Digital Personalized Health and Medicine
L.B. Pape-Haugaard et al. (Eds.)

doi:10.3233/SHTI200112

A Secure Multi-Party Computation Protocol for Time-To-Event Analyses

Lennart VOGELSANG [a,b,1], Moritz LEHNE [c], Phillipp SCHOPPMANN [a,b], Fabian PRASSER [c,d], Sylvia THUN [c,d,e], Björn SCHEUERMANN [a,b] and Josef SCHEPERS [c]
[a] *Department of Computer Science, Humboldt-Universität zu Berlin, Germany*
[b] *Alexander von Humboldt Institute for Internet and Society (HIIG), Berlin, Germany*
[c] *Berlin Institute of Health (BIH), Germany*
[d] *Charité – Universitätsmedizin Berlin, Germany*
[e] *Hochschule Niederrhein – University of Applied Sciences, Krefeld, Germany*

Abstract. The cryptographic method *Secure Multi-Party Computation* (SMPC) could facilitate data sharing between health institutions by making it possible to perform analyses on a "virtual data pool", providing an integrated view of data that is actually distributed – without any of the participants having to disclose their private data. One drawback of SMPC is that specific cryptographic protocols have to be developed for every type of analysis that is to be performed. Moreover, these protocols have to be optimized to provide acceptable execution times. As a first step towards a library of efficient implementations of common methods in health data sciences, we present a novel protocol for efficient time-to-event analysis. Our implementation utilizes a common technique called *garbled circuits* and was implemented using a widespread SMPC programming framework. We further describe optimizations that we have developed to reduce the execution times of our protocol. We experimentally evaluated our solution by computing Kaplan-Meier estimators over a vertically distributed dataset while measuring performance. By comparing the SMPC results with a conventional analysis on pooled data, we show that our approach is practical and scalable.

Keywords. secure multi-party computation; healthcare data; patient privacy; operational confidentiality of information

1. Introduction

The increasing abundance of digital health data holds great potential for medicine and healthcare. Paired with standardized data formats and analytic tools, this wealth of data can help to improve diagnosis, treatment and prevention as well as efficiency of care. However, health data is distributed across a large number of institutions, and the knowledge to be gained from these isolated data repositories is limited. This has led to the creation of a wide range of national and international data sharing initiatives. But laws and regulations such as the EU General Data Protection Regulation often prevent the disclosure and sharing of health data for privacy reasons.

Secure Multi-Party Computation (SMPC) is a cryptographic approach that can help to overcome this limitation and facilitate data sharing between health institutions. SMPC

[1] Corresponding Author, Lennart Vogelsang, Department of Computer Science, Humboldt-Universität zu Berlin, Rudower Chaussee 25, 12489 Berlin, Germany; E-mail: vogelsal@informatik.hu-berlin.de.

makes it possible to perform analyses on a *virtual data pool*, providing an integrated view of data that is actually distributed – without any of the participants having to disclose their private data [1].

Prior work on SMPC in healthcare includes a protocol for genome-wide association studies that keeps underlying geno- and phenotypes private [2], as well as a study that uses SMPC to investigate drug-target interactions in large pharmacological datasets [3]. Other applications include a disease surveillance system that ensures patient and provider privacy [4] and a method for privacy-preserving health record linkage [5].

2. Objective

Despite its potential as a privacy-preserving approach to health data sharing, SMPC is not yet widely used in health analytics. One reason for this is that SMPC is computationally demanding, and protocols therefore have to be highly optimized for acceptable performance.

As a first step towards a library of efficient SMPC implementations of common methods in health data sciences, we studied the problem of time-to-event analysis over vertically distributed data. In this scenario, one participant (e.g. a hospital) holds relevant clinical data (e.g. the primary diagnosis) while another participant (e.g. a medical practice or death register) holds data on relevant events (e.g. time of death or (re-)admission to hospital) of the same patient group. As a showcase, we present an efficient SMPC implementation of the Kaplan-Meier estimator for survival probabilities.

3. Methods

3.1. Secure Multi-Party Computation

Our implementation uses the Obliv-C framework [6], which provides an abstraction layer for the *garbled circuits* technique. With this widespread approach, two-parties can evaluate a common function without disclosing their inputs by mapping the function to a Boolean circuit [1]. Furthermore, we used a library providing additional primitives for sorting and secure memory that hides access patterns (ORAM) [7–9].

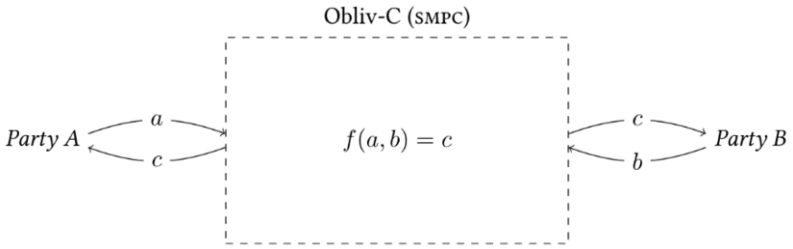

Figure 1. Example for the abstraction layer provided by Obliv-C.

Obliv-C provides an abstraction layer that imitates a Trusted Third Party (TTP). As illustrated in Figure 1, both parties *A* and *B* push their data *a* and *b* to the TTP, which calculates the function *f* and returns the output *c*. In reality, however, there is no TTP;

the computation and its inputs are encrypted and computations are executed distributively between both parties.

3.2. Efficient SMPC Protocol for Kaplan-Meier Estimators

One major drawback of SMPC is its computational overhead. To reduce this overhead, we split our problem into two phases: In the first, SMPC-based phase, computation is minimized and as much computation as possible is offloaded to the second, non-SPMC-based phase. As intermediate results from the SMPC program are revealed to the subsequent non-SMPC programs, special care has to be taken to make sure that these intermediate results do not leak private information.

In the particular case of the Kaplan-Meier estimator, the survival probabilities of individual patients should not be leaked, as this would create a direct mapping between the result and each patient's private information. Instead, values have to be aggregated so that only cumulated distinct data points are revealed. To aggregate time points in the SMPC phase, we used the secure memory (ORAM) mentioned above.

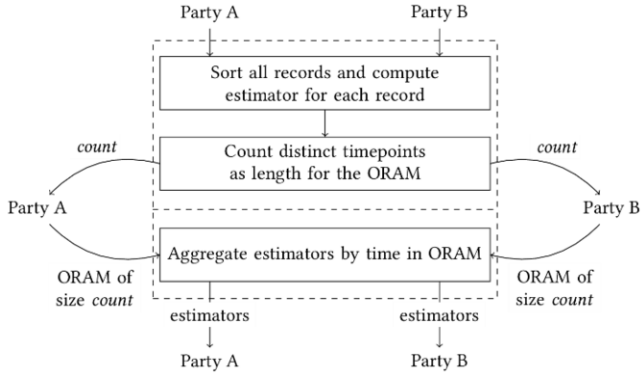

Figure 2. Schematic view of our algorithm implementing the Kaplan-Meier estimator.

One challenge with this approach is that the ORAM has to be initialized with a maximum size. While the number of patient records could be used as an upper bound, this adversely impacts performance, as ORAMs do not scale well. However, the length of the result is typically lower because time points get aggregated and not every patient has an event. The size of the ORAM can thus be limited to the number of distinct time points. We achieve this by iterating over the dataset in two (sub-) phases: first, we aggregate the values without storing them, only counting the number of distinct time points, revealing the count afterwards; then we initialize the ORAM and perform the calculations again, this time storing the results as shown in Figure 2. This does not impact privacy in any way, as the count is leaked at the end of the protocol anyway.

The source code of our SMPC protocol for Kaplan-Meier estimators is available at https://github.com/LevitatingOrange/secure-kaplan-meier.

3.3. Experimental Setup

In our experiments we used two datasets: (1) A synthetic dataset including patient demographics (date of birth, gender and postal code), case information (case number,

admission and discharge dates), coded diagnoses (ICD-10-GM) and procedures (German OPS codes); (2) a simulated dataset from a death registry. The datasets included a total of 3,927 patients with 5,110 cases. The patients shared a common identifier across both datasets (e.g. a social security or health insurance number).

We performed two experiments: (1) a "conventional" analysis in R (www.R-project.org) on the pooled datasets on one machine; (2) an analysis on distributed data with our SMPC protocol simulating two machines connected with a local area network (LAN, bandwidth: 119 Mbits/s, average latency: 0.347ms) and a wide area network (WAN, bandwidth: 31.3 Mbits/s, average latency: 5.11ms). We used two laptops for the LAN scenario: a six core Intel Core i7-8850H processor and 32 GB RAM machine and another with a four core Intel Core i7-4960HQ and 16 GB RAM. In the WAN scenario, we connected the former laptop to a virtual machine on a hosting provider with 12 GB RAM and a six core Intel Xeon E5-2680 processor.

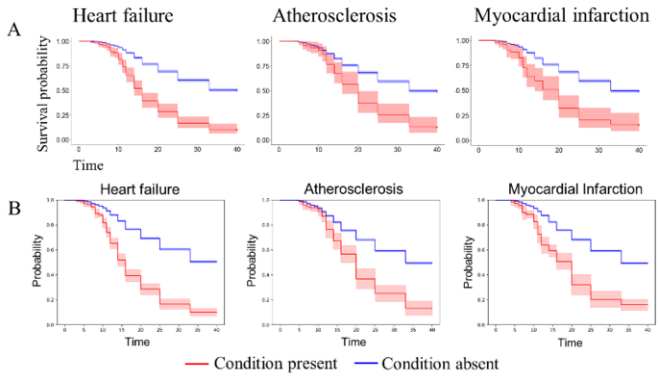

Figure 3. Kaplan-Meier plots obtained using (A) pooled data and (B) the SMPC protocol.

4. Results

Figure 3 shows the Kaplan-Meier graphs comparing patients with (vs. without) a diagnosis of heart failure (ICD-10 code: I50*), atherosclerosis (ICD-10 code: I70*) and myocardial infarction (ICD-10 code: I21*). As can be seen, the results obtained in both experiments were in high agreement.

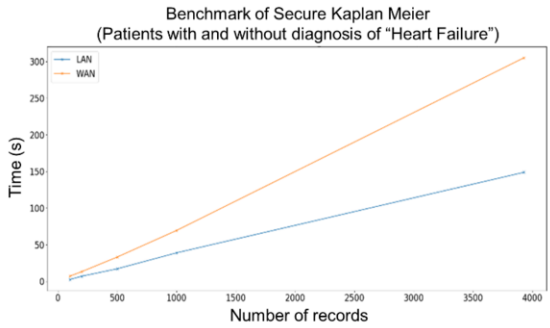

Figure 4. Execution times of the SMPC Kaplan-Meier estimator in the LAN and WAN scenario.

Figure 4 shows the execution times of the SMPC protocol. Both scenarios (LAN and WAN) showed a linear increase in computation time in relation to the number of records processed. The complete dataset of nearly 4,000 records was processed in about five minutes in the WAN scenario.

5. Discussion and Conclusion

In this article, we have presented an efficient SMPC implementation of the Kaplan-Meier estimator over vertically distributed data.

Slight differences in the survival probabilities resulting from the SMPC and the conventional implementation can be explained by the fact that Obliv-C only supports 32-bit single precision floating point arithmetic (rather than the 64-bit precision floating-point operations of the conventional analysis). Moreover, the experiments showed that our approach provides practical execution times. Even with very large datasets (>1,000,000 records), we expect computations to take not more than several hours or maximally a few days. This is acceptable in typical research projects, especially when considering that the alternative, in many cases, would be not to do the project at all.

In future work, we plan to implement additional SMPC protocols for common statistical methods in health analytics and test them in real-world scenarios. An interesting candidate would be an SMPC implementation of a Cox proportional hazard model for more advanced statistical analyses of survival times. In the long-term, SMPC protocols could be made available in an open source library, providing a collection of methods that researchers can choose from for their specific projects.

For long, SMPC remained a largely academic field, especially due to its sub-par performance compared with traditional analysis methods. With recent progress towards efficient SMPC protocols and the focus on data privacy and security, SMPC can play a large role in advancing analysis across distributed healthcare data.

References

[1] A.C. Yao, How to generate and exchange secrets, in: 27th Annu. Symp. Found. Comput. Sci. Sfcs 1986, 1986: pp. 162–167.

[2] H. Cho, D.J. Wu, and B. Berger, Secure genome-wide association analysis using multiparty computation, *Nat. Biotechnol.* **36** (2018) 547–551.

[3] B. Hie, H. Cho, and B. Berger, Realizing private and practical pharmacological collaboration, *Science*. **362** (2018) 347–350.

[4] K. El Emam, J. Hu, J. Mercer, L. Peyton, M. Kantarcioglu, B. Malin, D. Buckeridge, S. Samet, and C. Earle, A secure protocol for protecting the identity of providers when disclosing data for disease surveillance, *J. Am. Med. Inform. Assoc. JAMIA*. **18** (2011) 212–217.

[5] F. Chen, X. Jiang, S. Wang, L.M. Schilling, D. Meeker, T. Ong, M.E. Matheny, J.N. Doctor, L. Ohno-Machado, and J. Vaidya, Perfectly Secure and Efficient Two-Party Electronic-Health-Record Linkage, *IEEE Internet Comput.* **22** (2018) 32–41.

[6] S. Zahur, and D. Evans, Obliv-C: A Language for Extensible Data-Oblivious Computation, *ACR Cryptol. EPrint Arch.* (2015) 20.

[7] O. Goldreich, and R. Ostrovsky, Software Protection and Simulation on Oblivious RAMs, *J ACM.* **43** (1996) 431–473.

[8] S. Zahur, X. Wang, M. Raykova, A. Gascon, J. Doerner, D. Evans, and J. Katz, Revisiting Square-Root ORAM: Efficient Random Access in Multi-party Computation, in: 2016 IEEE Symp. Secur. Priv. SP, IEEE, 2016: pp. 218–234.

[9] J. Doerner, and A. Shelat, Scaling ORAM for Secure Computation, in: Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur. - CCS 17, ACM Press, 2017: pp. 523–535.