On Multi-user based Efficient Computation Outsourcing Scheme and its Application to Cloud

V. Sudarsan Rao¹ and N. Satyanarayana²

(Corresponding author: V. Sudarsan Rao)

Department of CSE, Khammam Institute of Technology and Sciences (KITS)¹
Telangana, India

Department of CSE, Nagole Institute of Technology and Sciences (NITS)²
Telangana, India

(Email: sudharshan.cse2008@gmail.com)

(Received Oct. 9, 2017; revised and accepted Apr. 13, 2018)

Abstract

The outsourcing process is computationally secure if it is performed without unveiling to the other external agent or cloud, either the original data or the actual solution to the computations. Secure multiparty computation computes a certain function without revealing their private secret information. In this paper, a new kind of outsourcing computing protocol is proposed which utilizes multi cloud servers view framework. This paper mainly adopts the Fully Homomorphic Encryption technique (FHE). In our proposed protocol, encrypted data by different users is transformed to cloud. The protocol being non-interactive between users, gives the comparatively lesser computational and communication complexity. The analysis of our proposed protocol is also presented at the end of the paper.

Keywords: Access Control; Circuit Computation; Cloud Computing; Privacy; Secret Information Parameters; Secure Outsourcing

1 Introduction

Beside the tremendous advantages of outsourcing, client faces some challenges by outsourcing the computational task to cloud [2, 3]. These are security, input-output privacy and verification of result. Consider a scenario where some mutually distrusted members are present, and they want to compute a complex function, which involves their own private inputs [6]. This scenario may be termed as secure multi-party computation. Suppose, $U_1, U_2, \ldots, U_m$ are $m$ users, and each posses a private number $n_1, n_2, \ldots, n_m$. Consider function is,

$$\text{FUNC} = f(n_1, n_2, \ldots, n_m),$$

which they want to co-operatively compute, but they don’t want to expose $n_i$ of corresponding $U_i$ to other users $U_j$, $i \neq j$ & $i, j \in (1, 2, \ldots, m)$. Also they should guarantee that FUNC should not be known by any of the unauthorized user. Its observable that the computation and communication complexities are mostly dependant on the complex nature of computation function. The scenario is shown as Figure 1.

![Figure 1: General computational outsourcing scenario](image)

Recently, as the development of cloud computing [33], users’ concerns about data security are the main obstacles that impedes cloud computing from wide adoption. These concerns are originated from the fact that sensitive data resides in public cloud [31], which is maintained and operated by untrusted cloud service provider (CSP) [21, 29]. The expectation of users is that the cloud should compute the function having the inputs as private parameters of users in the encrypted/transformed form.

Remaining paper organized as - Section 2 provides a general nomenclature for various secure outsourcing algorithms. Significant state-of-the-art protocols along with the motivation towards the problem and our contribution in this paper is summarized. Preliminaries are given in Section 3. Secure outsourcing using FHE scheme is given in Section 4. Experimental analysis are presented in Section 5. Section 6 presents our proposed scheme along
with correctness, security analysis and our experimental simulation results. Section 7 concludes the paper.

2 Secure Outsourcing Algorithms

2.1 Related Work

While outsourcing the private data functions to the cloud, there exist many problems and challenges. In past years, much research have been carried out to come up with various solutions for secure computational outsourcing. One solution was proposed by Gentry [16], in 2009 where a joint public key is used to encrypt their private input data and accordingly the notion was termed as Homomorphic encryption, which successively used in the secure outsourcing of practical complex problems. In the work by [1], authors have given a scheme where for encryption purpose, users’ public keys are utilized, and cloud will be able to compute the function having their private inputs. A more secure outsourcing was given by Halevi et al. [18] in 2011, that was a non-interactive method for secure outsourcing [8]. [23] given a new fully homomorphic scheme, multikey FHE, which applied bootstrapping concept for secure outsourcing of computations. ABE, introduced as fuzzy identity-based encryption in [25], was firstly dealt with by Goyal et al. [15]. Two distinct and interrelated notions of ABE were determined in [15]. Accordingly, several constructions supporting for any kinds of access structures were provided [13, 24] for practical applications [19, 32]. Atallah et al. [2] offered an structure for secure outsourcing of scientific computations e.g. multiplication of matrices. Although, the solution used the disguise technique and thus led to leakage of private information. Atallah and Li [3] given an efficient protocol to outsource sequence comparison with two servers in secure manner. Furthermore, Benjamin and Atallah [6] addressed the problem of secure outsourcing for widely applicable linear algebraic computations. Atallah and Frikken [4] further studied this problem and gave improved protocols based on the so-called weak secret hiding assumption. Recently, Wang et al. [28] presented efficient mechanisms for secure outsourcing of linear programming computation.

In [17], a novel paradigm for outsourcing the decryption of ABE is given. Compared with our work, the two lack of the consideration on the eliminating the overhead computation at attribute authority. In 2014, V.Sudarshan et al. [27] proposed an Attribute-Based Encryption mechanism, applied for cloud security. Recently Lai et al. [20] given a construction with verifiable decryption, which achieves both security and verifiability without random oracles. Their task supplements a redundancy with ciphertext and uses this redundancy for correctness checking. A.K.Chattopadhyay et al. [12] proposed the scheme which uses simple Boolean based encryption and decryption of the data files, which is low in computational cost.

Z. Cao et al. [11] remarked that Yu et al.’s scheme [30] has two inherent weaknesses:

1) It does not truly mitigate the client’s computational burden for key updates.  
2) It does not ensure confidentiality since the files uploaded to the cloud by the client are eventually not encrypted at all.

2.2 Motivation and Contribution

In the scenario of outsourcing private inputs or computational function to cloud, There exist hurdles in following two aspects - One is in the users’ or customers point of view, where they want to ensure the privacy of its input parameters and results. Another is to cloud servers point of view, where cloud entity is worried about feasibleness of encrypted/transformed inputs and operating on them. In computational outsourcing, users are not participating in the computational function, rather than they outsource the private problem along with parameters to the cloud, but users and cloud servers are not mutually trusted entities. Thus, users would not like to submit their private problem data inputs to the cloud. Thus, encrypting/transforming the private data prior to submission to cloud is a usual solution.

Our contribution in this paper is as -

- We have proposed protocol for secure and an efficient computational outsourcing to cloud. The protocol is completely non-interactive between users.
- We have performed the computational security analysis for our proposed system.
3 Preliminaries

This section discusses some of the significant preliminaries required for secure computational outsourcing.

3.1 Lattice-Based Encryption

As we know that the computational complexity as well as the input parameters’ privacy is mostly dependant on the encryption procedure adopted by user. Lattice-Based Encryption [9,10] is considered as secure against quantum computer attacks and much efficient as well as potent than RSA and Elliptic curve cryptosystems.

Lattice based cryptosystem, whose security is based on core lattice theory problems, was introduced by Miklos Ajtai, in 1996. In the same year, first lattice based public key encryption scheme (NTRU) was proposed. Later, much work and improvement [16] was carried out towards this direction involving some additional cryptographic primitives LWE (learning with errors).

3.2 Computational Verifiability

Various different solutions exist for secure computational outsourcing. Homomorphic encryption (HE) can be assumed as a better solution to secure outsourcing of scientific computations, but it is useful when the returned result can be trusted.

Lemma 1. It is infeasible to factorizing the $N$ in polynomial time if integer factorization in large scale in infeasible.

Proof. Assume $x$ is an adversary who is able to factorize a number $N$ into primes $p$ and $q$ of probable same bit length in polynomial time. Suppose this operations probability as $p'$. Each factor $fact_i$ of a number $N$ will at least posses two prime factors. So the probability $p_i'$ that the attacker can factorize it is almost lesser than $p'$. Thus the resultant probability that attacker can factorize $N$ is $\prod_{i=1}^{n} p_i' \leq (p')^n$. Now if $p'$ is negligible, the resultant probability is also negligible.

Definition 1. A matrix $M \in R^{n \times n}$ can be called as orthogonal if it is satisfying one of the equivalent conditions

1) $M \cdot M^T = M^T \cdot M = I_n$;

2) $M$ is invertible and $M^{-1} = M^T$.

4 Secure Outsourcing Using FHE

This section summarizes Sudarshan et al. scheme [26] for secure outsourcing of large matrix multiplication computations on cloud. First, the key space is being generated at client side, which will be utilized in further steps. Here, we have considered the scientific computation as ‘large matrix multiplication’, which the client needs to outsource to cloud server. Here the assumption taken is that the third party or cloud server is untrusted. Client needs to perform problem transformation step for secure outsourcing. Further, the computation inside cloud is performed. After getting the computed result, client will retransform it and get the original result for matrix multiplication problem. The complete description and steps involved in this scheme are summarized as below:

Algorithm 1 Secure Outsourcing using FHE

1. Begin
2. Generate secret key pair: $\{H,Y\}$
   where, $H$: is a Hadamard matrix [34] and $Y$: is a diagonal matrix selected randomly.
3. Consider, $M_1$ and $M_2$ are two large matrices, for which the multiplication needs to be computed, thus client will outsource this computation problem to cloud side.
4. Client computes,
   
   $M'_1 = H \times M_1 \times Y$
   $M'_2 = Y^{-1} \times M_2$

5. Client sends $M'_1$ and $M'_2$ to cloud server.
6. $Result' \leftarrow M'_1 \times M'_2$
7. The cloud server sends back the computed result to client side.
8. After getting the computed result, client will retransform it and get the original result for MM problem.

The procedure is given as below Algorithm

9. $Result \leftarrow H^{-1} \times Result'$
10. End

5 Experimental Analysis

This section presents our experimental analysis.

5.1 System Specifications

Our system specifications are as below:

- Software specifications
  OS Ubuntu 16.04 LTS, 64 bit; Python version 'Python 3.6.0'.

- Hardware specifications
  Processor Intel core i3 4030U CPU @1.90GHz x 4.

5.2 Our Results

We performed the experiments on varying sized secret key pair, arbitrary large sized matrices $M_1$ and $M_2$ as input. Problem parameters transformation/encryption, decryption and entire average execution time for executing the protocol is analysed. The graph for encryption phase for various sized input parameters is Figure 1.

The graph for decryption phase for various sized input parameters is in Figure 2.
The graph for overall algorithm execution for various sized input parameters is in Figure 3.

In above Performance graphs 1-3, the encryption, decryption and overall execution time (in seconds) for varying experimental instances of secret key matrix pair size dimensions is shown. The end results of execution performance for varying key sizes is presented as Table 1.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Dimensions</th>
<th>Exec Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>T(Encry)(in sec)</td>
</tr>
<tr>
<td>1.</td>
<td>4x4</td>
<td>0.0994174</td>
</tr>
<tr>
<td>2.</td>
<td>8x8</td>
<td>0.1260472</td>
</tr>
<tr>
<td>3.</td>
<td>16x16</td>
<td>0.1321644</td>
</tr>
<tr>
<td>4.</td>
<td>32x32</td>
<td>0.165747</td>
</tr>
</tbody>
</table>

| Tabular form |

6 Proposed Scheme

In this section, we have proposed an efficient secure computational outsourcing mechanism applicable for multi-users. The system model and proposed mechanism steps are given in subsections below.

6.1 System Model

The proposed system model is represented as in Figure 4. Notations used are given in Table 2.
Table 2: Notations used in proposed system

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1</td>
<td>First cloud server</td>
</tr>
<tr>
<td>CS2</td>
<td>Second cloud server</td>
</tr>
<tr>
<td>c_i</td>
<td>Ciphertexts (encrypted data of each customer/user U_i)</td>
</tr>
<tr>
<td>n</td>
<td>No. of users</td>
</tr>
<tr>
<td>α_i</td>
<td>Private input corresponding to U_i</td>
</tr>
<tr>
<td>ψ</td>
<td>Probability density function</td>
</tr>
<tr>
<td>q</td>
<td>Prime order</td>
</tr>
<tr>
<td>RAND_i</td>
<td>Random number for i^{th} user</td>
</tr>
<tr>
<td>C^FUn</td>
<td>Function circuit</td>
</tr>
<tr>
<td>R</td>
<td>Ring structure space</td>
</tr>
<tr>
<td>β</td>
<td>Final computed result</td>
</tr>
</tbody>
</table>

6.2 Protocol Steps

The proposed secure computational outsourcing protocol executes in the below phases

Algorithm 2 Key Gen() and Set up

1: Begin
2: Perform sampling for ring element space vector a_i \sim R_q^N, \forall i = (1, 2, \cdots, n); Ring element SK_i \sim \psi; α_i \sim ψ^N (ψ represents: probability density function), where
\[ ψ = \int_{-\infty}^{x} P(ξ) dξ \]
3: Key pairs of U_i: Public key (a_i.SK_i + 2α_i) \in R_q^N; Private key SK_i.
4: CS1 has its private no. as K_{CS1} & CS2 has its private no. as K_{CS2}.
5: U_i shares a random no. RAND_i with CS1.
6: Each user U_i initiates protocol and sends RAND_i,SK_i to CS2.
7: CS2 reckons K_{CS2},RAND_i,SK_i and sends back to CS1.
8: CS1 can get K_{CS2},SK_i by extracting RAND_i.
9: End

∀i \in (1, 2, \cdots, n), U_i uses Lattice based encryption method to encrypt its own problem input α_i. The sub-steps involved in this are as Algorithm-3.

Algorithm 3 Lattice based Encryption

1: Begin
2: First U_i perform sampling as: e_i \sim ψ^N, where, ψ is: probability density function(PDF), defined as
\[ ψ = \int_{-\infty}^{x} P(ξ) dξ \]
3: Next, each user U_i computes
\[ c_i = \langle u_i, e_i \rangle + α_i \in R_q \]
\[ c_i, = \langle a_i, e_i \rangle \in R_q \]
4: Further, it gives output as ciphertext,
\[ c_i = (c_i, c_i) \in R_q^N; (N = 2) \]
5: End

CS1 stores all ciphertexts coming from user U_i(1 ≤ i ≤ n), then further steps are given in Algorithm-4.

Production of the result by cloud servers will follow as steps of Algorithm-5.

6.3 Analysis of Proposed Scheme

Here, we have presented the correctness and security analysis of our proposed scheme.

- Correctness analysis:
  The correctness analysis of given scheme is as follows.

  **Theorem 1.** Due to Homomorphic properties of the transformed ciphertexts, the given scheme is correct.

  Let, P and Q are rings a function f : P \rightarrow Q will be ring homomorphism if ∀x_1, x_2 \in P.

  \[ f(x_1 + x_2) = f(x_1) + f(x_2); \]
  \[ f(x_1 * x_2) = f(x_1) * f(x_2). \]

- Security analysis:
  The security analysis of proposed scheme can be analysed as below.

  **Theorem 2.** As long as Lattice based encryption is secure and cloud servers CS1 and CS2 are noncolluding, the given protocol is secure enough.
Algorithm 4 Circuit Computation on Outsourcing
1: Begin
2: First, CS1 transforms the ciphertexts as $c_i \rightarrow c_i^{TR_1}$
   where, $(c_i^{TR_1}, c_i^{TR_2}) = (K_{CS1.c_i, K_{CS1.}(K_{CS2.SK_i}).c_i})$.
3: CS1 sends above $c_i^{TR_1}$ to CS2.
4: After receiving $c_i^{TR_1}$, CS2 again transforms $c_i^{TR_1}$ into $c_i^{TR_2} = (K_{CS2, K_{CS1.c_i}, K_{CS1.}(K_{CS2.SK_i}).c_i})$
   take, $K = K_{CS1, K_{CS2}}$ then, $c_i^{TR_2} = (c_0^{TR_2}, c_i^{TR_2}) = (K.c_i, K.SK_i.c_i)$
5: CS2 then reckons the ciphertext of result by transformed ciphertext of every user’s private i/p.
6: Additive oprn. for each add. gate
   $\Rightarrow c_i^{TR_2} \oplus c_j^{TR_2}$
   $\Rightarrow (c_i^{TR_2} - c_0^{TR_2}) \bigoplus (c_j^{TR_2} - c_0^{TR_2})$
   $\Rightarrow (K.K_{SK_1.c_i} - K.c_i \bigoplus (K.SK_i.c_i - K.c_i)$
   $\Rightarrow (K.(SK_1.c_i - c_0) \bigoplus (K.SK_i.c_i - c_0))$
   $\Rightarrow K.(SK_1.c_i - c_0) \bigoplus (SK_i.c_i - c_0)$
   $\Rightarrow K.[\alpha_i + \alpha_j]$
7: Multiplicative oprn. for every mul. gate -
   $\Rightarrow c_i^{TR_2} \otimes c_j^{TR_2}$
   $\Rightarrow (c_i^{TR_2} - c_0^{TR_2}) \bigotimes (c_j^{TR_2} - c_0^{TR_2})$
   $\Rightarrow (K.(SK_1.c_i - c_0) \bigotimes (K.SK_i.c_i - c_0)$
   $\Rightarrow (K.(SK_1.c_i - c_0) \bigotimes (K.SK_i.c_i - c_0))$
   $\Rightarrow K^2.(SK_1.c_i - c_0) \bigotimes (SK_i.c_i - c_0)$
   $\Rightarrow K^2.[\alpha_i \times \alpha_j]$
8: End

Algorithm 5 Production of Result
1: Begin
2: When CS2 performed gate by gate computation on circuit $\psi_{FUN}$, it gets some intermediate meta result, which is encrypted by $K_{CS1}$ and $K_{CS2}$ of the cloud servers CS1 and CS2.
   If $\beta = FUN(\alpha_1, \alpha_2, \ldots, \alpha_n)$ and let’s $\theta$ is the no. of multiplicative gates of $\psi_{FUN}$.
   then, $\beta^\theta = K_1^{\theta+1, \beta} = (K_{CS1}^{\theta+1, K_{CS2}})^\beta$
3: To provide results for each user, and ensure that only authorized user set must get final result [Assume, $U_\lambda$, $\lambda \in \{1, 2, 3, \ldots, n\}$ is authorized user set to access result], CS2 first sends $\beta^\theta$ to CS1.
4: CS1 removes $K_{CS1}^{\theta+1}$ and ties $RANDB_\lambda$ to compute $\beta_\lambda = RANDB_\lambda, K_{CS2}^{\theta+1, \beta}$
5: Then CS1 sends $\beta_\lambda$ to CS2.
6: CS2 finally removes $K_{CS2}^{\theta+1}$ and gets $\beta_\lambda = RANDB_\lambda, \beta$
7: Further CS2 sends it to authorized users set $U_\lambda, \lambda \in \{1, 2, 3, \ldots, n\}$.
8: End

Algorithm 6 Secure Results Reconstruction at Users’ side
1: Begin
2: For each $U_\lambda, \lambda \in \{1, 2, 3, \ldots, n\}$, it successfully gets the final result $\beta$ by depositing $RANDB_\lambda$.
3: End

In proposed protocol, each user $U_i$ encrypts its private input $\alpha_i$ with the help of its own public key, which is being produced by triggering lattice based encryption scheme. Further, $U_i$ sends $RANDB_i, SK_i$ to CS2. Then, CS2 reckons $K_{CS2}, RANDB_i, SK_i$ and sends back to CS1. Here, $U_i$’s private key is $SK_i$, which is protected by $RAND_i$. In the entire process, the user’s private keys are not being revealed.

After transferring computed results, cloud ensures in the protocol that only authorized user set must get final result; (Assume, $U_\lambda, \lambda \in \{1, 2, 3, \ldots, n\}$ is authorized user set to access result.)

6.4 Experimental Simulation Results
This section presents the simulation results on virtual cloud server. We used Microsoft Azure cloud server along with Android Studio 3.0.1 software platform SDK to perform simulation on client-cloud outsourcing model. JDK 9.0.1 is used for backend interface development. The machine used for simulation has specifications as OS: Windows 10; RAM size: 16 GB; NVIDIA GeForce GTX 1080 GPU with octa core i7 CPU.

The experimental simulation results obtained are given in Table 3.

In Table 3, we tested our scheme through various client’s private data size and key size parameters etc. The client’s private data can be in any form like integer, image, text, pdf file, video file etc. We obtained the time taken for encrypt private data, the time taken for upload operation on Azure cloud server and time taken to download and decrypt data on client side. Inside cloud scientific computational operations are being performed on transformed parameters. The overall representation is given in Figure 5.

6.5 Comparative Analysis
This section presents the comparison of our scheme with existing schemes on several factors/parameters. The representation is given in Table 4.

7 Conclusion and Future Work
When users have to compute some complex function, which involves their private inputs then to perform outsourcing is the possible scenario from user side. There exist hurdles in following two aspects: One is in the users’ or customers point of view, where they want to ensure the privacy of its input parameters and results. Another is to cloud servers point of view, where cloud entity is worried about feasibleness of encrypted/transformed inputs and operating on them.
Table 3: Experimental simulation

<table>
<thead>
<tr>
<th>Client data size</th>
<th>Key size</th>
<th>$T.T_{\text{Encry}}$</th>
<th>$T.T_{\text{Upload on Cloud Server}}$</th>
<th>$T.T_{\text{Download res and Decry}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 kB</td>
<td>16 bits</td>
<td>2 sec.</td>
<td>2 sec.</td>
<td>2.4 sec.</td>
</tr>
<tr>
<td>1 mB</td>
<td>32 bits</td>
<td>2.6 sec.</td>
<td>2.4 sec.</td>
<td>2.8 sec.</td>
</tr>
<tr>
<td>2 mB</td>
<td>32 bits</td>
<td>3 sec.</td>
<td>3.2 sec.</td>
<td>3.2 sec.</td>
</tr>
<tr>
<td>50 mB</td>
<td>64 bits</td>
<td>6 sec.</td>
<td>6.4 sec.</td>
<td>5.8 sec.</td>
</tr>
<tr>
<td>100 mB</td>
<td>64 bits</td>
<td>9.4 sec.</td>
<td>8.8 sec.</td>
<td>9 sec.</td>
</tr>
<tr>
<td>500 mB</td>
<td>128 bits</td>
<td>19 sec.</td>
<td>19.2 sec.</td>
<td>19 sec.</td>
</tr>
<tr>
<td>1 GB</td>
<td>128 bits</td>
<td>32.2 sec.</td>
<td>32 sec.</td>
<td>32.4 sec.</td>
</tr>
</tbody>
</table>

Figure 5: The overall performance

Table 4: Comparison with related work

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Feasible data size</th>
<th>$\text{Encry}()$ technique adopted</th>
<th>Download result and decrypt()</th>
<th>Users</th>
<th>Speed-up</th>
<th>Cloud Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>C. Wang et al. (2015)</td>
<td>Low and medium sized</td>
<td>Parameters transformation</td>
<td>Slow on large size data</td>
<td>Single user</td>
<td>Good for medium sized problem</td>
<td>Moderate</td>
</tr>
<tr>
<td>Jin Li et al. (2015)</td>
<td>Medium sized</td>
<td>Identity based encryption</td>
<td>Slow on large size data</td>
<td>Single user</td>
<td>Good up to medium sized problem</td>
<td>Moderate</td>
</tr>
<tr>
<td>Our construction</td>
<td>Medium to large sized</td>
<td>Lattice based encryption</td>
<td>Comparatively faster</td>
<td>Multi user supported</td>
<td>Better for large sized problem</td>
<td>Good</td>
</tr>
</tbody>
</table>
In this paper, we have constructed a scheme for secure outsourcing based on multi cloud servers. The computational complexity and security analysis is also given for our proposed system. Finding an efficient, practical and computationally secure outsourcing solution for various specific scientific problems will be our further research work.

References


Biography

V. Sudarsan Rao received his M.Tech in Computer Science and Engineering from JNTUH, Hyderabad, India, in 2010. Currently, he is a Researcher in the Department of Computer Science and Engineering of JNTUH, Hyderabad. His research interests include Cloud Computing and Security of provably secure symmetric encryption schemes, efficient software implementations of cryptographic primitives, pattern recognition, and Network Security.

Dr. N. Satyanarayana, M. Tech (CS), Ph.D (CSE): The Nagole Institute of Science and Technologies headed by Dr. N. Satyanarayana who possessed the highest qualification in engineering. His qualifications are M.Phil, AMIE(ET), M.Tech (CS), Ph.D(CSE), MISTE, MCSI. He obtained his Ph.D in Computer Science and had a remarkable record of merit in pursuit his engineering studies. He is young and dynamic. He has a disposition to inspire both students and staff to achieve their right goals. With his extensive and rich research experience he is able to run the institution on sound academic ground.