LINEAR PROGRAMMING CONCEPTS FOR SECURE AND PRACTICAL OUTSOURCING

Madesh Kumar S
PG Student
M Tech.
HKBKCE, Bangalore
madeshk612@gmail.com

***Abstract*— Internet computing technologies, like grid computing, enable a weak computational device connected to such a grid to be less limited by its inadequate local computational, storage, and bandwidth resources. However, such a weak computational device (PDA, smartcard, sensor, etc.) often cannot avail itself of the abundant resources available on the network because its data are sensitive [1]. Cloud computing promises greater flexibility in business planning along with significant cost savings by leveraging economies of scale in the IT infrastructure. It also offers a simplified capital and expenditure model for compute services as well as increased agility for cloud customers who can easily expand and contract their IT services as business needs change [2].**

The main objective of cloud computing enables customers with limited computational resources to outsource their large computation workloads to the cloud, and economically enjoy the massive computational power, bandwidth, storage, and even appropriate software that can be shared in a pay-per-use manner. Despite the tremendous benefits, security is the primary obstacle that prevents the wide adoption of this promising computing model, especially for customers when their confidential data are consumed and produced during the computation. One fundamental advantage of the cloud paradigm is computation outsourcing, where the computational power of cloud customers is no longer limited by their resource-constraint devices [4].

Outsourcing is to store the task or data which user wants it to do from outside their system such as cloud. By outsourcing the workloads into the cloud, customers could enjoy the literally unlimited computing resources in a pay-per-use manner without committing any large capital outlays in the purchase of hardware and software and/or the operational overhead therein. There are two adversarial behaviours that the helper software might engage in: intelligent and unintelligent failures. Intelligent failures occur any time that the helper chooses to deviate from its advertised functionality based on knowledge it gained of the inputs to the computation it is aiding [3].

From outsourced computation workloads often contain sensitive information. To combat against unauthorized information leakage, sensitive data have to be encrypted before outsourcing so as to provide end to end data confidentiality assurance in the cloud and beyond. On the other hand, the operational details inside the cloud are not transparent enough to customers. Besides, possible software bugs, hardware failures, or even outsider attacks might also affect the quality of the computed results. Without providing a mechanism for secure computation outsourcing, i.e., to protect the sensitive input and output information of the workloads and to validate the integrity of the computation result, it would be hard to expect cloud customers to turn over control of their workloads from local machines to cloud solely based on its economic savings and resource flexibility [4].

I. INTRODUCTION

Cloud Computing provides convenient on-demand network access to a shared pool of configurable computing resources that can be rapidly deployed with great efficiency and minimal management overhead [1]. One fundamental advantage of the cloud paradigm is computation outsourcing, where the computational power of cloud customers is no longer limited by their resource-constraint devices. By outsourcing the workloads into the cloud, customers could enjoy the literally unlimited computing resources in a pay-per-use manner without committing any large capital outlays in the purchase of both hardware and software and/or the operational overhead therein.

Despite the tremendous benefits, outsourcing computation to the commercial public cloud is also depriving customers' direct control over the systems that consume and produce their data during the computation, which inevitably brings in new security concerns and challenges towards this promising computing model [2]. On the one hand, the outsourced computation workloads often contain sensitive information, such as the business financial records, proprietary research data, or personally identifiable health information etc. To combat against unauthorized information leakage, sensitive data have to be encrypted before outsourcing [2] so as to provide end to-end data confidentiality assurance in the cloud and beyond. However, ordinary data encryption techniques in essence prevent cloud from performing any meaningful operation of the underlying plaintext data [3], making the computation over encrypted data a very hard problem. On the other hand, the operational details inside the cloud are not transparent enough to customers [4]. As a result, there do exist various motivations for cloud server to behave unfaithfully and to return incorrect results, i.e., they may behave beyond the classical semi-honest model. For example, for the computations that require a large amount of computing resources, there are huge financial incentives for the cloud to be "lazy" if the customers cannot tell the correctness of the output. Besides, possible software bugs, hardware failures, or even outsider attacks might also affect the quality of the computed results. Thus, we argue that the cloud is intrinsically *not secure* from the viewpoint of customers.

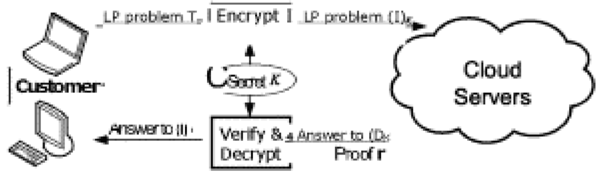
Recent researches in both the cryptography and the theoretical computer science communities have made steady advances in "secure outsourcing expensive computations" (e.g. [5] [10]). Based on Yao's garbled circuits [11] and Gentry's breakthrough work on fully homomorphism encryption (FHE) However, applying this general mechanism to our daily computations would be far from practical, due to the extremely high complexity of FHE operation as well as the pessimistic circuit sizes that cannot be handled in practice when constructing original and encrypted circuits. Focusing on engineering computing and optimization tasks, in this paper, we study practically efficient mechanisms for secure outsourcing of linear programming (LP) computations. Linear programming is an algorithmic and computational tool which captures the first order effects of various system parameters that should be optimized, and is essential to engineering optimization. It has been widely used in various engineering disciplines that analyze and optimize real-world systems, such as packet routing, flow control, power management of data centers, etc. [13]. Because LP computations require a substantial amount of computational power and usually involve confidential data, we propose to explicitly decompose the LP computation.

Specifically, we first formulate private data owned by the customer for LP problem as a set of matrices and vectors. This higher level representation allows us to apply a set of efficient privacy-preserving problem transformation techniques, including matrix multiplication and affine mapping, to transform the original LP problem into some arbitrary one while protecting the sensitive input/output information. One crucial benefit of this higher level problem transformation method is that existing algorithms and tools for LP solvers can be directly reused by the cloud server. Although the generic mechanism defined at circuit level, e.g. [9], can even allow the customer to hide the fact that the outsourced computation is LP, we believe imposing this more stringent security measure than necessary would greatly affect the efficiency. To validate the computation result, we utilize the fact that the result is from cloud server solving the transformed LP problem. In particular, we explore the fundamental duality theorem together with the piece-wise construction of auxiliary LP problem to derive a set of necessary and sufficient conditions that the correct result must satisfy. Such a method of result validation can be very efficient and incurs close-to-zero additional overhead on both customer and cloud server. With correctly verified result, customer can use the secret transformation to map back the desired solution for his original LP problem. We summarize our contributions as follows:

1) For the first time, we formalize the problem of securely outsourcing LP computations, and provide such a secure and practical mechanism design which fulfills input/output privacy, cheating resilience, and efficiency.
2) Our mechanism brings cloud customer great computation savings from secure LP outsourcing as it only incurs $0(nP)$ for some $2 < p < 3$ local computation overhead on the customer, while solving a normal LP problem usually requires more than $0(n^3)$ time [13].
3) The computations done by the cloud server shares the same time complexity of currently practical algorithms for solving the linear programming problems, which ensures that the use of cloud is economically viable.
4) The experiment evaluation further demonstrates the immediate practicality: our mechanism can always help customers achieve more than 30 x savings when the sizes of the original LP problems are not too small, while introducing no substantial overhead on the cloud.

The rest of the paper is organized as follows. Section II introduces the system and threat model, and our design goals. Then we provide the detailed mechanism description in Section III. Section IV and V give the security analysis and performance evaluation, followed by Section VI which overviews the related work. Finally, Section VII gives the concluding remark of the whole paper.

Fig. 1: Architecture of secure outsourcing linear programming problems in



Cloud Computing

II. PROBLEM STATEMENT

A. System and Threat Model

We consider a computation outsourcing architecture involving two different entities, as illustrated in Fig. 1: the *cloud customer,* who has large amount of computationally expensive LP problems to be outsourced to the cloud; the *cloud server* (CS), which has significant computation resources and provides utility computing services, such as hosting the public LP solvers in a pay-per-use manner.

The customer has a large-scale linear programming problem (I) (to be formally defined later) to be solved. However, due to the lack of computing resources, like processing power, memory, and storage etc., he cannot carry out such expensive computation locally. Thus, the customer resorts to CS for solving the LP computation and leverages its computation capacity in a pay-per-use manner. Instead of directly sending original problem (I), the customer first uses a secret K to map (I) into some encrypted version (DK and outsources problem *(I) lc* to CS. CS then uses its public LP solver to get the answer of (DK and provides a correctness proof F, but it is supposed to learn nothing or little of the sensitive information contained in the original problem description (I). After receiving the solution of encrypted problem (DK , the customer should be able to first verify the answer via the appended proof F. If it's correct, he then uses the secret K to map the output into the desired answer for the original problem (I).

Because LP computations require a substantial amount of computational power and usually involve confidential data, we propose to explicitly decompose the LP computation outsourcing into public LP solvers running on the cloud and private LP parameters owned by the customer. The flexibility of such a decomposition allows us to explore higher-level abstraction of LP computations than the general circuit representation for the practical efficiency.

B. Design Goals

To enable secure and practical outsourcing of LP under the aforementioned model, our mechanism design should achieve the following security and performance guarantees.

1) **Correctness:** Any cloud server that faithfully follows the mechanism must produce an output that can be decrypted and verified successfully by the customer.
2) **Soundness:** No cloud server can generate an incorrect output that can be decrypted and verified successfully by the customer with non-negligible probability.
3) **Input/output privacy:** No sensitive information from

the customer's private data can be derived by the cloud server during performing the LP computation.

4) **Efficiency:** The local computations done by customer should be substantially less than solving the original LP on his own.

C. Background on Linear Programming

An optimization problem is usually formulated as a mathematical programming problem that seeks the values for a set of decision variables to minimize (or maximize) an objective function representing the cost subject to a set of constraints. For linear programming, the objective function is an affine function of the decision variables, and the constraints are a system of linear equations and inequalities. Since a constraint in the form of a linear inequality can be expressed as a linear equation by introducing a non-negative slack variable, and a free decision variable can be expressed as the difference of two non-negative auxiliary variables, any linear programming problem can be expressed in the following standard form,

$$\text{minimize } c^T x \text{ subject to } \mathbf{Ax = b, x > 0}. \quad (1)$$

Here **x** is an *n x 1* vector of decision variables, **A** is an m x *n* matrix, and both **c** and **b** *are n x 1* vectors. It can be assumed further that m <*n* and that **A** has full row rank; otherwise, extras rows can always be eliminated from **A.**

In this paper, we study a more general form as follows,

$$\text{minimize } c^T x \text{ subject to } \mathbf{Ax = b, Bx > 0}. \quad (2)$$

In Eq. (2), we replace the non-negative requirements in Eq. (1) by requiring that each component of **Bx** to be non-negative, where **B** is an *n* x *n* non-singular matrix, i.e. Eq. (2) degenerates to Eq. (1) when **B** is the identity matrix. Thus, the LP problem can be defined via the tuple (I) = **(A, B, b, c)** as input, and the solution x as output.

III. THE PROPOSED SCHEMES

This section presents our LP outsourcing scheme which provides a *complete outsourcing* solution for — not only the privacy protection of problem input/output, but also its efficient result checking. We start from an overview of secure LP outsourcing design framework and discuss a few basic techniques and their demerits, which leads to a stronger problem transformation design utilizing affine mapping. We then discuss effective result verification by leveraging the duality property of LP. Finally, we give the full scheme description.

A. Mechanism Design Framework

We propose to apply problem transformation for mechanism design. The general framework is adopted from a generic approach [9], while our instantiation is completely different and novel. In this framework, the process on cloud server can be represented by algorithm **ProofGen** and the process on customer can be organized into three algorithms **(KeyGen, ProbEnc, ResultDec).** These four algorithms are summarized below and will be instantiated later.

KeyGen(lk) *{K} . This is a randomized key generation algorithm which takes a system security parameter k, and returns a secret key K that is used later by customer to encrypt the target LP problem.*

ProbEnc(K,(1)) *{(1 1. K} . This algorithm encrypts the input tuple (I) into (I) κ with the secret key K. According to problem transformation, the encrypted input (I) κ has the same form as (I), and thus defines the problem to be solved in the cloud.*

ProofGen(K) {(y, F)}. *This algorithm augments a generic solver that solves the problem (I) κ to produce both the output* **y** *and a proof* F. *The output* **y** *later decrypts to* **x,** *and* F *is used later by the customer to verify the correctness of* **y** *or* **x.**

ResultDec(K,(1), y, F) **{x,** *I}. This algorithm may choose to verify either* **y** *or* **x** *via the proof* F. *In any case, a correct output x is produced by decrypting* **y** *using the secret K. The algorithm outputs I when the validation fails, indicating the cloud server was not performing the computation faithfully.*

Note that our proposed mechanism provides us one-time-pad types of flexibility. Namely, we shall never use the same secret key *K* to two different problems.

B. Basic Techniques

Before presenting the details of our proposed mechanism, we study in this sub section a few basic techniques and show that the input encryption based on these techniques along may result in an unsatisfactory mechanism. However, the analysis will give insights on how a stronger mechanism should be designed. Note that to simplify the presentation, we assume that the cloud server honestly performs the computation, and defer the discussion on soundness to a later section.

1) Hiding equality constraints **(A, b):** First of all, a randomly generated m x m non-singular matrix Q can be part of the secret key *K.* The customer can apply the matrix to Eq. (2) for the following constraints transformation,

$$Ax = b = A'x = b'$$

where A' = QA and b' = Qb.

Since we have assumed that A has full row rank, A' must have full row rank. Without knowing Q, it is not possible for one to determine the exact elements of A. However, the null spaces of A and A' remains the same, which may violate the security requirement of some applications. The vector b is encrypted in a perfect way since it can be mapped to an arbitrary b' with a proper choice of Q.

Hiding inequality constraints (B): The customer cannot transform the inequality constraints in the similar way as used for the equality constraints. This is because for an arbitrary invertible matrix Q, Bx > **0** is not equivalent to QBx > **0** in general.

D. Result Verification

Till now, we have been assuming the server is honestly performing the computation, while being interested learning information of original **LP** problem. However, such semi-honest model is not strong enough to capture the adversary behaviors in the real world. In many cases, especially when the computation on the cloud requires a huge amount of computing resources, there exists strong financial incentives for the cloud server to be "lazy". They might either be not willing to commit service-level-agreed computing resources to save cost, or even be malicious just to sabotage any following-up computation at the customers. Since the cloud server promises to solve the **LP** problem (DK = **(A', B', b', c'),** we propose to solve the result verification problem by designing a method to verify the correctness of the solution y of (DK $_K$. The soundness condition would be a corollary thereafter when we present the whole mechanism in the next section. Note that in our design, the workload required for customers on the result verification is substantially cheaper than solving the **LP** problem on their own, which ensures the great computation savings for secure **LP** outsourcing.

The **LP** problem does not necessarily have an optimal solution. There are three cases as follows.

Normal: There is an optimal solution with finite objective value.

Infeasible: The constraints cannot be all satisfied at the same time.

Unbounded: For the standard form in Eq. (1), the objective function can be arbitrarily small while the constraints are all satisfied.

E. The Complete Mechanism Description

Based on the previous sections, the proposed mechanism for secure outsourcing of linear programming in the cloud is summarized below.

KeyGen(1^k): Let *K = (Q,M,r,A,7).* For the system initialization, the customer runs **KeyGen(1^k)** to randomly generate a secret *K,* which satisfies Eq. (4).

ProbEnc(K, (11.): With secret *K* and original LP problem (I), the customer runs **ProbEnc(K, (I))** to compute the encrypted LP problem (DK = **(A', B', b', c')** from Eq. (3).

ProofGen(K): The cloud server attempts to solve the LP problem (DK in Eq. (5) to obtain the optimal solution y. If the LP problem (DK has an optimal solution, F should indicate so and include the dual optimal solution (s, t). If the LP problem (DK is infeasible, F should indicate so and include the primal and the dual optimal solutions of the auxiliary problem in Eq. (8). If the LP problem (DK is unbounded, y should be a feasible solution of it, and F should indicate so and include the primal and the dual optimal solutions of Eq. (9), i.e. the auxiliary problem of the dual problem of (DK K.

ResultDec(K, (I), y, F): First, the customer verifies y and F according to the various cases. If they are correct, the customer computes **x** = My — r if there is an optimal solution or reports (I) to be infeasible or unbounded accordingly; otherwise the customer outputs I, indicating the cloud server was not performing the computation faithfully.

IV. SECURITY ANALYSIS

A. Analysis on Correctness and Soundness Guarantee

We give the analysis on correctness and soundness guarantee via the following two theorems.

Theorem *1: Our scheme is a correct verifiable linear programming outsourcing scheme.*

Proof: The proof consists of two steps. First, we show that for any problem (I) and its encrypted version (DK $_9$ solution y computed by honest cloud server will always be verified successfully. This follows directly from the duality theorem of linear programming. Namely, all conditions derived from duality theorem and auxiliary LP problem construction for result verification are necessary and sufficient.

Next, we show that correctly verified solution y always corresponds to the optimal solution **x** of original problem (I). For space limit, we only focus on the normal case. The reasoning for infeasible/unbounded cases follows similarly. By way of contraction, suppose **x = My — r** is not the optimized solution for (I). Then, there exists x* such that $c^T x^* <$ where Ax* = **b** and Bx* > **0.** Since x* = **My*** $_-$ **r**, it is straightforward that cTMy* _ $_e T_r$ = $_e T_{x^.}$ < $_e T_x$ = c^T My — c^Tr, where **A$_y$* = b'** and B$_y$* > **0.** Thus, y* is a better solution than y for problem (DK , which contradicts the fact that the optimality of y has been correctly verified. This completes the proof of **theorem 1.**

B. Analysis on Input and Output Privacy Guarantee

We now analyze the input and output privacy guarantee. Note that the only information that the cloud server obtains is *(DK* = **(A', B', 13', c').**

We start from the relationship between the primal problem (I) and its encrypted one $_K$. First of all, the matrix **A** and the vector b are protected perfectly. Because for V mxn matrix **A'** that has the full row rank and V n x 1 vector **b',**] a tuple **(Q, M, r)** that transforms (A, b) into (A', b'). This is straightforward since we can always find invertible matrices **Q, M** for equivalent matrices **A** and **A'** such that **A' = QAM,** and then solve r from b' = Q(b + Ar). Thus from **(A', b'),** cloud can only derive the rank and size information of original equality constraints A, but nothing else. Secondly, the information of matrix B is protected by = (B — **AQA)M.** Recall that the n x m matrix **A** in the

V. PERFORMANCE ANALYSIS

A. Theoretic Analysis

1) Customer Side Overhead: According to our mechanism, customer side computation overhead consists of key generation, problem encryption operation, and result verification, which corresponds to the three algorithms **KeyGen, ProbEnc,** and **ResultDec,** respectively. Because **KeyGen** and **ResultDec** only require a set of random matrix generation as well as vector-vector and matrix-vector multiplication, the computation complexity of these two algorithms are upper bounded via *$0(n^2)$.* Thus, it is straight-forward that the most time-consuming operations are the matrix-matrix multiplications in problem encryption algorithm **ProbEnc.** Since m < *n,* the time complexity for the customer local computation is thus asymptotically the same as matrix-matrix multiplication, i.e., *0(nP)* for some 2 < *p* < 3.

2) Server Side Overhead: For cloud server, its only computation overhead is to solve the encrypted LP problem (DK as well as generating the result proof F, both of which correspond to the algorithm **ProofGen.** If the encrypted LP problem (DK belongs to normal case, cloud server just solves it with the dual optimal solution as the result proof F, which is usually readily available in the current LP solving algorithms and incurs no additional cost for cloud (see Section III-D). If the encrypted problem (DK does not have an optimal solution, additional auxiliary LP problems can be solved to provide a proof. Because for general LP solvers, phase I method (solving the auxiliary LP) is always executed at first to determine the initial feasible solution proving the auxiliary LP with optimal solutions also introduces little additional overhead. Thus, in all the cases, the computation complexity of the cloud server is asymptotically the same as to solve a normal LP problem, which usually requires more than *$0(n^3)$* time [13].

Obviously, the customer will not spend more time to encrypt the problem and solve the problem in the cloud than to solve the problem on his own. Therefore, in theory, the proposed mechanism would allow the customer to outsource their LP problems to the cloud and gain great computation savings.

B. Experiment Results

We now assess the practical efficiency of the proposed secure and verifiable LP outsourcing scheme with experiments. We implement the proposed mechanism including both the customer and the cloud side processes in Matlab and utilize the MOSEK optimization [12] through its Matlab interface to solve the original LP problem (I) and encrypted LP problem

K. Both customer and cloud server computations in our experiment are conducted on the same workstation with an Intel Core 2 Duo processor running at 1.86 GHz with 4 GB RAM. In this way, the practical efficiency of the proposed mechanism can be assessed without a real cloud environment. We also ignore the communication latency between the customers and

VI. RELATED WORK

A. *Work on Secure Computation Outsourcing*

General secure computation outsourcing that fulfills all aforementioned requirements, such as input/output privacy and correctness/soundness guarantee has been shown feasible in theory by Gennaro et al. [9]. However, it is currently not practical due to its huge computation complexity. Instead of outsourcing general functions, in the security community, Atallah et al. explore a list of work [5], [7], [8], [10] for securely outsourcing specific applications.

B. Work on Secure Multiparty Computation

Another large existing list of work that relates to (but is also significantly different from) ours is Secure Multi-party Computation (SMC), first introduced by Yao [11] and later extended by Goldreich et al. [14] and many others.

C. Work on Delegating Computation and Cheating Detection

Detecting the unfaithful behaviors for computation outsourcing is not an easy task, even without consideration of input/output privacy. Verifiable computation delegation, where a computationally weak customer can verify the correctness of the delegated computation results from a powerful but untrusted server without investing too much resources, has found great interests in theoretical computer science community. The customer can then use the commitment combined with a sampling approach to carry out the result verification (without re-doing much of the outsourced work.)

VII. CONCLUDING REMARKS

In this paper, for the first time, we formalize the problem of securely outsourcing LP computations in cloud computing, and provide such a practical mechanism design which fulfills input/output privacy, cheating resilience, and efficiency. By explicitly decomposing LP computation outsourcing into public LP solvers and private data, our mechanism design is able to explore appropriate security/efficiency tradeoffs via higher level LP computation than the general circuit representation. We develop problem transformation techniques that enable customers to secretly transform the original LP into some arbitrary one while protecting sensitive input/output information. We also investigate duality theorem and derive a set of necessary and sufficient condition for result verification. Such a cheating resilience design can be bundled in the overall mechanism with close-to-zero additional overhead. Both security analysis and experiment results demonstrates the immediate practicality of the proposed mechanism.

We plan to investigate some interesting future work as follows: 1) devise robust algorithms to achieve numerical stability; 2) explore the sparsity structure of problem for further efficiency improvement; 3) establish formal security framework; 4) extend our result to non-linear programming computation outsourcing in cloud.

REFERENCES

[1] M. J. Attalla and J. Li, "Secure outsourcing of sequence comparisons," Int. J. Inf. Sec., vol. 4, no. 4, pp. 277–287, 2005.

[2] N. Gaurha and Dr. Michael, "Data Security in Cloud Computing Using Linear Programming", International Journal of Emerging Technology and Advanced Engineering, Vol. 2, Issue 7, 2012.

[3] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in Proc. of TCC, pp. 264–282, 2005.

[4] C. Wang, K. Pen & J. Wang, "Secure and Practical Outsourcing of Linear Programming in Cloud Computing," IEEE transactions on cloud computing , 2011.

[5] M Haungs, R Pandey, E Barr, and J. Barnes, "A fast connection-time redirection mechanism for internet application scalability", in *Proc. of STOC'87*, pp. 1–6, 1987.

[6] S Setty, R McPherson, J. Blumberg, and M Walfish, "Making Argument Systems for Outsourced Computation Practical", in *Proc. Of CRYPTO'10*, 2010.

[7] M Guirguis, A Bestavros, I Matta and Y Zhang, "Reduction of Quality (RoQ) Attacks on Internet End-Systems", in *Proc. of ASIACCS*, pp. 48–59, 2010.

[8] K Hashizume, D Rosado, E Medina and E B Fernandez, "An analysis of security issues for cloud computing", Journal of Internet Services and Applications 2013, 4:5

[9] Ramanna, "DFA-Based Functional Encryption: Adaptive Security from Dual System Encryption", in *Proc. of ICDCS'10*, 2010.

[10] Jiangtao & Mikhail," Secure and private collaborative linear programming", IEEE 2006.

[11] Andrew , "Protocols for Secure Computations", IEEE, 2007.

[12] M Shrivastava, "Security in Cloud Computing Using Linear Programming" , *International Journal Of Emerging Technology And Advanced Engineering (IJETAE)*, 2012.

[13] B Martini and R Choo, "An integrated conceptual digital forensic framework for cloud computing", Elsevier Ltd, 2012.

[14] S Qaisar and K Khawaja, "Cloud Computing: Network/Security Threats And Counter measures", Interdisciplinary Journal Of Contemporary Research In Business, Vol. 3, 2012.

[15] L Marchegiani, Enzo and Peruffo, "Revitalising the Outsourcing Discourse with in the Boundaries of Firms Debate", Business Systems Review, Vol.1,2012