Advances in Computing 2015, 5(1): 1-8
DOI: 10.5923/j.ac.20150501.01

Real Time Secure Video Transmission Using Multicore CPUs and GPUs

K. Ganesan[1], Jerin Geogy George[2,*], Nithin P. V.[2]

[1]TIFAC-CORE in Automotive Infotronics and School of Information Technology and Engineering, VIT University, Vellore, India
[2]TIFAC-CORE in Automotive Infotronics, VIT University, Vellore, India

Abstract The main difficulty in implementing security to real time videos is the processing time. Processing time is considerably high while ensuring security using single core processors. This paper is analyzing the processing capability of Multicore CPUs and GPUs to facilitate a secure video transmission. Six different techniques using spatial and frequency domain are implemented using these systems. Adjacent frames are secured using different techniques to improve security. The processed video is transmitted over LAN to a neighboring system to see whether a real time reproduction is possible or not. The paper also compares the processing time for real time as well as stored videos, with varying resolution. In the case of real time video security, GPU system was able to transmit 23 frames per second while single core CPU system was able to transmit only 2 frames per second. Multicore CPU system with 8 cores was able to transmit 8 frames per second. The resolution of the video transmitted was 320x240. When just security techniques were applied (video not transmitted) on a stored video of resolution 640x480, the performance of GPU system was 38.3 times better than single core CPU system and 7.7 times better than multicore CPU system.

Keywords Video security, CUDA GPU, Multicore, Video processing

1. Introduction

Cryptography is a method of protecting the information from undesirable persons by converting it into an unrecognizable form. With the advent of smart phones, tablets and other electronic gadgets, video calling is gaining more popularity. Many airborne surveillance cameras are being used for commercial as well as military purposes. Massive accumulation of data invites users to store data in cloud storage systems. In all these cases sensitive information is being transmitted from one place to another. Video-on-demand is another area where security is needed to prevent unauthorized people from accessing the video. Regarding the protection of video, which may be containing many frames per second (fps), the two techniques that are widely employed are scrambling and encryption. In scrambling, the pixel values of the image are swapped between the indexes whereas in encryption the pixel values are modified with some algorithm. To avoid high processing time normally scrambling is employed to provide protection to videos. Standard encryption algorithms could provide better security but at the cost of processing time [1, 2].

Parallel computing is a form of computing in which many operations are carried out simultaneously. Since the processing time has become an important factor in all computations, parallel computing platform is getting wide acceptance. Parallel computing platform can be based on Central Processing Unit (CPU) or Graphics Processing Unit (GPU). CPU based parallel computing can be accomplished through Multi-core systems, Multiprocessors, Computer clusters, Grid computers etc. GPUs are normally used for video rendering and graphics enhancement. But now GPU manufacturers have developed new platforms which enable GPU for more general purpose usage, not just for graphics or videos. Compute Unified Device Architecture (CUDA) programming developed by NVIDIA facilitates the GPUs for general purpose computing. Most of the modern computers, smartphones, tablets etc. are having multiple cores and many of them also contain GPUs. So the opportunity for parallel computing in modern digital world is vast [3-5].

This paper studies the feasibility of using Multi-core CPUs / GPUs to transmit a secure real time video and reproduce it at the receiving end without considerable delay and jitter. Different security techniques in spatial and frequency domain are applied on adjacent frames in order to improve the security. The paper conducts a comparative study between the performance of single core CPU, multicore CPUs and GPUs on the basis of the computation performed. The computation is increased by increasing the resolution of the video to be processed. The paper also analyses the possibility of incorporating encryption

* Corresponding author:
jeringeogygeorge@amaljyothi.ac.in (Jerin Geogy George)

Published online at http://journal.sapub.org/ac

techniques to secure real time videos.

The organization of the paper is as follows: In Section II, the methodology adopted for securing the videos using parallel computing is elaborated in detail. In Section III, the focus is on the results obtained. Finally, in section IV we enumerate our conclusions.

2. Methodology

The major steps involved in the secure video transmission are as follows.

- Capture a real time video or open a stored video.
- Extract each frame from the video.
- Scramble/encrypt the pixels in the frame using Multicore CPUs / GPUs.
- Transmit the processed frame to the client.
- Unscramble/decrypt the pixels of the received frame.
- Display the frame in real time or save the frame into a video file.

If the frames are displayed at a rate higher than the persistence of vision then the video can be shown in real time. This depends on both the transmission time and the processing time. Video is transmitted from the server to the client through Local Area Network (LAN) using socket programming.

2.1. GPU Based Parallel Computing

In order to process the frames using Graphics Processing Unit (GPU), Compute Unified Device Architecture (CUDA) library functions are used in the code which is programmed using C language. CUDA is a parallel computing platform developed by NVIDIA for general purpose GPU computing [6]. Image processing operations are performed using OpenCV library functions.

CUDA program allows us to use both CPUs and GPUs in one program. Part of the CUDA program written in C language runs in the CPUs (Host). The other part of the program runs in GPUs (Device) in parallel which is also written in C language but with some extensions to express parallelism. CUDA compiler compiles the code and splits it into pieces to be run on CPUs and GPUs. CUDA considers GPU as a coprocessor to the CPU with both of them having separate memories. CPU runs the main program and controls all the actions of GPU [6-8]. The major operations involved in a CUDA program are

- Moving the data from CPU memory to GPU memory.
- Allocating GPU memory.
- Invoking programs (kernels) in GPU that compute in parallel.
- Moving the data from GPU memory to CPU memory.

The part of the code which is to be parallelized using GPU is written as a device kernel. The codes inside the kernel get executed in parallel using multiple threads. The kernel is written in such a way that only a single thread is executing at a time. Kernel does not mention any thing about the level of parallelism. The number of threads that should be executed in a kernel is specified by the kernel call function. The maximum number of threads and blocks that can be scheduled depends on the compute capability of GPU. The threads are arranged in blocks in order to obtain high performance. The kernel is called from the host and is executed in the device. Before calling the kernel the data required for the operation is copied from the host memory to the device memory. The device memory is allocated according to the size of the data encountered. When the device finishes the calculations, the result is copied back from the device memory to the host memory. Thus the serial portions of program are run on CPUs and the parallel portions are run on GPUs [7, 8].

2.2. CPU Based Parallel Computing

In order to use parallelism using multiple cores, Open Multiprocessor (OpenMP) library functions are used in the code programmed using C language [3, 9]. Image processing operations are performed using OpenCV library functions.

OpenMP facilitates only user defined parallelization. User specifies the action to be taken by the compiler and system runs the program in parallel. Some compiler directives are used for this purpose. OpenMP constructs will not check for data dependencies, data conflicts, race conditions, deadlocks or any other situation that may give error output from the program. User should use the constructs carefully to avoid all these situations. Mainly, the parts of the code which contain loops with high iteration are parallelized [9, 10].

OpenMP program begins with a single thread (main thread) of execution. The parallel construct is added above the loop body which is to be parallelized. When the main thread encounters the parallel construct, it creates a team with additional threads. The number of threads to be created is mentioned in the construct [9]. The number of threads cannot exceed the maximum number of logical processors in the system. The task inside the parallel region is split among the threads. Each of the thread has a thread private memory and shared memory. At the end of the parallel construct only the master thread resumes the execution. There is an implicit barrier for all the other threads by the end of the construct [9, 10]. Only the loops that do not cause any data conflicts are parallelized using the above mentioned method. All the variables used inside the loop are also made independent in order to avoid data dependencies.

2.3. Security Techniques

Different spatial and frequency domain techniques are used to ensure security in the video. Adjacent frames are secured with different techniques to improve security. If all the frames in a video are secured using only a single technique, then it is easy to crack the security measures. The following are the various spatial and frequency domain techniques employed to secure the image frames in the video.

2.3.1. Arnold Transform

Arnold transform is used to scramble the pixels in a frame in spatial domain. Arnold transform is a process of clipping, splicing and realigning the pixel matrix of digital image. The new indexes for the pixels are calculated using equation (1) [11, 12].

$$\begin{pmatrix} \mathrm{x}' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} mod\ N \tag{1}$$

Here, (x’,y’) is the new index, which is calculated from the old index (x,y) and N is the size of the image matrix. If the image matrix is not a square matrix, then zeros are padded to make it as square matrix. By rearranging all the pixels to their new index, the frame gets scrambled.

Arnold transform is widely employed in digital image scrambling because of its periodicity. If the number of iterations used for scrambling is S, then the number of iterations to be used for unscrambling is (P-S). Here, P is the period of Arnold transform and it depends on the size of the image [11]. For each frame, the number of iterations (S) used for scrambling is varied in order to improve the security. The value of S is calculated from the pixels in the image frame. This value is transmitted along with the secured video and is used for unscrambling at the receiver side.

2.3.2. Cosine Transform Based Scrambling

The pixels in the frame are transformed into frequency domain using discrete cosine transform. Two dimensional discrete cosine transform is used for this purpose. The general equation for a 2D cosine transform is defined by equation (2) and equation (3) [13].

$$\begin{gathered} F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} A(i).\, A(j). \\ \cos\left[\frac{\pi u}{2N}(2i+1)\right] \cos\left[\frac{\pi v}{2M}(2j+1)\right].f\,(i.j) \end{gathered} \tag{2}$$

Here,

$$A(K) = \begin{cases} \frac{1}{\sqrt{2}} & for\ k = 0 \\ 1 & otherwise \end{cases} \tag{3}$$

Here, F(u,v) is the DCT coefficient and f(i,j) is the intensity of pixel at row i and column j.

A matrix containing integer numbers in ascending order as elements is scrambled using Arnold transform technique as mentioned earlier. The number of iterations used for scrambling is varied for each frame in order to improve security. This scrambled matrix is threshold to generate a matrix containing +1 and -1 as its elements. The matrix has the same size as that of the frame size. This matrix is multiplied with the matrix obtained by applying DCT to the video frame. Then the inverse DCT is applied on the obtained matrix. The new matrix obtained after IDCT is the scrambled matrix [14].

The frames are split into 8x8 blocks to perform discrete cosine transform. It has been found that in 8x8 blocks lots of information can be dropped without creating acceptable blocking artifacts [13, 15]. Selection of this size also provides better performance. The matrix containing +1 and -1 values is the key for unscrambling. If the same matrix operations are repeated again then the frame will get unscrambled.

2.3.3. Fourier Transform Based Scrambling

The pixels in the frame are converted into the frequency domain by FFT. In this domain some alterations are introduced and then the frame is converted back to the spatial domain. Before applying FFT, the single channel video frame is converted into two channel complex frame by adding a separate channel consisting of zeros. This is done because performing FFT will produce real values as well as imaginary values. Added channel is then used to store the imaginary values. The general equation for the 2D FFT used is defined by equation (4) [16].

$$F(x,y) = \sum_{M=0}^{M-1} \sum_{N=0}^{N-1} f(m,n)\, e^{-j2\pi\left(x\frac{m}{M}+y\frac{n}{N}\right)} \tag{4}$$

Here, F(x, y) is the FFT coefficient and f(m, n) is the intensity of pixel in row m and column n of the input image frame.

After applying FFT, the magnitude and phase of each matrix element is calculated. Then two matrices, namely, magnitude matrix and phase matrix are generated. The magnitude matrix is kept as the same but the phase matrix is permuted using Arnold transform technique. The number of iterations used for the permuting the phase matrix is calculated from the pixels in the frame. The new phase matrix is combined with the magnitude matrix to form the complex matrix. The real channel obtained after applying IFFT to the complex matrix gives the scrambled frame in the time domain [17].

The number of iterations (S) used for scrambling is transmitted along with the secured video and is used for unscrambling at the receiver side.

2.3.4. Cipher Block Chaining Encryption

To encrypt the video using cipher block chaining, a look up table is created. To create the look up table an 8 dimensional cat map is used. It is given by the equation (5) and equation (6) [14, 18].

$$\begin{bmatrix} A_{n+1} \\ B_{n+1} \\ C_{n+1} \\ D_{n+1} \\ E_{n+1} \\ F_{n+1} \\ G_{n+1} \\ H_{n+1} \end{bmatrix} = A \begin{bmatrix} A_n \\ B_n \\ C_n \\ D_n \\ E_n \\ F_n \\ G_n \\ H_n \end{bmatrix} \mathrm{mod}\ \ 256 \tag{5}$$

Here,

$$A=\begin{bmatrix}1&7&33&125&403&1119&2591&4279\\1&8&39&150&487&1356&3141&5182\\1&7&34&130&421&1171&2712&4476\\1&6&26&96&305&842&1948&3224\\1&5&19&63&192&520&1200&2000\\1&4&13&38&104&272&644&1056\\1&3&8&20&48&112&256&448\\1&2&4&8&16&32&64&128\end{bmatrix} \qquad (6)$$

Here, 8 initial conditions namely A_0, B_0, C_0, D_0, E_0, F_0, G_0, and H_0 are chosen. Equation (5) is applied repeatedly for 256 times to obtain 8 sequences of 256 unique set of values. Each time the value of n is incremented by 1. Each time the value of A_n is checked with its previous values. If the new value is equal to any of the previous value, then the new value is incremented till it is not equal to any of the previous values. After 256 operations, the sequence A (A_0, A_1, A_2,….A_{255}) will contain unique values. The sequence A along with its index value is used for encryption.

The pixel in the frame is matched against the values in the sequence A. There will be only one match since A contains unique set of values. Then the pixel is replaced with the index value of the matched value in sequence A. By doing same operation on all pixels, the entire frame gets encrypted. The same sequence can be used in reverse order to decrypt the video frame.

2.4. Socket Program

Socket program is used to transmit the secure video from the server system to the client system. Stream sockets which rely on TCP protocol to establish reliable connection is used for this purpose. The major steps involved in the data transmission are the following [19].

- Create socket (stream socket) at both server and client systems.
- Assign address to the sockets using bind function.
- Server listens to any connection request using listen function.
- Client uses connect function to request for a connection.
- Server accepts the connection request using accept function.
- Once the connection is established, send and receive functions are used to transmit the data.
- Close function is used to stop the data transmission and to close the connection.

Initially server and client systems are connected using a LAN cable. A local network is also established between the systems by manually setting the IP addresses.

2.5. Implementation

To ensure security, RGB image frames are captured from the video. Then the three channel image is split into single channel R, G and B images. Then each of these single channel images is secured using one of the above mentioned techniques. The technique used is decided by the image frame number, N. N%6 is computed and the corresponding security technique listed in Table 1 is applied.

Table 1. Security techniques

N%6	Security technique
0	Arnold transform
1	DCT scrambling
2	DFT scrambling
3	Cipher block chaining (CBC) encryption
4	CBC encryption + Arnold transform
5	CBC encryption + DCT scrambling

After applying the chosen security technique on the single channel image frames, they are combined to form the RGB image frame. This frame is transmitted to the client system using socket program through LAN cable.

At the receiving end the same process is repeated to reproduce the original image frame. The received secure video as well as the reproduced original video is displayed in real time using Open CV functions.

3. Results and Discussions

The techniques mentioned above were tested using both real time and stored videos. The processing time for Multi-core CPUs and GPUs were noted down. The same techniques were employed by varying the resolution for both the real time and stored videos. At the server and client side 2 Intel Xeon processors each with 4 cores (Total of 8 cores) were used for multicore programming. The clock speed of the processor is 2.4 GHz. For GPU programming, at the server and client side TESLA C2050 GPUs with 448 cores were used. The clock speed of the GPU processor used is 1.15 GHz. The double precision floating point performance of TESLA C2050 is 515 Gflops. Its single precision floating performance is 1.03 Tflops.

3.1. Sample Image Frames of Processed Video

The following figures show the sample image frames obtained when various security techniques were applied on the video.

Figure 2 and Figure 4 is seen as three different portions. This is because of the memory coalescing technique applied in Arnold transform scrambling and Fourier transform based scrambling. Memory coalescing improves the processing speed by minimizing the time spent on memory accesses [7]. Even though the secured video is seen as three portions, the scrambling achieved is good. In Figure 3 and Figure 5 some shades of the original image is seen. But they do not reveal any information about the original image. When the cipher block chaining (CBC) encryption was combined with the cosine transform based scrambling, a completely unrecognizable image was obtained. This is seen in Figure 7. The combination of CBC encryption and Arnold transform scrambling also gave a good secured video. Figure 6 shows the image frame obtained on applying this technique.

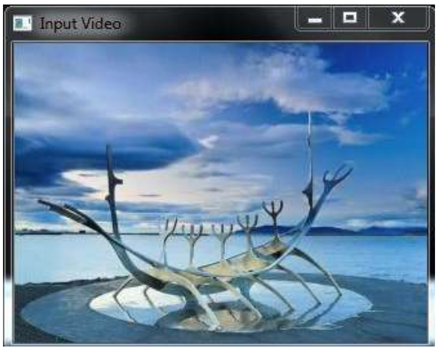

Figure 1. Image frame of input video

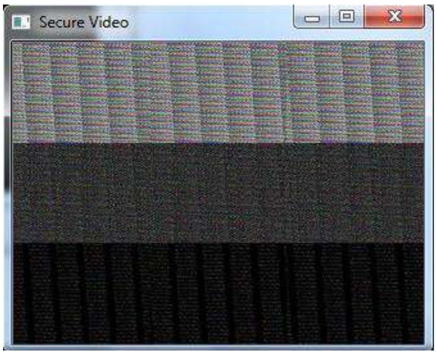

Figure 2. Image frame of the scrambled video using Arnold Transform

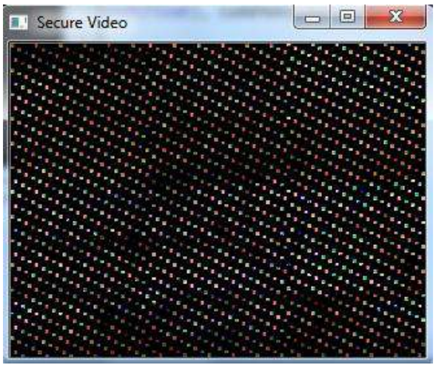

Figure 3. Image frame of the scrambled video using Discrete Cosine Transform

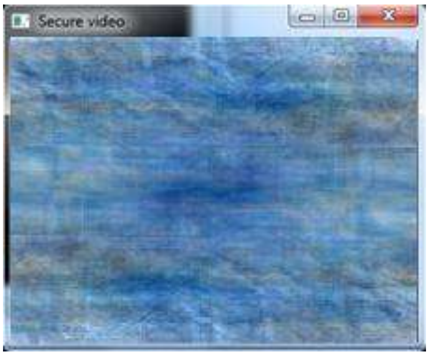

Figure 4. Image frame of the scrambled video using Discrete Fourier Transform

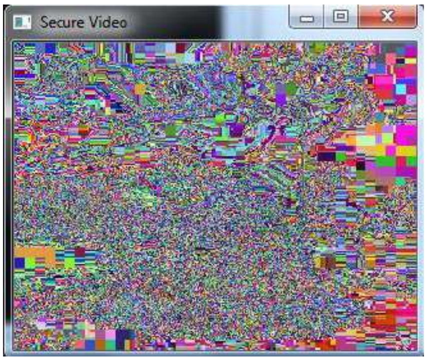

Figure 5. Image frame of the encrypted video using Cipher Block Chaining encryption

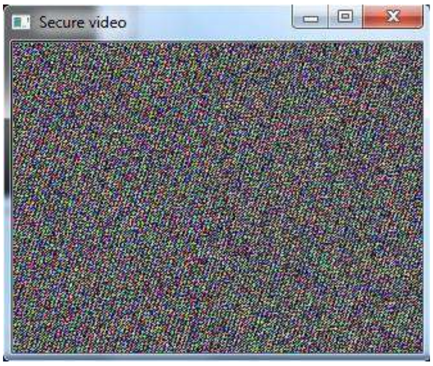

Figure 6. Image frame of the secured video using Cipher Block Chaining encryption and Arnold Transform scrambling

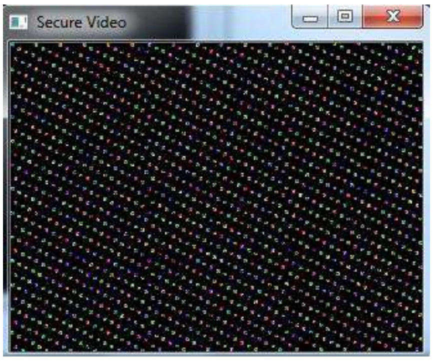

Figure 7. Image frame of the secured video using Cipher Block Chaining encryption and Discrete Cosine Transform scrambling

3.2. Comparison of Security Techniques

The frames are secured basically using three scrambling techniques and an encryption technique. In the three scrambling techniques employed, one is in time domain and the other two is in frequency domain. Arnold transform scrambling, the spatial domain technique, is not as strong as the frequency domain techniques employed since there is no domain shifting. Arnold transform scrambling is also a very common digital image scrambling technique. But the processing time for implementing this technique is low compared to the other techniques.

DCT scrambling and DFT scrambling are the frequency domain scrambling techniques used. Both techniques involve a shifting from time domain to frequency domain. But in DFT scrambling the image is converted into a two channel complex frame which again is split into magnitude and phase matrix. The scrambling is implemented by permuting the phase matrix. In DCT scrambling the scrambling is applied directly to the matrix obtain after converting to the frequency domain. So DFT scrambling is the secure technique comparing the two. But the processing time for this technique is more compared to the other.

Scrambling involves just swapping of the pixels in a frame while encryption modifies the pixels in the frame. So the Cipher block chaining (CBC) encryption technique is far more secure than the scrambling techniques employed, but at the cost of processing time. In CBC encryption each pixel level is replaced with a value that is calculated using a complex algorithm. Hence the processing time is much more compared to the scrambling techniques.

The last two techniques is a combination of the scrambling techniques mentioned above. These two techniques are the most secure techniques employed in this paper. CBC encryption plus Arnold transform scrambling involves an encryption and scrambling in time domain whereas CBC encryption plus DCT scrambling involves an encryption in time domain and then a scrambling in frequency domain.

3.3. Video Transmission

The secured video is transmitted to another system in the network and the original video is reproduced in the client system using single core CPU, multi core CPUs and GPUs. The number of frames considered for experimentation was 100. The time delay added between consecutive frames have small effect on the time taken by the systems listed in Table 2. Figure 8 is a chart plotted using the time taken by the systems listed in Table 2 in their respective order.

Table 2. Time taken for secure video transmission

	Time taken by CPU (1 thread)	Time taken by CPUs (16 thread)	Time taken by GPUs
Webcam video	49526 ms	15029 ms	4542 ms
Stored video	49156 ms	11762 ms	4384 ms

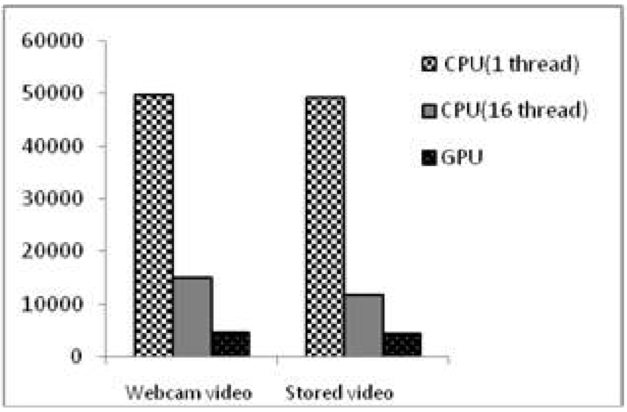

Figure 8. Performance comparison of the systems

GPU showed great speedup over single core CPU when100 frames of secured video of resolution 320 X 240 were transmitted. 23 frames of secured video were transmitted per second using GPUs whereas only 2 frames were transmitted per second using single core CPU. Multicore CPUs were able to transmit 8 frames of secured video per second. It was also found that processing a video taken from a camera takes more time than processing a stored video. This was due to the delay caused by the webcam to capture the frames.

3.4. Processing Time for Video with Different Resolution

In this case the security techniques were applied in the similar manner, but the resolution of the video was varied each time. This experiment was carried out in order to analyse the performance variation of the systems when the computation encountered by them was increased. Here only the time taken for applying the security techniques was noted. The number of frames considered was 100. Table 3 lists the processing time taken by each system for different resolutions. Figure 9 is a graph plotted using data in Table 3 to illustrate the performance variation when the computation was increased.

Table 3. Processing time for different resolution

	Time taken by CPU (1 thread)	Time taken by CPUs (16 thread)	Time taken by GPUs
160 x 120	9687 ms	3449 ms	2028 ms
240 x 180	20717 ms	6303 ms	2028 ms
320 x 240	39986 ms	10265 ms	2044 ms
480 x 360	86751 ms	20280 ms	3307 ms
640 x 480	172888 ms	34679 ms	4508 ms

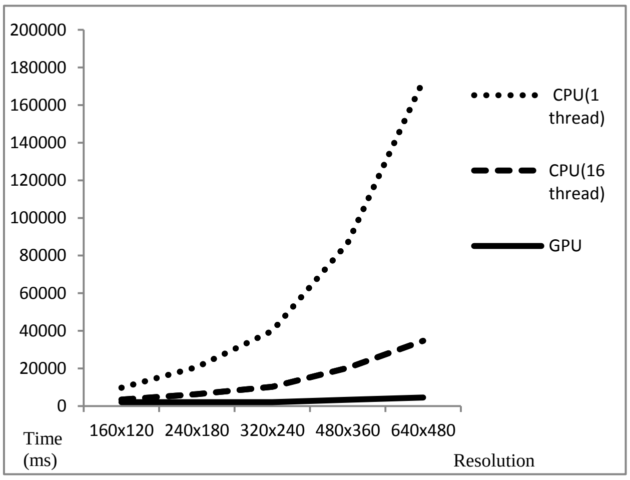

Figure 9. Processing time vs Resolution

It was observed that the processing time taken by GPU remains constant for processing videos up to 320x240 resolution. Above this resolution, as the amount of computation was increased, the processing times taken by GPUs tend to increase. The time listed in Table 3 was also affected by the delay given between each frames when they were displayed as a video. So Table 3 and Figure 9 are more suitable for a comparative study between the systems.

The performance improvement of GPUs over CPU can be seen from the figure 9. As the computation was increased the processing time taken by single core CPU had increased exponentially. The processing time of multicore CPUs also increased. But compared to single core CPU the processing time of multicore CPUs increased at a slower rate. If the processing time taken for the 640x480 video is analysed the performance of a GPUs is about 38.3 times better than the single core CPU and about 7.7 times better than the multicore CPUs.

Table 4. Mean and standard deviation of the processing time for different resolution

Resolution	Mean	Standard deviation
160 x 120	5055 ms	4074 ms
240 x 180	9683 ms	9792 ms
320 x 240	17432 ms	19960 ms
480 x 360	36779 ms	44101 ms
640 x 480	70692 ms	89781 ms

Table 4 gives the mean and standard deviation of the processing time taken by the systems to process frames of different resolution. It can be seen that as the resolution of the frames was increased the standard deviation had increased. This is due to the increasing difference in the processing time between the three systems with the resolution. This variation in the standard deviation values gives a good idea about the performance of GPU over the other systems, considering the amount of computation.

Amdahl’s law was used to calculate the level of parallelism achieved in the multicore program [20]. For video with 640x480 resolution, about 91% of the code was found to be parallelised. This amount was calculated, considering the upper limit of speed up given by Amdahl’s law. Since the amount of parallelism available in the code is above 90%, effective scalability is possible [20]. That means better speedup can be achieved by increasing the number of cores.

4. Conclusions

This paper analysed the processing capability of a multicore CPUs and GPUs to facilitate secure video transmission. Six different techniques in spatial and frequency domain were implemented using these systems. The secured video was transmitted over LAN to the neighbouring system and a real time reproduction was achieved.

The analysis proved that it is also not suitable to apply encryption on video using single core systems. The processing time was very high and it caused a lot of lag when video is reproduced. But with the processing capability of GPUs and multicore CPUs, encryption technique was incorporated along with some scrambling techniques to secure real time videos. The whole process was parallelised using Tesla C2050 GPUs and Intel Xeon (8 core) CPUs.

Applying security on real time videos is very difficult [21]. Even applying a simple scrambling technique on a real time video using single core system caused a lot of lag. With the help of GPUs and multicore CPUs, security was applied on real time videos. Parallelism using GPUs caused no lag whereas using multicore CPUs caused small lag which was very less compared to single core CPU. GPUs were able to transmit 23 frames of secured video per second while multicore CPUs transmitted 8 frames per second.

An increase in speed by about 11 times was achieved using GPUs to that of single core CPU when security is applied on real time video. Speed improvement of GPU over 8 cores CPUs was around 3.3 times in the same case. The performance showed some difference when considering a stored video. Speed improvement of GPUs over single core CPU remained the same but only 2.7 times improvement was observed for stored video processing using GPUs over multicore CPUs.

GPUs also showed better performance as the amount of computation was increased. Performance improvement was only 4.8 times over single core CPU when processing 160x 120 image frames. But it became 38 times when the resolution was changed to 640x480. So it can be concluded that it is always better to process high resolution videos using GPUs.

ACKNOWLEDGEMENTS

The authors, in particular Jerin Geogy George and Nithin P V would like to acknowledge the TIFAC-CORE in Automotive infotronics at VIT University, Vellore, India, for providing the necessary hardware, software and technical support in successfully implementing the present work.

REFERENCES

[1] Wenjun Lu, Avinash Varna, and Min Wu, 2011, Secure video processing: problems and challenges, Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference, 5856-5859.

[2] Jie Shen, 2009, Privacy-protection in real-time video communication, Embedded Software and Systems, ICESS '09. International Conference, 217-220.

[3] J. Reinders, 2007, Intel threading building blocks, O’Reilly, Media.

[4] Song Jun Park, 2009, An analysis of GPU parallel computing, DoD High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC), 365-369.

[5] George R Desrochers, 1987, Principles of parallel and multi

processing, Intertext Publications, McGraw-Hill.

[6] NVIDIA, 2012, CUDA C programming guide PG-02829-001_v5.0, http://docs.nvidia.com/cuda/cuda-c-programming-guide.

[7] Thomas True, 2012, Best practices in GPU based video processing, GPU technology conference, San Jose California.

[8] Pavel Karas, 2010, GPU acceleration of image processing algorithms, Centre for Biomedical image analysis.

[9] OpenMP.org, 2006, The OpenMP API specification for parallel programming, http://openmp.org.

[10] Ying Liu and Fuxiang Gao, 2010, Parallel implementations of image Processing algorithms on multi-core, Fourth International Conference on Genetic and Evolutionary Computing, pp.71-74.

[11] Fei Chen, Kwok-wo Wong, Xiaofeng Liao and Tao Xiang, 1960, Period distribution of the generalized discrete Arnold cat map for $N=2^e$, Information Theory, IEEE Transactions, Volume 59, Issue 5, 3249-3255.

[12] Mao-Yu Huang et al. 2010, Image encryption algorithm based on chaotic maps, Computer Symposium (ICS), International, Tainan.

[13] Anton Obukhov and Alexander Kharmalov, 2008, Discrete cosine transform for 8x8 blocks with CUDA, NVIDIA.

[14] K. Ganesan, G. Harisha Reddy, Sindhura Tokala and Raghava Monica Desur, 2013, Chaos based video security using Multicore framework, American Journal of Computer Architecture, 2(1): 1-7.

[15] Wenjun Zeng and Shawmin Lei, 2003, Efficient frequency domain selective scrambling of digital video. IEEE Transactions on Multimedia, Volume 5, issue 1, 118-129.

[16] L. Deng, Yu C L., Chakrabarti C. and Kim J, 2008, Efficient image reconstruction using partial 2D Fourier transform, Signal Processing Systems, SiPS, IEEE Workshop, 49-54.

[17] Arthur Gordon Mason, 1991, Video scrambling in frequency domain, Patent EP0406017 A1, Reese, L. C., and Welch, R. C, Lateral loading of deep foundations in stiff clay., J. Geotech. Engrg. Div., 101(7), 633–649.

[18] K. Ganesan and S. Ganesh Babu, Chaos based image encryption using 8D Cat Map (unpublished, private communication).

[19] Ming Xue and Changjun Zhu, 2009, The socket programming and software design for communication based on server/client, Circuits, Communications and Systems, PACCS ‘09. Pacific-Asia Conference, 775-777.

[20] Ami Marowka, 2012 “Extending Amdahl’s law for heterogeneous computing”, Parallel and Distributed Processing with Applications (ISPA), IEEE 10th International Symposium, 309-316.

[21] Gabor Feher, 2013, The price of secure mobile video streaming, 27th International Conference on Advanced Information Networking and Applications Workshops, 126-131.