International Journal of Computer Applications (0975 – 8887)
Volume 169 – No.8, July 2017

PFPS: Parallel File Protecting System

Sheetal Adagale
Student
RMD Sinhgad SOE, Pune

Aishwarya Sawant
Student
RMD Sinhgad SOE, Pune

Pratiksha Pandagale
Student
RMD Sinhgad SOE, Pune

Trupti Zanje
Student
RMD Sinhgad SOE, Pune

Kanchan M. Varpe
Guide
RMD Sinhgad SOE, Pune

ABSTRACT

Nowadays everything is digitized. We share information on network, even confidential information also get transferred in the form of files. So it require protection of data against corruption and fault tolerance. This needs secure and efficient way to protect private data from illegle usage. For that purpose we have proposed Parallel File protecting System using CPP (CPU Parallel protecting), CPUP (CPU Parallel Unprotecting), GPP(GPU Parallel Protecting), GPUP(GPU Parallel Unprotecting), HPP(Hybrid Parallel Protecting),

HPUP (Hybrid Parallel Unprotecting) which secures file with the help of SHA3, AES and BLAKE2b algorithm. In proposed system we optimized SHA3-256 and parallel AES algorithm for security purpose with GPU Parallelism and CPU Parallelism for high performance. Parallelism is achieved with the help of NVIDIA's GPU. Along with SHA3 Blake2b is used for fast secure hashing. We have achieved better speed and security with the help of Blake2b. Thus Parallel File Protecting System is secure system and it can be used in computer equipped with NVIDIA's GPU.

General Terms

Security, Algorithms, GPU, CPU, SHA3-256, Paralllel AES, Blake2b etc.

Keywords

Deduplication, integrity checking, protecting, unprotecting etc.

1. INTRODUCTION

Now days all the transactional processes are does online, many calculations does over on network therefore for providing efficient security to all types of data on network is very difficult. To overcome these drawback parallel computing concept is used in proposed system for providing efficient security to data using various security algorithms. This system uses combination of three algorithms such as parallel AES, SHA3-256 and blake2b. In proposed system parallel AES is used for parallel encryption and decryption process, whereas SHA3-256 and blake2b is used for key generation, hash function and achieving better speed in security. The SHA3-256 and blake2b both are cryptographic hash functions, but SHA3-256 is not efficient for fast software hashing for applications such as integrity checking and deduplication in file systems and cloud storage. Therefore blake2b and SHA3-256 algorithms are used together in this system for getting faster speed in system performance. In proposed system for achieving parallelism in encryption and decryption process GPU is used with the help of CUDA platform. In existing system SHA3-256 and parallel AES algorithms are used but they are not enough capable for improving IO performance and integrity checking, therefor this system uses Blake2b for overcoming existing system drawback with high speed performance.

The system uses six algorithms for achieving better security on individual devices as well as together such as (CPU Parallel Protecting), CPUP(CPU Parallel Unprotecting), GPP(GPU Parallel Protecting), GPUP(GPU Parallel Unprotecting), HPP(Hybrid Parallel Protecting) ,HPUP(Hybrid Parallel Unprotecting).

2. SYSTEM WORKING

This section consists of system architecture and workflow of the system. The System architecture is designed as shown in fig. 1.

To implement a secure and efficient file protecting system using parallel AES and SHA3-256 and Blake2b with the help of GPU. In this system users are authenticated on the basis of login details of the user. The authenticated user gets the access to the system and gets input file for further processing.

To secure input file the PFPS system has two main functional parts first is encryption and second is decryption both the phases work in a parallel way by using NVIDIA GPU. In encryption phase given input file is transfer to the main memory and applies all three security algorithms such as parallel AES, SHA3-256, Blake2b. Therefore while applying all these algorithms for encryption on data some bits are padded at the end of data. In padding some bits are added in the data for generating encrypted file. Then padded data is divided upto 128 bit blocks and encrypted. The parallel AES is a block cipher and it uses 128 bit key for encryption. Therefore each block goes through 11 rounds of encryption, with 4 steps such as Subbytes, ShiftRows, MixColumns, AddRoundKey. The rounding process takes much more time for encrypting the file therefore that round takes in parallel way with the help of GPU for reducing the total required time of rounding process. In rounds are takes for each and every block, after performing round on data encrypted file is generated. In the second part of the of the system i.e decryption takes encrypted file as a input and again divides that encrypted file into 128 bits of blocks. Therefore for getting decrypted file it goes through all 11 rounds which are takes in encryption process also. In decryption part exactly inversion process is done.

The system uses SHA3-256 with blake2-b for achieving better speed .Blake2 b algorithm provides more features than SHA3-256 such as 32% less RAM required than BLAKE, parallelism for many-times faster hashing on multicore or SIMD CPUs, Tree hashing for incremental update or verification of large files, personalization for defining a unique hash function for each application, minimal padding, which is faster and simpler to implement.

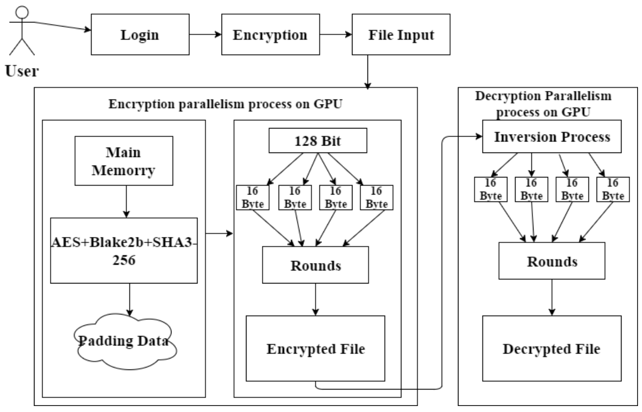

Fig.1 System Architecture.

For more understanding first understands the working of all the three algorithms in the next section.

3. ALGORITHMS

3.1 Parallel AES Algorithm

The parallel AES is cipher cryptographic algorithm. Also meant the substituting permutation network. It contains the several operations such as the given input replaced by the special substitution shuffling the rows and adding bit into it.

In this system parallel AES is achieved parallelism on blocks of data. Therefore all the blocks are encrypted and decrypted in a parallel way. In this algorithm following steps are follows for encryption process such as:

1. Byte Substitution (Sub Bytes)
2. Shift Rows
3. Mix Columns
4. Add Round Key

A. Byte Substitution

Byte substitution by based on the fixed table of input is in bytes and the result also will be the matrix of four columns and rows.

B. Shift Rows

The shifting processing carried out in the matrix by shifting the rows to the left one by one. In the shifting process if any entries are fall of then those entries shifted or reinserted right side of the row. It is important for the encryption of the data given as the input.

- The first row in the matrix is not shifted.
- While the second row shifted to one position from current position to the left.
- Then third row shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- One by one row shifted to the left and any rows fall of, it reinserted to the right.
- New matrix will the same number of bytes 16 but shifted as above with each other.

C. Mix Columns:

Mix the four bytes by using the functions using math. In this it get the four byte of inputs and writes the newly generated bytes in which it replaces the existing data. It will write the other matrix of newly generated data.

D. Add Round Key

The 16 bytes of the matrix are now considered as 128 bits and are XOR-ed to the 128 bits of the round key. If this is the last round then the output is the cipher text. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

E. Decryption process of Parallel AES algorithm

Decryption can be done in a similar way to encryption. First two arrays are defined. When a decryption needs to be performed, one array contains the key and the other one the cipher text.

It is the inversion process of encryption

1. Inversion of Add round key
2. Inversion of Mix columns
3. Inversion of Shift rows
4. Inversion of Byte substitution

The parallel AES algorithm:

```
1.  Cipher(byte in[16], byte out[16], key_array
    round_key[Nr+1])
2.  Begin
3.  byte state[16];
4.  state = in;
5.  AddRoundKey(state, round_key[0]);
6.  for i = 1 to Nr-1 stepsize 1 do
7.  SubBytes(state);
8.  ShiftRows(state);
9.  MixColumns(state);
10. AddRoundKey(state, round_key[i]);
11. end for
12. SubBytes(state);
13. ShiftRows(state);
14. AddRoundKey(state, round_key[Nr]);
15. end.
```

3.2 SHA3-256 Algorithm

The file contents hashed by SHA3 and the digest the message in it for the security purpose. To know the any modification into the files contents some data is digested into the file. The Message can be evenly partitioned into *n*-bit chunks, for this step padding is required. It uses the pattern 10 1: a1 bit, zero or more 0 bits and a last 1 bit. The last 1 bit is compulsory for the safety resistant of different data contents, that is, diverse hash functions are padded through the block. Without it, various hash functions of the data will be the same up to truncation to compute a hash, reset the state to 0, pad the data given as the input, and break it into pieces into n-bits the encrypted key depends on the SHA3-256 algorithm.

The SHA3-256:

```
1.  SHA3:=proc(message::string, messagetype::name:=text)
2.  Local n,m,l;
3.  If type(procname,'indexed')then
4.  n:=op(procname)
5.  else
6.  error "output length not specified"
7.  end if;
8.  if not n in{224,256,384,512}then
9.  error "%1 is not a valid output length",n
10. end if;
11. m:=messagebytes(message,messagetype);
12. l:=keccak(m,1600-2.n,n,hash);
13. bytestohexstring(l)
14. end proc;
```

3.3 Blake2b Algorithm

The Blake2b algorithm is used for the secure hashing. 224, 256, 384, and 512 bits size message digested into the file. The Maximum message length of hash function is at least 64 − 1bits. In addition, its compulsory to use Blake2b to explicitly handle hashing with is parallelizable to improve the performance trade-offs be suitable for insubstantial surroundings. The Blake2b pads the last data block if and if only it is required, with null bytes. If the data length is divided into blocks of size and in multiple of blocks, no extra padded bytes is added to it then it operates on 64-bit words and returns a 64-byte hash value. Blake2b does 12 rounds, against 16 and 14 respectively for Blake2b. Based on the security analysis performed so far, and on reasonable assumptions on future progress, it is unlikely that 16 and 14 rounds are meaningfully more secure than 12 and 10 rounds.

The Blake2b algorithm :

```
1. Input:
     M
     cbMessageLen: Number, (0..$2^{128}$)
     Key
     cbKeyLen: Number, (0..64)
     cbHashLen: Number, (1..64)
2. Output:
   Hash
    $h_0$ ← 0xcbbb9d5dc1059ed8
    $h_1$ ← 0x629a292a367cd507
    $h_2$ ← 0x9159015a3070dd17
    $h_3$ ← 0x152fecd8f70e5939
    $h_4$ ← 0x67332667ffc00b31
    $h_5$ ← 0x8eb44a8768581511
    $h_6$ ← 0xdb0c2e0d64f98fa7
    $h_7$ ← 0x47b5481dbefa4fa4
3.  $h_0$ ← $h_0$ xor 0x0101kknn
4.  cBytesCompressed ← 0
5.  cBytesRemaining ← cbMessageLen
6.  if (cbKeyLen > 0) then
7.  M ← Pad(Key, 128) || M
8.  cBytesRemaining ← cBytesRemaining + 128
9.  end if
10. while (cbBytesRemaining > 128) do
11. chunk ← get next 128 bytes of message **M**
12. cBytesCompressed ← cBytesCompressed + 128
13. cBytesRemaining ← cBytesRemaining - 128
14. Compress(h, chunk, cBytesCompressed, false)
15. end while
16. hunk ← get next 128 bytes of message **M**
17. cBytesCompressed←
    cBytesCompressed+cBytesRemaining
18. chunk ← Pad(chunk, 128)
19. Compress(h, chunk, cBytesCompressed, true)
20. Result ← first cbHashLen bytes of little endian state
    vector h
21. End Algorithm Blake2b
```

Blake2b is optimized for 64 bit platforms. Like SHA-2, Blake comes in two variants: one that uses 32-bit words, used for computing hashes up to 256 bits long, and one that uses 64-bit words, used for computing hashes up to 512 bits long. The core block transformation combines 16 words of input with 16 working variables, but only 8 words (256 or 512 bits) are preserved between blocks.

4. MATHEMATICAL MODEL

A mathematical model is a description of a system using mathematical concepts and language. A model may help to explain a system and to study the effects of different components, and to make predictions about behavior.

In this mathematical model gives the idea abot flow of the system. It shows that system uses six algorithms such as CPU, CPUP, GPU, CPUP, HPP, HPUP and its mathematical representation.

4.1 Mathematical Model:

1. S={f,E,D,GPU,CPU,HPP,CPP,CPUP,GPP,GPUP,O ,HPP,HPUP}
2. S = System
3. f = File
4. E = Encryption
5. D= Decryption
6. F(AES) = $\int_0^{\infty}$ (E) + (D)……(1) //AES Encryption and Decryption process
7. GPP=f + 16+(16 - f %16) + hashByteLen
8. // GPU file protecting
9. GPUP=cipherLen=cipherLen-hashByteLen-16-cipher[cipherLen-1] ….//GPU file Unprotecting
10. $GPU = \sum_{i=0}^{n}(GPP + GPUP)$ ..//GPU parallel file Encryption and Decryption
11. Where,
12. GPP=GPU Parallel Protecting
13. GPUP=GPU Parallel Unprotecting
14. CPP= f +1 6 + (16-f%16)+hashByteLen….. //CPU parallel file protecting
    CPUP=Hash(hashByteLen*8,cipher,(cipherLen-hashbyteLen)*8,hashValue)…..// CPU parallel file unprotecting

$$CPU = \sum_{i=0}^{n}(CPP + CPUP) \qquad …(3)$$

CPU file Encryption and Decryption

Where,

CPP-CPU Parallel Protecting
CPUP-CPU Parallel UnProtecting
HPP=plain+plainLen
hashByteLen,hashvalue,hashByteLen…//Hybrid parallel file protecting

HPUP = GipherLen * 1.0/(ratio+1)/16*16….//Hybrid parallel file unprotecting

Where,

HPP-Hybrid Parallel Protecting
HPUP-Hybrid Parallel Unprotecting

As per eq. 2 and 3

O = Output

$$HPU = \sum_{i=0}^{n} HPP + HPUP \qquad …(4)$$

$$O = \int_0^2 (CPU + GPU) \qquad …(5)$$

This mathematical model gives the mathematical formula's for implementing all the six algorithms in system. In this HPP and HPUP is a combination of CPP-GPP and CPUP-GPUP algorithms.

5. RESULTS

This section presents results accomplish by PFPS System with the help of Graphs also experimental discussion is included to show implemented system testing.In Fig. 2 shows the comparison between existing system and proposed system related to protecting or encryption process. It gives the speed of protecting in existing and proposed system. It compares the performance on the basis of two parameters such as total time which is required for protecting and file size. In this some comparison is if file size is 5.6kb then it takes 0.4ms in existing system and 0.3ms in proposed system, if file size is 20.4kb then it takes 0.5ms for existing system and 0.4ms for proposed system, if file size is 2.21mb then it takes time 0.08ms for existing system and 0.05 for proposed system. Therefore all the comparisons shows that proposed system gives better performance than the existing system.

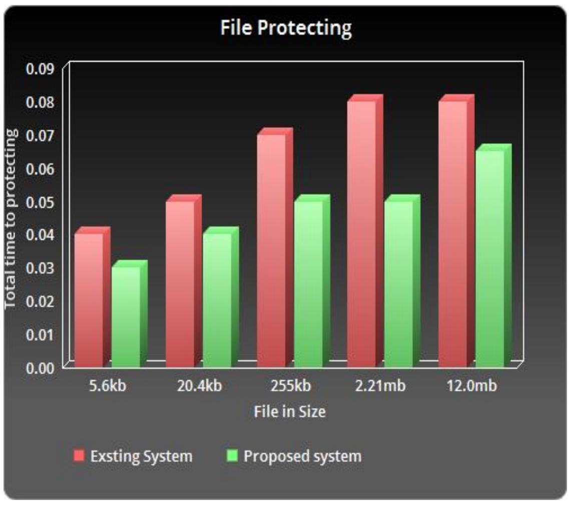

Fig.2 File Protecting

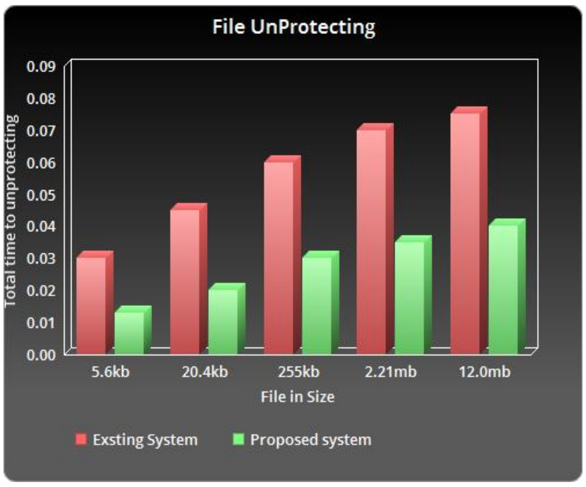

Fig.3 File Unprotecting

In Fig 3 shows the comparison between existing system and proposed system related to unprotecting or decryption process. It gives the speed of unprotecting in existing and proposed system.

It gives the speed of unprotecting in existing and proposed system.It compares the performance on the basis of two parameters such as total time which is required for protecting and file size. In this some comparison is if file size is 5.6kb then it takes 0.3ms in existing system and 0.1ms in proposed system, if file size is 20.4kb then it takes 0.5ms for existing system and 0.2ms for proposed system, if file size is 2.21mb then it takes time 0.08ms for existing system and 0.03 for proposed system. Therefore all the comparisons shows that proposed system gives better performance than the existing system.

- **Experimental Discussion**
  - **HW , SW used** :

  The PFPS system is mainly used for parallel encryption and decryption, therefore for performing this operations on data only CPU is not capable for performing these operations in a parallel way. Therefore this system uses NVIDIA GPU for handling all the parallel operations efficiently.

  The PFPS system uses CUDA platform for implementation with help of C/C++ programming language .It also uses Windows operating system and Visual Studio 2012 .This system uses only applies C/C++ programming language because CUDA platform uses only this language for doing parallel programming.

  - **System implementation platform**

    This system uses CUDA platform for implementation of the system. This system requires only CUDA platform because it uses only for parallel programming. It can be does all the operations in a parallel way and increasing the system performance.

  - System Testing using Snapshots

6. CONCLUSION

In this paper we have implemented PFPS:Parallel File Protecting System to provide security to the user files that are transferred or stored on the network against illegal usage. This system is mainly focused for protecting files for transferring or storing safely. It can secure any type of file such as audio, video, RAR etc.

For security purpose SHA3-256, parallel AES algorithm is applied with the help of CUDA platform. For improving IO management Blake2b algorithm in combination with optimized SHA3-256 is introduced in PFPS.

It increases the speedup performance in encryption and decryption as compared with existing system. This PFPS can be used as an application to protect files in a fastest way by parallelism over the huge storage web portals.

In future, algorithm Blake2b will need to optimize and also PFPS can be extended for cloud storage.

7. REFERENCES

[1] Xiongwei Fei , Kenli Li , Wangdong Yang , Keqin Li,A secure and efficient file protecting system based on SHA3 and parallel AES,Parallel Computing 52(2016) 106-132

[2] .X. Shi, F. Park, L. Wang, J. Xin, Y. Qi, Parallelization of a color-entropy preprocessed chan-vese model for face contour detection on multi-core cpu and gpu, Parallel Comput. 49 (2015) 2849.

[3] K. Li, J. Liu, L. Wan, S. Yin, K. Li, A cost-optimal parallel algorithm for the 01 knapsack problem and its performance on multicore cpu and gpu implementations, Parallel Comput. 43 (2015) 2742.

[4] .K. Li, W. Yang, K. Li, Performance analysis and optimization for SPMV on GPU using probabilistic modeling, IEEE Trans. Parallel Distrib. Syst. 26 (1) (2014) 196205.

[5] A. Pousa, V. Sanz, A. de Giusti, Performance analysis of a symmetric cryptographic algorithm on multicore architectures , in: Computer Science Technology Series-XVII Argentine Congress of Computer Science-Selected Papers, Edulp, 2012, pp. 5766.

[6] Hoang-Vu Dang, Bertil Schmidt ,The Sliced COO Format for Sparse MatrixVector Multiplication on CUDA-enabled GPUs computing,2011 W. D. Doyle, “Magnetization reversal in films with biaxial anisotropy,” in *Proc. 1987 INTERMAG Conf.*, 1987, pp. 2.2-1-2.2-6.

[7] Marcin Krotkiewski , 2011 , perfomance study of our CUDA based implementation of the symmetric 7-point and 27-point ,and the general 27-point stencil(3 3 3 convolution) for modern GPUs,2011

[8] C. Mei, H. Jiang, J. Jenness, CUDA-based AES parallelization with fine-tuned GPU memory utilization, in: 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), IEEE, 2010, pp. 17.

[9] S.A. Manavski, CUDA compatible GPU as an efficient hardware accelerator for AES cryptography, in: IEEE International Conference on Signal Processing and Communications, 2007 (ICSPC07), IEEE, 2007, pp. 6568.

[10] M. Biglari, E. Qasemi, B. Pourmohseni, Maestro: A high performance AES encryption/decryption system, in: 2013 17th CSI International Symposium on Computer Architecture and Digital Systems (CADS), IEEE, 2013, pp. 876-880. Available:

[11] Kenli Li, Wangdong Yang, and Keqin Li, Senior Member, IEEE Performance optimization using partitioned SpMV on GPUs and multicore CPUs,IEEE Transactions on Computers 64(9):2623-2636 September 2015.

[12] A. Mohd, Y. Jararweh, L.A. Tawalbeh, Aes-512: 512-bit advanced encryption standard algorithm design and evaluation, in: IAS, IEEE, 2011, pp. 292297.

[13] B. Liu, B.M. Baas, Parallel aes encryption engines for many-core processor arrays, IEEE Trans. Comput. 62 (3) (2013) 536547.

[14] J. Diaz, C. Munoz-Caro, A. Nino, A survey of parallel programming models and tools in the multi and many-core era, IEEE Trans. Parallel Distrib. Syst.23 (8) (2012) 13691386.

[15] T. Nhat-Phuong, L. Myungho, H. Sugwon, L. Seung-Jae, High throughput parallelization of AES-CTR algorithm, IEICE Trans. Inform. Syst. 96 (8) (2013)16851695.

[16] Q. Dong, J. Zhang, L. Wei, A sha-3 based rfid mutual authentication protocol and its implementation, in: 2013 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC), IEEE, 2013, pp. 15

[17] N. Moreira, A. Astarloa, U. Kretzschmar, Sha-3 based message authentication codes to secure IEEE 1588 synchronization systems, in: 39th Annual Conference of the IEEE on Industrial Electronics Society (IECON13), IEEE, 2013, pp. 23232328.

[18] J.S. Banu, M. Vanitha, J. Vaideeswaran, S. Subha, Loop parallelization and pipelining implementation of AES algorithm using OpenMP and FPGA, in:2013 International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN), IEEE, 2013, pp. 481485

[19] X. Shi, F. Park, L. Wang, J. Xin, Y. Qi, Parallelization of a color-entropy preprocessed chan-vese model for face contour detection on multi-core cpu and gpu, Parallel Comput. 49 (2015) 2849.

[20] C. JunLi, Q. Dinghu, Y. Haifeng, Z. Hao, M. Nie, Email encryption system based on hybrid aes and ecc, in: IET International Communication Conference on Wireless Mobile and Computing (CCWMC11), IET, 2011, pp. 347350

[21] A.Mohd, Y.Jararweh, L.A.Tawalbeh, Aes-512:512-bit advanced encryption standard algorithm design and evaluation,in:IAS,IEEE,2011,pp.292297.

[22] P.Maistri, F.Masson, R.Leveugle, Implementation of the advanced encryptio standard on gpus with the nvidia cuda framework,in:2011 IEEE Symposiumon Industrial Electronics and Applications (ISIEA), IEEE, 2011, pp.213217.

[23] C.-L.Duta, G.Michiu, S.Stoica, L.Gheorghe, Accelerating encryption algorithms using parallelism, in: 2013 19th International Conference on Control Systems and Computer Science (CSCS), IEEE, 2013, pp.549554.