Trusted Execution Environment for Decentralized Process Mining

Valerio Goretti[1], Davide Basile[1], Luca Barbaro[1], and Claudio Di Ciccio[1,2]

[1] Sapienza University of Rome, Italy, name.surname@uniroma1.it
[2] Utrecht University, Netherlands, c.diciccio@uu.nl

Abstract. Inter-organizational business processes involve multiple independent organizations collaborating to achieve mutual interests. Process mining techniques have the potential to allow these organizations to enhance operational efficiency, improve performance, and deepen the understanding of their business based on the recorded process event data. However, inter-organizational process mining faces substantial challenges, including topical secrecy concerns: The involved organizations may not be willing to expose their own data to run mining algorithms jointly with their counterparts or third parties. In this paper, we introduce CONFINE, a novel approach that unlocks process mining on multiple actors' process event data while safeguarding the secrecy and integrity of the original records in an inter-organizational business setting. To ensure that the phases of the presented interaction protocol are secure and that the processed information is hidden from involved and external actors alike, our approach resorts to a decentralized architecture comprised of trusted applications running in Trusted Execution Environments (TEEs). We show the feasibility of our solution by showcasing its application to a healthcare scenario and evaluating our implementation in terms of memory usage and scalability on real-world event logs.

Keywords: Collaborative information systems architectures · Inter-organizational process mining · TEE · Confidential computing

1 Introduction

In today's business landscape, organizations constantly seek ways to enhance operational efficiency, increase performance, and gain valuable insights to improve their processes. Process mining offers techniques to discover, monitor, and improve business processes by extracting knowledge from chronological records recorded by process-aware information systems, i.e., the *event logs* [13]. The vast majority of process mining contributions consider *intra-organizational* settings, in which processes are executed inside individual organizations. However, organizations increasingly recognize the value of collaboration and synergy in achieving operational excellence. *Inter-organizational* business processes involve several independent organizations cooperating to achieve a shared objective. Process mining can bring the advantages of transparency, performance optimization, and benchmarking in this context [2]. Since different process data owners feed separate mining nodes, this setting characterizes what we call *decentralized process mining*.

Companies, though, are reluctant to share private information required to execute process mining algorithms with external parties [25], thus hindering its adoption. Letting

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-61057-8_30

sensitive operational data traverse organizational boundaries introduces concerns about data secrecy, security, and compliance with internal regulations [27]. To address this issue, the majority of research endeavors have focused thus far on the alteration of input data or of intermediate analysis by-products, with the aim to impede the counterparts from reconstructing the original information sources [18,17,16,15]. These preemptive solutions have the remarkable merit of neutralizing information leakage by malicious parties a priori. Nevertheless, they entail an ex-ante information loss, thus compromising downstream process mining capabilities [18,17], or require the execution of computationally heavy protocols undermining scalability [16,15].

To overcome these limitations, we propose CONFINE, a novel approach and tool aimed at enhancing collaborative information system architectures with secrecy-preserving process mining capabilities. To secure information secrecy during the exchange and elaboration of data, our solution resorts to *Trusted Execution Environments* (TEEs) [30], namely hardware-secured contexts that guarantee code integrity and data confidentiality before, during, and after their utilization. Owing to these characteristics, CONFINE lets information be securely transferred beyond the organizations' borders. Therefore, computing nodes other than the information provisioners can aggregate and elaborate the original, unaltered process data in a secure, externally inaccessible vault. Also, CONFINE is capable of providing these guarantees while demanding scalable computational overhead.

The decentralized architecture of CONFINE supports a four-staged protocol: *(i)* The initial exchange of preliminary metadata, *(ii)* the attestation of the mining entities, *(iii)* the secure transmission and secrecy-preserving merge of encrypted information segments amid multiple parties, *(iv)* the isolated and verifiable computation of process discovery algorithms on joined data. We evaluate our proof-of-concept implementation against synthetic and real-world data with a convergence test followed by experiments to assess the scalability of our approach. Since TEEs operate with dedicated memory pages shielded from access by external entities (operating system included), thus entailing a hardware constraint on computation space, we endow our experiments with an analysis of memory usage, too.

The remainder of this paper is as follows. Sect. 2 provides an overview of related work. In Sect. 3, we introduce a motivating use-case scenario in healthcare. We present the CONFINE approach in Sect. 4. We describe the implementation of our approach in Sect. 5. In Sect. 6, we report on the efficacy and efficiency tests for our solution. Finally, we conclude our work and outline future research directions in Sect. 7.

2 Related Work

The scientific literature already includes noticeable contributions to process mining in a decentralized setting with a focus on data secrecy, despite the relative recency of this research branch across process mining and collaborative information systems. The work of Müller et al. [28] revolves around data privacy and security within third-party systems that mine data generated from external providers on demand. To safeguard the integrity of data earmarked for mining purposes, their research introduces a conceptual architecture that entails the execution of process mining algorithms within a cloud service environment, fortified with Trusted Execution Environments. Drawing inspiration from this foundational contribution, our research work seeks to design a decentralized approach

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-61057-8_30

characterized by organizational autonomy in the execution of process mining algorithms, devoid of synchronization mechanisms taking place between the involved parties. A notable departure from the framework of Müller et al. lies in the fact that here each participating organization retains the discretion to choose when and how mining operations are conducted. Moreover, we bypass the idea of fixed roles, engineering a peer-to-peer scenario in which organizations can simultaneously be data provisioners or miners. Fahrenkrog-Petersen et al. [18,17] theorize the PRETSA algorithms family, namely a set of event log sanitization techniques that perform step-wise transformations of prefix-tree event log representation into a sanitized output ensuring *k-anonimization* and *t-closeness*. While these algorithms effectively minimize information loss, they introduce targeted approximations within the original event log, which may compromise the exactness of process mining results or inhibit mining tasks. In contrast, our research proposes an architecture wherein secure computational vaults collect event logs devoid of upstream alterations and protect them at runtime, thus generating results derived directly from the original information source. Elkoumy et al. [16,15] present Shareprom. Like our work, their solution offers a means for independent entities to execute process mining algorithms in inter-organizational settings while safeguarding the proprietary input data from exposure to external parties operating within the same context. Shareprom's functionality, though, is confined to the execution of operations involving event log abstractions [3] represented as directed acyclic graphs, which the parties employ as intermediate pre-elaboration to be fed into secure multiparty computation (SMPC) [12]. As the authors remark, relying on this specific graph representation imposes constraints that may turn out to be limiting in a number of process mining scenarios. In contrast, our approach allows for the secure, ciphered transmission of event logs (or segments thereof) to process mining nodes. Moreover, SMPC-based solutions require computationally intensive operations and synchronous cooperation among multiple parties, which make these protocols challenging to manage as the number of participants scales up [37]. In our research work, individual computing nodes run the calculations, thus not requiring synchronization with other machines once the input data is loaded.

We are confronted with the imperative task of integrating event logs originating from different data sources and reconstructing consistent traces that describe collaborative process executions. Consequently, we engage in an examination of methodologies delineated within the literature, each of which offers insights into the merging of event logs within inter-organizational settings. The work of Claes et al. [10] holds particular significance for our research efforts. Their seminal study introduces a two-step mechanism operating at the structured data level, contingent upon the configuration and subsequent application of merging rules. Each such rule indicates the relations between attributes of the traces and/or the activities that must hold across distinct traces to be combined. In accordance with their principles, our research incorporates a structured data-level merge based on case references and timestamps as merging attributes. The research by Hernandez et al. [20] posits a methodology functioning at the raw data level. Their approach represents traces and activities as *bag-of-words* vectors, subject to cosine similarity measurements to discern links and relationships between the traces earmarked for combination. An appealing aspect of this approach lies in its capacity to generalize the challenge of merging without necessitating a-priori knowledge of the underlying semantics inherent to the logs

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-61057-8_30

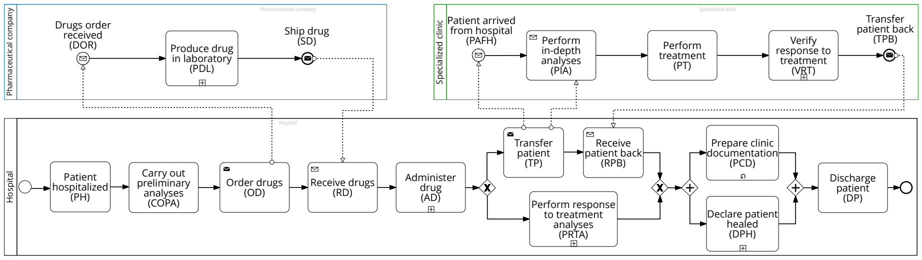

Pharmaceutical company
Drugs order received (DOR)
Produce drug in laboratory (PDL)
Ship drug (SD)
Specialized clinic
Patient arrived from hospital (PAFH)
Perform in-depth analyses (PIA)
Perform treatment (PT)
Verify response to treatment (VRT)
Transfer patient back (TPB)
Hospital
Patient hospitalized (PH)
Carry out preliminary analyses (COPA)
Order drugs (OD)
Receive drugs (RD)
Administer drug (AD)
Transfer patient (TP)
Receive patient back (RPB)
Perform response to treatment analyses (PRTA)
Prepare clinic documentation (PCD)
Declare patient healed (DPH)
Discharge patient (DP)

Fig. 1: A BPMN collaboration diagram of a simplified healthcare scenario

Table 1: Events from cases 312 (Alice) and 711 (Bob) recorded by the hospital, the specialized clinic, and the pharmaceutical company

Hospital		
Case	Timestamp	Activity
312	2022-07-14T10:36	PH
312	2022-07-14T16:36	COPA
711	2022-07-14T17:21	PH
312	2022-07-14T17:36	OD
711	2022-07-14T23:21	COPA
711	2022-07-15T00:21	OD
711	2022-07-15T18:55	RD
312	2022-07-15T19:06	RD
711	2022-07-15T20:55	AD
312	2022-07-15T21:06	AD
312	2022-07-15T22:06	TP
711	2022-07-16T00:55	PRTA
711	2022-07-16T01:55	PCD
711	2022-07-16T02:55	DPH
711	2022-07-16T04:55	DP
312	2022-07-16T07:06	RPB
312	2022-07-16T09:06	DPH
312	2022-07-16T10:06	PCD
312	2022-07-16T11:06	DP

Pharmaceutical company		
Case	Timestamp	Activity
312	2022-07-15T09:06	DOR
711	2022-07-15T09:30	DOR
312	2022-07-15T11:06	PDL
711	2022-07-15T11:30	PDL
312	2022-07-15T13:06	SD
711	2022-07-15T13:30	SD

Specialized clinic		
Case	Timestamp	Activity
312	2022-07-16T00:06	PAFH
312	2022-07-16T01:06	PIA
312	2022-07-16T03:06	PT
312	2022-07-16T04:06	VRT
312	2022-07-16T05:06	TPB

$T_{312} = \langle$ PH, COPA, OD, DOR, PDL, SD, RD, AD, TP, PAFH, PIA, PT, VRT, TPB, RPB, DPH, PCD, DP $\rangle$
$T_{711} = \langle$ PH, COPA, OD, DOR, PDL, SD, RD, AD, PRTA, PCD, DPH, DP $\rangle$

under consideration. However, it entails computational overhead in the treatment of data that can interfere with the overall effectiveness of our approach.

3 Motivating Scenario

To provide a running example and motivating scenario for our investigation, we focus on a simplified hospitalization process for the treatment of rare diseases. The process model is depicted as a BPMN diagram in Fig. 1 and involves the cooperation of three parties: a hospital, a pharmaceutical company, and a specialized clinic. For the sake of simplicity, we describe the process through two cases, recorded by the information systems as in Table 1. Alice's journey (**case 312**) begins when she enters the hospital for the preliminary examinations (patient hospitalized, PH). The hospital then places an order for the drugs (OD) to the pharmaceutical company for treating Alice's specific condition. Afterwards, the pharmaceutical company acknowledges that the drugs order is received (DOR), proceeds to produce the drugs in the laboratory (PDL), and ships the drugs (SD) back to the hospital. Upon receiving the medications, the hospital administers the drug (AD), and conducts an assessment to determine if Alice can be treated internally. If specialized care is required, the hospital transfers the patient (TP) to the specialized clinic. When the patient arrives from the hospital (PAFH), the specialized clinic performs in-depth analyses (PIA) and proceeds with the treatment (PT). Once the specialized clinic had completed the evaluations and verified the response to the treatment (VRT), it transfers the patient back (TPB). The hospital receives the patient back (RPB) and prepares the clinical documentation (PCD). If Alice has successfully recovered, the hospital declares the patient as healed (DPH). When Alice's treatment is complete, the hospital discharges the patient (DP). Bob (**case 711**) enters the hospital a few hours later. His hospitalization process is similar to Alice's. However, he does not need specialized care, and his case is only treated by the hospital. Therefore, the hospital performs the response to treatment analyses (PRTA) instead of transferring him to the specialized clinic.

Both the National Institute of Statistics of the country in which the three organizations reside and the University that hosts the hospital wish to uncover information on this inter-organizational process for reporting and auditing purposes via process analytics [21]. The involved organizations share the urge for such an analysis and wish to be able to repeat the mining task also in-house. The hospital, the specialized clinic, and the pharmaceutical company have a partial view of the overall unfolding of the inter-organizational process as they record the events stemming from the parts of their pertinence. In Table 1, we show cases 312 and 711 and the corresponding traces recorded by the hospital (i.e., T^H_{312} and T^H_{711}), the specialized clinic (i.e., T^S_{312} and T^S_{711}), and the pharmaceutical company (i.e., T^C_{312} and T^C_{711}). Those traces are projections of the two combined ones for the whole inter-organizational process: $T_{312} = \langle$PH, COPA, OD, DOR, PDL, SD, RD, AD, TP, PAFH, PIA, PT, VRT, TPB, RPB, DPH, PCD, DP$\rangle$ and $T_{711} = \langle$PH, COPA, OD, DOR, PDL, SD, RD, AD, PRTA, PCD, DPH, DP$\rangle$. Results stemming from the analysis of the local cases would not provide a full picture. Data should be merged. However, to safeguard the confidentiality of the information, the involved parties cannot give other organizations open access to their traces. The diverging interests (being able to conduct process mining on data from multiple sources without giving away the local event logs in-clear) motivate our research.

We remark that the problem we aim to solve spans across an array of domains beyond healthcare. It particularly applies to scenarios in which one or more parties are interested in process analytics outcomes based on data they bear but cannot be disclosed to the other process actors or to the miners. In the supply chain realm, e.g., the extraction of aggregate knowledge about trends and management guidelines is called for, but the acquisition of competitive advantage out of knowledge leakage must be prevented [33]. In personal informatics, company-wide work routine monitoring and analysis are desirable, though the details of individual participants should be sheltered from inquisitive inspections [31].

4 Design

Our goal is to enable the secure aggregation and elaboration of original, unaltered event logs from decentralized sources in dedicated environments that potentially lie beyond the individual organizations' information perimeter. With this objective in mind, we devise the `Secure Miner` component, which is capable of safeguarding data merge and processing by running certified code in an isolated execution vault. Thus, we decouple provisioning from treatment, and the two tasks can be carried out by distinct computing nodes. Here, we introduce CONFINE's key components, with a special focus on the `Secure Miner`.

The CONFINE architecture at large. Our architecture involves different information systems running on multiple machines. An organization can take at least one of the following roles: **provisioning** if it delivers local event logs to be collaboratively mined; **mining** if it applies process mining algorithms using event logs retrieved from provisioners. Figure 2 depicts the high-level schematization of the CONFINE framework. In our solution, each or-

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-61057-8_30

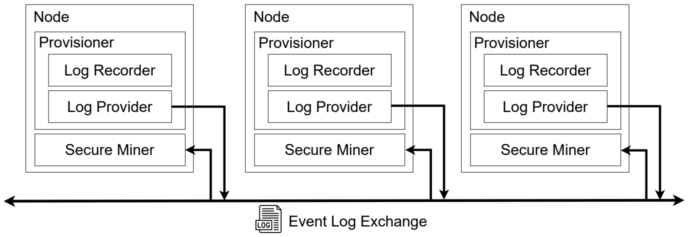


Fig. 2: The CONFINE high-level architecture

ganization hosts one or more nodes encompassing diverse components (the names of which will henceforth be formatted with a `teletype` font). Depending on the played role, nodes come endowed with a `Provisioner` or a `Secure Miner`, or both. The `Provisioner` component, in turn, consists of the following two sub-components. The `Log Recorder` registers the events taking place in the organizations' systems. The `Log Provider` delivers on-demand data to miners. The hospital and all other parties in our example record Alice and Bob's cases using the `Log Recorder`. The `Log Recorder`, in turn, is queried by the `Log Provider` for event logs to be made available for mining. The latter controls access to local event logs by authenticating data requests by miners and rejecting those that come from unauthorized parties. In our motivating scenario, the specialized clinic, the pharmaceutical company, and the hospital leverage `Log Providers` to authenticate the miner before sending their logs. The `Secure Miner` component shelters external event logs inside a protected environment to preserve data confidentiality and integrity. Notice that `Log Providers` accept requests issued solely by `Secure Miners`. Next, we provide an in-depth focus on the latter.

The Secure Miner. The primary objective of the `Secure Miner` is to allow miners to securely execute process mining algorithms using event logs retrieved from provisioners (the specialized clinic, the pharmaceutical company, and the hospital in our example). `Secure Miners` are isolated components that guarantee data inalterability and confidentiality.

Figure 3 illustrates a schematization of the `Secure Miner`, which consists of four sub-components: *(i)* the `Log Requester`; *(ii)* the `Log Receiver`; *(iii)* the `Log Manager`; *(iv)* the `Log Elaborator`. The `Log Requester` and the `Log Receiver` are the sub-components that we employ during the event log retrieval. `Log Requesters` send authenticable data requests to the `Log Providers`. The `Log Receiver` collects event logs sent by `Log Providers` and entrusts them to the `Log Manager`, securing them from accesses that are external to the `Secure Miner`. Miners of our motivating scenario, such as the university and the national institute of statistics, employ these three components to retrieve and store Alice and Bob's data. The `Log Manager` merges the event data locked in the `Secure Miner` to have a global view of the inter-organizational process comprehensive of activities executed by each involved party. The `Log Elaborator` executes process mining algorithms in a protected environment, inaccessible from the outside computation environment. In our motivating scenario, the `Log Manager` combines the traces associated with the cases of Alice (i.e., T^{H}_{312}, T^{S}_{312}, and T^{C}_{312}) and Bob (i.e, T^{H}_{711}, T^{S}_{711}, and T^{C}_{711}),

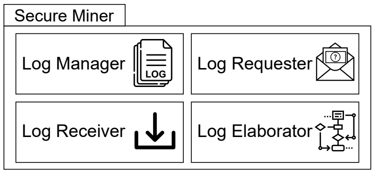


Fig. 3: Sub-components of the Secure Miner

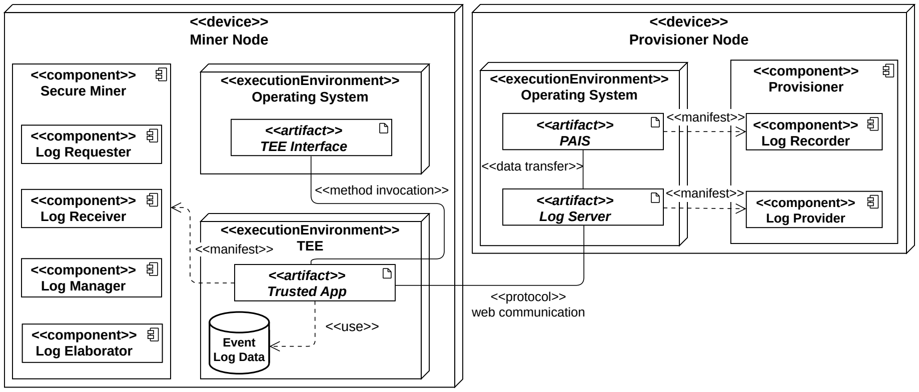


Fig. 4: UML deployment diagram of the CONFINE architecture

generates the chronologically sorted traces T_{312} and T_{711}, and feeds them into the Log Elaborator's mining algorithms (see the bottom-right quadrant of Table 1).

5 Realization

Thus far, we have outlined the main functionalities of each component at large. Here we discuss the technical aspects concerning the realization of our solution. We first present the technologies through which we enable the design principles in Sect. 4. Then, we discuss the CONFINE interaction protocol. Finally, we show the implementation details.

5.1 Deployment

Figure 4 depicts a UML deployment diagram [24] to illustrate the employed technologies and computation environments. We recall that the Miner and Provisioner nodes are drawn as separated, although organizations can host both. In our motivating scenario, e.g., the hospital can be equipped with machines aimed for both mining and provisioning.

Provisioner Nodes host the Provisioner's components, i.e., the Log Recorder and the Log Provider. The Process-Aware Information System (PAIS) manifests the Log Recorder [14]. The PAIS grants access to the Log Server, enabling it to retrieve event log data. The Log Server, on the other hand, embodies the functionalities of the Log Provider, implementing services that handle remote data requests and provide event log data to the miners. The Miner Node is characterized by two distinct *execution environments*: the Operating System (OS) and the Trusted Execution Environment (TEE) [30]. TEEs establish isolated contexts separate from the OS, safeguarding code and data through hardware-based encryption mechanisms. This technology relies on dedicated sections of a CPU that handle encrypted data within a reserved section of the main memory [11].

By enforcing memory access restrictions, TEEs aim to prevent one application from reading or altering the memory space of another, thus enhancing system security. These dedicated areas in memory are limited, though. Once the limits are exceeded, TEEs have

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-61057-8_30

to scout around in outer memory areas, thus conceding the opportunity to malicious readers to understand the saved data based on the reads and writes. To avoid this risk, TEE implementations often raise errors that halt the program execution when memory demand goes beyond the available space. Therefore, the design of secure systems that resort to TEEs must take into account that memory consumption must be kept under control.

We leverage the security guarantees provided by TEEs [22] to protect a `Trusted App` responsible for fulfilling the functions of the `Secure Miner` and its associated sub-components. Our `TEE` component ensures the integrity of the `Trusted App` code, protecting it against potential malicious manipulations and unauthorized access by programs running within the `OS`. Additionally, we utilize the isolated environment of the `TEE` to securely store event log data (e.g., Alice and Bob's cases). The `TEE` retains a private key in its inaccessible memory section, paired with a public key in a Rivest-Shamir-Adleman (RSA) [29] scheme for attestation (only the owner of the private key can sign messages in a way that is verifiable via the public key) and secure message encryption (only the owner of the private key can decode messages that are encrypted with the corresponding public key). In our solution, access to data located in the `TEE` is restricted to the sole `Trusted App`. Users interact with the `Trusted App` through the `TEE Interface`, which serves as the exclusive communication channel. The `Trusted App` offers secure methods, invoked by the `Trusted App Interface`, for safely receiving information from the `OS` and outsourcing the results of computations.

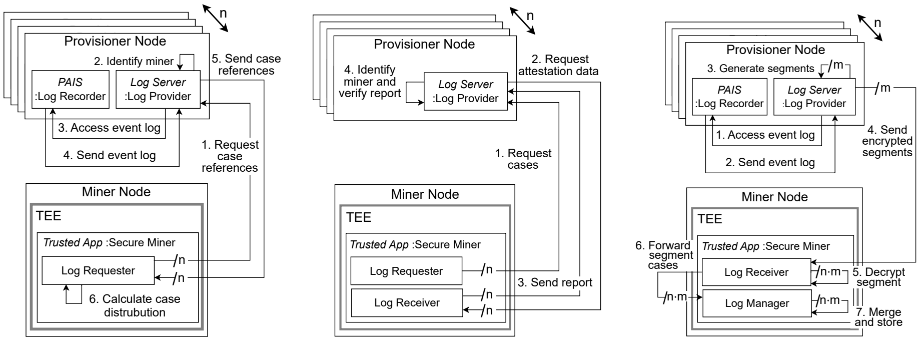
5.2 The CONFINE protocol

We orchestrate the interaction of the components in CONFINE via a protocol, which consists of four subsequent stages: *(i) initialization*, *(ii) remote attestation*, *(iii) data transmission*, and *(iv) computation*. These stages are depicted in Figs. 5(a) to 5(c) and 6, respectively. They are mainly enacted by a `Miner Node` (multiple instances of which can be deployed in a decentralized fashion) and n `Provisioner Nodes`. We assume their communication channel is reliable [9] and secure [23]. In the following, we describe each of the above phases in detail.

Initialization. The objective of the initialization stage is to inform the miner about the distribution of cases related to a business process among the `Provisioner Nodes`. At the onset of this stage, the `Log Requester` within the `Trusted App` issues n requests, one per `Log Server` component, to retrieve the list of case references they record (step 1 in Fig. 5(a)). Following sender authentication (2), each `Log Server` retrieves the local event log from the `PAIS` (3, 4) and subsequently responds to the `Log Requester` by providing a list of its associated case references (5). After collecting these n responses, the `Log Requester` delineates the distribution of cases. In the context of our motivating scenario, by the conclusion of the initialization, the miner gains knowledge that the case associated with Bob, synthesized in the traces T^{H}_{711} and T^{C}_{711}, is exclusively retained by the hospital and the specialized clinic. In contrast, the traces of Alice's case, denoted as T^{H}_{312}, T^{C}_{312}, and T^{S}_{312}, are scattered across all three organizations.

Remote attestation. The remote attestation serves the purpose of establishing trust between miners and provisioners in the context of fulfilling data requests. This phase adheres to the overarching principles outlined in the RATS RFC standard [8] serving as the foundation

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-61057-8_30



(a) Initialization (b) Remote attestation (c) Data transmission

Fig. 5: Unfolding example for the initialization, remote attestation and data transmission phases of the CONFINE protocol

for several TEE attestation schemes (e.g., Intel EPID[3] and AMD SEV-SNP[4]). Remote attestation has a dual objective: *(i)* to furnish provisioners with compelling evidence that the data request for an event log originates from a `Trusted App` running within a TEE; *(ii)* to confirm the specific nature of the `Trusted App` as an authentic `Secure Miner` software entity. This phase is triggered when the `Log Requester` sends a new case request to the `Log Server`, specifying: *(i)* the segment size (henceforth, *seg_size*), and *(ii)* the set of the requested case references. Both parameters will be used in the subsequent *data transmission* phase. Each of the n `Log Servers` commences the verification process by requesting the necessary information from the `Log Receiver` to conduct the attestation (2). Subsequently, the `Log Receiver` generates the attestation report containing the so-called *measurement* of the `Trusted App`, which is defined as the hash value of the combination of its source code and data. Once this report is signed using the attestation private key associated with the `TEE`'s hardware of the `Miner Node`, it is transmitted by the `Log Receiver` to the `Log Servers` alongside the attestation public key of the `Miner Node` (3). The `Log Servers` authenticate the miner using the public key and decrypt the report (4). In this last step, the `Log Servers` undertake a comparison procedure in which they juxtapose the measurement found within the decrypted report against a predefined reference value associated with the source code of the `Secure Miner`. If the decrypted measurement matches the predefined value, the `Miner Node` gains trust from the provisioner.

Data transmission. Once the trusted nature of the `Trusted App` is verified, the `Log Servers` proceed with the transmission of their cases. To accomplish this, each `Log Server` retrieves the event log from the `PAIS` (steps 1 and 2, in Fig. 5(c)), and filters it according to the case reference set specified by the miner. Given the constrained workload capacity of the `TEE`, `Log Servers` could be requested to partition the filtered event log into m distinct segments. Log segments contain a variable count of entire cases (3). The cumulative size of these segments is governed by the threshold parameter specified by the miner in the initial request (step 1 of the remote attestation phase, Fig. 5(b)). As an

[3] sgx101.gitbook.io/sgx101/sgx-bootstrap/attestation. Accessed: 05/04/2024.

[4] amd.com/en/processors/amd-secure-encrypted-virtualization. Accessed: 05/04/2024.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-61057-8_30

illustrative example from our motivating scenario, the Log Server of the hospital may structure the segmentation such that T^H_{312} and T^H_{711} are in the same segment, whereas the specialized clinic might have T^S_{312} and T^S_{711} in separate segments. Subsequently, the n Log Servers transmit their m encrypted segments to the Log Receiver of the Trusted App (4). The Log Receiver, in turn, collects the $n \times m$ responses in a queue, processing them one at a time. After decrypting a processed segment (5), the Log Receiver forwards the cases contained therein to the Log Manager (6). Data belonging to the same process instance are merged by the Log Manager to build a single trace (e.g., T_{312}) comprehensive of all the events in the partial traces (T^H_{312}, T^S_{312} and T^C_{312}). To do so, the Log Manager applies a specific *merging schema* (i.e., a rule specifying the attributes that identify a case) as stated in [10]. In our illustrative scenario, the merging schema to combine the cases of Alice is contingent upon the linkage established through their case identifier (312). We underline that our solution facilitates the incorporation of diverse merging schemas encompassing distinct trace attributes. The outcomes arising from merging the cases within the processed segments are securely stored by the Log Manager in the TEE.

Computation. The Trusted App requires all the provisioners to have delivered data referring to the same process instances. For example, when the hospital and the other organizations have all delivered their information concerning case 312 to the Trusted App, the process instance associated with Alice becomes eligible for the computation phase, illustrated in Fig. 6. The Log Manager forwards the cases earmarked for computation to the Log Elaborator (step 1). These cases may constitute either the entire merged event log or a subset thereof. The former setting entails a single computation routine, thus saving execution time but requiring a larger memory buffer in the TEE, whereas the latter necessitates multiple consecutive elaborations with a lower demand for space. Subsequently, the Log Elaborator proceeds to input the merged cases into the process mining algorithm (2). Notice that the above choice on the buffering of cases affects the selection of the mining algorithm to employ. If we elaborate subsequent batches, each containing a part of all merged cases, the mining algorithm must support incremental processing, enriching the output as new batches come along. An example of this class of algorithms is the HeuristicsMiner [36]. Otherwise, incrementality is not required. Ultimately, the outcome of the computation is relayed by the Log Elaborator from the TEE to the TEE Interface running atop the Operating System of the Miner Node (3). In our motivating scenario, the university and the national institute of statistics, serving as miners, disseminate the outcomes of computations, generating analyses that benefit the provisioners, although the original data are never revealed in clear. Furthermore, our protocol enables the potential for provisioners to have their own Secure Miner, allowing them autonomous control over the computed results. Notice that the CONFINE protocol does not impose restrictions on the post-computational handling of results.

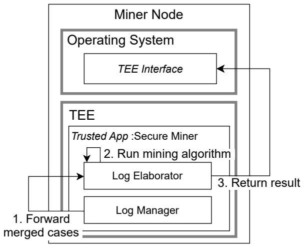


Fig. 6: Computation phase of the CONFINE protocol

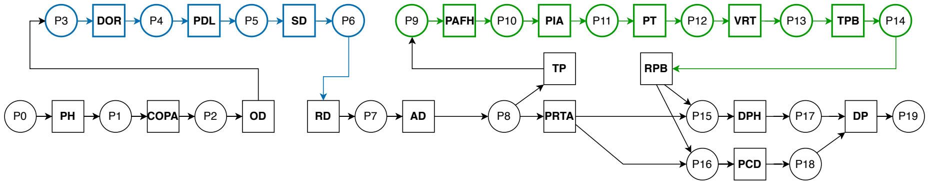


Fig. 7: HeuristicsMiner output with CONFINE

5.3 Implementation

We implemented the `Secure Miner` component as an Intel SGX[5] trusted application, encoded in Go through the EGo framework.[6] We resort to a TLS communication channel [34] between miners and provisioners over the HTTP web protocol to secure the information exchange. To demonstrate the effectiveness of our framework, we re-implemented and integrated the HeuristicsMiner discovery algorithm [36] within the `Trusted Application`. Our implementation of CONFINE, including the HeuristicsMiner implementation in Go, is openly accessible at the following URL: github.com/Process-in-Chains/CONFINE/.

6 Evaluation

In this section, we evaluate our approach through our implementation. We begin with a convergence analysis to demonstrate the correctness of the data exchange process. As discussed in Sect. 5.1, the availability of space in the dedicated TEE areas is subject to hardware limitations. Therefore, we focus on memory consumption, as exceeding those limits could lower the level of security guaranteed by TEEs. Thus, we gauge the memory usage with synthetic and real-life event logs, to observe the trend during the enactment of our protocol and assess scalability. We discuss our experimental results in the following. All the testbeds and results are available in our public code repository (linked above).

Output convergence. To experimentally validate the correctness of our approach in the transmission and computation phases (see Sect. 5), we run a *convergence* test. To this end, we created a synthetic event log consisting of 1000 cases of 14 events on average (see Table 2) by simulating the inter-organizational process of our motivating scenario (see Fig. 1)[7] and we partitioned it in three sub-logs (one per involved organization), an excerpt of which is listed in Table 1. We run the stand-alone HeuristicsMiner on the whole log, and processed the sub-logs through our CONFINE toolchain. As expected, the results converge and are depicted in Fig. 7 in the form of a workflow net [1]. For clarity, we have colored activities recorded by the organizations following the scheme of Table 2 (black for the hospital, blue for the pharmaceutical company, and green for the specialized clinic).

Memory usage. Figures 8(a) and 8(b) display plots corresponding to the runtime space utilization of CONFINE (in MegaBytes). Differently from Fig. 8(b), Fig. 8(a) excludes the

[5] sgx101.gitbook.io/sgx101/. Accessed: 05/04/2024.

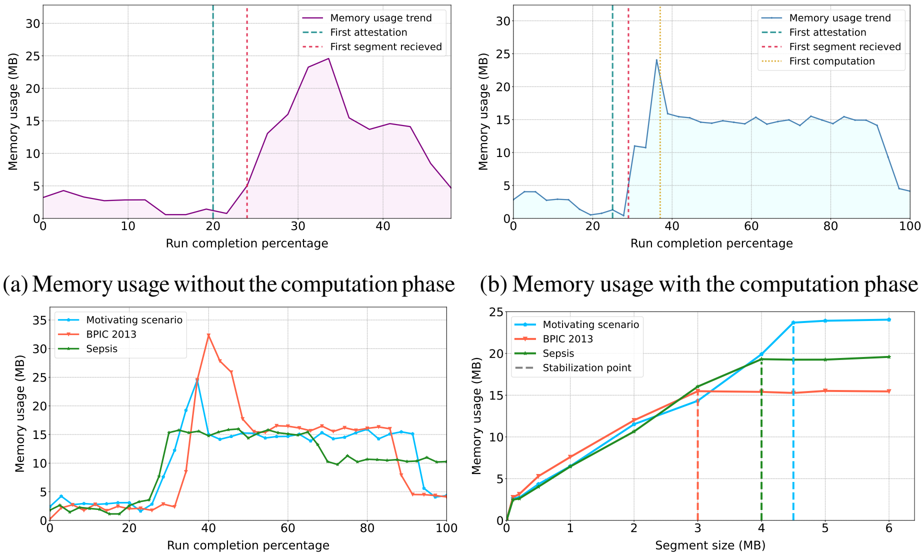
[6] docs.edgeless.systems/ego. Accessed: 05/04/2024.

[7] We generated the event log through BIMP (https://bimp.cs.ut.ee/). We filtered the generated log by keeping the sole events that report on the completion of activities, and removing the start and end events of the pharmaceutical company and specialized clinic's sub-processes.

Pre-print copy of the manuscript published by Springer (available at `link.springer.com`)
identified by DOI: `10.1007/978-3-031-61057-8_30`

Table 2: Event logs used for our experiments

Name	**Type**	**Activities**	**Cases**	**Max** events	**Min** events	**Avg.** events	**Organization** ↦ **Activities**
Motivating scenario	Synthetic	19	1000	18	9	14	$\mathscr{O}^P \mapsto 3$, $\mathscr{O}^C \mapsto 5$, $\mathscr{O}^H \mapsto 14$
Sepsis [26]	Real	16	1050	185	3	15	$\mathscr{O}^1 \mapsto 1$, $\mathscr{O}^2 \mapsto 1$, $\mathscr{O}^3 \mapsto 14$
BPIC 2013 [32]	Real	7	1487	123	1	9	$\mathscr{O}^1 \mapsto 6$, $\mathscr{O}^2 \mapsto 7$, $\mathscr{O}^3 \mapsto 6$



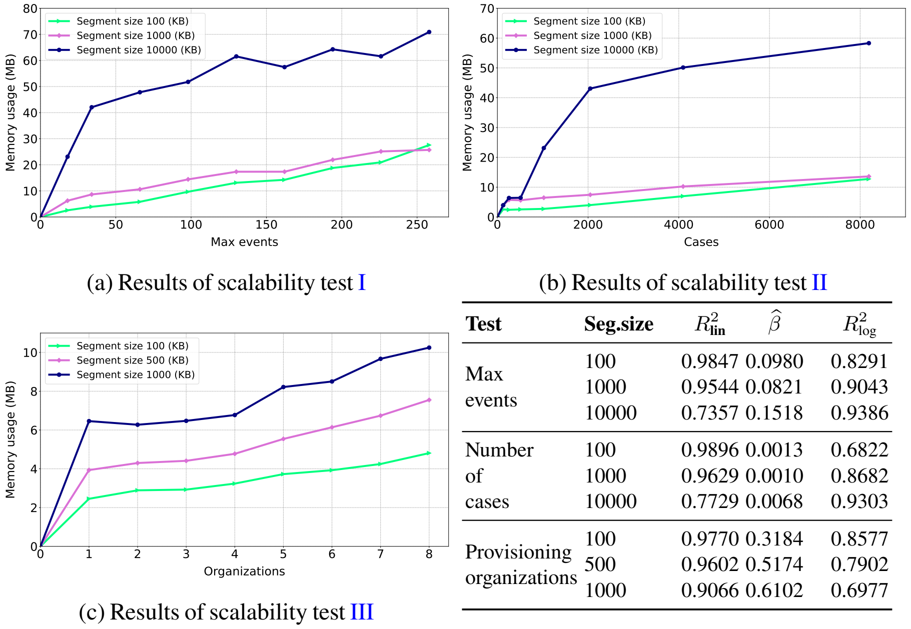
(a) Memory usage without the computation phase (b) Memory usage with the computation phase

(c) Memory usage with three event logs (d) Segment size impact on memory usage

Fig. 8: Memory usage test results

computation stage by leaving the HeuristicsMiner inactive so as to isolate the execution from the mining-specific operations. The dashed lines mark the starting points for the remote attestation, data transmission and computation stages. We held the segment size (*seg_size*) constant at 2 MegaBytes. We observe that the data transmission stage reaches the highest peak of memory utilization, which is then partially freed by the subsequent computation stage, steadily occupying memory space at a lower level. To verify whether this phenomenon is due to the synthetic nature of our simulation-based event log, we gauge the runtime memory usage of two public real-world event logs, too: Sepsis [26] and BPIC 2013 [32]. The characteristics of the event logs are summarized in Table 2. Since those are *intra-organizational* event logs, we split the contents to mimic an *inter-organizational* context. In particular, we separated the Sepsis log based on the distinction between normal-care and intensive-care paths, as if they were conducted by two distinct organizations. Similarly, we processed the BPIC 2013 log to sort it out into the three departments of the Volvo IT incident management system. Figure 8(c) depicts the results. We observe that the BPIC 2013 log demands the most memory during the initial stages, whereas the Sepsis log is associated with the least expensive run, but the polylines exhibit a matching shape with

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-031-61057-8_30



(a) Results of scalability test I

(b) Results of scalability test II

(c) Results of scalability test III

Test	Seg.size	R^2_{lin}	$\widehat{\beta}$	R^2_{log}
Max events	100	0.9847	0.0980	0.8291
	1000	0.9544	0.0821	0.9043
	10000	0.7357	0.1518	0.9386
Number of cases	100	0.9896	0.0013	0.6822
	1000	0.9629	0.0010	0.8682
	10000	0.7729	0.0068	0.9303
Provisioning organizations	100	0.9770	0.3184	0.8577
	500	0.9602	0.5174	0.7902
	1000	0.9066	0.6102	0.6977

(d) Scalability measurements

Fig. 9: Scalability test results

our synthetic dataset.To verify whether these trends are affected by the dimension of the exchanged data segments, we conducted an additional test to examine memory usage as the *seg_size* varies. Notably, the polylines displayed in Fig. 8(d) indicate a linear increment of memory occupation until a breakpoint is reached. After that, the memory in use is steady. These points, marked by vertical dashed lines, indicate that the *seg_size* value that allows the providers to send their whole log partition in a single segment.

Scalability. To examine the scalability of the `Secure Miner`, we focus on its capacity to efficiently manage an increasing workload in the presence of limited memory resources (as it is the case with TEEs). We set three distinct test configurations by varying our motivating scenario log. In particular, we considered (I) the maximum number of events per case, (II) the number of cases $|\widehat{\text{CID}}|$, and (III) the number of provisioning organizations $|\widehat{\mathcal{O}}|$ as independent integer variables. To conduct the test on the maximum number of events, we added a loop back from the final to the initial activity of the process model, progressively increasing the number of iterations $2 \leqslant x_{\circlearrowleft} \leqslant 16$ at a step of 2, resulting in $18+16\cdot(x_{\circlearrowleft}-1)$ events. Concerning the test on the number of cases, we simulated additional process instances so that $|\widehat{\text{CID}}| = 2^{x_{cid}}$ having $x_{cid} \in \{7,8,\dots,13\}$. Finally, for the assessment of the number of organizations, the test necessitated the distribution of the process model activities' into a variable number of pools, each representing a different organization ($|\widehat{\mathcal{O}}| \in \{1,2,\dots,8\}$). We parameterized the above configurations with three segment sizes (in KiloBytes): *seg_size* $\in \{100,1000,10000\}$ for tests I and II, and *seg_size* $\in \{100,500,1000\}$ for test III (the range is reduced without loss of generality to compensate the partitioning of activities into multiple organizations). To facilitate a more rigorous interpretation of the output trends across varying *seg_size*s, we employ two well-known statistical measures. As a primary measure of goodness-of-fit, we employ the coefficient of determination R^2 [5], which assesses the degree to which the observed data adheres to the linear (R^2_{lin}) and logarithmic ($R^2_{\log}$) regressions derived from curve fitting approximations. To delve deeper into the analysis of trends exhibiting a high R^2_{lin}, we consider the slope $\widehat{\beta}$ of the approximated linear regression [4].

Table 9(d) lists the obtained measurements, which we use to elucidate the observed patterns. Figure 9(a) depicts the results of test I, focusing on the increase of memory utilization when the number of events in the logs grows. We observe that the memory usage trends for *seg_size* set to 100 and 1000 (depicted by green and lilac lines, respectively) are almost superimposable, whereas the setting with *seg_size* $= 10000$ (blue line) exhibits significantly higher memory usage. With *seg_size* assigned with 100 and 1000, R^2_{lin} approaches 1, signifying an almost perfect approximation of the linear relation. With these settings, $\widehat{\beta}$ is very low yet higher than 0, thus indicating that memory usage is likely to continue increasing as the number of maximum events grows. The configuration with *seg_size* $= 10000$ yields a higher $R^2_{\log}$ value, thus suggesting a logarithmic trend, hence a greater likelihood of stabilizing memory usage growth rate as the number of maximum events increases. In Fig. 9(b), we present the results of test II, assessing the impact of the number of cases on memory consumption. As expected, the configurations with *seg_size* set to 100 and 1000 demand lower memory than settings with *seg_size* $= 10000$. The R^2_{lin} score when *seg_size* is assigned with 100 and 1000 indicate a strong linear relationship between the dependent and independent variables compared to the trend with *seg_size* $= 10000$, which is better described by a logarithmic regression ($R^2_{\log} = 0.9303$). Differently from test I, the $\widehat{\beta}$ score associated with the linear approximations with *seg_size* set to 100 and 1000 approaches 0, indicating that the growth rate of memory usage as the number of cases increases is negligible. In Fig. 9(c), we present the results of test III, on the relation between the number of organizations and memory usage. The chart shows that memory usage trends increase as provisioning organizations increase for all three segment sizes. The R^2_{lin} values for the three *seg_size*s are very high, indicating a strong positive linear correlation. The test with *seg_size* $= 100$ exhibits the slowest growth rate, as corroborated by the lowest $\widehat{\beta}$ (0.3184). For the configuration with *seg_size* $= 500$, the memory usage increases slightly faster ($\widehat{\beta} = 0.5174$). With *seg_size* $= 1000$, the overall memory usage increases significantly faster than the previous configurations ($\widehat{\beta} = 0.6102$). We derive from these findings that the `Secure Miner` may encounter scalability issues when handling settings with a large number of provisioning organizations. Further investigation is warranted to determine the precise cause of this behavior and identify potential mitigation strategies.

In the next section, we discuss other future endeavors stemming from our work.

7 Conclusion and Future Work

In this paper, we described CONFINE, a decentralized approach to process mining. Based upon TEEs, it guarantees the secrecy and confidentiality of data transmitted to and elaborated by processing nodes outside the perimeter of the event log providers. Our research can spur a number of future investigations and improvements in the field. First, we

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-61057-8_30

aim to enhance our solution by readjusting it to the relaxation of underlying assumptions we made, including fair conduct by data provisioners, the absence of injected or maliciously manipulated event logs, the exchange of messages through reliable communication channels where no loss or bit corruption occurs, and the existence of a universal clock for timestamps. Also, we are extending our analysis with formal proofs of soundness and completeness of the protocol. Our future work encompasses the integration of usage control policies that specify rules on event logs' utilization, too. We plan to design enforcement and monitoring mechanisms to achieve this goal following the principles adopted in [6,7]. We remark that a possibly severe threat to data secrecy lies in the reconstruction of the original input information back from the mining output. Keeping this aspect in mind is crucial to determine the mining algorithm to be embedded in the `Secure Miner`. Studies in this regard have been conducted, among others, in [35,19]. Integrating the proposed recommendations with CONFINE paves the path for future investigations. Finally, we acknowledge that the focus of our implementation is on a specific process discovery task. Nevertheless, our approach has the potential to seamlessly cover a wider array of discovery techniques as well as other process mining functionalities like conformance checking and performance analysis. Showing their integrability with our approach, and drawing guidelines on the use of different algorithms, are research directions we plan to follow.

Acknowledgments. The authors thank Giuseppe Ateniese for the fruitful discussion and insights. This work was partly funded by MUR under PRIN grant B87G22000450001 (PINPOINT), the Latium Region under PO FSE+ grant B83C22004050009 (PPMPP), and Sapienza University of Rome under grant RG123188B3F7414A (ASGARD).

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: ICATPN. pp. 407–426 (1997)
2. van der Aalst, W.M.P.: Intra-and inter-organizational process mining: Discovering processes within and between organizations. In: PoEM. pp. 1–11 (2011)
3. van der Aalst, W.M.P.: Federated process mining: Exploiting event data across organizational boundaries. In: SMDS 2021. pp. 1–7 (2021)
4. Altman, N., Krzywinski, M.: Simple linear regression. Nature Methods pp. 999–1000 (2015)
5. Barrett, J.P.: The coefficient of determination—some limitations. The American Statistician pp. 19–20 (1974)
6. Basile, D., Di Ciccio, C., Goretti, V., Kirrane, S.: Blockchain based resource governance for decentralized web environments. Frontiers in Blockchain p. 1141909 (2023)
7. Basile, D., Di Ciccio, C., Goretti, V., Kirrane, S.: A blockchain-driven architecture for usage control in solid. In: ICDCSW. pp. 19–24 (2023)
8. Birkholz, H., Thaler, D., Richardson, M., et al.: Remote ATtestation procedureS (RATS) Architecture (2023)
9. Cachin, C., Guerraoui, R., Rodrigues, L.E.T.: Introduction to Reliable and Secure Distributed Programming (2. ed.). Springer (2011)
10. Claes, J., Poels, G.: Merging event logs for process mining: A rule based merging method and rule suggestion algorithm. Expert Syst. Appl. **41**(16), 7291–7306 (2014)
11. Costan, V., Devadas, S.: Intel SGX explained. Cryptology ePrint Archive (2016)
12. Cramer, R., Damgård, I., Nielsen, J.B.: Secure Multiparty Computation and Secret Sharing. Cambridge University Press (2015)
13. De Weerdt, J., Wynn, M.T.: Foundations of process event data. In: Process Mining Handbook, pp. 193–211. Springer (2022)

14. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018)
15. Elkoumy, G., Fahrenkrog-Petersen, S.A., et al.: Secure multi-party computation for inter-organizational process mining. In: BPMDS/EMMSAD. pp. 166–181 (2020)
16. Elkoumy, G., Fahrenkrog-Petersen, S.A., et al.: Shareprom: A tool for privacy-preserving inter-organizational process mining. In: BPM (PhD/Demos). pp. 72–76 (2020)
17. 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)
18. Fahrenkrog-Petersen, S.A., van der Aa, H., Weidlich, M.: Optimal event log sanitization for privacy-preserving process mining. Data Knowl. Eng. **145**, 102175 (2023)
19. Fahrenkrog-Petersen, S.A., Kabierski, M., van der Aa, H., Weidlich, M.: Semantics-aware mechanisms for control-flow anonymization in process mining. Inf. Syst. **114**, 102169 (2023)
20. Hernandez-Resendiz, J.D., Tello-Leal, E., Marin-Castro, H.M., et al.: Merging event logs for inter-organizational process mining. In: New Perspectives on Enterprise Decision-Making Applying Artificial Intelligence Techniques, pp. 3–26. Springer (2021)
21. Jans, M., Hosseinpour, M.: How active learning and process mining can act as continuous auditing catalyst. Int. J. Accounting Inf. Systems **32**, 44–58 (2019)
22. Jauernig, P., Sadeghi, A.R., Stapf, E.: Trusted execution environments: Properties, applications, and challenges. IEEE Secur. Priv. **18**(2), 56–60 (2020)
23. Kamil, A., Lowe, G.: Understanding abstractions of secure channels. In: FAST. pp. 50–64 (2010)
24. Koch, N., Kraus, A.: The expressive power of UML-based web engineering. In: IWWOST02. vol. 16, pp. 40–41 (2002)
25. Liu, C., Li, Q., Zhao, X.: Challenges and opportunities in collaborative business process management: Overview of recent advances and introduction to the special issue. Inf. Syst. Front. **11**, 201–209 (2009)
26. Mannhardt, F.: Sepsis cases - event log (2016). https://doi.org/10.4121/UUID:915D2BFB-7E84-49AD-A286-DC35F063A460
27. Müller, M., Ostern, N., Koljada, et al.: Trust mining: analyzing trust in collaborative business processes. IEEE Access pp. 65044–65065 (2021)
28. Müller, M., Simonet-Boulogne, A., Sengupta, S., Beige, O.: Process mining in trusted execution environments: Towards hardware guarantees for trust-aware inter-organizational process analysis. In: ICPM. pp. 369–381 (2021)
29. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems (reprint). Commun. ACM **26**(1), 96–99 (1983)
30. Sabt, M., Achemlal, M., Bouabdallah, A.: Trusted execution environment: What it is, and what it is not. In: 2015 IEEE TrustCom/BigDataSE/ISPA. pp. 57–64 (2015)
31. Sinik, T., Beerepoot, I., Reijers, H.A.: A peek into the working day: Comparing techniques for recording employee behaviour. In: RCIS. vol. 476, pp. 343–359 (2023)
32. Steeman, W.: BPI challenge 2013, incidents (2013). https://doi.org/10.4121/UUID:500573E6-ACCC-4B0C-9576-AA5468B10CEE
33. Tan, K.H., Wong, W.P., Chung, L.: Information and knowledge leakage in supply chain. Inf. Syst. Frontiers **18**(3), 621–638 (2016)
34. Thomas, S.A.: SSL and TLS Essentials: Securing the Web. Wiley (2000)
35. von Voigt, S.N., Fahrenkrog-Petersen, S.A., et al.: Quantifying the re-identification risk of event logs for process mining - Empiricial evaluation paper. In: CAiSE. pp. 252–267 (2020)
36. Weijters, A.J.M.M., van der Aalst, W.M.P., Alves De Medeiros, A.K.: Process mining with the HeuristicsMiner algorithm (2006)
37. Zhao, C., Zhao, S., Zhao, M., et al.: Secure multi-party computation: Theory, practice and applications. Inf. Sci. **476**, 357–372 (2019)

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-61057-8_30

References

Goretti, Valerio, Davide Basile, Luca Barbaro, and Claudio Di Ciccio (2024). "Trusted Execution Environment for Decentralized Process Mining". In: *CAiSE*. Ed. by Giancarlo Guizzardi, Flávia Maria Santoro, Haralambos Mouratidis, and Pnina Soffer. Vol. 14663. Lecture Notes in Computer Science. Springer, pp. 509–527. ISBN: 978-3-031-61056-1. DOI: 10.1007/978-3-031-61057-8_30.

BibTeX

```
@InProceedings{   Goretti.etal/CAiSE2024:TEEforProcessMining,
  author        = {Goretti, Valerio and Basile, Davide and Barbaro, Luca and
                  Di Ciccio, Claudio},
  booktitle     = {CAiSE},
  title         = {Trusted Execution Environment for Decentralized Process
                  Mining},
  year          = {2024},
  pages         = {509--527},
  crossref      = {CAiSE2024},
  doi           = {10.1007/978-3-031-61057-8_30},
  keywords      = {Collaborative information systems architectures;
                  Inter-organizational process mining; TEE; Confidential
                  computing}
}
@Proceedings{     CAiSE2024,
  title         = {Advanced Information Systems Engineering - 36th
                  International Conference, CAiSE 2024, Limassol, Cyprus,
                  June 3-7, 2024, Proceedings},
  year          = {2024},
  editor        = {Giancarlo Guizzardi and Fl{\'{a}}via Maria Santoro and
                  Haralambos Mouratidis and Pnina Soffer},
  isbn          = {978-3-031-61056-1},
  publisher     = {Springer},
  series        = {Lecture Notes in Computer Science},
  volume        = {14663}
}
```