SN Computer Science (2020) 1:188
https://doi.org/10.1007/s42979-020-00198-9

ORIGINAL RESEARCH

# Risk Management and Privacy Violation Detection in the PoSeID-on Data Privacy Platform

Paulo Silva[1] · Rui Casaleiro[1] · Paulo Simões[1] · Nuno Antunes[1] · Marilia Curado[1] · Edmundo Monteiro[1]

Received: 6 March 2020 / Accepted: 17 May 2020 / Published online: 3 June 2020

**Abstract**
As personal data establishes itself as one of the main resources of our digital society, ways of controlling, monitoring, managing and securing personally identifiable information have become crucial. PoSeID-on is an H2020 European Union project that targets this need. In this paper, we present and discuss PoSeID-on's risk management and personal data analyser strategies, which support core functions of the data privacy platform being developed. In addition to describing the architecture of such functions, we provide experimental results concerning their operation.

**Keywords** Data protection · Privacy · Natural language processing · Personally identifiable information

## Introduction

Processing personal data with the aim of extracting information and knowledge is crucial to an increasing number of applications. Willingly or unwillingly, people are continuously feeding data processors and/or data controllers with items of personal information and, in most cases, user control over who does what with someone's personal data is not practical or inexistent, despite EU's General Data Protection Regulation (GDPR) [1].

In this context, the PoSeID-on H2020 European Project is developing a platform for ensuring protection of and user control over Personally Identifiable Information (PII), in environments where multiple data processors deal with data pertaining to a potentially high number of data subjects. Two key functions of this platform are Risk Management and Personal Data Analysis.

The novelty and relevance of this solution are highlighted by the lack of similar approaches. Therefore, this work is motivated by the need to give data subjects the possibility of transparently and safely controlling their data. To accomplish this, the Risk Management Module (RMM) and the Personal Data Analyser (PDA), together with privacy-enhanced dashboard (PED), aim to empower data subjects and reduce the unnecessary exposure of their PII. This is achieved by allowing the data subject to authorize which service provider's functionality has access and to which specific PII the authorization applies. At the same time, the RMM and PDA autonomously monitor such interactions and warn the data subjects of potentiation privacy risks. The presentation and discussion of the achievements and capabilities of the RMM and the PDA are the main objectives and contributions of this paper.

The paper is organized as follows. "The PoSeID-on Platform" section provides a brief overview of the PoSeID-on platform. Risk management is addressed in "Risk Management" section. "Privacy Violation Detection" section discusses privacy violation detection. "Experimental Results" section presents and analyses some of the obtained results. Conclusions and directions for further work are presented in "Conclusion" section.

This article is part of the topical collection "Privacy, Data Protection and Digital Identity" guest edited by Fernando Boavida, Andrea Praitano and Georgios V. Lioudakis.

Paulo Silva
pmgsilva@dei.uc.pt

[1] Department of Informatics Engineering, CISUC, University of Coimbra, Coimbra, Portugal

## The PoSeID-on Platform

The goal of the PoSeID-on Project ("Protection and control of Secured Information by means of a privacy-enhanced dashboard") [2] is to develop a system for personal data protection, in line with GDPR with respect to digital security.

The project aims to design, implement and validate a privacy-enhancing dashboard for personal data protection, a platform that manages all the personal data transactions between a data subject (owner of personal data) and private or public entities acting as data controllers or data processors. All relevant information shall be made available to users via a user-friendly web dashboard that allows them to track PII, manage PII access permissions, and view the risk level stemming from their data exposure. In order to reduce identity fraud and protect the privacy of users, access to the dashboard is to be made available through electronic identification (eID) accounts only, in line with the Electronic Identification, Authentication and Trust Services (eIDAS) regulation [3].

The PoSeID-on solution is based on technologies such as blockchain, smart contracts, and cloud computing, enabling users to manage personal data and data access authorizations in an easy, secure and auditable way. PoSeID-on aims at impacting society as a whole, as it leads to increased trust in the digital market, in addition to supporting fundamental rights in the digital society.

Through smart contracts, the project aims to meet the need of data confidentiality, inviolability, and access control for data subjects. Through the blockchain technology, references to PII shall be managed and exchanged securely. The blockchain technology was selected due to two main reasons. First and foremost, there was the need to maintain an irrevocable record of PII transactions, including permissions handling and all kinds of operations involving PII processing, for providing full control to PII owners, for accountability, and for legal assurances. On the other hand, there was need to allow multiple entities to share data and to contribute to data processing, without relinquishing control over their own databases, or without relying on a central datastore. By agreeing to participate in the PoSeID-on system, users benefit from full control over their PII, and third parties can provide an auditable ledger of all their PII-related operations to users and regulators. Moreover, it should be highlighted that no PII is ever stored in the blockchain that only stores information on permissions and on PII handling.

Figure 1 illustrates the overall PoSeID-on architecture, identifying the various system components. Table 1 lists the conceptual components and the respective short description.

## Risk Management

Risk detection is performed by combining machine learning algorithms, that analyse multiple sources of information about transactions, user-level behaviour and system-level behaviour, in the form of component logs. When the data subject provides explicit consent, transaction-specific data and PII will also be sent to the Risk Management Module (RMM) and used to complement this analysis. High risk levels may trigger alerts to PoSeID-on administrators and data subjects, depending on the RMM settings.

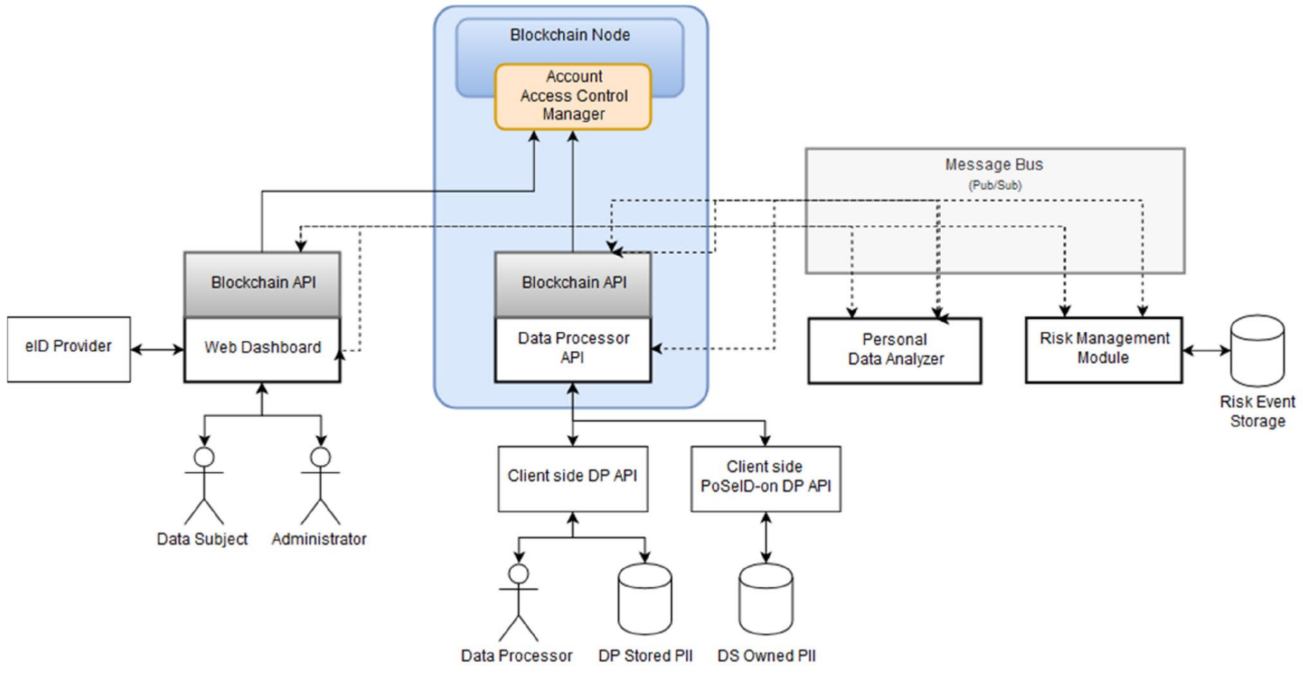


Fig. 1 PoSeID-on overall architecture

**Table 1** PoSeID-on architecture components

| Component | Description |
|---|---|
| Data subjects, data processors, and administrators | Primary target of PoSeID-on platform end-users |
| Dashboard | Interface for data subjects and administrators |
| Data processor API | Access point for data processors to send/receive requests |
| Client-side data processor API | Connector to data processor's internal information systems |
| Permissioned blockchain and smart contracts | Blockchain implementation where only authorized parties can propose changes. Serves as a back-end for PII access management within the PoSeID-on platform |
| Blockchain API | Abstraction layer that allows modules to access and interact with the blockchain |
| Risk Management Module | Detects operational anomalies which may translate to security and privacy risks |
| Personal Data Analyser | Detects and evaluates privacy risks within PII transactions |
| eID provider | Authenticates users in the PoSeID-on platform |
| Data Subjects' PII Repository | PoSeID-on's storage for PII owned by the data subject (e.g. not belonging to a data processor, introduced manually by the data subject into PoSeID-on) |
| Message bus | Messaging module for PoSeID-on's components communication |

The used anomaly detection approach is based on system log analysis [4–6] and follows the framework by Lyu et al. [7]. This framework comprises of four main steps:

- *log collection*—logs in PoSeID-on are delivered through the message bus directly to the RMM, using a custom message protocol. The messages contain the log, structured according to Graylog Extended Log Format (GELF) [8];
- *log parsing*—for the log parsing step, Drain [9], a state-of-the-art online log parsing approach based on fixed depth trees, was implemented. In order to integrate Drain into the RMM, a Java implementation of the algorithm was developed, in order to fully integrate it in the Spark Streaming [10] pipeline;
- *feature extraction*—once logs haven been parsed, each log is assigned to one of the log templates derived from the Drain parsing step it is then necessary to create numerical feature vectors which will be provided to the machine learning models performing the anomaly detection. The current feature vector consists of the occurrence count of each log template in a time window, for each specific block id identified.
- *anomaly detection*—for the anomaly detection step, in initial phases of the RMM deployment, unsupervised learning models are more favourable due to the lack of operational data, such as K-Means and Principal Component Analysis, which do not require previous labelled data.

RMM's architecture (Fig. 2) is based on the Lambda Architecture proposed by Nathan Marz [11]. This architecture considers three layers: batch layer, speed layer, and serving layer.

The batch layer manages a master dataset which is used for historic risk analysis, taking advantage of having a large collection of data, which time span will depend on how long the RMM is allowed to retain data, provided the data subject has given explicit consent for this purpose. The speed layer deploys the models created by the batch layer in addition to the clustering models, and analyses the stream of data in real-time, using the data that arrives between batch analysis. This layer reasons over the data and, in case an anomaly is detected, dispatches a recommendation action request to the serving layer. Finally, the serving layer is in charge of receiving the results from both the batch and speed layers and notifying the respective entities about risk exposure, through the Dashboard interface.

## Privacy Violation Detection

The Personal Data Analyser (PDA) is used to control personal data in a transaction, with the aim of discovering all previously non-identified personal data, such as personal data for which there is no data subject authorization. The PDA's objective is, then, the detection of privacy violations. Figure 3 presents the PDA module architecture and displays its inner components and interactions with external modules.

The PDA resorts to natural language reasoning (NLR)/natural language understanding (NLU) to semantic and syntactic recognition, with the objective of identifying and classifying named entities such as persons or places. This is supported by state-of-the-art NLP tools like Stanford CoreNLP [12] and spaCy [13], which are then fine-tuned and improved for this particular context.

The requests arriving from the message bus to the request processor are processed and dispatched to other components

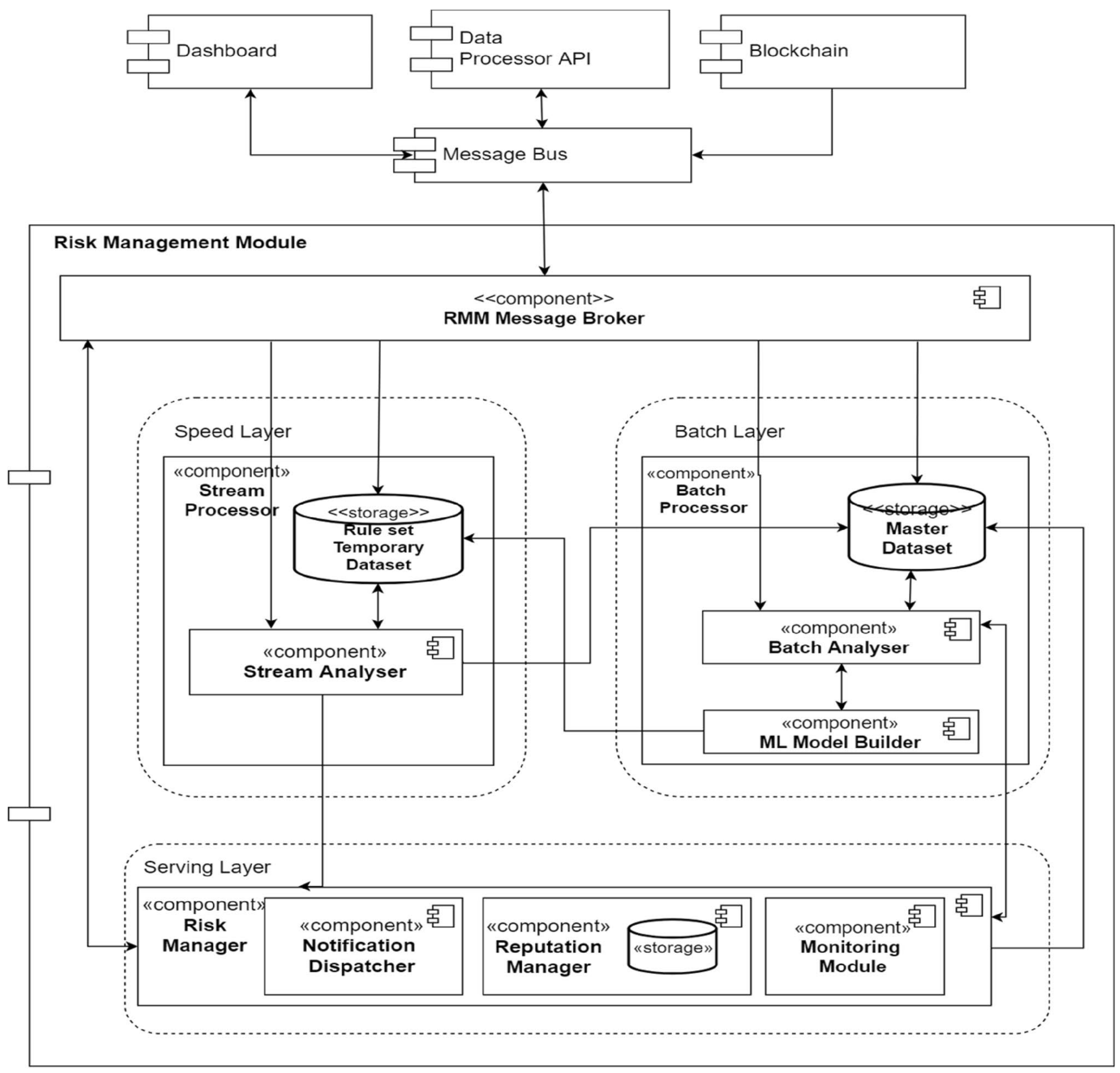

Dashboard
Data Processor API
Blockchain
Message Bus
Risk Management Module
<<component>> RMM Message Broker
Speed Layer
«component» Stream Processor
<<storage>> Rule set Temporary Dataset
«component» Stream Analyser
Batch Layer
«component» Batch Processor
<<storage>> Master Dataset
«component» Batch Analyser
«component» ML Model Builder
Serving Layer
«component» Risk Manager
«component» Notification Dispatcher
«component» Reputation Manager
«storage»
«component» Monitoring Module

**Fig. 2** RMM architecture

according to the data type (e.g. direct messages, RPC inputs, PDF, TXT, or other types of structured information). The metadata extractor retrieves all the associated metadata information and feeds it to the next component. The Internal parsers are intended to extract information and all data available inside each file or data structure being analysed. The files can be any structured information, PDF, XML, HTML, URL, CSV or TXT.

The final stage of the PDA flow is the Machine Learning (ML) reasoning unit. In this component, machine learning algorithms determine whether or not privacy risks are at stake. In order to determine privacy risks, the PDA uses Fuzzy Logic (with membership functions) and additional sets of rules. The assessment is performed with a combination of available data: the validation of the transaction's PII (i.e. based on the platform's allowed PII types), the existence or not of previously non-identified PII (i.e. through the NLP pipeline), the reputation of involved data processors (i.e. a normalized score), the level of sensitiveness of each PII attribute (i.e. sensitive or quasi-identifier) and the data retention periods (i.e. verification of minimum and maximum data retention times). The users' permissions are already verified in the Data Processor API (described in Fig. 1).

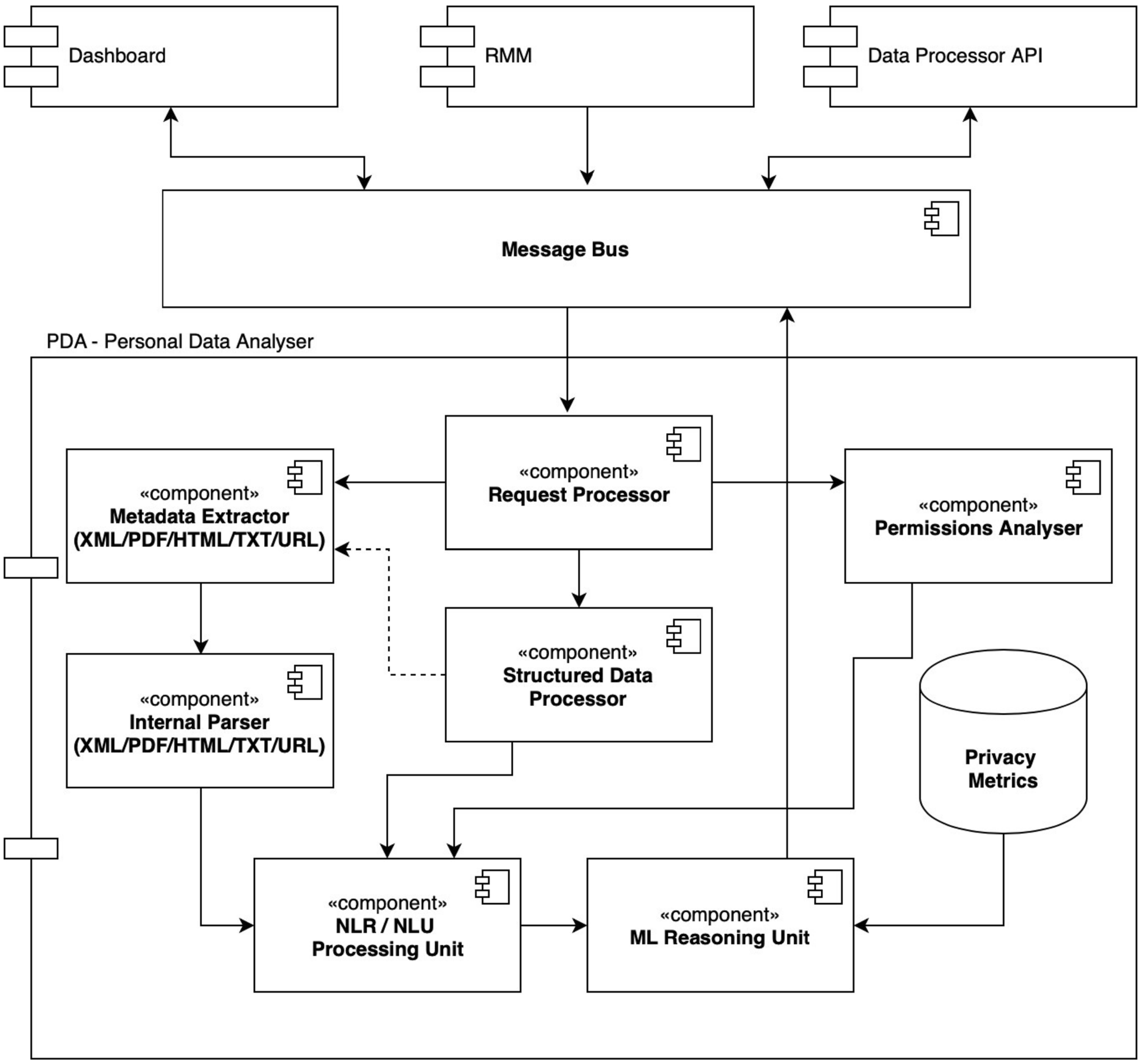

Dashboard
RMM
Data Processor API
Message Bus
PDA - Personal Data Analyser
«component» Metadata Extractor (XML/PDF/HTML/TXT/URL)
«component» Request Processor
«component» Permissions Analyser
«component» Internal Parser (XML/PDF/HTML/TXT/URL)
«component» Structured Data Processor
Privacy Metrics
«component» NLR / NLU Processing Unit
«component» ML Reasoning Unit

**Fig. 3** PDA architecture

The current version of the PDA (developed in Python version 3) provides a functional NLP pipeline capable of identifying named entities such as persons' names, cities, countries, locations, birth dates and others. It is currently using a hybrid approach where NLTK [14], Stanford CoreNLP, spaCy and regular expressions are used in conjunction to find PII. The main steps of the NLP pipeline are tokenization, part of speech tagging, named entity recognition. Recurring to the Python library Pika [15] to interact with RabbitMQ, the PDA is fully integrated with the Message Bus and successful delivery and validation of messages was achieved between RMM and PDA.

The current performance metrics available (although not final) show that one instance of the PDA is capable of processing at least 100 messages per second without engaging the NLP pipeline, and at least 3 messages per second when the NLP pipeline is called. These numbers refer to a local development environment. The final and optimized version, in a production environment, with more powerful hardware, is expected to highly increase the number of processed messages per second.

## Experimental Results

PoSeID-on's Risk Management Module and Personal Data Analyser are currently under evaluation in the project's staging environment, along with the other modules that comprise PoSeID-on's interim implementation. The end of the evaluation phase is scheduled for August 2020, and the lessons learnt will be fed into the final implementation of envisaged platform, according to plan. In this section, we provide some information on the performed tests and respective results.

### Risk Management Module

Evaluation of RMM anomaly detection functionality was, and still is, a challenging task. First of all, by the time of evaluation, no real data was being handled within PoSeID-on. In order to test the module, a public dataset was chosen, based on the type and structure of the logs. Furthermore, the staging environment, run in Minikube and used to deploy the PoSeID-on platform, suffered several setbacks during the time of testing, from mid-December 2019 to early January 2020.

To the best of the authors knowledge, no datasets from PII management systems are available and, as such, an alternative had to be found. Moreover, by using an alternative dataset, we cannot be sure that the obtained results will be equivalent to the ones that will be obtained under production conditions, although we can validate the anomaly detection approach and pipeline, since the log anomaly detection approach is generic.

The chosen dataset was the Hadoop Distributed File System (HDFS), made available by the LogPAI research team [16]. The HDFS log set was generated in a private cloud environment using benchmark workloads, and manually labelled through handcrafted rules to identify the anomalies. The logs are sliced into traces according to block ids. Then, each trace associated with a specific block id is assigned a ground-truth label: normal/abnormal. The log has a timespan of 38.7 h, 11,175,629 messages, and 575,061 blocks, of which 558,223 are normal blocks and 16,838 are anomalous blocks.

The chosen data set has three important characteristics, that allow to use the exact same approach that RMM uses for PoSeID-on system logs:

- *it is a distributed system*—logs are generated by several distributed components;
- *there is an identifier associated with each log*—similarly to PoSeID-on, logs have an identifier corresponding to the block each operation is referring to, and a single block can be the target of multiple operations or logs, much like PoSeID-on will have an identifier associated to a Data Subject or Data Processor;
- *anomalies are labelled according to the identifier*—the dataset is labelled in respect to blocks, such as a window of the RMM will label a window of logs corresponding to a data subject or data processor as anomalous or not.

Therefore, since the feature building step is independent from the log specificity—as it is based on an occurrence count of dynamically generated log templates during the parsing step, by means of the Drain algorithm—it is safe to assume that the approach taken works for the PoSeID-on system. Accuracy of results, however, cannot be generalized as it will depend on the frequency and nature of the produced logs in PoSeID-on.

The machine where the RMM was deployed was the same machine where the staging environment ran. A Cassandra and a RabbitMQ node ran on the same machine, to deliver the logs through the message queue, such as it would happen in the PoSeID-on deployment. Applications running on this machine remained the same for all runs of the experiments. The characteristics of the machine are the following: brand and model – Dell Precision 5820 Tower X-Series; operating system – Ubuntu 18.04.1 LTS; processor—Intel(R) Core (TM) i9-7920X CPU @ 2.90 GHz; RAM – 132 GB, 2666 MHz, DDR4; graphics—NVIDIA Quadro P1000, 4 GB GDDR5; storage—1 TB 7200 rpm SATA Hard Drive.

The RMM was packaged into a Java Archive (JAR) file containing all necessary dependencies and ran with the following Java Virtual Machine (JVM) configuration: Maximum memory allocation (-Xmx) – 64 GB; Initial memory allocation (-Xms) – 64 GB. In addition, Spark was run in local cluster mode and as such the enforced configuration is of one Driver, with the Driver performing all the execution, instead of an Executor. The following parameters were set for running the RMM in local mode: Spark Logical CPU Cores – 24; Driver Memory – 48 GB.

In order to evaluate the anomaly detection approach, a few tests were run using a single RMM instance. The full HDFS dataset was used for testing. Since Spark Streaming does not allow windowing by timestamp, a script was run to read the HDFS dataset file and send each log one by one, according to the timestamp, to the message queue. The RMM instance fetched this information every 5 s from the queue. Since the dataset spanned over approximately 39 h, in order to minimize testing times, a speed-up parameter was added to the instance. For all testing cases, this speed-up was of 4 times faster than the real time. Messages were sent 4 times faster to the queue and processing happened 4 times faster than the parameters defined for the test. Although this increases the load on the system, it was a necessary trade-off in order to shorten testing times. The results are not affected by this, since all time-based operations were scaled accordingly.

For the tests, the K-means algorithm parameters were set as follows: cluster initialisation – random; number of clusters per model – 4, 8, 12, 16; K-means half-life – 24 training batches.

As for the starting clusters centres, these were set using a random seed. The same seed was used for all models running inside the same RMM instance, for result comparison. Regarding the Descriptive Statistics setup, the 100,000 most recent distances were saved for each cluster of each model. The parameters for K-Means initialization and Descriptive Statistics were kept constant across all tests. Since it is not possible to be sure that a predicted anomaly is in fact an anomaly or not, all received logs were used for training of the model.

The parameters that suffered variations during testing were the ones related to the pipeline and anomaly detection approach, more specifically:

- *whether or not feature vector overlap was considered for training* when overlapping is used, the length of the window is larger than the length of the slide, which means that some values in the window are new (the ones that correspond to the most recent slide) and the remaining ones are reused from the previous window;
- *what percentage of points a cluster must have to be considered anomalous* since small clusters are hypothesized to contain anomalies, a threshold must be set to define what is a small cluster. This is done by defining a percentage of points to be considered anomalous and this number is then divided by the number of clusters. The result is used as the threshold per cluster;
- *Z-score threshold* How many standard deviations a vector's distance to the cluster centre must be over the cluster mean in order to be considered an anomaly;
- *Window length* window is considered for training and analysis. Shorter windows will be able to detect point anomalies, while larger windows will detect collective anomalies.

Sliding length is another potential parameter, but since it is necessary to have predictions output under a user session, a fixed value 5 min was used, as this is the expected duration of a typical user session.

In a first set of tests, T01, overlapping feature vectors were used. Nevertheless, models trained under these circumstances failed to detect any anomalies, due to model overtraining resulting from the repetition of anomalous patterns. Thus, subsequent tests (T02.1, T02.2, T03.1, and T03.2) did not use overlapping. In all tests, anomalies were considered in the training phase. Table 2, below, summarizes the parameters used in the various tests.

Figure 4 presents the results obtained in tests T02.1 through T03.2, for the various number of clusters per model under consideration (i.e. 4, 8, 12, 16 clusters).

The values for precision were quite high in all the tests, which means that the detection of anomalies is quite effective. Moreover, a combination of a Z-Score threshold of 2.0 and a window of 30 min achieved very good values of Recall and F1-Score for a model with 4 clusters, in test T02.2. It is also possible to see that an increase in the number of clusters seems to negatively affect tests in most cases, which is an indicator that using a small number of clusters may be the best option, performance and result wise. For the reasons explained previously, these results cannot be generalized for every situation and dataset without further testing with different datasets and the real PoSeID-on system logs. These aspects will continue to be explored in subsequent tests, throughout the whole duration of PoSeID-on's evaluation phase.

Current performance analysis with a single RMM instance demonstrates that the module can handle windows of 10,000 logs and output results every 5 min. These are preliminary results and the module can be further optimized to take advantage of parallelization in the future.

## Personal Data Analyser

In this sub-section, we provide some experimental results with the aim of evaluating the use of Named Entity Recognition (NER) as a way to identify, monitor and validate personally identifiable information. The results of this evaluation

**Table 2** Parameters used in the various tests

| Test | Overlapping | Anomaly % | *Z*-score threshold | Window (min) | Training periodicity (min) |
|---|---|---|---|---|---|
| T01 | Yes | 3 | 2.5 | 30 | 40 |
| T02.1 | No | 3 | 2.5 | 30 | 40 |
| T02.2 | No | 3 | 2.0 | 30 | 40 |
| T03.1 | No | 3 | 2.5 | 60 | 70 |
| T03.2 | No | 3 | 2.0 | 60 | 70 |

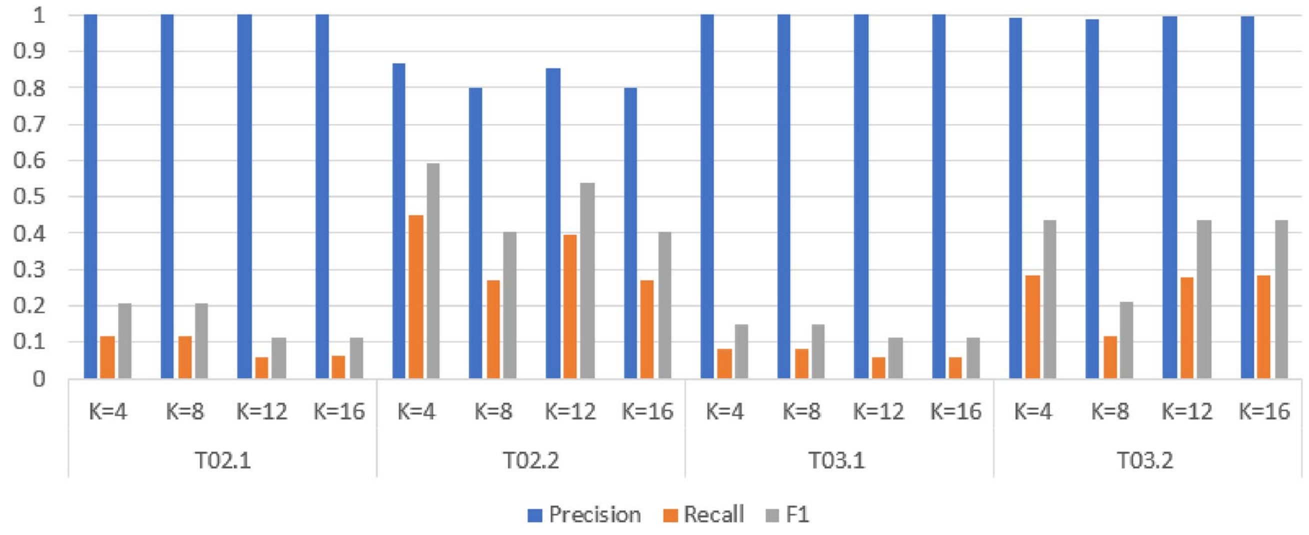


**Fig. 4** Precision, Recall, and F1 for 4, 8, 12, and 16 clusters

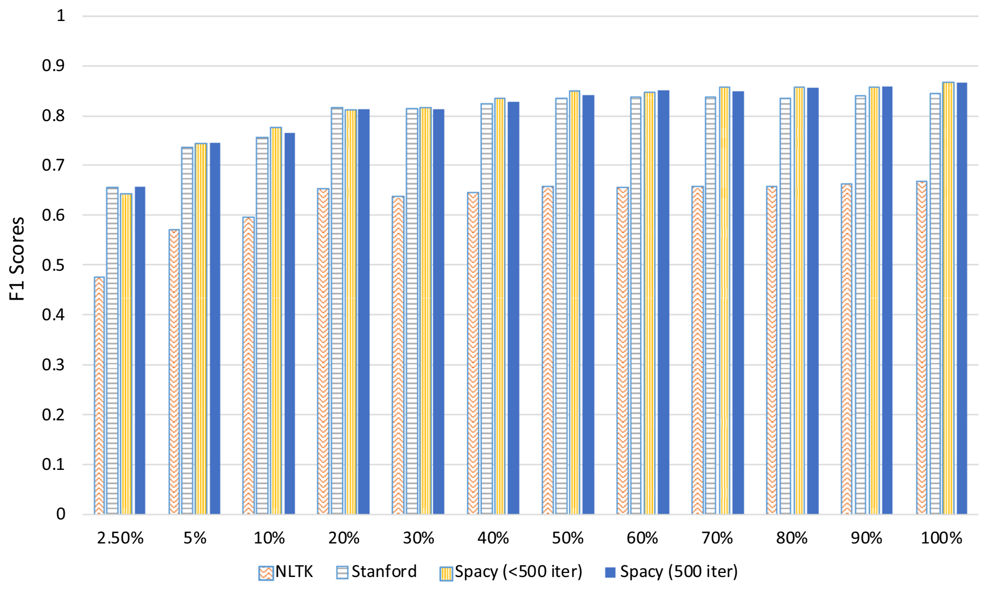
allowed us to determine which tool performs the best and offers more flexibility to our use case. For example, determining which tool achieves the highest F1 Scores in Named Entity Recognition. In our experiments, we used three of the most well-known Natural Language Processing tools (NLTK [17], Stanford CoreNLP [18], and SpaCy [19]) in two stages, with four different types of data (described next).

In the first stage, which we call the General Model stage, the effectiveness of the tools was evaluated in a generic dataset. The used dataset was retrieved from Kaggle [20] and based on the Groningen Meaning Bank semantically annotated corpus.

The dataset used in the first stage experiment had 1.354.149 tokens. Therefore, to evaluate performance of the NLP tools, as well as the models' classification capabilities using data with different sizes, the dataset was sliced into smaller portions (5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100%). For each portion, it was then applied the 70% and 30% proportion rule for training and validation, respectively.

Figure 5 provides the results for the $F_1$ scores for the various NLP tools and for the case of the General Model. In the figure, it is possible to note the increasing tendency up to the 20% size dataset and stabilizing (with very small variations) after that point. From the 20% size to the fully sized dataset, there is only a slight improvement of almost 0.03 in the score.

**Fig. 5** $F_1$ scores for the general model (NLTK, Stanford CoreNLP and SpaCy)



After having analysed all the obtained results, it was possible to draw a few overall conclusions regarding the three analysed tools. In our experiments, the lowest F1 Scores were obtained with NLTK. Instead, with higher F1 Scores, Stanford CoreNLP and spaCy presented similar results. The highest F1-Score was seen in spaCy, with a small margin. The results indicate that without any comprehensive tuning of the model training settings, spaCy provides the best results, while still leaving room for possible improvement if a specific configuration is applied (e.g. shorter training times by setting fewer iterations). On the other hand, the lack of tuning options in NLTK is the likely cause of the lower results in our experiments.

In the second stage, which we call the Contract-based Model stage, the created dataset was a fusion between contracts available in online sites [21, 22] and other contracts from the U.S Department of Defense (DoD) [23], containing personally identifiable information that was manually labelled beforehand.

Figure 6 shows the precision, recall, and $F_1$ scores obtained while evaluating the models that were created. It is possible to observe that NLTK's scores were approximately 0.45. On the other hand, SpaCy and Stanford CoreNLP reached higher F1 Scores (approximately 0.90). Moreover, the difference between these two is approximately 0.01, with Stanford CoreNLP providing better results.

The required time to train the models in the two stages varied greatly. The dataset size was an influencing factor – more data, more training time required. NLTK training sessions were all completed within 5 min, Stanford Core NLP up to 135 min, and spaCy up to 6077 min. Therefore, by choosing a hybrid approach and using these tools in our solution, we find a balance between model training effort and model classification performance.

The presented experiments allowed us to assess the effectiveness of three different NLP tools and their NER sub-tasks in discovering PII, and demonstrated how the proposed approach can effectively be used as a privacy-enhancing technology. Different machine learning models were trained and evaluated with generic datasets as well as contracts with PII (the latter ones having been manually labelled). To counter time-consuming manual labelling, we are currently assessing the feasibility and reliability of using. Mostly, AI—a synthetic data generator which, according to its developers, produces realistic data that resemble the characteristics and diversity of actual people [24].

## Conclusion

In this paper, we have presented and discussed two crucial modules of the PoSeID-on personal data protection platform. The risk management module is used for evaluating and managing privacy and operational risks within the PoSeID-on system, through machine learning-based analysis of operational logs and PII exchanges, and managing a risk score associated with each data processor. This can be used to advise on which service should eventually be disabled in case of anomalies or high exposure to privacy risks. On the other hand, to detect and prevent anomalies and misbehaved transactions, the Personal data analyser is used to monitor personal data transactions and related warnings generated by the blockchain platform. A warning is generated each time a transaction does not comply with pre-defined rules (e.g. permissions or data processor

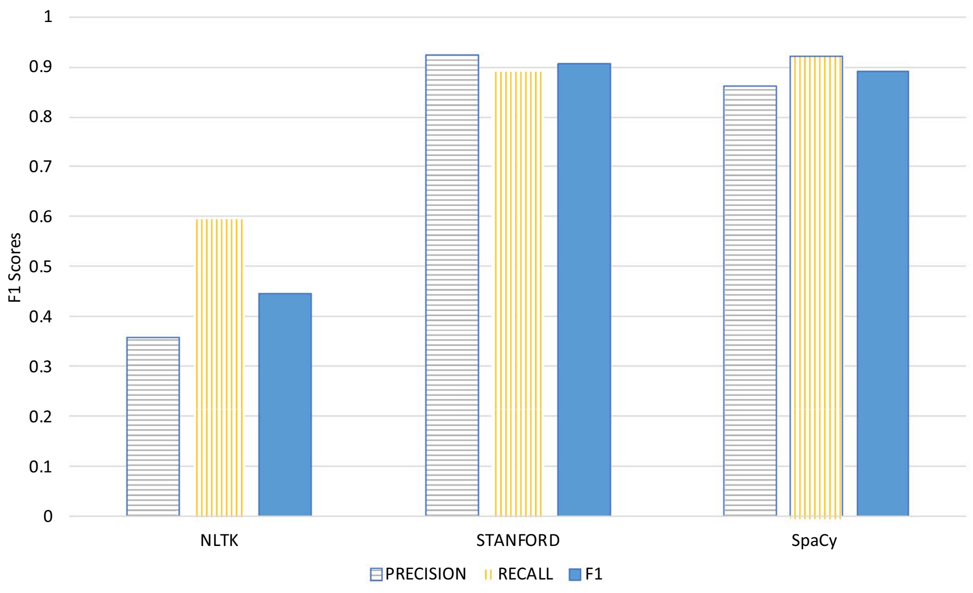


**Fig. 6** $F_1$ scores for the contract-based model (NLTK, Stanford CoreNLP and SpaCy)

internal reputation). These modules were subject to implementation and initial assessment, and some of the obtained results were presented in the paper in order to illustrate the referred modules' operation. The PoSeID-on platform is now under intense study and evaluation, and this will continue in the immediate future, with the aim of improving the efficacy and efficiency of the risk management and privacy violation detection functionality.

**Funding** The work presented in this paper was partially carried out in the scope of the PoSeID-on project—Protection and control of Secured Information by means of a privacy-enhanced dashboard, Grant Agreement Number: 786713. H2020-DS-2016-2017/DS-08-2017.

**Availability of Data and Material** All the data used in this work are publicly available. The data sources are listed in the references.

## Compliance with Ethical Stanards

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Council E. EU Regulation 2016/679 of the European Parliament and of the Council, "On the protection of natural persons with regard to the processing of personal data and on the free movement of such data", Directive 95/46/EC (General Data Protection Regulation): Official Journal of the European Union; 2016.
2. PoSeID-on project. Protection and control of secured information by means of a privacy enhanced dashboard. https://www.poseidon-h2020.eu Accessed Jan 2020.
3. Council E. EU Regulation No. 910/2014 of The European Parliament and of the Council, "On electronic identification and trust services for electronic transactions in the internal market", Directive 1999/93/EC: Official Journal of the European Union; 2014.
4. Astekin M, Zengin H, Sözer H. Evaluation of distributed machine learning algorithms for anomaly detection from large-scale system logs: a case study. In: 2018 IEEE international conference on big data (big data); 2018. p. 2071–7.
5. Lin Q, Zhang H, Lou J, Zhang Y, Chen X. Log clustering based problem identification for online service systems. In: 2016 IEEE/ACM 38th international conference on software engineering companion (ICSE-C); 2016. p. 102–11.
6. Liu Z, Qin T, Guan X, Jiang H, Wang C. An integrated method for anomaly detection from massive system logs. IEEE Access. 2018;6:30602–11.
7. He S, Zhu J, He P, Lyu MR. Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th international symposium on software reliability engineering (ISSRE); 2016. p. 207–18.
8. Google. Graylog extended log format—GELF. http://docs.graylog.org/en/3.0/pages/gelf.html. Accessed Jan 2020.
9. He P, Zhu J, Zheng Z, Lyu MR. Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE international conference on web services (ICWS); 2017. p. 33–40.
10. Apache Spark. https://spark.apache.org/streaming/. Accessed Jan 2020.
11. Lambda Architecture. http://lambda-architecture.net. Accessed Jan 2020.
12. Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D. The Stanford CoreNLP natural language processing toolkit. In: Association for computational linguistics (ACL) system demonstrations; 2014. p. 55–60.
13. spaCy: Industrial-strength natural language processing. https://spacy.io. Accessed Jan 2020.
14. Bird SEK, Loper E. Natural language processing with python. Boston: O'Reilly; 2009.
15. Pika—a RabbitMQ (AMQP 0-9-1) client library for python. https://github.com/pika/pika. Accessed Jan 2020.
16. LogPAI—Log analytics powered by AI. http://www.logpai.com/. Accessed Jan 2020.
17. Bird S, Klein E, Loper E. Natural language processing with python. Boston: O'Reilly; 2009.
18. Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D. The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. Baltimore: Association for Computational Linguistics; 2014. p. 55–60. https://doi.org/10.3115/v1/P14-5010.
19. ExplosionAI. spaCy—industrial-strength natural language processing. 2019. https://spacy.io/. Accessed Jan 2020.
20. Kaggle. An online community of data scientists and machine learners. 2019. https://www.kaggle.com/ Accessed Jan 2020.
21. Metrolink. Metrolink is Southern California's premier regional passenger rail system serving over 55 stations across the region. 2019. https://www.metrolinktrains.com/globalassets/about/contracts/may-26-2019/contract-no.-sp452-16-conformed-contract-fully-executed.pdf. Accessed Jan 2020.
22. Texas Department of Information Resources. 2019. Our mission is to provide technology leadership, technology solutions. https://dir.texas.gov/View-Search/Contracts-Detail.aspx?contractnumber=DIR-TSO-4101. Accessed Jan 2020.
23. Official website for U.S. Department of Defense. https://www.defense.gov/Newsroom/Contracts. Accessed Jan 2020.
24. MostlyAI. Creating AI-generated synthetic data. 2019. https://mostly.ai/. Accessed Jan 2020.