Towards Practical Secure Cloud Health Monitoring

Muthuramakrishnan Venkitasubramaniam †, Tolga Soyata*
*Dept. of Electrical and Computer Engineering †Dept. of Computer Science
University of Rochester University of Rochester

***Abstract*—The Patient Protection and Affordable Care Act is one of the most significant government efforts to generalize the use of electronic medical records (EMRs) and to incentivize the development of innovative technologies that can help curb the rising US healthcare costs. Moving datacenters to the cloud is a clear mechanism for healthcare providers to reduce costs. However, privacy of personal medical data is a major concern when processing data off-site. Fully Homomorphic Encryption (FHE), discovered by Gentry in 2009 allows for data to be *stored* and *computed* in encrypted form. Given the tremendous potential FHE has to revolutionize cloud computing, a lot of the recent works have focussed on implementing these encryption schemes. However, most implementations are too slow for most applications. We present a technique that demonstrates feasibility of using state-of-the-art FHE implementations for concrete applications. Through simulations, we demonstrate that our method yields about 20x speedup in a sample application. This is a significant step towards practical FHE-based medical remote monitoring.**

I. Introduction

Cloud computing could be a viable option to reduce costs associated with EMRs by outsourcing the storage of medical data to cloud operators, such as Amazon Web Services, GoogleCloud, and Microsoft Azure. However, even if the hospitals are willing to embrace cloud computing, cloud operators are reluctant to sign a Business Associate Agreement (BAA) and accept to store Personal Health Information (PHI) due to the high risks associated with a potential breach of data ($50K to $1.5M penalties depending on the type of the violation).

In a breakthrough result, Gentry [1] presented a novel technique that allows encrypted data to be processed securely: *fully homomorphic encryption* (FHE). Unfortunately, current FHE schemes are computationally expensive while requiring huge storage even for a relatively small amount of raw data. In [**?**], we show with an open source implementation of a FHE scheme called HElib [2], we demonstrate that it may be practical for some basic applications. One such application involves streaming sensor data to the cloud and comparing the values to a threshold.

II. Case Study : Long Term Health Monitoring

As a case study, we chose to implement an ECG-oriented application: detecting prolongation of the QT interval. Prolongation of the QT interval (shown in Figure 1) may be genetic (Long QT syndrome, LQTS) or drug-induced. Monitoring is typically conducted in the hospital, but this gives an incomplete picture of the patient's QT interval since they are not participating in their normal daily activities. This makes remote QT_c surveillance of an ambulatory patient a good candidate application for our proof-of-concept.

In our case study we upload FHE-encrypted ECG data in an online manner from a patient to the cloud, and have the cloud compute over the encrypted data and transmit the result to the patient's doctor in real-time. In our particular case-study, we assume that the cloud receives two values, QT and RR, pictured in Figure 1 for every heart-beat cycle. These values will be extracted from the ECG monitor[1] and transmitted from the patient's end in encrypted form. The patient's QT_c is then computed (homomorphically, in the cloud) as $QT_c = \frac{QT}{\sqrt[3]{RR/sec}}$ (Fridericia's formula [3]). The still-encrypted QTc values then need to be compared to a threshold value, such as 500ms. The function we are interested in, then, is:

$$f(\mathrm{QT}, \mathrm{RR}) = [\mathrm{QT}^3 > (500\mathrm{ms})^3(\mathrm{RR}/\mathrm{sec})], \qquad (1)$$

III. A New Approach

Since FHE allows for computing over encrypted data, the obvious approach is to send encryptions of the data elements to the cloud, homomorphically evaluate f from Equation 1 on each element in the incoming stream, and then compute the "OR" of the result of the computations (again, homomorphically). As we show in our experimental results, this solution is computationally costly. This is because homomorphic operations are inherently incredibly expensive. In fact, the known FHE schemes have a different cost model where performing a multiplication operation homomorphically is typically far more expensive than an addition operation and the cost of multiplication grows significantly with the multiplication-depth (i.e., a cascaded set of multiplications).

A simple calculation will show that in order to do this following the naive approach we need a depth $d = \mathrm{DEPTH}_f + \log n$ to process n data elements, where DEPTH_f is the

[1]Using standard digital-signal processing methods

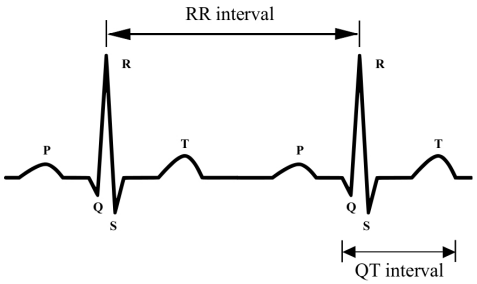

Fig. 1. Normal sinus rhythm. The QT interval represents the time for the ventricular recovery phase of the heart. Prolongation of QT (relative to the RR interval) indicates an increased risk for life-threatening events.

multiplication-depth of f. The main contribution of our approach is to show how we can significantly improve the computational efficiency by relying on an alternative representation of the computation that will significantly reduce the depth of the computation. In addition, our method will be inherently parallelizable and have small input locality.

Our high-level idea is to choose a suitable computational model that is reasonably powerful and then rely on existing HE schemes or develop new schemes that allow for both (1) homomorphic computation of all functions in the chosen computational model and (2) aggregation of the results collected over a period of time. Towards this we first represent the function f as a *branching program* instead of a circuit. A branching program is a directed acyclic graph with a special start node s and final node t where each edge is labeled with either an input bit or its negation. The result of the computation is true if there is a path from the start to the final node traversing only edges for which the assignment sets the value on the edge true.

Next, we show how to use an FHE scheme to evaluate a branching program and aggregate the results of the computation over streaming data. Using elementary linear algebra we can show that evaluating a branching program is equivalent to evaluating the determinant of a particular matrix. More precisely, the detereminant will be $f(x)$ for the matrix corresponding to input x. Given the matrix representation of two inputs x_1 and x_2, computing the "AND" of $f(x_1)$ and $f(x_2)$ now reduces to simply multiplying the matrices corresponding to the inputs, since $det(AB) = det(A)det(B)$.

Our overall approach is to use the inputs QT and RR in encrypted to form to first generate elements of the matrix for each element homomorphically and then multiply the matrices corresponding to all elements in the data stream. The main benefit of our approach is due to the very low multiplicative depth of our computation. In fact, the depth of our computation will be $\log n$ giving a saving of DEPTH_f

IV. Performance Evaluation

To evaluate our approach we compare the running times of the naive approach to our approach using the open-source implementation HELib [2] of the FHE scheme due Brakerski, Gentry and Vaikuntanathan [4]. Since the HELib library is not thread-safe, to compare the performance of the two approaches we must simulate the computation to estimate the time taken on parallel machines. To accomplish this, we first perform real benchmarks of each individual operation on a single thread, and then use that time to estimate the computational costs for our parallel experiments. In our estimate we assume that each parallel machine has instantaneous access to the input encryptions and results of computations from other machines. (In essence, we ignore the data transfer and sharing costs.)We ran our simulation for processing 10 and 10240 samples. Both approaches improved with the number of processors. However, the matrix approach was consistently better than the naive approach by a factor of 20. The main reason for this is that the cost of computation increases significantly with the depth of the computation and the depth of the naive approach is significantly higher than that of the matrix approach. For a more detailed discussion of our results see [**?**]

V. Optimizations and Scalability

Our proposed approach works for any boolean function that can be represented as a branching program. The size of the matrix corresponds to the size of the graph. For the equation in our case study, we construct a branching program of size ~ 600.

We observe that if the matrix representing the function is sparse, then we can efficiently encode the elements of the matrix that will facilitate multiplication. In particular, a band matrix with a small band can be efficiently multiplied using homomorphic operations. A band matrix has non-zero entries only in a band around the main diagonal. Furthermore, multiplying two band matrices results in a band matrix with a slightly larger band. An advantage of using branching programs is that there is notion of width for a branching program which directly translates to the width of the corresponding band matrix. Furthermore, a fundamental theorem due to Barrington [5], from complexity theory, states that the complexity of functions computable by small width branching programs is exactly the set of circuits with small depth (referred to in the literature as NC^1). The branching program we construct for our case study has width 10, and the corresponding band matrix has width 9.

VI. Conclusions and Future Work

In theory, FHE is a solution, but in practice it is too slow. We have presented a technique that improves computational efficiency and scalability of FHE for a large set off applications, resulting in 20x speedup for a representative application. Our work shows that for applications where we would like to perform some sort of detection on a stream of encrypted data can be performed efficiently by formulating the detection algorithm via a branching programs. Furthermore, it suggests that to achieve scalability the right cloud architecture would be one that facilitates generation of encrypted matrices from encrypted inputs and multiplication of encrypted matrices using homomorphic operations.

References

[1] Craig Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, 2009, pp. 169–178.

[2] Shai Halevi and Victor Shoup, "Algorithms in helib," *IACR Cryptology ePrint Archive*, vol. 2014, pp. 106, 2014.

[3] Louis Sigurd Fridericia, "Die Systolendauer im Elektrokardiogramm bei normalen Menschen und bei Herzkranken," *Acta Medica Scandinavica*, vol. 53, pp. 469–486, 1920.

[4] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *ITCS*, 2012, pp. 309–325.

[5] David A. Mix Barrington, "Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1," *J. Comput. Syst. Sci.*, vol. 38, no. 1, pp. 150–164, 1989.