Secure Multi-Party Computation for Inter-Organizational Process Mining

Gamal Elkoumy[1], Stephan A. Fahrenkrog-Petersen[2], Marlon Dumas[1], Peeter Laud[3], Alisa Pankova[3], and Matthias Weildich[2]

[1] University of Tartu, Tartu, Estonia
{gamal.elkoumy,marlon.dumas}@ut.ee
[2] Humboldt-Universität zu Berlin, Berlin, Germany
{fahrenks,weidlima}@hu-berlin.de
[3] Cybernetica, Tartu, Estonia
{peeter.laud,alisa.pankova}@cyber.ee

Abstract. Process mining is a family of techniques for analysing business processes based on event logs extracted from information systems. Mainstream process mining tools are designed for intra-organizational settings, insofar as they assume that an event log is available for processing as a whole. The use of such tools for inter-organizational process analysis is hampered by the fact that such processes involve independent parties who are unwilling to, or sometimes legally prevented from, sharing detailed event logs with each other. In this setting, this paper proposes an approach for constructing and querying a common type of artefact used for process mining, namely the time-annotated directly-follows graph (DFG), over multiple event logs belonging to different parties, in such a way that the parties do not share the event logs with each other. The proposal leverages an existing platform for secure multi-party computation, namely Sharemind. Since a direct implementation of DFG construction in Sharemind suffers from scalability issues, the paper proposes to rely on vectorization of event logs and to employ a divide-and-conquer scheme for parallel processing of sub-logs. The paper reports on an experimental evaluation that tests the scalability of the approach on real-life logs.

Keywords: Process Mining · Privacy · Secure Multi-Party Computation

1 Introduction

Analysing business processes based on event logs extracted from information systems through process mining techniques [1] becomes increasingly popular in industry. It enables organizations to optimize their processes and achieve their strategic goals, e.g., realizing cost-efficient process execution. However, in practise, many business processes are not restricted to a single organization, but are executed by several collaborating organizations. We call such processes inter-organizational business processes. An example of such a process is given in Figure 1, which illustrates the ground handling of an aircraft, which includes at least the airline and the ground handler, e.g., the airport.

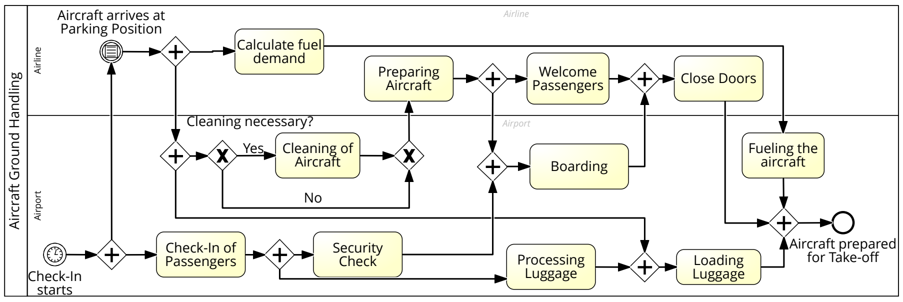

Fig. 1: The example of an aircraft ground handling process.

Due to confidentiality issues and privacy regulations, such as GDPR, it is not possible for organizations to share their process data with each other [?]. Exchanging event logs could reveal personal information of customers or expose business secrets. As a consequence, common techniques for process mining cannot be employed for inter-organizational business processes, despite the fact that these processes can have a large impact of the core operation of a business. For the aforementioned scenario, for instance, it is well-known that the orchestration of ground handling activities is of crucial importance for both involved parties [36]. It determines the number of flights an airport can operate as well as the number of flights an airline can offer per aircraft.

In this paper, we focus on the question of how to enable process mining for inter-organizational business processes without requiring the involved parties to share their private event logs or trust a third party. To this end, we propose an architecture for process mining based on *secure multi-party computation* (MPC) [?]. In essence, MPC aims at the realization of some computation over data from multiple parties, while exposing only the result of the computation, but keeping the input data private. We consider the setting of an MPC platform where the involved parties upload their event logs to a network of compute nodes. Before the upload, secret sharing algorithms locally split each single data value into different parts (i.e., shares) that are then stored at different nodes. Since each share does not provide any information about the original data, the uploaded event log is encrypted and exposed neither to the platform operator nor other involved parties. Nonetheless, the MPC platform enables the computation over the encrypted data through protocols for result sharing among the nodes.

We realise the above architecture to answer analysis queries that are common in process mining. That is, we show how to construct a time-annotated directly-follows graph that serves as a starting point for process discovery algorithms [?] and the detection of performance bottlenecks [?]. We implement our proposed architecture using the Sharemind platform [11]. In order to avoid scalability issues that would be imposed by a naive implementation, we employ vectorization of event logs and propose a divide-and-conquer scheme for parallel processing of sub-logs. We demonstrate the effectiveness of these optimisations in an experimental evaluation with real-world event logs.

The remaining paper is structured as follows. Section 2 lays out related work and the background for this work. Section 3 introduces our architecture for privacy-preserving inter-organizational process mining along with the optimizations needed for efficient implementation. An experimental evaluation is presented in Section 4, before Section 5 concludes the paper.

2 Background and Related Work

The work we will present in this paper relates to the topics of privacy-preserving process mining, cross-organizational process mining and secure multi-party computation. In the remainder of this section we will introduce related work out of these areas of research.

2.1 Privacy-preserving Process Mining

In the area of privacy-preserving process mining[33] two general approaches have been established[20]: anonymizing the event data and directly performing privatized process mining techniques. For the anonymization of event logs from one organization techniques such as *PRETSA*[21], an algorithm to ensure the established privacy metrics k-anonymity[39] and t-closeness[28], exist. These metrics[43], based on data similarity, are widely adopted and offer protection against certain attacks, like the disclosure of the identity of individuals involved in the dataset. The necessity of privacy-preserving process mining, due to novel legal development such as the GDPR, was recently discussed in [31]. Approaches based on cryptography[15,35] have also been proposed.
On the side of privatized process mining methods for prrivacy-preserving process discovery[30,40,41] and the privacy-aware discovery of resource roles[34] have been established. Recently the Both privatized process mining and techniques for anonymizing event logs have been made available for a large audience with the tool ELPaaS[6]. The tool and all techniques mentioned above, with the exception of [41], a concerned with privacy-preserving process mining for one organization and have not yet adopted or evaluated for a cross-organizational setting. While is focussed on a cross-organizational setting [41] it is only focused on generating process models, while our aim is answering a wide range of queries about a business process.

2.2 Cross-Organizational Process Mining

The specific problem of privacy in cross-organizational process mining has been addressed in [29]. Liu et. al provide a framework for privacy-preserving cross-organizational process mining based on the assumption that a trusted third party exists and mostly focus on the discovery of process models. Our plan is to provide a solution for scenarios without a trusted third party and lays the focus on computing queries over distributed event logs in a privacy-preserving manner. Other approaches of cross-organizational process mining, that do not consider privacy aspects are presented in [37,46]. Such as [29] it also aiming in the discovery of process models. Another line for research[3,14,42] in cross-organizational process mining attends to compare the same process in different organizations.

2.3 Secure Multi-Party Computation

Secure Multi-party Computation (MPC) [24] is a cryptographic functionality that allows n parties to cooperatively evaluate $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ for some function f, with the i-th party contributing the input x_i and learning the output y_i, and no party or an allowed coalition of parties learning nothing besides their own inputs and outputs.

There exist a few different approaches for constructing MPC protocols. Using the inherently 2-party *garbled circuits* (GC) approach [44,45], one of the parties encrypts each gate of the Boolean circuit representing f, and sends it to the other party, together with the keys corresponding to both parties' inputs. The second party decrypts a part of the representation of each gate, and learns the output of f in the end. Garbled circuit based protocols have small round complexity, but tend to require more bandwidth than some of the next approaches.

Homomorphic encryption (HE) [19,32] can be used to perform computations (mostly linear) on data without seeing it, and threshold homomorphic encryption [17] can be used to build a full MPC protocol. These primitives have been used for privately implementing the Alpha Algorithm for process discovery [40,41]. Any computations on encrypted data are enabled by *fully homomorphic encryption* (FHE) schemes [23]. While FHE-based approaches minimize the communication between parties, they are very computation-intensive.

Homomorphic secret sharing [7,38] is currently the most common basis for MPC protocols [16,22]. In such protocols, the arithmetic or Boolean circuit representing f is evaluated gate-by-gate, constructing secret-shared outputs of gates from their secret-shared inputs. Each evaluation requires some communication between parties (except for addition gates), hence the depth of the circuit determines the round complexity of the protocol. On the other hand, there exist protocols with low communication complexity [12,4,18], allowing the secure computation of quite complex functions f, as long as the circuit implementing it has a low multiplicative depth.

The complexity of MPC protocols is heavily dependent on the number of parties jointly performing the computations. Hence the typical deployment of MPC has a relatively small number of *computation parties* (typically just 2 or 3) actually running the protocols for evaluating gates, while an unbounded number of parties may contribute the inputs and/or receive the outputs of the computation [25]. There exist frameworks that support such deployments of MPC, they have APIs to simplify the development of privacy-preserving applications [5]. One of such frameworks is Sharemind [11], whose main protocol set [12] is based on secret-sharing among three computing parties. In this paper, we build on top of Sharemind and its large number of primitive protocols and subroutines [9], but our techniques are also applicable to other secret sharing based MPC systems.

Sharemind framework simplifies our work by offering the SecreC language [8] for programming privacy-preserving applications, abstracting away these details of cryptographic protocols that make sense to be abstracted away.

3 Multi-Party Computation based Process Mining

This section introduce our techniques for process mining based on secure multi-party computation. To this end, in Section 3.1, we first clarify our model for cross-organizational process mining including the required input data and the obtained analysis results. We then introduce our architecture for realizing the respective analysis using secure multi-party computation in Section 3.2. In Section 3.3, we focus on techniques for vectorization and parallelization to improve the efficiency of our approach.

3.1 Model for Cross-organizational Process Mining

We consider a model in which an event log $L = \{t_1, \ldots, t_n\}$ is defined as a set of traces, each capturing the single execution of a business process. We define a trace t as a finite sequence of events $t = \langle e_1, \ldots, e_m \rangle$. An event e is defined as $e = (i, a, ts)$. It represents a single execution of an activity a at time ts, where i is a unique event identifier, i.e., $i = i'$ implies that $(i, a, ts) = (i', a', ts')$. We assume that events in a trace are ordered by their timestamp.

However, for a cross-organizational business process, an event log that records the process execution from start to end is commonly not available. Rather, different parties record sub-logs that contain sub-traces, built of events that denote activity executions at the respective party. To keep the notation concise, we consider a setting in which two parties, I_a and I_b, execute a cross-organizational process, e.g., the airport and the airline in our motivating example. Then, each of the two parties records an event log, denoted by L_a and L_b. A trace $t = \langle e_1, \ldots, e_m \rangle \in L$ of the cross-organizational process is materialized as two sub-traces, $t_a \in L_a$ and $t_b \in L_b$, each being defined as the order-preserving projection of trace t on the events that denote activity executions at the respective parties I_a and I_b. We assume that each activity can only be executed by one of the parties, so that the projection of the log into sub-logs is defined unambiguously.

For the above setting, we consider the scenario that the parties I_a and I_b want to answer some analysis queries Q over the cross-organizational event log L, yet *without* sharing their logs L_a and L_b with each other. More specifically, we focus on analysis queries that can be answered on a time-annotated directly-follows-graph (DFG) of the cross-organizational process. In essence, the DFG captures the frequencies with which the executions of two activities have been observed to directly follow upon each other in a trace. Moreover, we consider temporal annotations of the directly-follows dependencies in terms of time between the respective activity executions. A time-annotated DFG thereby enables conclusions on the main paths of process execution, rarely executed paths, as well as the location of major delays in processing.

@Stephan, could you clarify that DFG would be kept secret in sharemind and we reveal only the answers for the queries?

Formally, the time-annotated DFG is captured by an $|A| \times |A|$ matrix, where A is the set of all possible activities of the process. Each cell contains a tuple (c, Δ). The counter c represents the frequency with which a directly-follows dependency has been observed in L, i.e., for the cell (a_1, a_2) it is the number of times that two events $e_1 = (i_1, a_1, ts_1)$ and $e_2 = (i_2, a_2, ts_2)$ follow each other directly in some trace of L. Furthermore, Δ is the total sum of the time that passed by between all occurrences of the respective events, i.e., $ts_2 - ts_1$ for the above events.

In cross-organizational process mining, the aforementioned time-annotated DFG cannot be computed directly, as this would require the two parties to share and integrate their sub-logs.

3.2 MPC Architecture for Process Mining

In order to enable inter-organizational process mining based on a time-annotated DFG *without* requiring parties to sharing their event logs with each other, we propose an architecture for secure multi-party computation (MPC). As outlined in Figure 2, we rely on a platform for MPC that takes the event logs of the participating parties, i.e., L_a and L_b, as encrypted input. In the MPC platform, the respective data is then prepared in several steps in order to answer analysis queries over the time-annotated DFG.

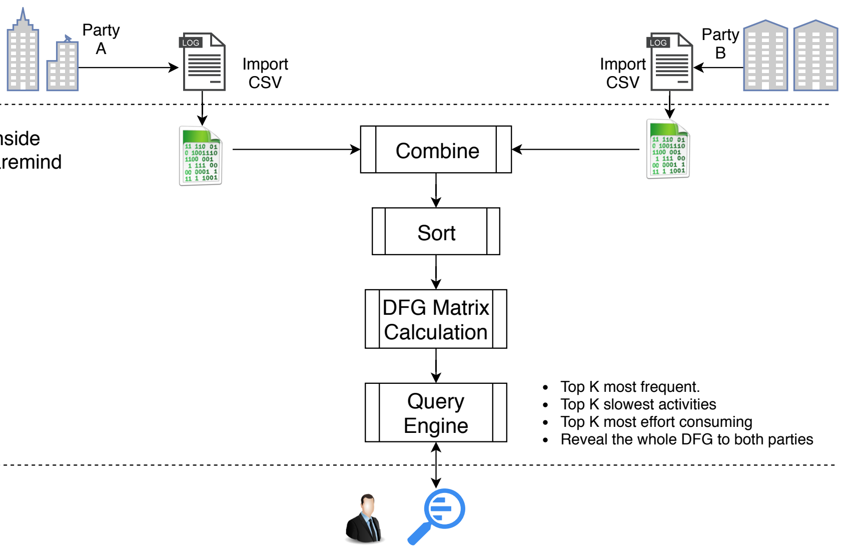

Fig. 2: Number of variants retained for k-anonymity.

We summarize the functionality realized in the MPC platform as follows:

Combination While the input parties upload their respective event logs L_a and L_b to the MPC platform, these get encrypted and distributed to the computing parties. So it is ensured, that nobody can access the secret data. After the logs upload we append one of the logs to the other so that the unsorted log L is created.

Sorting To ensure an efficient calculation of the annotated DFG G we sort the merged log L, such that all event $e \in t$ of a trace $t \in L$ are group together and ordered by their respective execution time ts. For this purpose we rely on the network sorting algorithm[10], that ensures efficient sorting without revealing secret information.

Matrix calculation After sorting L we can construct the DFG G inside the MPC platform, while keeping the DFG G secret and not revealing any information.

Query answering A query Q requests a subset S of the annotated DFG G. The MPC platform generates S based on the query Q, e.g., the top-5 most common activities, and only reveals this subset S to the output parties. By publishing only the subsets S to pre-defined output parties we limit the information an output party can learn about the process.

3.3 Scalability Considerations

Cross-organizational process mining using the above general architecture might suffer from scalability issues. The reason being the use of multi-party computation to realise the functionality to combine and sort the event logs and to calculate the annotated DFG, which is inherently distributed and applied to homomorphic encrypted data. Hence, even for functions that have a generally low run-time complexity ($\mathcal{O}(n)$ for the combination, $\mathcal{O}(n \log(n))$ for the sorting, $\mathcal{O}(n^2)$ for the calculation of the annotated DFG), there is a non-negligible overhead induced by MPC. For instance, a naive realisation of the Quicksort algorithm to sort the events of the combined log would require $\mathcal{O}(n \log(n))$ rounds of communication between the distributed nodes and $\mathcal{O}(n \log(n))$ value comparisons per round [**?**]. Against this background, we consider two angles to improve the efficiency of the analysis, namely vectorization and parallelization.

Vectorization. Contemporary MPC platforms provide efficient implementations of basic functions to process distributed, encrypted data. For instance, Sharemind supports bitwise secret sharing that represents a data value as a bitwise combination of vectors that are distributed among the nodes. Based thereon, the platform offers efficient implementations of vector-based functions, such as those for matrix multiplication [26,27]. For instance, the multiplication of two bit vectors can be realized in one round that requires $\mathcal{O}(6k)$ interactions between nodes, where k is the number of bits required to encode the value.

To leverage the efficient implementation of such basic functions, we employ vectorization and represent events as a bit vector over the set A of all possible activities of the process. To mask the actual number of possible activities, the set over which the vector is defined may include padding, i.e., some additional placeholders. Following this representation, an event corresponds to a binary vector in which at most one bit is set to one. Then, counting the number of directly-follows dependencies in the computation of the annotated DFG can be traced back to vector multiplication.

Matthias, rework the following

Parallelization. An additional approach to improve the runtime performance common for secure-multi party computation is the usage of parallelization. To ensure efficient parallelization it is important to divide the data from the input parties such that all data chunks can be used independent from each other. In our scenario this can be done by bundling chunks of cases and distributing them to several machine. For each chunk a DFG can be generated. The different DFGs can then be united to one DFG that represents the entire business process.

However, such an approach comes with the question of determining the size of the chunks. We consider that a chunk has size m representing the number of cases that are at least part of that chunk. In a multi-party computation setting the result of the processing of each chunk would not be protected. Therefore, proceeding only a small number of cases in one chunk might reveal a lot of information about the cases inside the chunk and is critical from a privacy perspective. On the other hand comes a small chunk size m with the advantage of higher performance, since it would be possible to distribute the work over more machines. This leads to an important trade-off between runtime performance and privacy considerations.

To enable an efficient splitting of the event log into chunks, it in necessary that all traces have the same length. If this condition is true, the sorted event log can just be split up at certain points, e.g. exactly in the middle if the aim is to generate two chunks and the number of traces is odd. Otherwise a split into chunks would require scanning the whole event log for the right point to split it, something that would come the computational cost of $\mathcal{O}(n)$, with n being the events in the log. Therefore, we apply a padding to the traces in the log, so that all traces have the same length, the length of the longest trace. Such a padding can be done by each input party, if they pad their part of the trace to the length of the max length of their respective longest sub trace. Since such a preprocessing comes with lower costs, than performing additional operations in a MPC setting. Nonetheless, we need to ensure that the padding has no influence on the generated DFG. Therefore, we propose padding with a bit vector, representing the activity, containing only zeros. Thanks to our vectorization approach such as padding has no influence on the resulting DFG.

4 Evaluation

In this section we will provide an evaluation for our introduced approaches based on the following research questions:

Q1: How do different event logs characteristics influence the performance of the secure multi-party computation?

Q2: Does the proposed approach scale up with increasing the number of parallel chunks and what would be the effect of scalability on both the execution time and communication overhead between the computing parties?

4.1 Datasets

To evaluate our approaches we run the experiment using the following real-world event logs:

BPIC 2017 This event log captures the loan application process from a dutch financial institute. This event logs contains traces with a medium length, but is the largest investigated log in terms of number of events.

Traffic Fines This event logs captures a process to handle the collection of fines from traffic law violations from a local police from Italy. It consists of rather short traces with a small number of different activities.

Credit Requirement This event log captures credit requirement process information in a dutch bank. The event log contains of short traces with only 8 unique events.

All of the introduced event logs capture a process from a single organization and not a cross-organizational business process. We decided to do so, because the existing public synthetic event logs[13] for cross-organizational business processes consist only of a marginal number of events. Therefore, we have chosen event logs that contain a large number of cases and vary in the length of the traces, see table 1. For the BPIC 2017, the data took about 5 hours till time out, we decide to selected 1000 traces randomly from the event log so we the log file could converge in a reasonable file.

To simulate a cross-organizational setting we use Round Robin approach to assign each activity of each event log to one party, assuming the business process is executed in a two party setting. We distribute the activities equally between both parties, so that each party executed half of the activity types for each log.

Event Log	No. of events	No. of cases	Avg. No. of events in case	Max No. of events in case	Min No. of events in case
BPIC 2017	1,202,267	31,509	38.16	180	10
Traffic Fines	561,470	150,370	3.73	20	2
Hospital Log	150,291	1,143	131.488	1814	1

Table 1: Event Logs for Evaluation

4.2 Experimental Setup

To answer the above questions, we benchmark the performance of the proposed approach. We consider latency, throughput and the communication overhead as performance measures.

Latency. We define latency as the amount of time needed to transform the 2 parties' event log securely into a DFG matrix. We report both the total execution time and the execution time of the chunk based sort. That would help to study how would help to study how the implementation of the proposed approach would be in a map/reduce like scheme. In subsection 4.3, the experiments show that changing the number of parallel chunks affects the chunk based sort only but the rest of the pipeline has a fixed execution time for the same dataset.

Throughput. We define throughput as the number of events that the system can process in a given amount of time. In our experiment, we report the number of events per minute as a measure of throughput.

Communication Overhead. In multi-party computation, the computing parties are communicating together so they can compute the final results securely. We define the communication overhead as the amount of data sent and received between the computing parties during the execution overhead. We report the values for each server. The communication overhead metric gives insights for how the system would react in case the computing parties are remotely installed.

In the above performance measures, we perform the experiments until we have the DFG matrix, because it is the most sophisticated and time consuming portion of the pipeline, as it requires communicating both parties to build the matrix. Furthermore, once we have the DFG in secret shares, we can calculate the required queries in short time.

We provide the source code of our implementation on *Github*[4]. To implement our approaches we used the SecreC programming language from Cybernetica. Our implementation is run on the multi-party computation platform Sharemind[5].

To evaluate the performance of an approach we report the average maximum values for latency and the average value for both throughput the communication overhead. To calculate the mean execution time we run each experimental setting over 5 times and calculate the maximum execution time of the three servers. We used *Nethogs*[6] to measure the communication overhead and we report the average value per server.

Hardware. We run our experiments in a setting with three symmetrical servers as computation parties. These physical servers are connected through a local network. All of them use the same Sharemind setup. Each server has AMD Processor 6276 and 192 GB RAM and all the servers are connected using 1 gigabit Ethernet switch.

4.3 Results

Latency Benchmark. In fig 3, we show the execution time to calculate the DFG matrix securely with different number of parallel chunks. We plot a bar for each chunk size. We split each bar to present the execution time of parallel sort in blue and the rest of the execution time in orange. From fig 3, we can see that the execution time decreases with increasing the number of parallel chunks due to the execution of sort in parallel for each chunk separately. Because the

[4] https://github.com/Elkoumy/SecureMPCBPM
[5] https://sharemind-sdk.github.io
[6] https://github.com/raboof/nethogs

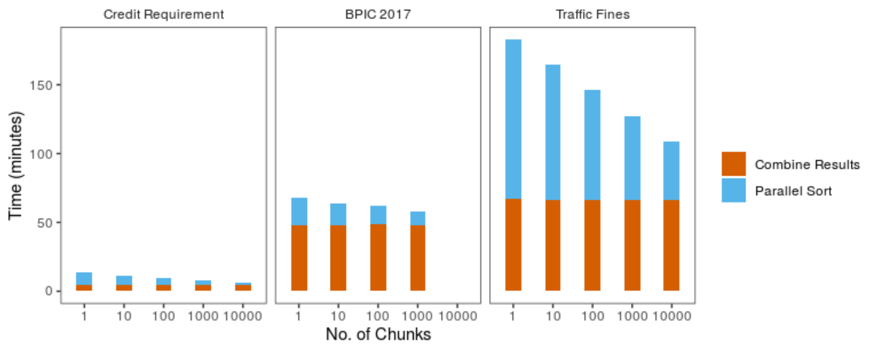

Fig. 3: Latency Benchmark: Execution Time vs no. of Chunks.

sort complexity is $O(nlogn)$, and after dividing data into chunks, the sorting complexity per chunk is $O((n/c)\ log\ (n/c))$ where c is the number of parallel chunks. The execution time when using a single chunk, the sort is slower as c equals 1 and by increasing the number of chunks the factor n/c decreases which reduces the execution time. Furthermore, from fig 3, we can observe that the execution time of the parallel sort, the blue part of the bars, changes with changing the number of parallel chunks and the combine results is almost fixed for each dataset.

An interesting observation is that, both Credit Requirement and Traffic Fines datasets as the same number of events, and different number of unique events. The number of unique events increases the number of bits used to represent the event in the binary representation for the outer product step. Increasing the number of unique events per log makes the result matrix bigger which adds computation overhead. The same thing could be observed with the BPIC 2017 dataset as it has larger number of unique events.

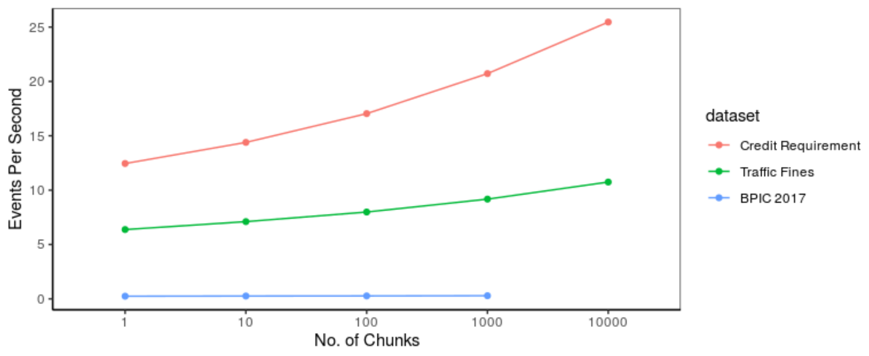

Fig. 4: Throughput Benchmark: Events per Second vs no. of Chunks.

Throughput Benchmark. In fig 4, we report the number of events per second for different number of chunks. We can observe that, for datasets with smaller average number of events per trace, as in the Traffic Fines log, the throughput increase largely with increasing the number of chunks. However, increasing the average number of events per trace, as in both Credit Requirement log and BPIC 2017 log, makes the throughput grows slower. The reason for that is with the larger average of events per trace, the size of the data grows rapidly because of padding. We choose padding to avoid revealing the number of events per each trace.

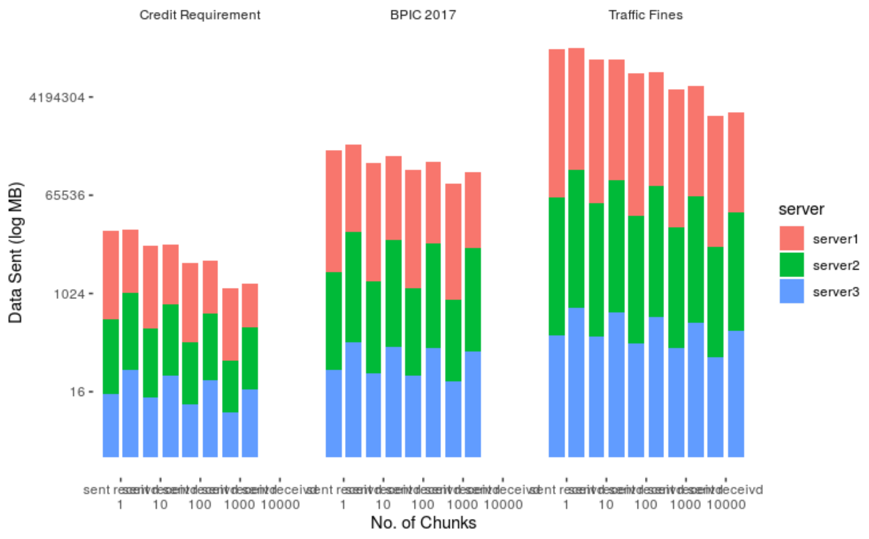

Fig. 5: Communication Benchmark: Data Sent and Received vs no. of Chunks.

Communication Overhead Benchmark. In fig 5 , we present the amount of data sent and received at each server. We can observe that the communication overhead decreases with increasing the number of chunks. There are differences between the values reported for the different servers, because our implementation inside Sharemind uses asymmetry of comparison protocols.

Also, we can observe from fig 5 that the communication overhead increases with increasing the number of unique events per event log. That happens as the outer product for larger matrix size costs more communications.

5 Conclusion

This paper introduces a framework for inter-organizational process mining based on secure multi-party computation. Our approach enables organizations to perform process mining on shared business processes with revealing only the number of traces per log, the number of unique events and the maximum number of events per a trace. To ensure a scalable execution of the secure multi-party computation we introduce techniques based on vectorization and divide-and-conquer. Our evaluation on real world event logs shows that these technique improve the performance of the secure multi-party computation and scale up under different characteristics of event logs.

Several directions for future research arise such as combining our approach with privacy guarantees like differential privacy to comply with stronger privacy requirements. Additionally, it could be explored how an adversarial input can be prohibited from gaining knowledge by manipulating the secure multi-party computation with false input data.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016). https://doi.org/10.1007/978-3-662-49851-4, https://doi.org/10.1007/978-3-662-49851-4
2. van der Aalst, W.M.: Intra-and inter-organizational process mining: Discovering processes within and between organizations. In: IFIP Working Conference on The Practice of Enterprise Modeling. pp. 1–11. Springer (2011)
3. Aksu, Ü., Schunselaar, D.M., Reijers, H.A.: A cross-organizational process mining framework for obtaining insights from software products: Accurate comparison challenges. In: 2016 IEEE 18th Conference on Business Informatics (CBI). vol. 1, pp. 153–162. IEEE (2016)
4. Araki, T., Furukawa, J., Lindell, Y., Nof, A., Ohara, K.: High-throughput semi-honest secure three-party computation with an honest majority. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 805–817. ACM (2016). https://doi.org/10.1145/2976749.2978331, https://doi.org/10.1145/2976749.2978331
5. Archer, D.W., Bogdanov, D., Lindell, Y., Kamm, L., Nielsen, K., Pagter, J.I., Smart, N.P., Wright, R.N.: From keys to databases—real-world applications of secure multi-party computation. The Computer Journal **61**(12), 1749–1771 (2018)
6. Bauer, M., Fahrenkrog-Petersen, S.A., Koschmider, A., Mannhardt, F., van der Aa, H., Weidlich, M.: Elpaas: Event log privacy as a service. In: Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019 co-located with 17th International Conference on Business Process Management, BPM 2019, Vienna, Austria, September 1-6, 2019. pp. 159–163 (2019)
7. Blakley, G.R., et al.: Safeguarding cryptographic keys. In: Proceedings of the national computer conference. vol. 48 (1979)
8. Bogdanov, D., Laud, P., Randmets, J.: Domain-polymorphic programming of privacy-preserving applications. In: Proceedings of the Ninth Workshop on Programming Languages and Analysis for Security. p. 53. ACM (2014)
9. Bogdanov, D., Laur, S., Talviste, R.: A practical analysis of oblivious sorting algorithms for secure multi-party computation. In: Bernsmed, K., Fischer-Hübner, S. (eds.) Secure IT Systems - 19th Nordic Conference, NordSec 2014, Tromsø, Norway, October 15-17, 2014, Proceedings. Lecture Notes in Computer Science,

vol. 8788, pp. 59–74. Springer (2014). https://doi.org/10.1007/978-3-319-11599-3_4, https://doi.org/10.1007/978-3-319-11599-3_4

10. Bogdanov, D., Laur, S., Talviste, R.: A practical analysis of oblivious sorting algorithms for secure multi-party computation. In: Nordic Conference on Secure IT Systems. pp. 59–74. Springer (2014)
11. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A framework for fast privacy-preserving computations. In: European Symposium on Research in Computer Security. pp. 192–206. Springer (2008)
12. Bogdanov, D., Niitsoo, M., Toft, T., Willemson, J.: High-performance secure multi-party computation for data mining applications. Int. J. Inf. Sec. **11**(6), 403–418 (2012). https://doi.org/10.1007/s10207-012-0177-2, https://doi.org/10.1007/s10207-012-0177-2
13. Borkowski, M., Fdhila, W., Nardelli, M., Rinderle-Ma, S., Schulte, S.: Event-based failure prediction in distributed business processes. Information Systems (2017)
14. Buijs, J.C., van Dongen, B.F., van der Aalst, W.M.: Towards cross-organizational process mining in collections of process models and their executions. In: International Conference on Business Process Management. pp. 2–13. Springer (2011)
15. Burattin, A., Conti, M., Turato, D.: Toward an anonymous process mining. In: 2015 3rd International Conference on Future Internet of Things and Cloud. pp. 58–63. IEEE (2015)
16. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 11–19. ACM (1988). https://doi.org/10.1145/62212.62214, https://doi.org/10.1145/62212.62214
17. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. Lecture Notes in Computer Science, vol. 2045, pp. 280–299. Springer (2001). https://doi.org/10.1007/3-540-44987-6_18, https://doi.org/10.1007/3-540-44987-6_18
18. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7417, pp. 643–662. Springer (2012). https://doi.org/10.1007/978-3-642-32009-5_38, https://doi.org/10.1007/978-3-642-32009-5_38
19. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory **31**(4), 469–472 (1985)
20. Fahrenkrog-Petersen, S.A.: Providing privacy guarantees in process mining. In: (CAiSE Doctoral Consortium 2019), Rome, Italy, June 3-7, 2019. pp. 23–30 (2019)
21. Fahrenkrog-Petersen, S.A., van der Aa, H., Weidlich, M.: PRETSA: event log sanitization for privacy-aware process discovery. In: International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019. pp. 1–8 (2019)
22. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In: Coan, B.A., Afek, Y. (eds.) Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC '98, Puerto Vallarta, Mexico, June 28 - July 2, 1998. pp. 101–111. ACM (1998). https://doi.org/10.1145/277697.277716, https://doi.org/10.1145/277697.277716

23. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. pp. 169–178. ACM (2009). https://doi.org/10.1145/1536414.1536440, `https://doi.org/10.1145/1536414.1536440`
24. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA. pp. 218–229. ACM (1987). https://doi.org/10.1145/28395.28420, `https://doi.org/10.1145/28395.28420`
25. Kamm, L.: Privacy-preserving statistical analysis using secure multi-party computation. Ph.D. thesis, University of Tartu (2015)
26. Kerik, L., Laud, P., Randmets, J.: Optimizing mpc for robust and scalable integer and floating-point arithmetic. In: International Conference on Financial Cryptography and Data Security. pp. 271–287. Springer (2016)
27. Laud, P., Pankova, A.: Privacy-preserving frequent itemset mining for sparse and dense data. In: Nordic Conference on Secure IT Systems. pp. 139–155. Springer (2017)
28. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: 2007 IEEE 23rd International Conference on Data Engineering. pp. 106–115. IEEE (2007)
29. Liu, C., Duan, H., Zeng, Q., Zhou, M., Lu, F., Cheng, J.: Towards comprehensive support for privacy preservation cross-organization business process mining. IEEE Trans. Services Computing **12**(4), 639–653 (2019). https://doi.org/10.1109/TSC.2016.2617331, `https://doi.org/10.1109/TSC.2016.2617331`
30. Mannhardt, F., Koschmider, A., Baracaldo, N., Weidlich, M., Michael, J.: Privacy-preserving process mining - differential privacy for event logs. Business & Information Systems Engineering **61**(5), 595–614 (2019)
31. Mannhardt, F., Petersen, S.A., Oliveira, M.F.: Privacy challenges for process mining in human-centered industrial environments. In: 2018 14th International Conference on Intelligent Environments (IE). pp. 64–71. IEEE (2018)
32. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 223–238. Springer (1999)
33. Pika, A., Wynn, M.T., Budiono, S., ter Hofstede, A.H., van der Aalst, W.M., Reijers, H.A.: Towards privacy-preserving process mining in healthcare (2019)
34. Rafiei, M., van der Aalst, W.M.: Mining roles from event logs while preserving privacy. In: International Conference on Business Process Management. pp. 685–697. Springer (2019)
35. Rafiei, M., von Waldthausen, L., van der Aalst, W.M.P.: Ensuring confidentiality in process mining. In: Proceedings of the 8th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2018), Seville, Spain, December 13-14, 2018. pp. 3–17 (2018)
36. Schmidberger, S., Bals, L., Hartmann, E., Jahns, C.: Ground handling services at european hub airports: development of a performance measurement system for benchmarking. International Journal of Production Economics **117**(1), 104–116 (2009)
37. Schulz, K.A., Orlowska, M.E.: Facilitating cross-organisational workflows with a workflow view approach. Data & Knowledge Engineering **51**(1), 109–147 (2004)

38. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
39. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **10**(05), 557–570 (2002)
40. Tillem, G., Erkin, Z., Lagendijk, R.L.: Privacy-preserving alpha algorithm for software analysis. In: 37th WIC Symposium on Information Theory in the Benelux/6th WIC/IEEE SP Symposium on Information Theory and Signal Processing in the Benelux (2016)
41. Tillem, G., Erkin, Z., Lagendijk, R.L.: Mining encrypted software logs using alpha algorithm. In: SECRYPT. pp. 267–274 (2017)
42. Van Der Aalst, W.M.: Configurable services in the cloud: Supporting variability while enabling cross-organizational process mining. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 8–25. Springer (2010)
43. Wagner, I., Eckhoff, D.: Technical privacy metrics: A systematic survey. ACM Comput. Surv. **51**(3), 57:1–57:38 (2018). https://doi.org/10.1145/3168389, `https://doi.org/10.1145/3168389`
44. Yao, A.C.: Protocols for secure computations. In: 23rd annual symposium on foundations of computer science (sfcs 1982). pp. 160–164. IEEE (1982)
45. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). pp. 162–167. IEEE (1986)
46. Zeng, Q., Sun, S.X., Duan, H., Liu, C., Wang, H.: Cross-organizational collaborative workflow mining from a multi-source log. Decision support systems **54**(3), 1280–1301 (2013)