ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Resolving large and Complex Matrix Factorization Problem using Cloud Platform as a Service

R.Jeyaseelan[*1], N.Pandeeswari[2], Dr. P.Ganeshkumar[3]

[*1] PG Scholar, [2]Assistant Professor, [3]Professor, Department of Information Technology, PSNA College of Engineering and Technology, Dindigul, India

Jai.seelan98@gmail.com

Abstract

Cloud computing enables the resource constrained clients lacks economically, to use the huge computational power of cloud resources to outsource their complex computational tasks. However, outsourcing original input to the public cloud causes severe anxiety over security risks. The input/output privacy has to be maintained and also, proof has to be generated to verify the result against the malicious cloud server.The matrix factorization (MF) which requires enormous amount of resources to complete its task; is pretty common in engineering and economics computational task such as text mining and analysis. The proposed scheme is motivated to design a secure, resilient and efficient outsourcing of MF to the public cloud server. The concept behind defending the input matrix is by applying the transposition and permutations on the original matrix to acquire encrypted matrix. Then, the result returned from the cloud is decrypted and proof verification is done to ensure the correctness of cloud server. This paper exhibits that the how securely and efficiently proposed protocol outsources the MF problem onto the cloud and then verifying the result against malicious cloud server. Extensive theoretical and experimental analysis shows that this protocol is extremely efficient and widely applicable for practical use.

Keywords: Cloud computing, matrix factorization, robust cheating resistance, secure outsourcing, Monte carlo verification

Introduction

Cloud computing is the recent innovative technology is defined as [1] providing on demand network access to a large pool of computing platform deployed with greater efficiency and little management overhead. With the efficient computing paradigm, the clients' lacks due to the limited computational resources are encouraged to utilize the cloud computing utility. Instead of setting up their own computing platform with huge amount of resources, the clients can utilize the computing platform provided through one of the cloud services, Platform as a service (PaaS) on pay per use manner. Despite with the beneficial services, outsourcing client's original information to the malicious cloud server may bring security risks and challenges [1]. Before outsourcing original information to the public cloud, the original information has to be transferred to its encrypted version. Since the large scale matrix factorization problem is computationally complex [3] to solve with restricted amount of resources. The proposed paper aims to outsource the MF problem into the public cloud to use the on demand computing platform with greater amount of resources. And also, it is significant to ensure the input/output privacy. The original input matrix protected by doing transposition and permutations. Then the encrypted matrix is forwarded to the cloud server. The result come back from the cloud server is decrypted at client side. Due to the openness of cloud computing platform, result produced by cloud server is not yet ensured as correct. Rather, the computation inside cloud is not transparent [4] and no guarantee to the quality of the computational result. In addition, some accidental reasons such as [1] software bugs and hardware failures may produce false computation result. Consequently, to necessitate the correctness of the computing utility, it is optimal to ensure the correctness of result. The proposed work contributes to design a protocol which is secure, robust and efficient to outsource the MF problem to the public cloud. This protocol design has four phases specified in order namely key generation, MF encryption, MF decryption, and result verification. Key generation phase is used to generate a secret key for every

ISSN: 2277-9655
Impact Factor: 1.852

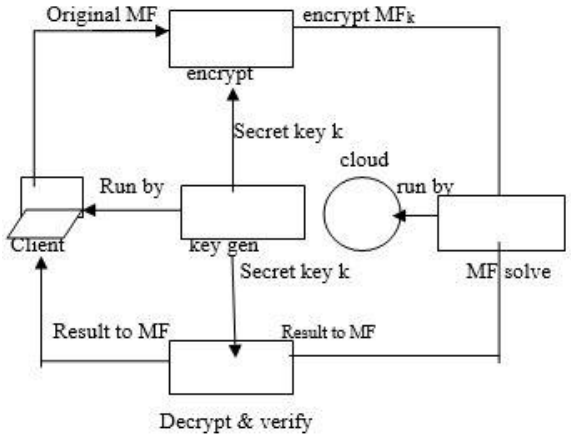
instance of matrix. The second phase, MF encryption is to transfer the original matrix into encrypted matrix with the help of the generated secret key. The third phase, MF decryption is to transfer the result which is in decrypted form to original form. The MF encryption and decryption is done at client side only. The fourth phase is result verification, which involves multiplying the decrypted factor matrices and verifies the result with original input matrix. The cloud computing utility can be able to work [1] with the encrypted input value. So, the malicious attacker cannot observe any details about user information.

CHALLENGES. The following are some of the notable challenges outsourcing computational problem to the public service provider. Client's input/output data privacy is the first challenge. The outsourced computational problems and the results to these problems often contain sensitive information. To solve these problem means user to hide the original data from the cloud, client needs to encrypt their data before outsourcing and decrypt the returned result from the cloud after outsourcing.

The second challenge is the verification of the result returned by the cloud, because the cloud may be return the incorrect value. The outsourcing factorization protocol should satisfy some aspects: secure, verifiable, and efficient.

MAIN CONTRIBUTIONS. This paper addresses the issue of how to outsource MF to a remote malicious cloud server while ensuring correctness, maintaining data input/output privacy, realizing result verifiability, and improving computational efficiency. matrix multiplication and matrix factorization of square matrices are essentially the same problem. An important challenge in encrypting the input matrix M is therefore to avoid multiplying the original matrix M with general matrices, for avoiding a complexity that is the same as inverting M itself. By applying permutation functions, this paper describes a way of multiplying M with special matrices, where matrix product can be computed in $L(n^2)$ time. Besides, the challenge in the result verification step is also to avoid general matrix multiplication, since the validity of a returned matrix can be easily checked by taking a product of that matrix with the original input M, and check whether an identity matrix is obtained. By introducing Monte Carlo verification algorithm, the proposed protocol is able to verify the correctness of the returned result in $L(n^2)$ time. Based on permutation technique and Monte Carlo technique, the client can reduce its original $L(n^{2.373})$ work to $L(n^2)$ work by outsourcing MF to a cloud. Moreover, experimental evaluation is also provided to show that the proposed protocol is able to allow the client to outsource MF to a cloud and gain substantial computation savings.

ORGANIZATION. This paper also proceeds as, introduces some essential preliminaries. we describe our protocol with detailed techniques. And give some related analysis and performance evaluation,



SECURE MF OUTSOURCE SYSTEM MODEL

System Construction

System Model. We consider the secure MF out-sourcing system model, as illustrated in above diagram client with low computational power wants to outsource the original MF to a cloud service provider, who has massive computational power and special software's. In order to protect input privacy, the client encrypts the original MF using a secret key K to get a MF problem, written as MF_K . Later, the encrypted MF_K is given to the cloud for a result. Once the cloud receives MF_K , the computation is carried out with software's; then the cloud sends back the result to MF_K . The cloud also sends back a proof Γ that tries to prove the returned result is indeed correct and the cloud does not cheat. On receiving the returned result, the client decrypts the returned result using the secret key K to get the result to the original MF. Meanwhile, the client checks whether this result is correct: if yes, accepts it; if no, just rejects it.

Threat model. The security threats faced by the outsourcing system model primarily come from the behavior of the cloud. Generally, there are two levels of threat models in outsourcing: semi-honest cloud model and malicious cloud model. In the semi-honest cloud model, the cloud correctly follow the protocol specification. However, the cloud records all the information it can access, and attempts to use this to learn information that should remain private. While in the malicious cloud model, the cloud can arbitrarily deviate from the protocol specification. The malicious cloud may just return a random result to the client to

ISSN: 2277-9655
Impact Factor: 1.852

save its computing resources, while hoping not to be detected by the client. Therefore, an outsourcing protocol in the malicious cloud model should be able to handle result verification. In this paper, we assume that the cloud is malicious. The proposed protocol should be able to resist such a malicious cloud.

The Matrix Factorization Problem Model: The matrix factorization is defined as decomposing the original matrix into a product of factor matrices in the form, lower triangular matrix and upper triangular matrix. Matrix factorization maps two different elements to a joint space of dimensionality assume the two different elements are user and item. The matrix explains the association [5] between item and user is the input matrix; AB.It is quite difficult [5] to compute the factor matrices. A well known technique called as singular value decomposition (SVD) and gradient descent method [5] can be used to identify factor matrices.

Let AB be the original matrix that is factorized as A and B matrices.

AB=A .B

Input matrix AB of size m x n is factored as m x q and q x n;

Matrix 1:

A of size m x q

Matrix 2:

B of size q x n;

Such that A X B = AB

MF protocol design:

Since MF is very large computational task, the clients are encouraged to use the on demand cloud computing utility on pay per use manner. To safely outsource their problem into

Public cloud server, the proposed scheme aims to design a protocol which is safe, secure, robust and efficient. The protocol design has four phases and they are specified in orderly manner.

1. Secret random key generation.
2. MF encryption.
3. MF decryption.
4. Result verification.

Secret random key generation:

The protocol design starts with secret random key is generated using java random secret key generator algorithm.

Design goals. We identify the following goals that the outsourcing protocol should satisfy.

- *Correctness.* If both the client and the cloud follow the protocol honestly, the MF can be indeed fulfilled by the cloud and the client gets a correct result to the original MF.
- *Security.* The protocol can protect the privacy of the client's data. On one hand, given the encrypted MF_K problem, the cloud cannot get meaningful knowledge of the client's input data, which is referred to as *input privacy*. On the other hand, the correct result to the original MF is also hidden from the cloud, and this is called as *output privacy*.
- *Robust Cheating Resistance***.** The correct result from a faithful cloud server must be verified successfully by the client. No false result from a cheating cloud server can pass the verification with a non-negligible probability.
- *Efficiency.* The local computation done by the client should be substantially less than the computation of the original MF on his own. In addition, the amount of computation on computing the encrypted MF_K should be as close as possible to that on computing the original MF.

Framework. Syntactically, a secure MF outsourcing protocol should contain five sub-algorithms: 1) the algorithm for key generation Key Gen, 2) the algorithm for MF encryption MF Enc, 3) the algorithm for solving MF_K problem MF Solve, 4) the algorithm for MF decryption MF Dec, and 5) the algorithm for result verification Result Verify.

One significant difference between this framework and the traditional encryption framework is that in this case both encryption and decryption process occur in the client side. This eliminates the expensive public key exchange process in the traditional encryption framework. Therefore, this framework is able to efficiently realize one-time-pad type of flexibility. That is to say, Key Gen will be run every time for a new outsourced matrix instance to enhance security. Once we have this framework, we just need to work out the details of these five sub-algorithms, they are,

Protocol Construction

In this section, each part of the framework for secure outsourcing of Matrix Factorization will be individually solved.

ISSN: 2277-9655
Impact Factor: 1.852

Matrix Factorization in the Cloud

Algorithm Procedure MFK -in-the-Cloud

Input : Y.
Output: $R= Y^{-1.}$

1: On input the encrypted matrix Y, the cloud then invokes any matrix inversion algorithm to compute

$R= Y^{-1.}$

2: The cloud then sends matrix R back to the client.

Algorithm 1

Matrix Factorization Encryption

The second phase of the protocol is Matrix Factorization encryption which is used to convert the original matrix into encrypted matrix by taking transposition and permutations[5] on the original matrix which makes the original information secure. If the secret key generated in the above phase is k; then algorithm for second phase is described as follows:

Input: secret key k and input matrix AB, m x n;
Output: encrypted matrix.

1. multiply input matrix by secret key k
   k. AB = k (AB) => (kA). B
2. Take transpose of the matrix.
   $(k(AB))^T = (B)^T . (kA)^T$
3. Do permutation to swap the row elements.

Store the first row of *AB* matrix in a temporary array T having size as m*1;

```
For i =2 to m
begin
For j=1 to n
 begin
 move: ABij-> ABi-1, j;
 end;
 end;
```

3. Store the value in matrix T as the last row of the input matrix;

```
For j=1 to m
begin
move: t [1] [j] -> AB[m] [j];
end;
```

Matrix Factorization Decryption

The input given to the Matrix Factorization solver is MFk that produce the factor matrices C and D such as C= $(B)^T$ and D= $(kA)^T$

The third phase of protocol design is decryption of result which returns from the cloud server. The decryption algorithm is normally the reverse of corresponding encryption algorithm.
The following algorithm explains the decryption procedure.

Input: the factor matrices C and D;
Output: The original factor matrices A and B
1. Change the order of multiplication;
   C.D -> D.C = > $(k(AB))^T = (B)^T (kA)^T$ -> $(kA)^T (B)^T$
2. Take transpose of factor matrices.
   1st matrix: **$(kA)^T$ -> kA; m x q**
   2nd matrix: **$(B)^T$ -> B; q x n**
2.Divide all the elements of 1st matrix by secret key k;
3.Do permutation to swap the row elements.
3.1To retrieve the first factor matrix A:
3.1.1 store the last row element of matrix A in an temporary array T1 of size (m x 1);

```
3.1.2 for i=1 to m-1
        begin
        for j=1 to q
        begin
        move: Aij -> Ai+1,j
        end;
        end;
```

3.1.3 Store the elements of T1 as the last row of the A;

```
        i=m;
        for j=1 to q
        begin
        move T1ij->Aij
        end;
```

3.2To retrieve the second factor matrix B;
3.2.1Store the last row element of matrix B in an temporary array T2 of size (q x 1);

```
3.2.2 for i=1 to n-1
        begin
        for j=1 to q
        begin
        move: Bij -> Bi+1, j
        end;
        end;
```

ISSN: 2277-9655
Impact Factor: 1.852

3.2.3 Store the elements of T2 as the last row of the B;

```
i=q;
for j=1 to n
begin
move T2ij->Bij
end;
```

Final result is obtained .The resultant factor matrices are A and B.

Result verification

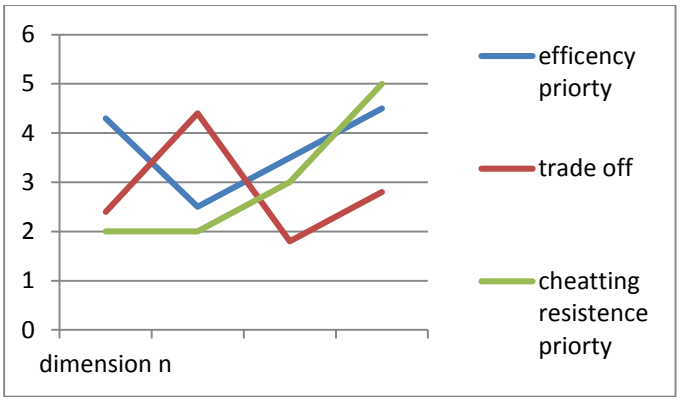
Input: The factor matrices and original matrix.
Output: Boolean value; true or false;

```
1. Boolean proofverification (matrix A, matrix B, matrix AB)
{
    if ( A . B = = AB)
    then
    return 'True'
    else
    return 'False'
}
```

The platform as a service of cloud computing provides a computing platform to do the computational task.

Performance Evaluation

CLIENT SIDE OVERHEAD. The client side overhead is generated by running four sub-algorithms: Key Gen, MFC Enc, MFC Dec, and Result Verify. It is evident that Key Gen takes time O(n). In MFC Enc, applying (4) to efficiently compute Y, it only takes time O(n2).Likewise, the time consumed by MFC Dec is O(n2). As to Result Verify, the time is dominated by computing R×(Xr), which takes time O(n2). CLOUD SIDE OVERHEAD. For the cloud, its only computation overhead is generated by running MFC Solve. The cloud can apply any existing matrix inversion algorithm. As mentioned before, from the complexity point of view, matrix multiplication and matrix inversion of square matrices are essential the same problem, we have that the computational overhead in the client side will be less than that in the cloud side for a sufficiently large n. The theoretical results indicate that the proposed protocol is able to allow the client to outsource MF to the cloud and gain substantial computation savings. This claim will be further validated by our experiments in the next subsection.



Conclusion

In this paper, we have designed a protocol for out-sourcing of MF to a malicious cloud. We have shown that the proposed protocol simultaneously fulfills the goals of correctness, security (input/output privacy), robust cheating resistance, and high efficiency. With MF already well rooted in scientific and engineering fields, the proposed protocol can be deployed individually or serve as a primitive building block, based on which some higher level secure outsourcing protocols are constructed. We also introduced a Monte Carlo verification algorithm to handle result verification. Its superiority in designing inexpensive result verification algorithm for secure outsourcing is well demonstrated. Directions to launch further research include: 1) establishing formal security framework for MF outsourcing problem; 2) adding result verification for some early protocols, which do not handle result verification, as a counter offensive to malicious cloud; 3) identifying new meaningful scientific and engineering computational tasks and then designing protocols to solve them.

Refrences

[1] *Xinyu Lei, Xiaofeng Liao, Senior Member, IEEE, Tingwen Huang, Huaqing Li, and Chunqiang Hu "Outsourcing Large Matrix Inversion Computation to A Public Cloud" . IEEE Transactions On Cloud Computing, vol. x, no. x, x 2013*

[2] *Cong Wang, Student Member, IEEE, Kui Ren, Member, IEEE, Jia Wang, Member, IEEE, and Karthik Mahendra Raje Urs, "Harnessing the Cloud for Securely Outsourcing Large-scale Systems of Linear Equations"*

[3] *Zhengping Qiany Xiuwei Cheny Nanxi Kang\ Mingcheng Chen\ Yuan Yuz Thomas Moscibroday Zheng Zhangyy Microsoft Research Asia, \ Shanghai Jiaotong*

ISSN: 2277-9655
Impact Factor: 1.852

University, z Microsoft Research Silicon Valley "MadLINQ: Large-Scale Distributed Matrix Computation for the Cloud"

[4] Sun Microsystems, Inc., "Building customer trust in cloud computing with transparent security," 2009, online at https://www.sun.com/offers/details/sun transparency.xml.

[5] Yehuda Koren, Yahoo Research, Robert Bell and Chris Volinsky, AT&T Labs—Research "Matrix Factorization Techniques for Recommender Systems" Published by the IEEE Computer Society 0018-9162/09/$26.00 © 2009 IEEE

[6] http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/. Matrix equation

[7] X. Zhang, Z. Qian, Y. Ren, and G. Feng, "Watermarking with flexible self-recovery quality based on compressive sensing and compositive reconstruction," Information Forensics and Se-curity, IEEE Transactions on, vol. 6, no. 4, pp. 1223–1232, 2011

[8] Y. Lindell and B. Pinkas, "Secure multiparty computation for privacy-preserving data mining," Journal of Privacy and Confidentiality, vol. 1, no. 1, p. 5, 2009.

[9] R. Durstenfeld, "Algorithm 235: random permutation," Com-munications of the ACM, vol. 7, no. 7, p. 420, 1964.

[10] D. E. Knuth, The art of computer programming. addison-Wesley, 2006.

[11] R. Freivalds, "Probabilistic machines can use less running time," Information Processing, vol. 77, pp. 839–842, 1977.

[12] R. Motwani and P. Raghavan, Randomized algorithms. Cam-bridge university press, 1995.

[13] J. Camenisch, S. Hohenberger, and M. Pedersen, "Batch verification of short signatures," Advances in Cryptology–EUROCRYPT 2007, pp. 246–263, 2007.

[14] V. Strassen, "Gaussian elimination is not optimal," Numerische Mathematik, vol. 13, no. 4, pp. 354–356, 1969.

[15] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," Journal of symbolic computation, vol. 9, no. 3, pp. 251–280, 1990.