International Journal of Scientific Research & Engineering Trends
Volume 3, Issue 3, May-2017, ISSN (Online): 2395-566X

Encrypted and Unencrypted Computation for Abstract Machine

Thripthi.P.Balakrishnan
PG Scholar
Shree Venkateshwara Hi-Tech Engg College, Gobi
Email: thripthi.p.b@gmail.com

Mr. S.Vijayanand, ME
Assistant Professor
Shree Venkateshwara Hi-Tech Engg College, Gobi
Email: anand.vijay87@gmail.com

Dr. T. Senthil Prakash
Professor & HOD
Shree Venkateshwara Hi-Tech Engg College, Gobi
Email: jtyesp@yahoo.co.in

***Abstract* – Security is a major issue nowadays. Many cryptographic algorithms are used for maintaining data security. For performance as well as for physical security reasons, it is often advantageous to realize cryptographic algorithms in hardware. This proposed system based on heterogeneous abstract machine for encrypted and unencrypted compute program. In this abstract machine encryption and decryption of data is done in same memory location. The following situations are considered: several users upload data encrypted with a public-key FHE, a server carries out computations on the encrypted data and then sends them to an agency who has a decryption key for the FHE. In this situation, one approach to reduce the storage requirement is to use AES encryption to encrypt data, and then perform homomorphism computations on cipher texts after converting to FHE-cipher texts.**

***Keywords* – Abstract Machine, Encrypted Computation, One Instruction Set Computer, Heterogeneous Computer, Homomorphic Encryption.**

I. INTRODUCTION

There are many computing paradigms such as cloud computing have become increasingly popular as they allow outsourcing computation to a typically more powerful or dedicated set of machines. concern with such paradigms, however, is the privacy of the outsourced data. While outsourcing several attacks may happen. cryptographic primitives such as homomorphic encryption can be leveraged to address those privacy concerns, and eventually return control of the data back to the legitimate information owner. As soon as fully homomorphic encryption (FHE) became theoretically possible, the academic interest in FHE applications has increased accordingly. In addition, partial homomorphic encryption (PHE) has recently been leveraged for verifiable computation. PHE schemes are more practical than their FHE. Indeed, PHE schemes typically require straightforward operations on ciphertexts, such as modular multiplication, which can be implemented very efficiently.

The proposed abstract machine defines a universal computer for processing encrypted and unencrypted data together within the same program memory space. Data encryptions are generated using Paillier PHE, and only homomorphic addition is natively supported. Universal computation, however, requires support for both addition and multiplication notably; we devised a PHE operating model with a single instruction that supports both encryption and decryption within itself; this effectively enables unifying re-encryption with the executing program.

II. EXISTING SYSTEM

Ascend proposed; a block cipher accelerator decrypts all oblivious RAM read operations before instructions and data move into the processor caches, while data evicted from these caches are re-encrypted before oblivious memory storage. The processor contains a shared block cipher key and the chip itself is considered tamper-proof.

Aegis propose a secure processor, which supports integrity attestation of executed software similar to Trusted Platform Modules (tamper-evident execution), and provides privacy protection for off-chip memory using non-malleable symmetric encryption (tamper-resistant execution). The Aegis chip contains a permanent private key that is necessary to decrypt other symmetric keys used to protect data and instructions within program binaries to provide privacy and integrity protection from potentially malicious processes running on the same system with elevated privileges. In this case, a memory encryption engine protects all traffic between the processor and the system main memory. In the same direction, the hardware enforced isolation discussed in provides integrity protection even in case the operating system kernel is under attack, using a cryptographic coprocessor provisioned with secret keys during fabrication

Drawback of existing system

- Privacy protection only off-chip memory space.
- Re-encrypted before memory storage make problem.
- Malicious processes run on same system.

III. PROPOSED SYSTEM

Cryptoleq's, a new programming language based on a single instruction computer architecture, which processes homomorphism data natively. Cryptoleq's defines a universal computer for processing encrypted and unencrypted data together within the same program memory space. Data encryptions are generated using Paillier PHE, and only homomorphism addition is natively supported. Universal computation, however, requires support for both addition and multiplication, and in this work, Simulate multiplication with heuristically obfuscated re-encryption implemented using Cryptoleq's instructions. Cryptoleq's supports programs written without privacy protections, as well as protected execution using encrypted data under full encryption or heuristic obfuscation modes, depending on the need to multiply encrypted values.

1. A practical framework:

- With extended assembly language, compiler, and emulator for executing Cryptoleq's programs on different platforms.
- Cryptoleq's is heterogeneous, allowing mixing encrypted and unencrypted instruction operands in the same program memory space.
- Programming with Cryptoleq's is facilitated using an enhanced assembly language that allows the development of any advanced algorithm on encrypted data sets.

2. Advantages of proposed system

- Simple to use.
- Lower computation cost.
- Execution of Cryptoleq's program on different platforms.
- Easy to allocate memory space.

3. Overall operation

Data to be send is first send to abstract machine. The abstract machine performs the computation (here multiplication) in memory segments in sectors**.** The computed data is then encrypted and received by sender. Sender sends this encrypted data to receiver who decrypts it using a decryption key.

Likewise receiver can send data back to sender. Receiver also has to send data first to abstract machine. Abstract machine will perform the same operations and send encrypted data back to receiver.

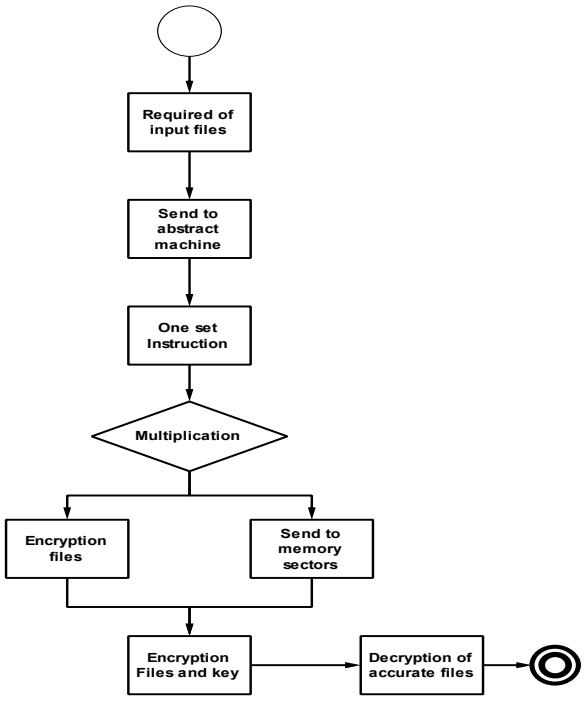

Fig.1. Overall operations

4. Memory organization

The memory is organized as a collection of sectors (fig. 2). Each sector is a collection of continuous segments (i.e. sequences of memory cells) and all cell addresses within one sector share the same *s* value, while all cell addresses within one segment have sequential *t* values. Incrementing a *t* value by a unit returns the next cell address.

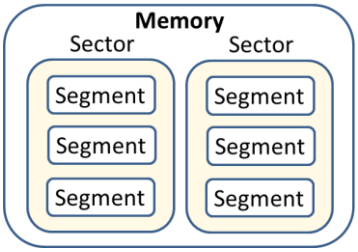

Fig.2. Memory organization

IV. SYSTEM DESCRIPTION

1. System Architecture

The abstract machine is a processor model operating on a sequence of memory cells. Each memory cell has an address and a value, while any cell value may also be used as a memory address. The memory is organized as a collection of sectors. Each sector is a collection of continuous segments (i.e. sequences of memory cells).

The abstract machine performs the computation (here multiplication) in memory segments in sectors**.** The computed data is then encrypted and received by sender. Sender sends this encrypted data to receiver who decrypts it using a decryption key.

International Journal of Scientific Research & Engineering Trends
Volume 3, Issue 3, May-2017, ISSN (Online): 2395-566X

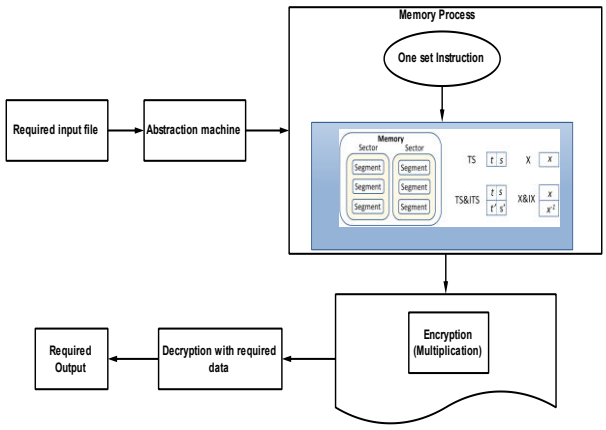

Fig.2. System Architecture

2. Modules

- Design of abstraction machine
- Design of memory using segment.
- Implement one set instruction of computer.
- Decryption of Required Process.

2.1. Design of abstract machine

Abstract machines that model software are usually thought of as having very high-level operations. For example, an abstract machine that models a banking system can have operations like "deposit," "withdraw," "transfer," etc. set computer, capable of performing general-purpose computation on encrypted programs.

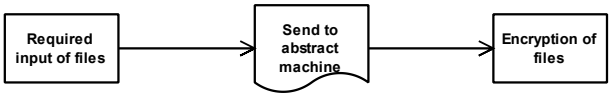

Fig.3. Design of abstract machine

2.2. Design of memory using segment

The design of memory based on segment of sector. In this approach, the memory is organized as a collection of sectors. Each sector is a collection of continuous segments (i.e. sequences of memory cells) and all cell addresses within one sector share the same s value, while all cell addresses within one segment have sequential t values. Incrementing a t value by a unit returns the next cell address.

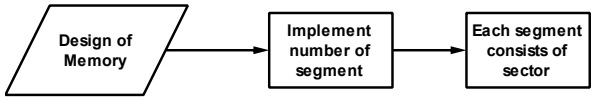

Fig 4: Design of memory

2.3. Implement one instruction set computer

One instruction set computer (OISC) is a computer architecture which supports only one instruction and is able to perform universal computation. It operates on memory organized as a sequence of memory cells, while processor instructions and data reside in unified memory space, following the von- Neumann model. There exist three OISC categories (i) Transport Triggered Architecture Machines, (ii) Bit Manipulation Machines, and (iii) Arithmetic Based Machines.

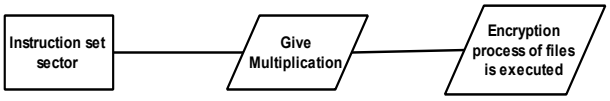

Fig.5. Implement one instruction set computer

2.4. Decryption of Required Process

In this module the encrypted data from abstract machine received is then decrypted to retrieve the correct data.

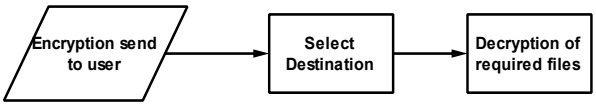

Fig.6. Final process

V. Related Works

1. Efficient Architectures for implementing Montgomery Modular Multiplication and RSA Modular Exponentiation on Reconfigurable Logic

The fundamental operation of the algorithm is modular exponentiation which is achieved by repeated modular multiplications. The Montgomery modular multiplication algorithm is often used to perform these calculations [6]. However, the high bit lengths required to provide adequate security (1024 bits is considered secure against attack in the near future), mean a high hardware throughput is difficult to achieve.

2. Improved Delegation of Computation Using Somewhat Homomorphism Encryption to Reduce Storage Space

This scheme is far from being practical because of its large computational cost and large cipher texts. Since then, considerable efforts have been made to devise more efficient schemes. However, most FHE schemes still have very large cipher texts. One approach to reduce the storage requirement is to use AES encryption to encrypt data, and then perform homomorphism computations on cipher texts after converting to FHE-cipher texts.

3. De Trust: Defeating Hardware Trust Verification with Stealthy Implicitly-Triggered Hardware Trojans

Hardware Trojans (HTs) inserted at design time by malicious insiders on the design team or third-party intellectual property (IP) providers pose a serious threat to the security of computing systems. Researchers have proposed several hardware trust verification techniques to mitigate such threats, and some of them are shown to be able to effectively flag all suspicious HTs implemented in the Trust-Hub hardware backdoor benchmark suite[3]. No doubt to say, adversaries would adjust their tactics of attacks accordingly and it is hence essential to examine whether new types of HTs can be designed to defeat these hardware trust verification techniques.

VII. CONCLUSION

In this paper, we have presented a new computational model based on the concept of single instruction architecture, able to execute programs whose instruction operands have been encrypted using Paillier PHE scheme. Universal computation is achieved by introducing a software function, which adds multiplication to the abstract machine's native addition and subtraction operations. This function is expressed using the only available instruction. We have also developed an enhanced assembly language to facilitate the development of complex programs, in addition to a compiler and an emulator.

This model allows for several future improvements with regards to performance and security. The former can be improved through the introduction of high-radix representations (e.g. Montgomery), and advanced runtime techniques (such as automatic detection of open values to replace homomorphic multiplication with plaintext addition).

ACKNOWLEDGMENT

We offer our humble thanks at the sacred feet of the Almighty and also take great pleasure in expressing my profound gratitude to all those who have helped for this work.

REFERENCES

[1] Thripthi. P. Balakrishnan, Mr. S. Vijayanand, ME and Dr.T.Senthil Prakash, "Efficiently Verifiable Computation On Encrypted Data" in International Journal of Science and Engineering Research, 2016, Vol 4 Issue 12 December-2016 3221 5687, (P) 3221 568X.

[2] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM side channels and their use to extract private keys," in Proc. ACM Conf. Comput. Commun. Secur. (CCS), 2012, pp. 305–316.

[3] N. G. Tsoutsos, C. Konstantinou, and M. Maniatakos, "Advanced techniques for designing stealthy hardware trojans," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–4.

[4] K.-M. Chung, Y. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Advances in Cryptology*. Heidelberg, Germany: Springer, 2010, pp. 483–501.

[5] A. Daly and W. Marnane, "Efficient architectures for implementing montgomery modular multiplication and RSA modular exponentiation on reconfigurable logic," in *Proc. ACM/SIGDA Int. Symp. Field- Program. Gate Arrays*, 2002, pp. 40–49.

[6] T. Blum and C. Paar, "Montgomery modular exponentiation on reconfigurable hardware," in *Proc. 14th IEEE Symp. Comput. Arithmetic*, Apr. 1999, pp. 70–77.

[7] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc.ACM Symp. Theory Comput.*, 2009, pp. 169–178.

AUTHOR PROFILE

Thripthi P. Balakrishnan
is a PG scholar at Shree Venkateshwara Hi-Tech Engineering College, Gobi, Tamilnadu. She received her B.Tech. degree in Computer Science and Engineering from MES College of Engineering, Kuttippuram, Kerala. She has worked as lecturer at KMCT College of Engineering, Kerala.