Automatic Protocol Selection in Secure Two-Party Computations

Florian Kerschbaum
SAP Research
Karlsruhe, Germany
florian.kerschbaum@sap.com

Thomas Schneider
EC SPRIDE, TU Darmstadt
Darmstadt, Germany
thomas.schneider@ec-spride.de

Axel Schröpfer
SAP Research
Karlsruhe, Germany
axel.schroepfer@sap.com

Abstract

Performance of secure computation is still often an obstacle to its practical adoption. There are different protocols for secure computation that compete for the best performance. In this paper we propose automatic protocol selection *which selects a protocol for each operation resulting in a mix with the best performance so far. Based on an elaborate performance model, we propose an optimization algorithm and an efficient heuristic for this selection problem. We show that our mixed protocols achieve the best performance on a set of use cases. Furthermore, our results underpin that the selection problem is so complicated, that a programmer is unlikely to manually make the optimal selection. Our proposed algorithms nevertheless can be integrated into a compiler in order to yield the best (or near-optimal) performance.*

1 Introduction

Secure two-party computation allows two parties to compute a function f over their joint, private inputs x and y, respectively without revealing their private inputs or relying on a trusted third party. Afterwards, no party can infer anything about the other party's input except what can be inferred from her own input and the output $f(x, y)$. Secure computation has many applications, e.g., in the financial sector, and has been successfully deployed in commercial and industrial settings [4].

Performance is still often an obstacle to practical adoption of secure computation, even in the widely used semi-honest security model. A number of protocols compete for the best performance in this model. Recently, the garbled circuit implementation of [3] has achieved top performance in a number of applications, but still garbled circuits have some inherent limitations, e.g., due to the large circuit size of some functionalities such as multiplication. In this paper we propose a different approach. Instead of relying on a single protocol we mix protocols. Then, based on an extended performance model we *automatically* select the best protocol for a sub-operation. In all prior works this selection has been performed manually, e.g., [1, 2]. We present two algorithms for the protocol selection problem – an optimization based on integer programming and a heuristic.

In our work we test three hypotheses:

- Our mixed protocols are faster than a pure garbled circuit implementation.
- The results of our heuristic and the optimum found by integer programming are close.
- The protocol selection problem is too complicated to be solved manually by the programmer.

2 Secure Computation Protocols

We integrate two protocols for performing secure two-party computations – garbled circuits and additively homomorphic encryption where each variable x of bit length l is shared between the two parties such that $x = x_A + x_B \bmod 2^l$. Both protocols are generic, i.e., they can securely implement any ideal functionality. Nevertheless they have different performance characteristics as shown by the performance evaluations in [2, 8].

Our methods to convert between garbled circuits and homomorphic encryption are similar to those of previous works [2, 5], but more efficient as we use shorter random masks. To use x as input for a garbled circuit, we extend the inputs of the garbled circuit computing f with an l-bit addition circuit to which A provides input x_A and B provides input x_B, i.e., the slightly larger garbled circuit computes $f(\ldots, x_A + x_B \bmod 2^l, \ldots)$. Similarly, we can convert the output z of a sub-functionality that has been computed using garbled circuits into secret shares z_A, z_B that can later on be used for secure computations using homomorphic encryption. For this, we extend the output of the garbled circuit with an l-bit subtraction circuit whose subtrahend is a randomly chosen l-bit value z_A provided by A. We modify the garbled circuit protocol such that only B obtains the output $z_B = z - z_A$.

3 Cost Model

In order to choose which operation to implement using which protocol we need to compare their costs. By cost we mean the (wall clock) run-time of the protocol. Since the protocol can be composed from sub-protocols of both protocol types – garbled circuits and homomorphic encryption – we need to be able to assess their performance while taking care of additional conversion costs. We base our cost model on the model of [8] which can (reasonably) reliably forecast the protocol run-time for both types of protocols. The accuracy of the forecast mainly determines the effectiveness of our approach.

4 Optimal Partitioning

Given the cost model described we can define the problem of an optimal partitioning of the operations into the protocol types. Consider a compiler that translates a programming language into the intermediate language described in citeSKM11. In order to construct a cost-optimal (i.e., the fastest) protocol it needs to assign each operation of the intermediate language a protocol type, also considering the conversion costs.

We setup the problem formulation as follows. Let the elements x_i correspond to the left hand-side variable assigned in an operation. We denote with $\mathbb{X}$ the set of these elements (variables). The operator mapping function op maps x_i to the right hand-side operators of that operation. The cost function $a(x_i)$ corresponds to the costs for computing x_i using garbled circuits and $b(x_i)$ to the costs using homomorphic encryption, respectively. The cost functions $c(x_i)$ and $d(x_i)$ correspond to the costs for converting x_i from homomorphic encryption to garbled circuits and vice-versa, respectively. The set $\mathbb{Y} \subseteq \mathbb{X}$ of instructions will be implemented using garbled circuits; the set $\mathbb{X} \setminus \mathbb{Y}$ using homomorphic encryption. We formally define the problem as follows:

Definition 1 (Problem Definition) Let there be a set $\mathbb{X}$ of elements $x_1, \ldots, x_n$. Let there be a function $op(x_i)$ mapping x_i to a set $\mathbb{F}_i \subseteq \mathbb{X}$. Let there be four cost functions $a(x_i)$, $b(x_i)$, $c(x_i)$, and $d(x_i)$. Find the subset $\mathbb{Y} \subseteq \mathbb{X}$ that optimizes the following cost function

$$\begin{array}{c}\sum_{\{x|x\in\mathbb{Y}\}} a(x) + \sum_{\{x|x\in\mathbb{X}\setminus\mathbb{Y}\}} b(x) + \\ \sum_{\{x|x\in\mathbb{X}\setminus\mathbb{Y},\exists y.y\in\mathbb{Y},x\in op(y)\}} c(x) + \\ \sum_{\{x|x\in\mathbb{Y},\exists y.y\in\mathbb{X}\setminus\mathbb{Y},x\in op(y)\}} d(x).\end{array}$$

We search for the best solution to the partitioning problem defined above using an optimization algorithm. $0,1$-integer programming is a suitable candidate. In $0,1$ integer programming there are variables $\vec{z}$ for which an assignment is sought which minimizes a linear objective function $c(z)^T \vec{z}$ subject to certain constraints.

Integer programming is NP-complete and can become very slow for large instances. We therefore also implement a heuristic using a greedy algorithm. We start with all operations executed as garbled circuits. Then we consecutively scan each operation in a loop. If the overall cost decreases when converting this operation to homomorphic encryption we do so. We repeat until no more operations are converted.

5 Use Cases

In order to validate the complexity of manual partitioning and the cost advantage of our algorithmic approach, we consider three use cases for secure computation from the literature: joint economic-lot-size, biometric identification, and data mining. The secure joint economic lot-size problem describes a two-party scenario between a vendor and a buyer of a product. Both try to agree on a joint lot-size q for production and shipping. As described in [7], the confidentiality-preserving computation of q^* can be reduced to secure division. As our use case we consider both division algorithms, the Newton-Raphson variant described in [8] and the long division variant described in [7].

Comparing and matching biometric data is a highly privacy-sensitive task in systems that are widely used in law enforcement, including fingerprint-, iris-, and face-recognition systems. These biometric identification systems contain two phases. A first distance-computation phase calculates distances between the client's information (a vector of M samples) and the N entries (resp., their vectors) in the database. A second matching phase determines the ϵ-closest database entry, i.e., the entry that has the minimal distance in a maximum range ϵ comparing to the biometric information of the client. As our use case we consider an algorithm for biometric identification, computing the distances using Euclidean distance as metric which is commonly used for fingerprints and faces.

A well known algorithm for decision tree learning is the ID3 algorithm. A privacy-preserving classification variant of ID3 – described in [6] as one of the first privacy-preserving data mining algorithms – enables new applications where multiple private databases can be used to act as training set (e.g., medical databases). The authors of [6] use entropy to compute the best attributes, with the privacy-preserving computation of the natural logarithm as the basis operation. As our use case we consider an algorithm to compute the natural logarithm.

6 Evaluation

In all experimental settings, both partitioning mechanisms for computing optimal mixed protocols result in the best performance, including the previously mentioned four

pure garbled circuit cases. In 8 of 24 settings, the mixed protocols result in an average of 20% less runtime. The largest improvement is 31% lower runtime compared to the protocol entirely implemented as garbled circuit (Euclidean distance, short-term security, WAN).

Both optimization approaches result in mixed protocols that perform, in almost half of all experimental settings, noticeably better than pure protocols. While the heuristic only requires seconds to compute the partitioning per use case and setting, the integer program requires several hours using the LP solver SoPlex[1] on the aforementioned server hardware. The heuristic, in comparison to the integer program, tends to reduce the number of blocks of consecutive operations with the same protocol type. For long division and natural logarithm, over all settings, the ratio between number of blocks and number of operations is less than 0.025, while it is more than 0.279 (i.e., larger by a factor of 10) for the integer program.

Mixed protocols are heavily fragmented in order to achieve the optimal performance results. We obtain a wide spectrum of fragmentations. Although there seem to be patterns in some areas, it is difficult to infer a general conclusion that can be used to manually derive a partitioning with similar performance. Even unrolled operation blocks that are identical on the operation level, result in different partitionings within the same setting and use case. One such example is the natural logarithm; operations that are part of the main loop last from the middle of the algorithm until the (third) last operation. One could assume that there would be a rather intuitive relation between single operations in the intermediate language and both types of discussed protocols. Intuitively, for shared values (which we designed to be part of the homomorphic encryption model), operations can be assumed to be fast, if they are executed as local operations that do not use cryptographic algorithms (e.g., addition or multiplication by a constant). Similarly, garbled circuits could be supposed to perform faster than homomorphic encryption for comparing two secret values. To the contraty, we show that the relations are rather complex. For Newton-Raphson and short-term security, both algorithms assign the majority of subtraction operations to homomorphic encryption, since these operations can be implemented locally without communication. In contrast, for long division in the same security setting, both algorithms assign the majority of subtraction operation to garbled circuits.

7 Conclusions

In this paper we have presented algorithms for the automatic selection of a protocol – garbled circuits or homomorphic encryption – in secure two-party computation. Based on a performance model our algorithms minimize the costs of a mixed protocol. We present an evaluation based on three use cases from the literature: secure joint economic lot-size, biometric identification, and data mining. Our results support that mixed protocols perform better than pure garbled circuit implementations. Our results also support that our heuristic is close to the optimization algorithm based on integer programming. Furthermore, our detailed analysis of the experiments also revealed that there is no discernible pattern of the selection.

Acknowledgements This work was supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE and by the Hessian LOEWE excellence initiative within CASED.

[1] version 1.6.0, available at http://soplex.zib.de/

References

[1] M. Blanton, P. Gasti. Secure and Efficient Protocols for Iris and Fingerprint Identification. *European Symposium on Research in Computer Security (ESORICS)*, 2011.

[2] W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, I. Wehrenberg. TASTY: Tool for Automating Secure Two-partY computations. *ACM Computer and Communications Security (CCS)*, 2010.

[3] Y. Huang, D. Evans, J. Katz, L. Malka. Faster Secure Two-Party Computation Using Garbled Circuits. *USENIX Security Symposium*, 2011.

[4] F. Kerschbaum, A. Schröpfer, A. Zilli, R. Pibernik, O. Catrina, S. de Hoogh, B. Schoenmakers, S. Cimato, E. Damiani. Secure Collaborative Supply Chain Management. *IEEE Computer 44 (9)*, 2011.

[5] V. Kolesnikov, A.-R. Sadeghi, T. Schneider. Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima. *Cryptology And Network Security (CANS)*, 2009.

[6] Y. Lindell, B. Pinkas. Privacy Preserving Data Mining. *Journal of Cryptology 15(3)*, 2002.

[7] R. Pibernik, Y. Zhang, F. Kerschbaum, A. Schröpfer. Secure Collaborative Supply Chain Planning and Inverse Optimization - The JELS Model. *European Journal of Operational Research (EJOR) 208(1)*, 2011.

[8] A. Schröpfer, F. Kerschbaum. Forecasting Run-Times of Secure Two-Party Computation. *International Conference on Quantitative Evaluation of Systems (QEST)*, 2011.

[9] A. Schröpfer, F. Kerschbaum, G. Müller. L1 - An Intermediate Language for Mixed-Protocol Secure Computation. *IEEE Computer Software and Applications Conference (COMPSAC)*, 2011.