

Improving Practical Performance on Secure and Private Collaborative Linear Programming

Rafael Deitos and Florian Kerschbaum
SAP Research
Karlsruhe, Germany
{r.deitos, florian.kerschbaum}@sap.com

Abstract—Although information sharing in supply chain management dramatically reduces costs, parties are reluctant to do so, mainly due to fear of information leakage. Solutions based on secure multi-party computation guarantee privacy and correctness, at an extra communication and computation cost. Efficiency is a major concern for practical solutions. The best known solution with public indexes selection is due to Li and Atallah. It has the drawback of requiring a secure permutation on every round of the protocol, which turns to be very expensive. We therefore, propose a probability-based technique to reduce the number of secure permutations required by such solution, improving overall practical performance.

Keywords-secure linear programming; secure multi-party computation; supply-chain optimization

I. INTRODUCTION

The problem of protecting against the business partner is an emerging security trend. While in the past business partners were separated by physical boundaries and transactions were only bound to paper and goods, now the business partner may have access to internal systems [1]. Studies revealed that several security incidents involve business partners and this is a significant growing trend over the last five years [2]. Examples of networked business collaboration include vendor-managed inventory or virtual organizations for business web services [3].

Information asymmetry is one of the major sources of inefficiency in managing supply chains (SC) [4]. Effects include wrong investment in capacity, misallocation of resources, the *Bullwhip Effect* [5], reduced customer service, and unnecessary additional costs. Information sharing about inventory levels, sales, demand forecasts, etc. in SC management (SCM) can dramatically improve the performance and reduce costs [6]. The cause for such improvement is not information sharing, *per se*, but, rather, because shared information improves decision-making [4]. Consider the case of SC optimization [8] where a certain number of companies want to collaboratively plan production, transportation and warehousing. Information sharing, i.e. a centrally computed plan, leads to a global optimum compared to the local optima achieved by individual planning and then forwarding orders (upstream planning). It can therefore significantly reduce overall costs and avoid the negative side-effects of informa-

tion asymmetry. However, the necessary data to be shared for optimality includes production costs and capacities whose revelation negatively impacts the negotiation position.

Linear programming (LP) [9] provides the means to mathematically formulate and efficiently solve real-world problems. Technically, SC optimization can be modeled as LP [10]. To determine a SC wide master plan (SCMP) minimizing total cost of fulfilling final customer demand [11], [12], a deterministic, multi-period mixed integer programming model as in [10] can be employed. The most popular method for solving LP problems is the simplex [9]. Nevertheless, traditional LP does not consider security, i.e. privacy of the input. Fear that a SC partner will take advantage of private information and/or fear that such information will leak to a competitor are sufficient reasons to explain the reluctance of the partners in sharing private information, even when sharing would bring positive results for both sides. It is therefore necessary to securely compute LP.

Reputation systems or sticky policies could be used, but they offer no guarantees [1]. Secure Multi-Party Computation (SMC) can help solving this issue by realizing protocols with guaranteed security [13], [14]. However, practicality is a major concern for such protocols. Regarding running time, for example, [15] reports that the distributed solution runs 60,000 times slower than local execution. Computing secure LP has two main research directions. The basic different is how the pivot index is hidden. The work from Li and Atallah [17] is the best known protocol for public indexes selection, but it has problems.

Improving performance of the solution from [17] is the focus of this work. Section II contains a detailed description of its secure LP protocol and identifies two basic efficiency hindrances namely two-party scenario limitation and need for permutation on every iteration of the protocol. Extensions to the multi-party scenario and further reduction of the complexity were already addressed by the authors in [1]. However, even using algorithms with reduced round complexity, the costs of permuting on every iteration represent a major performance restriction. Section III introduces a probability-based scheme to reduce overall number of permutations significantly improving practical performance. Finally, last considerations and conclusions are discussed on Section IV.

II. STATE OF THE ART

The simplex algorithm is the most common method for solving LP problems (Please refer to [9] for details on the algorithm). A concrete example on how to model a SCMP using LP is given in [10]. The objective function and the constraints are organized in a matrix, in a tableau form, and the process mainly consists of an iteration on the main step of pivoting. An element of such matrix is selected according to some optimization rule and then included in a computation with every other element of the matrix to form a new matrix. The process repeats until an optimal solution is found, given the problem is bounded and a solution exists.

The literature for privacy preserving LP applied to SCM is rather scarce. However, two contributions for secure LP protocols in collaborative SCM are of interest. The main difference regards the way the index of the pivot element is hidden. Toft [16] introduced primitives for secure multi-party computation based on secret variables as well as high level protocols for secure LP. The index is kept as an encrypted variable and the protocols are proved to be secure against passive adversaries. However, indexing on secret variables is difficult and very expensive since every operation must be on the size of the entire array (or matrix) which is also very communication intensive. Li and Atallah [17] proposed two-party secure and private collaborative LP protocols where the matrix is permuted by each and every party, such that no one knows the entire permutation and therefore, the index of the pivot element can be chosen publicly because it is a random choice in the permuted matrix.

A. Secure and Private Collaborative Linear Programming

The security model from [17] considers *honest-but-curious* adversaries, i.e. they follow the protocol and attempt to learn more information than allowed from the execution. The problem of secure collaborative LP is defined as

$$\begin{array}{l} \text{minimizes } c^T \cdot x = -z_0 \\ \text{subject to } A \cdot x = b, x \geq 0 \end{array} \Rightarrow D = \begin{bmatrix} c^T & -z_0 \\ A & b \end{bmatrix}$$

where A is a $m \times n$ matrix, b is an m -dimensional vector, c and x are n -dimensional vectors, and the T stands for transpose. The system matrix D is specified in the canonical form of a LP [9], and it, and consequently A , b , c and z_0 , are all additively shared or split among two parties, Alice and Bob (refer to [17] for details).

D is an $(m+1) \times (n+1)$ matrix. The secure LP from [17] is based on the simplex algorithm from [9] with the difference that D is additively split by Alice and Bob. It can be summarized as a sequence of the following steps: (i) blind-and-permute; (ii) find incoming variable; (iii) test for unboundedness; (iv) find outgoing variable; (v) pivot; and (vi) output the result.

Performing the pivot operation requires both parties to know the index from the matrix in which to pivot. However, knowing such index may leak information about the original matrix unnecessarily. Take, for example, the case where an index is selected twice. One can exclude certain matrices as possible inputs by observing repetitions. Therefore in order to not reveal additional information about the matrices, repetitions need to be hidden, and a *blind-and-permute* protocol is defined. Consider Alice and Bob additively split D and each have private permutations for columns and rows. They jointly permute D in a way that the final permutation is known to neither of them, and in the end, the permuted matrix \bar{D} is, again, additively split using different randomness. Now, it no longer matters if they know a certain index in \bar{D} because there is no way of relating that index position to the corresponding position in the original matrix D . This final permutation is jointly computed so that the solution of the original system is kept. The *blind-and-permute* protocol is executed before every pivot operation and takes $\mathcal{O}(mn)$ modular exponentiations. The cost of the matrix pivoting protocol depends on the split multiplication protocol from [21] and is bounded by $\mathcal{O}(mn)$ modular exponentiations. Preventing information leakage by permuting D causes the indexes of the basic variables, which are responsible for determining the solution of the LP, to be permuted as well. Determining the solution of the LP includes recovering the original indexes of the basic variables. The cost of the indexes recovering protocol is $\mathcal{O}(kn)$ modular exponentiations, where k is the number of permutations applied, equal to the number of iterations.

Overall, the cost of the secure LP protocol is about $\mathcal{O}(kmn + kln + klm + lmn)$ modular exponentiations. Since $n \geq m$, the total cost becomes $\mathcal{O}(k(mn + ln) + lmn)$ [17]. SC optimization often considers LP problems of significant size, e.g. a medium problem has a number of constraints m of around 2000 [9]. Due to the dependency on the number of iterations k , invoking the *blind-and-permute* protocol on every iteration is very expensive.

III. IMPROVING SECURE LP: REDUCING THE NUMBER OF PERMUTATIONS

In the protocol from [17], privacy is partly guaranteed by the use of additively split data, which enables the selection of indexes for the pivot operation to be made publicly. Nevertheless, public selection reveals the position of the pivot element and therefore, disclose information. Furthermore, every row corresponds to a single constraint and every column to a certain coefficient, and if a row or column index is selected twice, information is disclosed as well. Enforcing the *blind-and-permute* protocol on *every iteration* and invoking the *indexes recovering* protocol after the last one, guarantees randomization of the indexes before every selection so that no information can be gained, preserving privacy of the input.

The cost of invoking the permutation protocol has already been discussed in [1], and represent an important parcel of the overall cost of the secure LP protocol. As an example, it has the same cost than the pivot operation, i.e., $\mathcal{O}(kmn)$, which is the most expensive operation. The number of iterations for solving a given LP can not be changed, i.e. it depends on the problem and varies according to it. The relation between number of iterations and number of permutations, i.e. permuting on *every* iteration, constitutes a major performance issue which may be avoided. We propose a probability-based solution for reducing the overall number of permutations at the cost of disclosing such number, which is now different and smaller than the number of iterations, opposing to Li and Atallah’s proposal.

There are three main sources of information leakage when considering public pivot index selection: selection of two or more pivot elements (i) in the same row; (ii) in the same column; or (iii) in the exact same position. Solving a LP with simplex may yield a number of iterations greater than the number of variables, or constants, which makes difficult to guarantee no indexes repetitions. The selection of two consecutive pivot elements in the same column cannot occur in the traditional simplex, since the tableau includes the columns of the basic variables and the algorithm enforces all elements in the pivot column to be zero after the pivot operation. However, rows can be selected twice, consecutively.

To reduce the number of permutations it is necessary to determine the probability distribution of rows and columns repetition in an execution of the simplex algorithm. Note that case (iii) is a combination of cases (i) and (ii). The procedure comprises design and implementation of a generic SCMP problem simulator. It generates a random SC, i.e. with a specific number of stages, nodes and links, and automatically models the problem (mathematically), according to the LP theory [10]. The resulting LP, in the canonical form, is the input for a simplex tool, and the output is the optimal values for the given objective function, i.e. minimize the overall cost of the SCMP. Several SC configurations were simulated, covering a considerable range of iterations, e.g. from 3 to 6 stages with 3 to 30 nodes. Figure 1 depicts the rows or columns index repetition distribution probability as an average of 100 problems for each of the 3 stages case configuration.

The number of repetitions r , including rows or columns, ranged from 0 to 84, for that case. The probability of low index repetition frequency decreases with increasing number of iterations, e.g. $P(r = 0)$. On the contrary, the probability of higher index repetition frequencies increases with increasing number of iterations, e.g. $P(r = 84)$. Moreover, the rate in which the number of rows or columns indexes repetition grows is smaller than the number of iterations grow. The probability of repeating rows or columns do not exceed 10%, except for $r = 0$, which is the no repetition case.

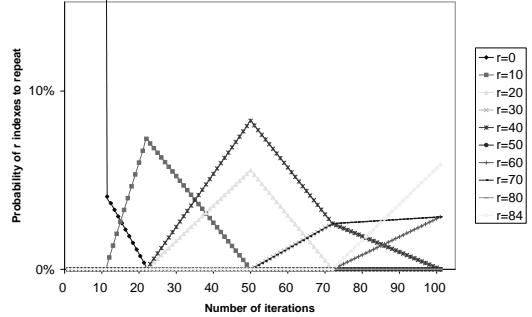


Figure 1. Probability distribution of row or column index repetition

The solution from [17] uses a performance restrictive policy regarding the number of permutations. The costs could be reduced if permuting before the first iteration, and then only when a row or column index is to be repeated. However, this approach would leak the exact number of repetitions, which may be used to infer about the original matrix, i.e. disclose extra information. The privacy requirements demand a more elaborated solution.

A. Concept

We introduce the concept of a *coin flip* to preserve the privacy of the input and reduce the number of permutations during the secure LP protocol execution. The coin is flipped before every iteration with a given probability and decides if the protocol will or will not permute the system matrix. Tossing the coin is performed jointly using a standard common coin-tossing protocol, i.e. no party can determine the outcome of the coin toss by itself. Examples for such protocol can be found in [22]. Assume two possible reasons for permuting: (i) an index is to be repeated (requirement from LP problem); and (ii) the coin flip decided to do so (requirement artificially introduced). The probability distribution for (i) can be extracted from practical experiments (Figure 1), and the probability distribution for (ii) is mathematically estimated.

Parties collaboratively computing secure LP are able to observe two output variables: (i) the number of iterations; and (ii) the number of permutations (if different than (i)). This is inevitable. If the coin flip probability is estimated such that a party can not distinguish the reason of a given permutation, i.e. row or column index repetition or coin flip decision, no additional information can be inferred regarding the real number of repeated indexes. The smaller the probability, the lower is the number of permutations, and consequently, the better is the overall performance of the privacy preserving LP protocol. Although it is impossible to distinguish the reason for the permutations, little information can still be gained, i.e. the number of permutations (or its complement, the number of iterations without permutations), but less than if no coin is being flipped.

B. Mathematical Analysis

The probability distribution of rows or columns indexes repetition is already known (Figure 1). We want to estimate the smallest probability for the coin flip which guarantees the privacy of the input, i.e. improve the performance of secure LP with less information as possible being revealed. We introduce the following measurement idea: *Quantify the number of bits needed to represent all the possible matrices obtained by the observation of a certain number of permutations and iterations during the computation of any given LP, defined as NoB* . The value of NoB must be maximum when permuting on every iteration, i.e. as less information as possible can be obtained, and minimal when only permuting on rows or columns indexes repetitions. There is a point where the combination of both probabilities, i.e. coin flip and indexes repetitions, results on a NoB value that is as close as possible to the maximum. We argue that such point is considerably smaller than the number of iterations, which yields a reduction on the number of permutation and hence, an improvement on the overall performance of any secure LP protocol with public index selection.

The probability of having m permutations and l iterations is formally represented by $P(pr = m, i = l) = \sum_{n=0}^m P(r = n, i = l) \cdot P(c = m - n, i = l)$, where $P(r = n, i = l)$ is the probability of n repetitions (due to rows or columns indexes repetition) and l iterations, obtained from experimental results. The probability of permutations due to coin flip decision and l iterations is determined by $P(c = m - r, i = l)$. The term $P(c = m - r, i = l)$ is defined as a function of the coin flip probability, number of iterations, repetitions and permutations. It can be formally expressed by $P(c = m - n, i = l) = C_{m-n}^{l-n} \cdot p^{m-n} \cdot (1-p)^{l-m}$, where C_{m-n}^{l-n} stands for combination, l is the iteration number, m is the permutation number, n is the repetition number and p stands for the probability of the coin flip.

$P(c = m - n, i = l)$ depends on three terms. (i) The combination of possible remaining places that a permutation could occur, excluding the ones where it has already happened, due to an index repetition or a coin flip decision (C_{m-n}^{l-n}). (ii) The coin flip probability itself. And (iii) the probability that the coin flip indicates that no permutation is needed. Generally speaking, the larger the number of repetitions or permutations, the lower the chances of having more permutations due to coin flip. It is also expected that as the probability p increases, the probability $P(c = m - r, i = l)$ increases.

Putting all the information together, the number of bits necessary to represent the matrices obtained by the observation of a certain number of permutations and iterations is quantified by $NoB = \sum_{m=0}^{l_{max}} \sum_{l=0}^{l_{max}} P(pr = m, i = l) \cdot \log \sum_{n=0}^m N \cdot P(r = n, i = l)$, where N is the number of bits necessary to represent the system matrix. The value of NoB is expected to be maximal when $p = 1$, which means

that the coin flip indicates permutation on every iteration. On the other hand, it is expected that NoB assume its minimum value when $p = 0$, i.e. all the permutations are caused by row or column indexes repetitions.

Next, NoB needs to be validated. As an example, consider two cases for the number of iterations l : (i) small l ; and (ii) large l . In case (i), for a fixed small l , the probability of repetitions to occur, $P(r = n, i = l)$, decreases if the number of repetition increases. In case (ii), for a fixed large l , $P(r = n, i = l)$ increases if the number of repetition increases. In both cases NoB outputs as expected.

After validation, we computed NoB using experimental values for $P(r = n, i = l)$ and variations on the coin flip probability p . Assume $N = 8$ (number of bits to represent the system matrix) and a number of iterations $0 \leq l \leq l_{max}$, where $l_{max} = 101$ with step size 1 (see Figure 1). The coin flip probability range is $0 \leq p \leq 1$, with a step size of 0.1. Figure 2 shows the information leakage, in percentage, for any given coin flip probability. The information leakage is a function defined as the inverse of NoB .

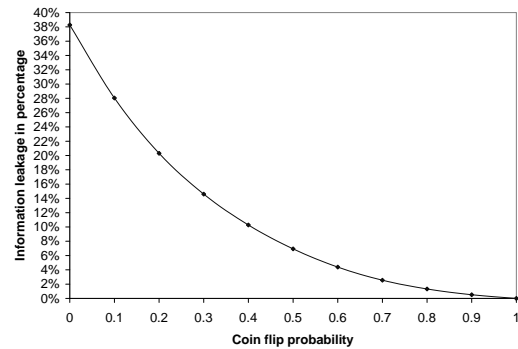


Figure 2. Percentage of information leakage due to index repetition

NoB increases when the probability of the coin flip increases. The higher the permutation probability (due to indexes repetitions or coin flip), the more difficult is to represent the matrices obtained by the observation of a certain number of permutations and iterations. Less information is revealed (see Figure 2) and therefore, privacy of the input is preserved. If $p = 1$, i.e. coin flip demands permutations on every iteration, i.e. equivalent to [17], no information is leaked. On the other hand, if $p = 0$, i.e. all the permutations are triggered by index repetitions, NoB is minimal and around 38% of information is leaked (from Figure 2).

Experiments revealed that with a coin flip probability of $0.6 \geq p \geq 0.7$, NoB reaches a value that is very close to the value when $p = 1$. It is not necessary to permute on every iteration l , but with a frequency of approximately 60% of l , the privacy of the input is preserved. The introduction of the coin flip concept reduces the constants of the permutation protocol ($\sim 40\%$), and hence, improves the overall performance, in practice. Ultimately, the privacy can be controlled by the coin flip probability, set off line.

IV. CONCLUSION

We considered the problem of secure and private collaborative linear programming for supply chain management, e.g. SCMP problem, and its performance issues regarding number of permutations. The best known solution for the public selection of the pivot index is [17], and it has a drawback of requiring a *blind-and-permute* protocol on every iteration of the simplex algorithm.

We introduced a scheme, based on the probability of a coin flip, to reduce the number of permutations. Such scheme is able to control the relation between performance and privacy of a secure simplex protocol, regarding the number of permutations. It is also able to reduce about 40% of such number, which definitely improves performance of the protocol with guaranteed security.

Moreover, we realize that this work can be extended in several directions. The exact speed up that the whole private and collaborative secure linear programming algorithm would gain with the introduction of the flip coin scheme could then be explored in future work.

ACKNOWLEDGMENT

The developments presented in this paper were partly funded by the European Commission through the ICT program under Framework 7 grant FP7-213531 to the SecureSCM project.

REFERENCES

- [1] F. Kerschbaum and R. J. Deitos, "Security against the business partner," in *SWS '08: Proceedings of the 2008 ACM workshop on Secure web services*. New York, NY, USA: ACM, 2008, pp. 1–10.
- [2] W. Baker, D. Hylender, and A. Valentine, "2008 data breach investigations report," Rep., 2008.
- [3] R. Deitos, F. Kerschbaum, and P. Robinson, "A comprehensive security architecture for dynamic, web service based virtual organizations for businesses," in *SWS '06: Proceedings of the 3rd ACM workshop on Secure web services*. New York, NY, USA: ACM, 2006, pp. 103–104.
- [4] V. Deshpande, M. Atallah, M. Blanton, K. Frikken, J. Li, and L. Schwarz, "Secure collaborative planning, forecasting, and replenishment," Tech. Rep., 2006.
- [5] H. Lee, P. Padmanabhan, and S. Whang, "The bullwhip effect in supply chains," *Sloan Management Review*, vol. 38, pp. 93–102, 1997.
- [6] G. Cachon and M. Fisher, "Supply chain inventory management and the value of shared information," *Management Science*, vol. 46, no. 8, pp. 1032–1048, 2000.
- [7] H. L. Lee and S. Whang, "Information sharing in a supply chain," *International Journal of Manufacturing Technology and Management*, vol. 1, pp. 79–93, 2000.
- [8] R. Pibernik and E. Sucky, "Centralised and decentralised supply chain planning," *International Journal of Integrated Supply Management*, vol. 2, no. 1/2, pp. 6–27, 2006.
- [9] G. B. Dantzig and M. N. Thapa, *Linear Programming 1: Introduction*, P. Glynn, Ed. Springer-Verlag New York, LLC, 1997.
- [10] R. Pibernik and E. Sucky, "An approach to inter-domain master planning in supply chains," *International Journal of Production Economics*, vol. 108, no. 1-2, pp. 200–212, July 2007.
- [11] A. J. Clark and H. Scarf, "Optimal policies for a multi-echelon inventory problem," *Manage. Sci.*, vol. 50, no. 12 Supplement, pp. 1782–1790, 2004.
- [12] E. A. Silver, D. F. Pyke, and R. Peterson, *Inventory Management and Production Planning and Scheduling, 3rd Edition*, 1998.
- [13] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*. New York, NY, USA: ACM, 1987, pp. 218–229.
- [14] A. C. Yao, "Protocols for secure computations," *Proceedings of the 23rd Annual IEEE Symposium on Foundations*, 1982.
- [15] F. Kerschbaum, D. Dahlmeier, A. Schröpfer, and D. Biswas, "On the practical importance of communication complexity for secure multi-party computation protocols," *ACM Symposium on Applied Computing*, 2009.
- [16] T. Toft, "Primitives and applications for multi-party computation," Ph.D. dissertation, University of Aarhus, 2007.
- [17] J. Li and M. J. Atallah, "Secure and private collaborative linear programming," *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2006.
- [18] I. B. Damgard and M. J. Jurik, "A length-flexible threshold cryptosystem with applications," in *In proceedings of ACISP'03, LNCS series*, vol. 2727, 2003, pp. 350–364.
- [19] D. Boneh and M. Franklin, "Efficient generation of shared rsa keys," in *In Advances in Cryptology—CRYPTO 97*, vol. 1294, 1997, pp. 425–439.
- [20] N. Gilboa, "Two party rsa key generation," in *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1999, pp. 116–129.
- [21] K. B. Frikken and M. J. Atallah, "Privacy preserving route planning," in *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. New York, NY, USA: ACM, 2004, pp. 8–15.
- [22] B. Schneier, *Applied Cryptography*. John Wiley & Sons, 1996.