Performance Analysis of Real Time Implementations of Voice Encryption Algorithms using Blackfin Processors

Cristina-Loredana Duta, Laura Gheorghe and Nicolae Tapus
Department of Computer Science and Engineering, University Politehnica of Bucharest, Bucharest, Romania

Keywords: Speech Processing, Voice Encryption, Digital Signal Processor, Grain V1, Trivium, Mickey 2.0, Blackfin Processor.

Abstract: A large part of the latest research in speech coding and speech encryption algorithms is motivated by the need of obtaining secure military communications, to allow effective operation in a hostile environment. Since the bandwidth of the communication channel is a sensitive problem in military applications, low bit-rate speech compression methods and high throughput encryption algorithms are mostly used. Several speech encryption methods are characterized by very strict requirements in power consumption, size, and voltage supply. These requirements are difficult to fulfill, given the complexity and number of functions to be implemented, together with the real time requirement and large dynamic range of the input signals. To meet these constraints, careful optimization should be done at all levels, ranging from algorithmic level, through system and circuit architecture, to layout and design of the cell library. The key points of this optimization are among others, the choice of the algorithms, the modification of the algorithms to reduce computational complexity, the choice of a fixed-point arithmetic unit, the minimization of the number of bits required at every node of the algorithm, and a careful match between algorithms and architecture. This paper describes the performance analysis on Digital Signal Processor (DSP) platform of some of the recently proposed voice encryption algorithms, as well as the performance of stream ciphers such as Grain v1, Trivium and Mickey 2.0 (which are suited for real time voice encryption). The algorithms were ported onto a fixed point DSP, Blackfin 537, and stage by stage optimization was performed to meet the real time requirements. Memory optimization techniques such as data placement and caching were also used to reduce the processing time. The goal was to determine which of the evaluated encryption algorithms is best suited for real time secure communications.

1 INTRODUCTION

Speech represents the fundamental form of communication between humans. There are two methods to represent the speech: through its message content (as information) and as an acoustic waveform (the signal which carries the message information).

In the last years, due to the advancements in communication technology and the increasing demand of speech based applications, security has become an important aspect. The purpose of secure communication is to overcome unwanted disclosure and unauthorized modifications while transmitting speech through insecure channels.

The redundancy of the language plays an important role in secure speech communication systems such that if the language is highly redundant, an intruder can decipher much easier the information. Traditional solutions to ensure communications confidentiality were based on scrambling techniques (which include simple permutations and affine transformations in frequency or time domain). Due to the fact that in the last decade, the computing power has quickly increased, these scrambling algorithms became vulnerable to attacks. In this context, many real-world cryptographic implementations shifted to integrating encryption with compression algorithms in order to reduce the size of the signal before encryption and to eliminate the redundancy.

In general, there are four main categories of speech encryption: frequency domain scrambling, time domain scrambling, amplitude scrambling and two-dimensional scrambling (combination of time-domain and frequency-domain scrambling). In the

Duta, C-L., Gheorghe, L. and Tapus, N.
Performance Analysis of Real Time Implementations of Voice Encryption Algorithms using Blackfin Processors.
DOI: 10.5220/0005744601570166
In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, pages 157-166
ISBN: 978-989-758-167-0

transform domain, there are many speech encryption methods. For instance, methods such as fast Fourier transform, discrete cosine transform and wavelet transform are widely used. Recently, some new voice encryption methods were developed based on chaotic maps and on circular transformations.

Speech encryption algorithms can also be classified into digital encryption and analogue encryption methods. Analogue encryption operates on the voice samples themselves. The main advantage of analogue encryption is the fact that no modem or voice compression method is required for transmission. Moreover, the quality of the voice which is recovered is independent of the language. This type of encryption is recommended to be used for the existing analog channels such as telephone, satellite or mobile communication links.

Digital encryption performs as a first step the digitization of the input voice signal. Then, the digitized signal will be compressed to produce a bit stream at suitable bit rate. The resulting bit stream will be encrypted and transmitted through insecure channels. This type of encryption ensures high voice quality, low distortion and is considered cryptanalytically stronger than analogue encryption.

Complex digital speech encryption algorithms were developed due to the appearance of Very Large Scale Integration (VLSI) and DSP chips and are nowadays used in applications such as voice activated security, personal communication systems, secure voice mail and so on. A part of these applications require devices that have limited resources, which means that their implementation is dependent on constraints such as memory, size and power consumption. In this context, because of the advantages offered, DPSs represent the best solution for obtaining high performance speech encryption, under real time requirements. Moreover, hardware cryptographic algorithms are more physically secure, which makes it hard for an attacker to read information or to modify it.

The purpose of this paper was to optimize and to compare the performance of six speech encryption algorithms which can be easily embedded in low power, portable systems and which can be used in real time. This paper focuses on the following speech encryption methods: three stream ciphers (Mickey 2.0, Grain v1, Trivium), scrambling encryption algorithm, Robust Secure Coder (RSC) algorithm, encryption algorithm based on chaotic map and Blowfish algorithms. An important aspect presented in this paper is solving the problem of optimizing the implementations of previously mentioned voice encryption algorithms on DSP platforms. All the algorithms were ported onto a fixed point DSP and a stage by stage optimization was performed to meet the real time requirements. The goal was to determine which of the evaluated encryption algorithms is best suited for real time secure communications (in terms of performance).

This paper is organized as follows. The necessary background for our work is presented in Section 2. Related work is described in Section 3. Details regarding the architecture and implementation of voice encryption algorithms are presented in Section 4. The experimental results for the un-optimized code and for the optimized code of the speech encryption algorithms are described in Section 5. Conclusions are summarized in Section 6 together with our future work.

2 BACKGROUND

This section includes a brief description of Mixed Excitation Linear Prediction (MELP), a speech coding algorithm, of stream ciphers such as Mickey v2, Trivium, Grain v1.0, of recently developed voice encryption algorithms and the description of general aspects of DSP architectures.

2.1 MELP Algorithm

Voice coders are widely used in digital telecommunications systems to reduce the required transmission bandwidth.

Since the late 1970s, vocoders have been implemented using linear prediction which is a technique of representing the spectral envelope, a method conducting to linear predictive coding (LPC) (Tremain, 1982). The main disadvantage of LPC method is the fact that sometimes it sounds buzzy or mechanical because of the inability to reproduce all kinds of voiced speech using a simple pulse train.

MELP vocoder (McCree, 1996) and (Supplee, 1997) is based on LPC model, but has some additional features such as: mixed-excitation, pulse dispersion, adaptive spectral enhancement and aperiodic pulses. The mixed-excitation reduces the buzz which is in general encountered in LPC vocoders. Aperiodic pulses ensure easy transitions between unvoiced and voiced segments of the signal. More exactly, the synthesizer can reproduce, without having tonal noises inserted, erratic glottal pulses. The pulse dispersion is, in general, implemented using a filter, which disperses the excitation energy with a pitch period. This feature is important for synthetic speech, because the harsh

quality of it is reduced. The filter for adaptive spectral enhancement provides a more natural quality to the outputted speech signal, by improving the match between natural and synthetic waveforms.

2.2 Trivium Stream Cipher

Trivium stream cipher (Canniere, 2006) has 80-bit initialization vector (IV) and a key of 80 bits and can generate a keystream up to 2^{64} bits. The secret state of the algorithm has 288 bits which includes three non-linear feedback shift registers of different lengths: 93, 84 and 111 bits. As the majority of stream ciphers, Trivium has two phases: the setup of key and IV phase and the keystream generation phase. The keystream generation operates in each clock cycle on three input bits and produces one output bit. The initialization includes 1152 steps of the clocking procedure. The algorithm is designed such that it allows improvement of the throughput using parallelization (64 iterations can be computed at once), without increasing the area necessary for implementation.

2.3 Mickey 2.0 Stream Cipher

Mickey 2.0(Babbage, 2006) is a synchronous stream cipher, which stands for "Mutual Irregular Clocking KEYstream generator". The cipher works with an IV with length up to 80 bits and accepts keys of 80 bits length. Mickey produces the ciphertext by performing bitwise XOR between the plaintext and the keystream bits. The keystream sequence can be of maximum 2^{40} bits. The state of the algorithm includes two 100-bit shift registers (one nonlinear and the other linear) which are clocked one by the other in an irregular mode. The designers have also created a version of the cipher (MICKEY1-28 2.0) which accepts an IV up to 128 bits and a 128-bit key. Regarding the implementation, the authors mention that they were able to generate, using a PC with 3.4 GHz Pentium 4 processor, 10^8 bits in approximately 3.81 seconds.

2.4 Grain V1 Stream Cipher

Grain stream cipher (Hell, 2007) uses an 80-bit key and a 64-bit IV. This initial version was revised because vulnerabilities were discovered in its structure and a new version Grain v1 was created which includes two stream ciphers: one for 128-bit keys (with 80-bit IV) and one for 80-bit keys (with 64-bit IV). These ciphers include a non-linear feedback register and a linear feedback register which are coupled using lightweight boolean functions. Even though, the Grain family ciphers design includes an ingenious multiplication of throughput speed, this feature increases the space consumed.

2.5 Scrambling Speech Encryption Algorithm

In (Ravikrindi, 2011), the authors use for speech encryption a scrambling technique. For this, they have developed a software program in assembly language programming techniques for Digital Signal Processor ADSP 2181 (which is a 16 bit fixed point processor).

The algorithm works as follows. The first step is to acquire the voice signals, then to digitally code them and store the values in the memory of the processor (128 samples are stored). The scrambling of the speech signal is performed using Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) techniques.

The next step is to perform decoding, in order to obtain the speech signal again, which will be transmitted to the receiver. After applying FFT, the signal is converted into spikes (in frequency domain).

The signal spikes will be stored in the memory and based on circular buffers, their positions will be interchanged, ensuring in this manner the scrambling of the original signal (first 64 samples are displaced into next 64 samples position and vice versa). After scrambling, IFFT is applied to convert the signal from frequency domain, back into the time domain.

The last step is to convert the digital signal into the analog signal and to transmit it. Figure 1 illustrates the speech encryption process, where ADC represents the Analog-to-Digital Converter and DAC represents the Digital-to-Analog Converter.

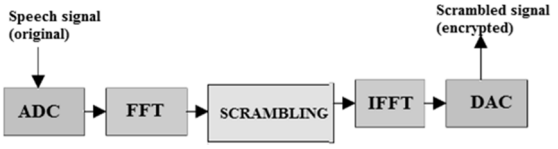

Figure 1: Speech encryption process (Scrambling).

The receiver performs the same steps previously described, obtaining at the end of the process the original speech signal. The original signal is obtained when the spikes (mentioned in the encryption process) are placed into their original positions, process which happens using the circular buffers. Figure 2 shows the speech decryption

process.

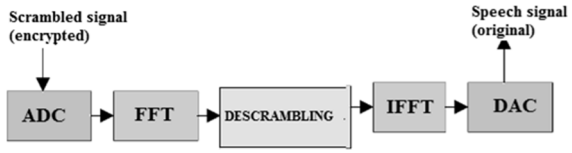

Figure 2: Speech decryption process (Descrambling).

2.6 Robust Secure Coder (RSC) Speech Encryption Algorithm

In (Babu, 2012), the authors present their scheme for speech encryption, called RSC, which includes MELP compression algorithm, Triple Data Encryption Standard (3DES) encryption algorithm and a Forward Error Correction (FEC) algorithm. Figure 3 present the speech encryption process. The original speech is passed through MELP encoder in frames of 22.5 milliseconds. The frame is coded into 54 bits of compressed speech frame. Because MELP algorithm ensures FEC for unvoiced mode only, the designers of RSC use 10 parity bits to ensure error correction only for voiced mode. The 54 bits of speech previously compressed and the 10 bits of FEC are given as input for 3DES encryption process. The result is 64 bits of encrypted and compressed speech, which will be transmitted to the receiver.

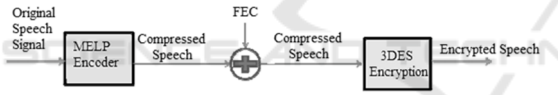

Figure 3: Speech encryption process (RSC algorithm).

The decryption process is illustrated in Figure 4 and is similar with the encryption process. The 64 bits of encrypted speech are given as input for 3DES decryption process, resulting 64 bits which will then enter in the FEC. 10 of 64 bits are used to correct errors and the rest of 54 bits are separated and given as input to MELP decoder. The output of the decoder is a synthesized speech frame of 22.5 milliseconds.

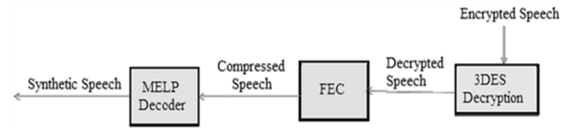

Figure 4: Speech decryption process (RSC algorithm).

2.7 Chaotic Map and Blowfish Speech Encryption Algorithms

In (Ulkareem, 2012), the authors describe a new solution to encrypt speech signal which includes chaotic encryption algorithm based on logistic map and Blowfish encryption algorithm. The advantage of using chaotic function is that it increases the security of the algorithm and the complexity of the encryption and decryption functions. The solution proposed by the authors' works as follows. For the encryption process, which is described in Figure 5, as a first step, the raw speech signal is divided into frames, each frame containing 256 values. Then, the speech frames chosen for encryption are decomposed using wavelet packet transform (WPT), to determine the decomposed frames coefficients of the level 2. At the end of this process, 256 coefficients are found for each selected frame. The next step is to use chaotic logistic map to generate a chaotic key which will be XORed with each frame value. The Blowfish algorithm provides two parts of 128 chaotic encrypted values which are merged to obtain an encrypted frame of 256 values.

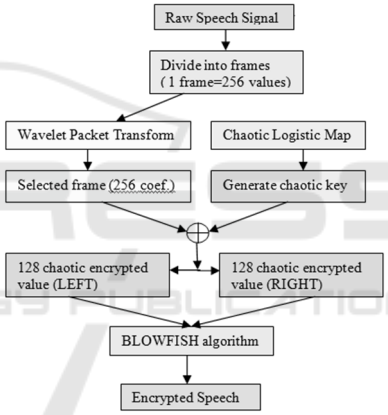

Figure 5: Speech encryption process (Chaotic map and Blowfish algorithms).

The decryption process is illustrated in Figure 6. In the first step, the encrypted speech signal is divided into frames (each containing 256 values) and each frame is split into left part (128 values) and right part (128 values). The two parts are given as input for Blowfish algorithm and then the decrypted frames are XORed with the chaotic key. At the end of the Blowfish decryption process, a frame of 256 values is obtained, which is passed through Inverse WPT (IWPT) restoring in this way the decrypted speech signal.

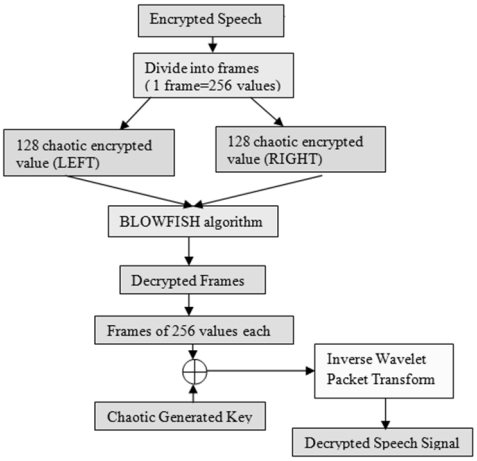

Figure 6: Speech decryption process (Chaotic map and Blowfish algorithms).

2.8 DSP Architecture

In general, speech coding and speech encryption algorithms include intensive processing operations and for this reason it's recommended to implement them on dedicated DSP which have instructions to handle these types of computations.

Several real-time speech encryption applications are characterized by very tight requirements in size, voltage supply and power consumption. In order to fulfill these constraints, thorough optimization should be performed at all levels, starting from algorithmic level, then at system and circuit architecture level, and also at layout and design of the cell library.

A block diagram of embedded DSP architecture is shown in Figure 7. This contains the processor core, the peripherals, the memory and others (Direct Memory Access controller, event controller etc.). The DSP core includes Data Address Generator (DAG), Arithmetic Logic Unit (ALU), register sets and sequencer.

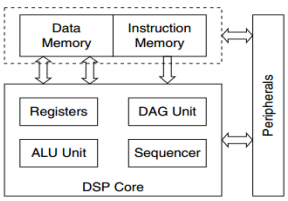

Figure 7: A diagram of DSP architecture.

The most important aspect regarding the DSP is to decide between floating point and fixed point computational core. Very fast implementations can be obtained by using floating-point processors, but these are not bit-exact. In this context, the best option for real time implementations is a specialized, fixed-point processor and this is what we have used for the implementations of previously described speech encryption algorithms.

We have chosen for our project's implementation Blackfin ADSP-BF537 (ADSP-BF357, 2013) processor core architecture, because it combines: flexible single instruction multiple data capabilities for parallel computations, an orthogonal RISC-like microprocessor instruction set, zero-overhead loops, a dual-MAC (Multiply and Accumulate) signal processing engine, and multiple timed features into a single instruction set. Blackfin contains an internal ADC and is much faster than microcontrollers. VisualDSP++ software can be used to simulate the behavior of the DSP chip. Unfortunately, even with this specialized DSP, optimization techniques are still necessary in the implementation of stable speech encryption algorithms with real-time performances.

3 RELATED WORK

To give a better perspective about the importance and utility of our project this section describes the results obtained by other researchers and other developed speech encryption algorithms.

(Wollinger, 2000) presents the results obtained after comparing the optimized implementations of Advanced Encryption Standard (AES) finalists on DPS. The evaluated algorithms were Mars, RC6, Rijndael, Serpent and Twofish and the implementations were done using TMS320C64x platform. The test scenarios included single-block mode and multi-block mode (two blocks at a time were encrypted) and the results were measured in: the number of cycles (necessary for the encryption process of each algorithm), the throughput (Mbit/sec) and memory usage.

In (Good, 2008), the hardware performances of the eSTREAM competition finalists are presented. The framework designed by the authors takes into considerations the following evaluation elements: compactness, throughput, power consumption, simplicity and scalability. Their evaluation shows that the simplest algorithm is Mickey128, the most flexible one is Trivium and that Grain80 offers the best results for two samples application of future wireless network and low-end of radio frequency identification tags/ wireless sensor network nodes.

In paper (Servetti, 2002), the authors propose a speech encryption technique which uses low

complexity perception based on partial schemes. The speech signal is compressed using ITU-T G.729 standard (ITU-t, 1996) and the result is divided into two classes: one is encrypted and the other is unprotected. Also, there are two level partial encryption techniques used, one low protection (for eavesdropping prevention) and one high-protection (for full encryption of the compressed bit stream).

In (Chen, 2007), a speech encryption method based on vector quantization (VQ) of LPC coefficients is presented. The secret key is generated using the indices of VQ corresponding to the neighboring frames derived from the natural speech's characters. (Girin, 2007) presents an encryption algorithm based on time-trajectory model of the sinusoidal components corresponding to voiced speech signals. This method uses the amplitude and phase parameters of the discrete cosine functions which are applied for each voiced segment of the speech.

A method for speech encryption based on augmented identity matrix is presented in (Tingting, 2009). Enhanced encryption can be achieved by analyzing the redundancy parameters of the coded speech signal, with low computation complexity in real time applications. In (Merit, 2012), a voice encryption method called "DES with Random permutation and Inversion" is described, which solves the problem of penetrating the RPE-LTP vocoder by the encrypted voice. The solution proposed ensures secure communication in Global System for Mobile Communications (GSM) and a good compatibility to all GSM networks.

The authors describe in (Kaur, 2012) a new speech encryption algorithm which integrates a personalized time domain scrambling scheme and is based on four level of hash based encryption. They encrypt the original signal four times using different algorithms (repositioning of bits, using twice random number generation and amplitude ascending ordering) at each level. Based on their experimental results, it can be seen that the proposed algorithm ensures a high level of security.

In (Knezevi, 2013) the authors illustrate how signal processing techniques can be used to design and implement cryptographic and security applications. Implementations on DSP processors of well known hash functions, public-key algorithms are described in detail. Moreover, using the special features of DSP processor, they present a key derivation technique and other methods of pre-processing data which can be very useful in performing side-channel attacks.

4 IMPLEMENTATION

In this section we present the details regarding real-time and offline implementations of six speech encryption algorithms (described in Section 2).

A point of interest in this area is getting DSP microprocessors, embedded in a system which performs speech encryption and decryption. Implementation in C is structured and easy to follow and can be an important starting point for implementing these algorithms on various platforms.

The algorithms were implemented at the beginning in C language using Microsoft Visual Studio2012, which makes the software processor independent and can be linked with any processor if the corresponding assembler is accessible. After we thoroughly analyzed their functionality, the code was transferred to the integrated development environment, called VisualDSP++, on a DSP platform. In the first stage, the algorithms were tested offline, using a single processor so that we could verify if the implementations remain functional even when their included in this new environment.

After we implemented the speech encryption algorithms on a DSP platform, we optimized the programs so that it allows real-time communication. This was done using two fixed point DSPs from Analog Devices, ADSP-BF537, as it can be seen in Figure 8. The communication between the DPSs is done using a serial transmission (through UART) and MELP algorithm was used to compress the speech signal. This block diagram is not available for the implementation of scrambling speech encryption algorithm (where the digital signal is given as input for the FFT function without compressing it) and for encryption algorithm based on chaotic map and Blowfish cipher (it uses WPT for compression).

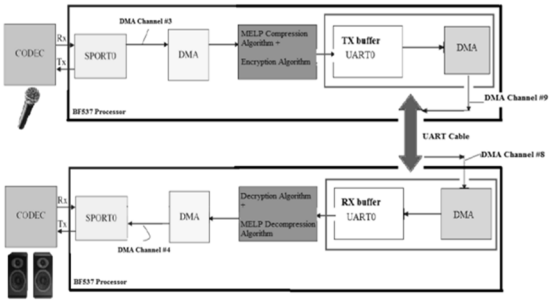

Figure 8: Real-time communication on BF537 block diagram.

Several optimizations were necessary to meet the real time requirement of completing all computation processes within frame duration. The source code of speech encryption algorithms were thoroughly optimized at the C Level.

5 OPTIMIZATION AND EXPERIMENTAL RESULTS

Since most embedded systems are real-time systems, code optimization in terms of execution speed is an important performance index which will result in lower power consumption. It is always easier to use a C compiler optimization. However, in cases where we have to save more MIPS or memory processing, assembly code optimization is the only way to achieve this level of performance. The development cost of writing 100% of the entire program in assembly language, far exceeds the performance gain.

A better approach is to start writing code in C, then create a detailed profile to identify time critical code sections and replace these code segments with code in assembly language. A common rule 80% - 20% says that 80% of processing time is used for 20% of code. So if we can identify those 20% of the code and optimize it using assembly language, significant performance gains can be achieved. A method for identifying the region of interest is to use the statistical profiler (for EZ-KIT) or the linear profiler (for a simulator) that exist in the VisualDSP++. Running the applications in VisualDSP++ for the first time, generated the execution times (in milliseconds) seen in Table 1, for compressing and encrypting a single frame.

Table 1: Execution time per frame before code optimization.

Speech Encryption Algorithm	Compression Algorithm	Execution time/frame (in ms)
Trivium	MELP	168.312
Mickey v2.0	MELP	160.125
Grain v1 - 80 bit key	MELP	166.129
Grain v1 -128 bit key	MELP	168.934
Scrambling	-	162.331
RSC	MELP	172.624
Chaotic map& Blowfish	WPT	174.753

We started by applying different optimization techniques at C level (Table 2) then we applied different hardware optimizations which can be seen in Table 3. The execution time (given in milliseconds) decreased significantly for all encryption algorithms as it can be seen in Table 4. All performed computational processes lasted more than 170 ms before code optimization, result which is inacceptable, given the time available for a frame (22.5 ms). After the optimization, the execution time per frame was reduced to less than 21.5 ms for all implemented algorithms.

Table 2: C level optimizations.

Optimization technique
Enable: optimization for C code, automatic inlining, interprocedural optimization from VisualDSP++ options
Use pragma for optimizing loops
Use pragma for data alignment Use pragma for different memory banks Use pragma for no alias
Use volatile and static data types
Use arithmetic data types (int, short, char, unsigned int, unsigned char, unsigned short)
Using runtime C/C++ and DSP libraries
Use pragma to optimize for speed
Using intrinsic functions and inline assembly
Profile Guided Optimization (PGO) - the compiler uses the data collected during program execution for an optimization analysis. PGO informs the compiler about the functions that affect branch prediction, improve loops transformation and reduce code size.

Table 3: Hardware level optimizations.

Optimization technique
Special addressing modes – using different data sections (for add() function)
Using assembly code
Using hardware loops
Using parallel instructions
Using software pipeline

Table 4: Execution time per frame after C level and hardware level optimizations.

Speech Encryption Algorithm	Execution time/frame (in ms) C level	Execution time/frame (in ms) Hardware level
Trivium	71.034	15.884
Mickey v2.0	61.977	6.349
Grain v1 - 80 bit key	67.893	11.815
Grain v1 -128 bit key	70.102	14.209
Scrambling	65.340	10.053
RSC	74.298	18.644
Chaotic map& Blowfish	78.112	21.115

As it can be seen in Table 4, the smallest execution time per frame (in ms) at C level is obtained for the implementations of Grain v1 (80 bit

and 128 bit key) and by the scrambling algorithm (approximately 65.5 ms). Mickey v2.0 algorithm has the smallest execution time at hardware level, only 6.439 ms. The highest execution time is obtained at software level and at hardware level by chaotic map &Blowfish algorithm (78 ms, respectively 21 ms).

Memory optimization techniques such as caching and data placement were also used to bring down the processing time. It was not possible to include the entire data inside the internal RAM of the DSP. For this reason, less frequently accessed data was kept in SDRAM which was comparatively slower. Frequently accessed functions were cached. A better optimization is achieved by writing the functions which are computationally intensive in assembly language. The result is a substantial reduction of the number of cycles needed for computation.

In Table 5, the CPU time for optimized and non-optimized speech encryption implementations is shown for each time consuming function.

Table 5: CPU time for optimized and non-optimized algorithms implementations.

Algorithm	Time consuming functions	No Optimi-zation	With Optimiza-tion
Trivium	Multiplying the algebraic normal form of two boolean functions	3.25 Mcycles	77 Kcycles
Mickey v2.0	Register clocking	2.0 Mcycles	55 Kcycles
	Keystream derivation		
Grain v1 - 80	Initialization phase	2.95 Mcycles	84 Kcycles
Grain v1 -128	Initialization phase	3.20 Mcycles	98 Kcycles
Scrambling	FFT	2.80 Mcycles	96 Kcycles
	IFFT		
RSC	Forward Error Correction	3.55 Mcycles	120 Kcycles
Chaotic map & Blowfish	Chaotic key generation	4.2 Mcycles	155 Kcycles
	Subkey generation for Blowfish		

For Trivium algorithm the most consuming function includes the multiplication of two Boolean functions using their algebraic normal form. Optimizing this function reduces the number of cycles with more than 2 Mcycles. For Mickey v2.0 cipher there are two time consuming functions: register clocking and keystream derivation, which if optimized save approximately 1.5 Mcycles. The initialization phase is the most consuming for Grain v1 algorithm (80 bit key or 128 bit key). Optimizing this function the number of cycles decreases with almost 2 Mcyles. For the scrambling technique, the FFT and IFFT functions consume approximately 3 Mcycles. After the optimization, these functions require less than 1 Mcycles. RSC algorithm has only one time consuming function, which is the Forward Error Correction. It's optimization is significant, from 3.55 Mcycles to 120 Kcycles. The cipher which uses chaotic map and Blowfish algorithm for voice encryption has two important functions: chaotic key generation and subkey generation for Blowfish. The reduction of clock cycles is high, from 4.2 Mcycles to 155 Kcycles.

Based on the results in Table 5, we can calculate the Clock Rate Reduction (CRR). This is defined as in equation (1), where X represents the number of clock cycles consumed by original code (before optimization) and Y is the number of clock cycles consumed by the optimized code. The CRR values for all implemented encryption algorithms can be seen in Table 6. As it can be observed, the best CRR was obtained for Trivium algorithm (41.21%), followed by Mickey v2.0 (35.36%). The smallest CRR was obtained for chaotic map and Blowfish algorithm, 26.09%.

$$CRR = \frac{(X - Y)}{Y} \times 100\% \quad (1)$$

Table 6: CRR value for all speech encryption algorithms.

Algorithm	CRR
Trivium	41.21%
Mickey v2.0	35.36%
Grain v1 - 80	34.11%
Grain v1 -128	31.65%
Scrambling	28.16%
RSC	28.58%
Chaotic map & Blowfish	26.09%

We have also performed subjective analysis for the offline and real-time implementation of the evaluated speech encryption algorithms and the results are shown in Figure 9.

In subjective analysis, the encrypted speech signal is listened and the quality of it will be determined only based on the listener's opinion. Ten listeners have graded the six algorithms. Each person has listened to 10 distinct audio files and then they gave grades on a scale of 0 to 10. As it can be seen from Figure 9, the scores confirm the fact that

offline speech encryption algorithms' implementations have better performances than the real-time implementations. "O" stands for offline implementation and "RT" stands for real-time implementation. The best results in subjective analysis were for offline Mickey v2.0 algorithm (9.1), followed very closely by offline RSC (8.92) and by offline Trivium (8.91). The worst results were for real-time implementations, especially of algorithms such as Grain v1 (7.14), RSC (7.32) and Trivium (7.34).

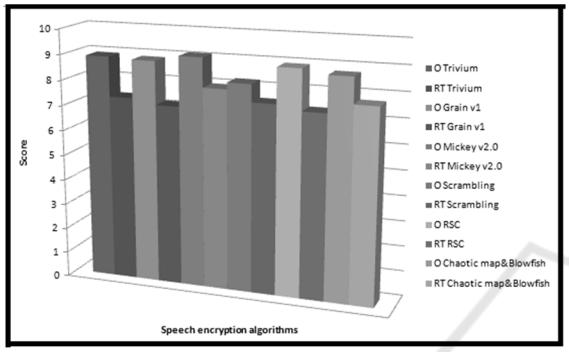

Figure 9: Scores for offline and real-time implementations.

6 CONCLUSIONS

Because secure communications are extremely important nowadays, we presented in this paper six speech encryption techniques that can be used in real-time applications. The implementations were performed in C and the code was ported on Blackfin ADSP-BF537. Thorough optimization of the code was carried out to achieve real-time implementations. The optimization process was performed step by step by using different methods which includes compiler tools, small function inline expansion, and intrinsic functions and so on. We were able for all algorithms to reduce the execution time per frame to less than 21.5 milliseconds. The results are ideal and meet the needs of engineering applications: real-time implementations of speech encryption algorithms on hardware platforms. However, it can be seen that the best results (smallest number of cycles and execution time per frame) were obtained for Mickey v2.0 stream cipher implementation and the worst results were obtained for the encryption algorithm proposed in (Ulkareem, 2012), which is based on chaotic map and Blowfish cipher.

Taking into consideration the subjective analysis results, it can be observed that offline implementations of the algorithms provide better speech quality than real-time implementation. All in all, the proposed speech encryption algorithms have a good audio quality, which is an extremely important feature for communication applications.

This work can be extended to developing hardware products which can be used to ensure secure real-time communications. Also, the algorithms we have selected for speech encryption can be implemented on other DSP platforms, which will offer more optimization methods and better performances.

ACKNOWLEDGEMENTS

The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/187/1.5/S/155536, by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/134398, and by the program Partnerships in priority areas – PN II carried out by MEN-UEFISCDI, project No. 47/2014.

REFERENCES

Tremain, T. E., 1982. The Government Standard Linear Predictive Coding LPC-10. In: *Speech Technology*, pp.40-49.

McCree, A., Kwan, T., George, E. B.,Viswanathan, V., 1996.A 2.4 kbit's MELP coder candidate for the new U.S. Federal Standard. In: *Acoustics, Speech, and Signal Processing, IEEE International Conference*, vol.1, pp. 200-203.

Supplee, L. M., Cohn, R. P., Collura, J. S.,McCree, A., 1997. MELP: the new Federal Standard at 2400bps. In: *Acoustics, Speech, and Signal Processing, IEEE International Conference*, vol.2, pp. 1591-1594.

Canniere, C., Preneel, B., 2006.Trivium Specifications. In: *eSTREAM, ECRYPT Stream Cipher Project*.

Babbage, S., Dodd, M., 2006. The stream cipher MICKEY 2.0. In *eSTREAM, ECRYPT Stream Cipher Project*.

Hell, M., Johansson, T., Meier, W., 2007. Grain- A Stream Cipherfor Constrained Environments. In: *International Journal of Wireless and Mobile Computing*, vol. 2, pp. 86-93.

Ravikrindi, R., Nalluri, S., 2015. Digital Signal Processing, Speech encryption and decryption, availableat https://www.scribd.com/doc/23336087/

11-speech-encryption-and-decryption (Accessed: July 2015).

Babu, A. A., Yellasiri, R., 2012.Symmetric Encryption Algorithm in Speech Coding for Defence Communications. In: *Journal of Computer Science & Information technology*, vol. 4, pp. 369-376.

Ulkareem, M., Abduljaleel, I. Q., 2013.Speech Encryption Using Chaotic Map and Blowfish Algorithms. In: *Journal of Basrah Researches*, vol. 39, no. 2, pp. 68-76.

ADSP-BF537 Blackfin Processor Hardware Reference manual, Revision 3.4 (2013).

Wollinger, T. J., Wang, M., Guajardo, J., Paar, C., 2000. How Well Are High-End DSPs Suited for the AES Algorithms? In: *Proceedings of the Third Advanced Encryption Standard Candidate Conference*, pp. 94-105.

Good, T., Benaissa, M., 2008.Hardware performance of eStream phase-III stream cipher candidates. In: *State of the Art of Stream Ciphers (SASC)*, pp. 163-173.

Servetti, A., De Martin, J.C., 2002.Perception-based partial encryption of compressed speech. In: *IEEE Transactions on Speech and Audio Processing*, vol.10, pp. 637 – 643.

ITU-t recommendation g.729, 1996, coding of speech at 8 kbit/s using conjugate-structure algebraic-codeexcited-linear-prediction (cs-acelp).

Chen, N., Zhu, J., 2007.Robust speech watermarking algorithm. In: *Electronics Letters*, vol. 3, pp. 1393-1395.

Girin, L., Firouzmand, M., Marchand, S., 2007.Perceptual Long-Term Variable-Rate Sinusoidal Modeling of Speech. In: *IEEE Transactions on Audio, Speech, and Language Processing*, vol.15, pp. 851 – 861.

Tingting, X., Zhen, Y., 2009.Simple and effective speech steganography in G.723.1 low-rate codes. In: *International Conference on Wireless Communications & Signal Processing*, pp. 1 – 4.

Merit, K., Ouamri, A., 2012.Securing Seech in GSM Networks using DES with Random Permutation and Inversion Algorithms. In:*International Journal of Distributed and Parallel Systems (IJDPS)*, vol.3, no.4, pp.157-164.

Kaur, H., Sekhon,G. S., 2012.A Four Level Speech Signal Encryption Algorithm. In: *International Journal of Computer Science and Communication (IJCSC)*, vol. 3, no. 1, pp. 151-153.

Knezevi, M., Batina, L., Mulder, E., Fan, J., Gierlichs, B., Lee, Y. K., Maes, R., Verbauwhede I., 2013.Signal Processing for Cryptography and Security Applications.In *Handbook of Signal Processing Systems*, pp. 223-241.

Olausson, M.,Dake, L., 2003.The ADSP-21535 Blackfin and Speech Coding.In*Proceedings of the Swedish System-on-chip Conference 2003*.

Blackfin DSP Instruction Set Reference, 2002.

ADSP-21535 Blackfin DSP Hardware Reference, 2002.

ITU-t recommendation on g.723.1, 1996, dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s.

ETSI GSM Fullrate Speech Codec for Analog DevicesBlackfin, Bayer DSP Solutions, 2008.

Bertini, G.,Fontata, F., Gonzalez, D.,Grassi, L., Magrini, M., 2004.Voice Transformation Algorithms with Real Time DSP Rapid Prototyping Tools, unpublished.

Shaked, Y., Cole, A. L., 2004. Implementation of MELP based Vocoder for 1200/2400 bps, The EE Project Contest 2000, Technion Signal and Image Processing Lab, unpublished.