secure connection must have agreed upon the cryptographic key beforehand. If this key is somehow compromised, the whole system is rendered useless. Clearly, key distribution is a critical concern. The traditional method of key distribution has been through use of couriers and secure mail. Without entering into a discussion of ethics associated with this approach, security is based upon loyalty and honesty. With some minor modifications, this traditional approach is still used by most commercially available secure communications systems.

To circumvent the problems associated with the manual key distribution, Public Key cryptography was developed [5,6]. Unlike single-key cryptosystems where the same key is used for encryption and decryption, Public Key cryptography is based on a two-key solution: an encryption key and a decryption key. The novelty of this technology is that knowledge of the encryption key reveals no information about the decryption key.

The user of a Public Key cryptosystem keeps the decryption key secret and makes the encryption key available to everybody (hence the name Public Key). When one wishes to send secure information, one encrypts the information using this public encryption key. From that moment, the only person who can decrypt this information is the intended recipient, i.e., the user of the Public Key cryptosystem who holds the corresponding secret decryption key. Therefore, the Public Key cryptosystem can be used as an electronic courier to exchange session keys. The basics for implementing this key exchange are discussed later.

To successfully address the major problem encountered in conventional single-key cryptosystems like DES, i.e., setting up secure connection between users that have never previously communicated, Diffie and Hellman proposed a cryptographic system with an encryption algorithm, E, and a decryption algorithm, D, with the following requirements:

1. D(E(M)) = M.
2. It is exceedingly difficult to deduce D from E.
3. E cannot be broken by a chosen plaintext attack.

Under these conditions there is no reason why E cannot be made public, hence the name Public Key Cryptography. A general description of a typical approach is discussed next.

The plaintext, regarded as a bit string, is divided into blocks. Each plaintext message, M is represented as an integer between 0 and n-1. The message is encrypted by raising the plaintext message to the e^{th} power modulo n. In other words, the ciphertext, C, is the remainder of M^e/n.

Decrypting the ciphertext and recovering the original plaintext message, M, is accomplished by raising it to the d^{th} power modulo n. The encryption and decryption algorithms, E and D respectively, satisfy requirement 1 above as follows:

- $C \equiv E(M) \equiv M^e \pmod{n}$, and
- $M \equiv D(M) \equiv C^d \pmod{n}$.

An examination of the above relationships reveals that the encryption key is the pair of positive numbers (e, n), and the decryption key is (d, n). The encryption key is made public by the user and the decryption key is kept secret. The encryption and decryption keys are computed as follows.

First, two large random prime numbers p and q are generated. The product of these two numbers forms the modulo which is made public, but the factors p and q are kept secret. The security of this algorithm depends on the enormous difficulty associated with factoring n. If the cryptanalyst could factor n, which is kept public, he could find p and q and decipher any message encrypted with the key. Fortunately mathematicians have been trying to factor large numbers for over 300 years without success [7].

Testing a number for primality could be implemented by employing a probabilistic algorithm due to Solovay and Strassen [8]. A random number *a* is chosen from a uniform distribution on {1, ..., *b*-1} and the following test is performed:

- gcd (*a*,*b*) = 1, and
- $J<a,b> \equiv a^{(b-1)/2} \pmod{b}$,

where J<.> stands for the Jacobi symbol. If for several values of *a* the above relationship is satisfied, then *b* is almost certainly prime. Each repetition of this algorithm for different values of ' *a* ' has a 50% chance of failure. This probability should be viewed as "*if* *b* is a composite number then this algorithm will definitely fail, whereas, there is a 50% chance that *b* may be prime". Clearly such a test must be implemented many times to ensure an accurate result.

The decryption key, d, is chosen to be a large random number which is prime with respect to (p-1)(q-1), i.e., gcd (d, (p-1)(q-1)) = 1.

The encryption key, e, is computed from p, q, and d to be the multiplicative inverse of d modulo (p-1)(q-1), i.e., exd=1 (mod(p-1)(q-1)).

A mathematical proof that the encryption and decryption algorithms work, given that the keys are generated properly, is given in [6].

Given the state of the art in computing machinery and state of technology, what level of difficulty is associated with breaking this algorithm. Consider an implementation with a modulo of 512 binary digits, equivalent to a decimal number which has approximately 154

decimal digits or 10^{154}. For comparison purposes the total number of atoms in the known universe is estimated to be approximately 10^{80}.

The fastest known factoring algorithm is due to Richard Schroeppel, which can factor n in approximately:

$10^{(sqr(ln(n)*ln(ln(n))))}$

steps. Using this method and assuming that each operation takes 1 ìs we get approximately $3x10^{31}$ years to factor n (there are approximately 31,556,952,000,000 ìs per year), thus satisfying requirements 2 and 3 above.

5 Digital Signal Processing Based System Architecture

The proposed system implements all signal processing functions by employing the architecture shown on **Figure 3.** The Computational Module implements the processing functions, i.e., system management, Vocoding, cryptographic key management and Encryption/Decryption. This module consists of a high-speed (12 MIPS and above) DSP microprocessor, called the Processing Unit, and two memory banks, a fast one, and a slow one.

5.1 Off-Line System Operation

This mode of operation is entered at the beginning of every secure voice or data communication to establish

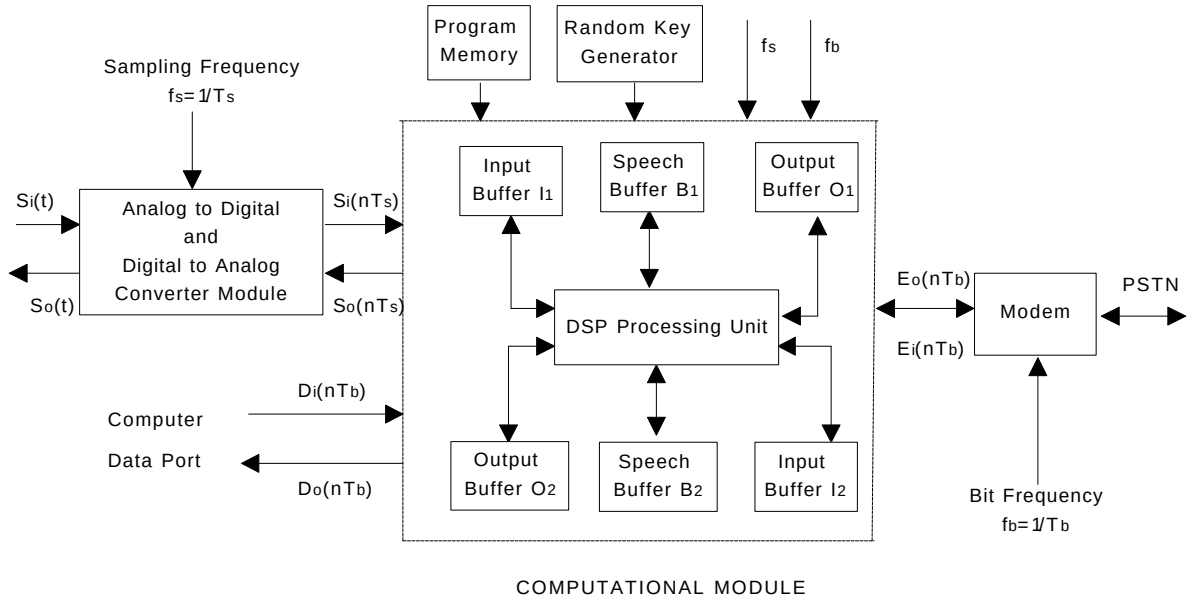

Figure 3: Integrated Secure Digital Voice and Data System

common cryptographic keys for the encryption and decryption processes. In this mode, the Public Key cryptosystem is used as an electronic courier to exchange session keys. The basics of this key exchange are as follows. First, each system provides the other with its own public encryption key. Then, each system utilising its Random Key Generator (see Figure 3), generates a random session key to be used later for encrypting voice and data information. To complete a security level connection, these session keys must be exchanged between the two systems. To do so, each system encrypts its session key using the public encryption key that was just received from the remote unit. This encrypted information is sent back and each system, using its secret decryption key, recovers the session key of the other. Thus, the session keys are safely exchanged and a secure communications channel is established. Then, the systems switch to the faster DES cryptosystem.

5.2 On-Line System Operation

On-Line is the real time operating mode of the system that implements the Vocoding and data encryption functions. In this mode, the Processing Unit collects voice samples and constructs speech frames. Every such frame corresponds to 22.5 msec of speech, i.e., 180 voice samples, and while the samples of one frame are collected in buffer I_1 of Figure 3, the Processing Unit is processing the previous frame. Buffer B_1 is used by the Processing Unit to store the compressed voice samples, i.e., for a 2.4 kbps Vocoder each uncompressed voice frame in buffer I_1 is represented by 54 bits in buffer B_1, 53 of which contain voice information and one bit is used to maintain frame synchronisation. Then, the data of buffer B_1 are encrypted and transferred to buffer O_1 and from there to the communications device.

In the opposite direction, the compressed and encrypted voice data are transferred from the communications device to buffer I_2. From there, they are decrypted and stored in buffer B_2, one decryption frame at a time. The digital synthesis of the voice samples is implemented by the Processing Unit that reads the data of buffer B_2, synthesises, and stores the uncompressed voice samples in the output buffer O_2, a frame at a time.

The overall system operates with two basic frequencies. The first, $f_s=1/T_s$, determines the data transfer between the A/D, D/A Converter Module and the Computational Module. The second, $f_b=1/T_b$, determines the data transfer rate between the Computational Module and the communications device and, in case of data encryption, the data transfer rate between the computer data port and the Computational Module.

The frequency f_s , used only for secure voice operation, is determined by the Nyquist criterion and is a function of the bandwidth of the voice signal. For narrowband voice with 3.2 kHz bandwidth, f_s= 8 kHz. The communications frequency f_b is determined by the Vocoder data rate and the speed of the communications device, i.e., 2.4 kbps or 4.8 kbps or 9.6 kbps.

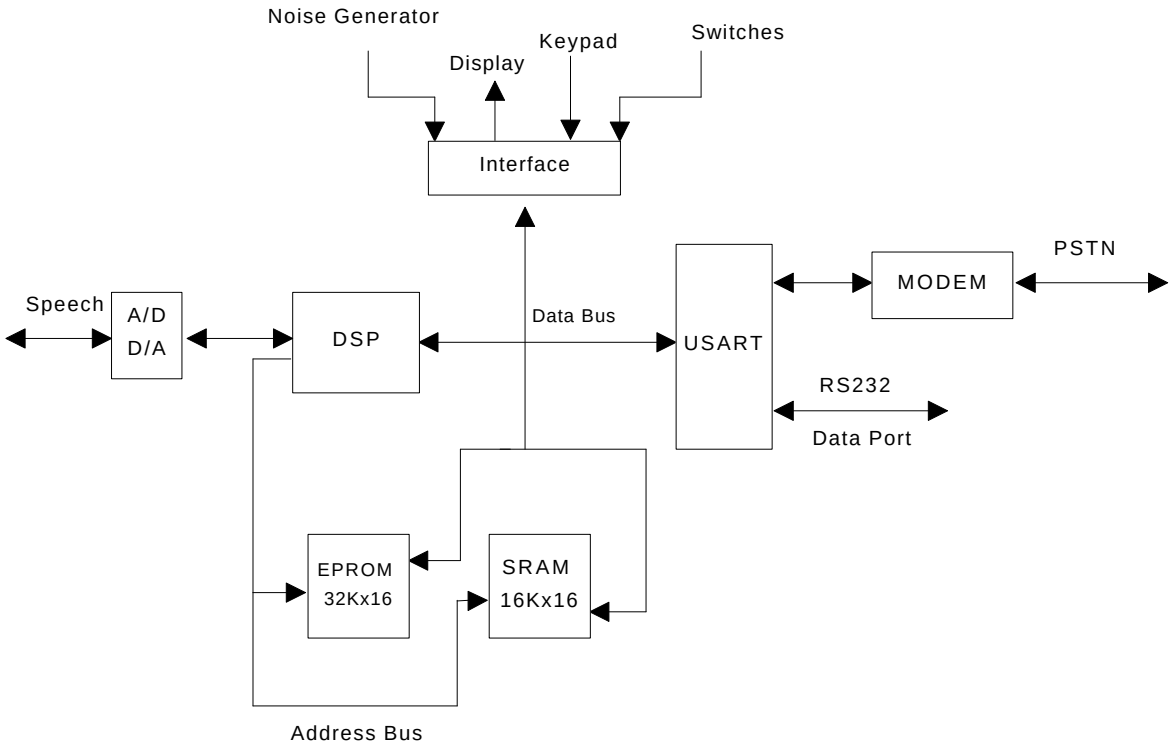

Figure 4: DSP Based Integrated Secure Voice and Data System Implementation

The operation of the overall system is interrupt driven. The two frequencies, f_s and f_b, are used to generate these interrupts to the Processing Unit. For every interrupt generated with frequency f_s, the Processing Unit executes the following functions:

(a) Reads a digital voice sample $S_i(nT_s)$ from the A/D converter and transfers it to buffer I_1.
(b) Writes a sample of synthetic speech $S_o(nT_s)$, if available, from buffer O_2 to the D/A converter.

Additionally, for every interrupt generated with frequency f_b, the Processing Unit executes the following functions:

(a) In case of secure data operation, it transfers an input data sample $D_i(nT_b)$ from the computer data port to buffer B_1, and a decrypted data sample $D_o(nT_b)$, if available, from buffer B_2 to the computer data port.
(b) Transfers an encrypted data sample $E_o(nT_b)$, if available, from buffer O_1 to the communications device, and reads a received encrypted data sample $E_i(nT_b)$, if available, from the communications device into buffer I_2.

When the Processing Unit is not executing one of the above interrupts, it performs the following functions:

(a) In case of secure voice operation, it analyses speech using the data stored in buffer I_1, and synthesises digital speech using the compressed voice samples available in buffer B_2.
(b) Data encryption of the compressed voice or data samples stored in buffer B_1, and decryption of the encrypted data stored in buffer I_2.

The architecture described above increases the security of the overall system since the data transferred to the communications device are **definitely encrypted**. This is not the case in conventional architectures which employ separate computational modules for the encryption, Vocoding, and system control functions (see Figure 1). This is so, since an accidental or intentional (tampering) malfunction of the System Control Module could result in unencrypted data to be transferred to the communications device and transmitted to the communications medium.

6 System Design and Implementation

6.1 Hardware Design

The backbone of the proposed architecture shown in Figure 4 is a high-speed DSP, such as the Texas Instruments TMS320C25 that can execute 12 MIPS. The system also comprises of some peripheral blocks, i.e., the communications module, the A/D, D/A module, and the interface to the control panel, the system display, the random number generator, and the Tamper and Auto-Zero switches. The USART of Figure 4 allows for both synchronous and asynchronous communications

protocols to be used between the DSP and the modem or DSP and the computer data port. In the design shown in Figure 4, the DSP based Central Processing Unit is supported by the necessary memories, 16Kx16 high-speed static RAM and 32KX16 low-speed EPROM. The high-speed memory is used for program execution and data buffering, as described earlier. The low-speed EPROM memory is used for storage of look-up tables and executable code. Upon power-up, portions of this code are transferred to the high-speed memory for execution (Cache memory). The A/D and D/A conversions are implemented by a single chip codec like the TCM2916. Voice samples transferred between the codec and the DSP are ì-law or á-law compressed, thus allowing 8 bit samples to have a 13 bit dynamic range. Decompression of these samples is implemented by the DSP software with the use of look-up tables. An important design issue is to avoid data buffer overflows. This is achieved by using the same clock source for the A/D conversions as well as for data transmission to the communications medium. Alternatively, a slightly slower A/D conversion rate and data filling by the USART could be employed. Finally, the I/O Interface hardware involves the use of decoding logic and bi-directional buffers, all of which can be implemented in a single VLSI chip.

6.2 System Software

The software design is based on the real-time Kernel software which is used to handle and prioritise all system interrupts. In the off-line mode of operation interrupts are generated from the system interface (I/O), while in the on-line mode of operation from the A/D and D/A conversions module, the communications module, and the system interface.

The major software tasks in the off-line mode of operation are the system interface monitoring and the cryptographic key management. The system interface software utilises wait-states and executes from the slow EPROM memories, while the computationally intensive key management routines are transferred to the fast SRAMs for execution. Cryptographic key management consists of two major software modules. The encryption and decryption routines, and the cryptographic key generation routines. Encryption and decryption is executed with every secure communication since an encryption of the local random session key and a decryption of the remote system's session key are required. Their implementation consumes approximately 1 K words of memory on the TMS320C25, and their execution approximately 9 secs. Most of this execution time (90%) is allocated to the decryption process and more specifically to the (a*b)mod(c) calculation. The new cryptographic key generation routines are required to compute a new set of Public Keys: (e,n) and (d,n). Their implementation consumes approximately 2.5 K words of TMS320C25 memory. Their execution time varies from a few seconds to over half an hour, depending on how quickly the random numbers generated by the noise generator meet the primality criteria.

In the on-line mode of operation the compression/decompression of the voice samples exchanged between the DSP and the codec, the speech analysis and synthesis, the encryption/ decryption, and the system interface (I/O) monitoring are implemented. Among these tasks the most computationally intensive is the speech analysis and synthesis, utilising approximately 80% of the system's computational power. The DES based 64 bit block encryption and decryption utilises less than 8% of the processor's computational capacity at the speed of 2.4 kbps. In terms of program space requirements, speech analysis requires 3.3 K words, speech synthesis 3 K words, and DES encryption/decryption 1.5 K words.

An important design consideration is to avoid overflow of the circular system buffers shown in Figure 3. Considering that every speech frame is 180 samples, buffers I_1 and O_2 are set to 518 words each. Also, since every DES encryption block is 64 bits, buffers B_1, O_1, I_2, and B_2 are set to 128 words each. Finally, it is important to note that the DES mode of operation implemented in the commercial version of SecLine-Plus is the Output Block Feedback (OFB) mode. This mode offers increased immunity to bit errors likely to occur in the noisy communications channel (telephone lines). To ensure that a unique Initial Value (IV) is used with every encryption block, a 64 bit counter with a random starting value is used.

7 Conclusion

A single microprocessor chip approach to secure voice and data communications systems has been presented.

This architecture offers increased security, ease to future upgrades (modification of encryption algorithms), reduced system size, lower power consumption and cost. All this is achieved at the expense of increased software complexity which, however, can be accommodated by the high power DSPs available today.

[1] P. Papamichalis: *"Practical Approaches to Speech Coding,"* Englewood Cliffs, NJ: Prentice-Hall, 1987.

[2] W. Hess: *"Pitch Determination of Speech Signals,"* N.York: Springer-Verlag, 1983.

[3] G. Troullinos: *"A Software Based Approach to Secure Voice Communications,"* IEEE , ICECS '96 , October 1996.

[4] Data Encryption Standard, FIPS Publication 46, National Bureau of Standards, 1977.

[5] W. Diffie, and M.E. Hellman: *"New directions in Cryptography,"* IEEE Transactions on Information Theory, Vol. IT-22, pp. 644-654, November 1976.

[6] R. Rivest, A. Shamir, and L. Adleman: *"A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,"* Communications of the ACM, Vol. 21, No. 2, pp. 120-126, February 1978.

[7] Donald Knuth: *"Seminumerical Algorithms,"* Vol. 2, The art of Computer Programming, Addison Wesly, 1981.

[8] R. Solovay, and V. Strassen: *«A Fast Monte-Carlo Test for Primality,"* SIAM Journal on Computing, Vol. 40, No. 52.