

Building A Privacy-Preserving Benchmarking Enterprise System

Florian Kerschbaum
SAP Research, Karlsruhe, Germany
florian.kerschbaum@sap.com

Abstract

Benchmarking is the process of comparing one's own performance to the statistics of a group of competitors, named peer group. It is a common and important process in the business world for many important business metrics, called key performance indicators (KPI). Privacy is of the utmost importance, since these KPIs allow the inference of sensitive information. Therefore several secure multi-party computation (SMC) protocols for securely and privately computing statistics of KPIs have recently been developed. These protocols are the basic building block for a privacy-preserving benchmarking system, but in order to complete an enterprise system that offers a benchmarking service to its customers more problems need to be solved. This paper addresses two remaining problems: peer group formation and protocol orchestration.

We first analyze how peer group participation impacts privacy and vice-versa. Given current network performance limitations we conclude that in order for KPIs to remain private one subscriber can participate in at most one peer group.

Peer group formation is the process of forming sensible peer groups out of the set of subscribers. A sensible peer group is one that is useful for benchmarking, i.e. a group of similar companies, under the constraint that one subscriber can participate in at most one peer group. We characterize subscribers by a set of discrete criteria and therefore view the automatic peer group formation as a data clustering problem. A data clustering algorithm customized for automatic peer group formation is required to build clusters whose size does not fall below a minimum threshold. We present a high-performance modification of k -means clustering that takes the minimum cluster size as an additional parameter which might be of independent interest. In a simulation we evaluate its practical applicability to automatic peer group formation. Our final approach is the first automatic peer group formation algorithm for an enterprise benchmarking system.

Polling-based protocol orchestration allows the subscribers to remain passive clients, i.e. require no inbound

connection, e.g. through a company firewall. We show through simulation that such a polling-based orchestration can be expected to complete within one polling interval.

1 Introduction

Benchmarking is the process of comparing one's own key performance indicators to the statistics of one's peer group. A key performance indicator (KPI) is a statistical quantity measuring the performance of a business process. Examples from different company operations are make cycle time (manufacturing), cash flow (financial) and employee fluctuation rate (human resources). A peer group is a group of (usually competing) companies that are interested in comparing their KPIs based on some similarity of the companies. Examples formed along different characteristics include car manufacturers (industry sector), Fortune 500 companies in the United States (revenue and location), or airline vs. railway vs. haulage (sales market).

Privacy (of the KPIs) is of utmost importance, since KPIs allow the inference of sensitive information. Therefore several secure multi-party computation (SMC) protocols have been recently developed that can privately compute the necessary statistics, e.g. [1, 9, 10]. SMC guarantees that no protocol participant will learn more than what he can infer by her input and the output of the protocol, i.e. the other parties' inputs remain confidential.

Our benchmarking enterprise system builds on top of the protocols of [9]. The communication pattern of this protocol matches the model of our enterprise system with a central entity as the hub of communication offering the benchmarking service to its customers. We call the central entity service or platform provider, since it runs the platform allowing the subscribers to access the benchmarking service. This is necessary to provide benchmarking statistics to customers as soon as they join the system. The central platform keeps a database of the computed statistics of the peer group. In the SMC protocol the service provider therefore becomes a participant without input, i.e. the service provider is not supposed to learn the KPIs of the sub-

scribers. Privacy against the service provider meets the expectations of the customers, since they do not want to trust a third party with their KPIs.

This paper addresses the problem of building an enterprise system based on the SMC protocols. We approach the problem of building enterprise system that offers such a (privacy-preserving) benchmarking service systematically from an end user's perspective. Section 3 reviews the use cases of our benchmarking enterprise system. From the analysis of these use cases it follows that privacy is only one of the desirable features. An important unsolved problem is that of grouping companies into sensible peer groups: peer group formation.

This paper first describes two models for peer group participation: single and multiple. In the single peer group model a party can participate in only one peer group whereas in the multiple peer group model the subscriber can participate in more than one. We show in Section 4.3 by conservative estimation of the parameters that the multiple peer group model cannot be realized while preserving the privacy of the KPIs.

We classify the subscribers by characteristics, such that similar companies have similar characteristics. Automatic peer group formation can then be viewed as a data clustering problem. Section 4.4 is dedicated to the description of our specialized data clustering algorithm and its evaluation. The algorithm is a high-performance modification of k-means clustering that allows specifying the minimum cluster size to be formed. Although constrained k-means clustering [3] already provides this feature our algorithm scales to the problem size we require and might be of independent interest. We named our algorithm k,l-means clustering where l is the minimum cluster size.

The paper concludes by a comparison of the running times of the SMC protocol depending on their orchestration model. A notification based orchestration allows the protocols to be executed in the optimal, i.e. minimal, running time, but require the service provider to be able to contact the subscribers. This requires the subscriber to allow an inbound connection through his firewall ("to punch a hole in his firewall") which is undesirable in many deployments. Polling enables the customers to inquire in intervals about the status of the protocol and participate when requested to without any inbound connection, but due to the sleeping intervals between inquiries the overall running time is no longer optimal. Section 5 shows a polling algorithm that allows the participants to complete a SMC protocol within one polling interval. We verify this by simulation.

Overall the paper describes the challenges faced when building a privacy-preserving benchmarking enterprise system. Building SMC protocols is only one challenge to be solved in this process. This paper contributes:

- a use case based analysis for a benchmarking enter-

prise system which is not limited to privacy-preserving systems.

- an analysis of peer group models under privacy constraints which clearly shows that the only practical model is the single peer group model.
- the first automatic peer group formation algorithm.
- a comparison of polling and notifications for protocol orchestration.

2 Economic Motivation for Privacy

The privacy requirement of the benchmarking enterprise system is designed for the economic advantage of the service provider. Two advantages can be separated: customer acceptance and competitive advantage.

Privacy is anticipated to increase customer acceptance of benchmarking. The intuition is that customers are reluctant to share business critical data and privacy-preserving benchmarking can alleviate the risk. This in turn leads to more potential customers, a larger market size and, actually, to larger revenue.

Privacy can also provide a competitive advantage. The risk and associated costs (e.g. insurance premiums) of sharing KPIs to engage in benchmarking can be lowered by privacy. Thereby offering a higher benefit to customers, justifying a higher price or increasing market share.

Also given the possibility of privacy-preserving benchmarking with similar results to and the same price as non-privacy-preserving benchmarking, following the argument of [8], there is no reason to engage in non-privacy-preserving benchmarking.

3 Use Cases

This section describes the three use cases for the (privacy-preserving) benchmarking enterprise system. The first use case is registration where a company (customer) wants to join the platform and become a subscriber. To satisfy the secure computation protocol we need to involve a certificate authority in the process. This use case is of minor importance to the remainder of the paper and is therefore described only briefly. The second use case is statistics retrieval where a customer retrieves the statistics of its peer group. The third use case is the computation of statistics where the database of statistics is actually computed.

The participants of the first use case are a company, the service provider and a certificate authority. The certificate authority and the service provider are considered mutually distrustful, but have a contract in place in order to execute a

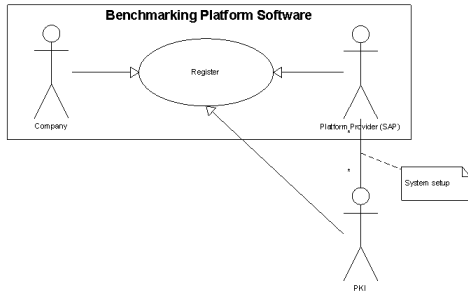


Figure 1. Registration use case

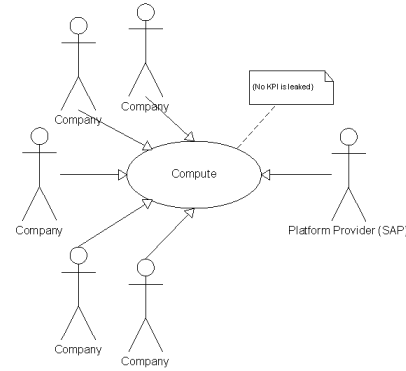


Figure 3. Statistics computation use case

registration protocol between the customer and the combination of the two of them. The first use case is depicted in figure 1.



Figure 2. Statistics retrieval use case

The participants of the second use case are a single customer and the service provider. The customer retrieves statistics of a peer group from the service provider’s database. The database decouples the computation of the statistics from the retrieval, i.e. the actual benchmarking process. This enables the customer to retrieve statistics of peer groups she is not participating in. In fact, she does not have to participate in any peer group in order to retrieve statistics. She can therefore start to retrieve statistics right after registration and does not need to wait for a synchronized run of the SMC protocol before she is able to use the service. Furthermore, the automatic peer group formation algorithm directs her to the correct peer group in the benchmarking platform. Figure 2 shows the second use case.

The participants of the third use case are a group of customers and the service provider. The exact composition of the group will be discussed later in the sections on peer group models, but at the very least all members of the peer group need to participate. They can e.g. then engage in the secure computation protocol of [9]. For all protocols the orchestration model is important and we will compare notifications and polling in Section 5. Statistics computation is depicted as a use case in figure 3.

4 Peer Group Formation

Peer group formation is the process of computing the peer groups from the set of subscribers at a given point in time. Peer group formation creates a mapping between subscribers and peer groups. A peer group has a minimum size for its statistics to be meaningful in the benchmarking process. This minimum size is larger than one, since a customer wants to compare to its competition and not just itself. Therefore a peer group always contains multiple subscribers. A subscriber can be in one or more peer groups. Consequently two peer group models can be distinguished: single and multiple.

In the single peer group model the customer is part of exactly one peer group. In the multiple peer group model the customer can be part of one or more peer groups. The peer group model has implications on the communication pattern and the privacy of the KPIs.

4.1 Single Peer Group Model

In the single peer group model the subscriber maps to one and only one peer group. The privacy of the KPI is protected by the size of the peer group. If the statistics computation is private, no individual KPI is being leaked. The service provider may know the peer group of a subscriber without a privacy breach. The service provider only needs to contact the subscribers that are members of a peer group to compute its statistics, since the lack of communication with non-members does not reveal information the service provider may not have.

Let k be the number of KPIs, p the number of peer groups, n the number of customers and m_i the number of customers in peer group i . In the single peer group model,

it holds that

$$n = \sum_i^p m_i$$

The single peer group model's statistics computation protocol communication cost has a lower bound of $\Omega(nk)$, since each KPI must be computed separately (k) and the sum of members of the peer groups is n (see equation above).

4.2 Multiple Peer Group Model

In the multiple peer group model the customer can be part of one or more peer groups. The privacy of the customer's KPI is at risk, if the service provider knows which customer participates in which peer group.

Denote customer's X_i participation in peer group j by $\lambda_{i,j} = 1$, else $\lambda_{i,j} = 0$. Let Λ denote the matrix of $\lambda_{0,0}, \dots, \lambda_{n,p}$. Figure 4 shows an example of such a peer group participation matrix.

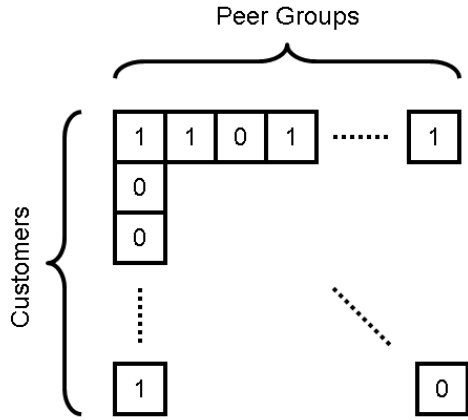


Figure 4. Peer Group Participation Matrix

Let $x_{i,j,k}$ be customer X_i 's value of the k -th KPI in the j -th peer group. Let $\vec{\beta}_{j,k} = (x_{1,j,k}, \dots, x_{n,j,k})$. The computation of the sum of a KPI per peer group can be written as

$$s\vec{u}m_k = \Lambda \vec{\beta}_{j,k}$$

The computation of the sums $s\vec{u}m_k$ for all peer groups for the k -th KPI is equivalent to the computation of average, if the values in Λ are divided by the peer group size or if the peer group size is known and the results are divided by the peer group size.

If Λ (or a selected rows of Λ) are invertible, then

$$\Lambda^{-1} s\vec{u}m_k = \vec{\beta}_{j,k}$$

Since $s\vec{u}m_k$ is public, Λ must either remain private for $\vec{\beta}_{j,k}$ to remain private or all partial matrices of Λ that contain all

members (1s) of each contained peer group row (e.g. the intersections of columns 2, 3, 5 and rows 1, 4, 5, such that rows 1, 4, 5 have no 1's outside of columns 2, 3, 5) must be non-invertible.

Λ must remain private to anyone, including the service provider. The communication pattern must not reveal Λ to the service provider and therefore every customer has to participate in the computation of every peer group. The multiple peer group model's statistics computation protocol communication cost has a lower bound of $\Omega(nkp)$, since each KPI (k) must be computed separately for each peer group (which now communicates with all participants).

4.3 Comparison

A necessary condition for Λ to be invertible (or pseudo-invertible) is $p \geq n$. This never holds in the single peer group model. Also no partial matrix that satisfies the condition above is invertible in the single peer group model, due to the fact that for all variables of a peer group there is at most one equation. Peer group formation uses non-sensitive criteria, that might even be public already, as input. A regular, non-privacy-preserving computation of peer groups is therefore preferable, since its computation and communication cost is lower.

For the statistics computation the single peer group model offers a better lower bound on the communication cost. For evaluation we use data we consider realistic for real-world applications given today's business software market. Considering this non-functional requirement for the number of customers and $p = o(n)$, the multiple peer group model places high burdens on the practicality of the protocols. Assume that only one encrypted value of size 256 bytes needs to be transferred per KPI, peer group and customer. For 200 KPIs and 100000 customers, one customer must transfer over for 4 GB per peer group. Even under very low impact assumptions this results in 16 TB per subscriber and 1600 PB for the service provider. This is unpractical for a real-time enterprise system under current network conditions.

A practical privacy-preserving benchmarking platform must therefore work in the single peer group model.

4.4 Automatic Peer Group Formation

Peer group formation is the task of grouping related companies, very much like data clustering. This task of peer group formation should be performed by the platform provider.

First each company is classified by a number of criteria, examples include revenue, number of employees, continent of headquarters, industry sector, or legal form. Then

```

1 means[] := random datapoint[k]
2 do
3   size[] := 0[k]
4   flag[] := false[n]
5   for i := 1 to n
6     cluster[i] := index of closest means[]
7     size[i] := size[i] + 1
8   do
9     reassign := false
10    for i := 1 to k
11      if size[i] < l
12        min := index of closest datapoints[] with
13          cluster[min] != i
14          flag[min] == false)
15        size[i] := size[i] + 1
16        size[cluster[min]] := size[cluster[min]] - 1
17        flag[min] := true
18        cluster[min] := i
19        reassign := true
20    while reassign
21    for i := 1 to k
22      recompute means[]
23  until means[] stabilize

```

Table 1. k,l-means clustering algorithm

each criterion is divided into a fixed number of discrete subclasses, e.g. for number of employees: 0 to 10, 11 to 100, 101 to 1000, 1001 to 10000, more than 10000. Let m be the number of criteria, then each company forms a data point in m -dimensional space. We can use existing data clustering algorithms to form peer groups.

4.4.1 K,L-Means Clustering

We propose using the k-means clustering algorithm [11], but need to adapt it to support a minimum cluster size. Recall that the minimum cluster size is necessary to protect the privacy of the individual participants and to create useful peer groups. Too small peer groups (e.g. just two participants) reveal the parties' KPI values and are not particularly useful for benchmarking. A solution called constrained k-means clustering using linear programming has been proposed in [3]. Nevertheless this solution does not scale to our requirements. Our test case using 10000 companies and 1000 clusters leads to a linear programming model with 10 million variables (and even more constraints). Such a large problem can only be solved using special hardware and we expect the problem size to increase by a 100 for real-world applications. Therefore a different algorithm is needed.

The k-means algorithm starts by randomly choosing k cluster centers. Then each data point is mapped to the closest cluster center and the cluster center is recomputed as the mean of its associated data points. The algorithm is continued until the cluster centers stabilize, i.e. the distance between two iterations is below a threshold. In the best case the cluster centers stabilize.

We propose a small extension in the form of a greedy

algorithm to k-means clustering. First, we fix another parameter l which is the minimum cluster size. After each data point has been assigned to a cluster center, we process each cluster. If a cluster does not have l data points assigned to it, we assign it the closest data points that

1. are not yet assigned to it
2. have not been reassigned in this iteration

The second condition prevents two cluster centers competing for a certain data point reassigning it alternatively and resulting in an infinite loop. We reassign data points until each cluster has at least l data points, then the cluster centers are recomputed. The final algorithm, which we call k,l-means clustering, is depicted in table 1 as pseudocode. Our addition to the regular k-means algorithm is confined to the lines 8-20 and described in greater detail than the remaining algorithm.

4.4.2 Evaluation

We assumed 10 criteria each divided into 5 subclasses, i.e. n data points in a 10-dimensional space. Then we choose companies according to a probability distribution. We evaluated our k,l-means clustering algorithm for benchmarking on two distributions. First, we chose each data point individually and uniformly. Then we chose first k cluster centers and choose $\frac{n}{k}$ data points with a fixed distance of 1 for each cluster. In the second distribution there exists a clustering with an average distance of 1 from the cluster center and a variance of 1. We use the Euclidean distance for measuring distance.

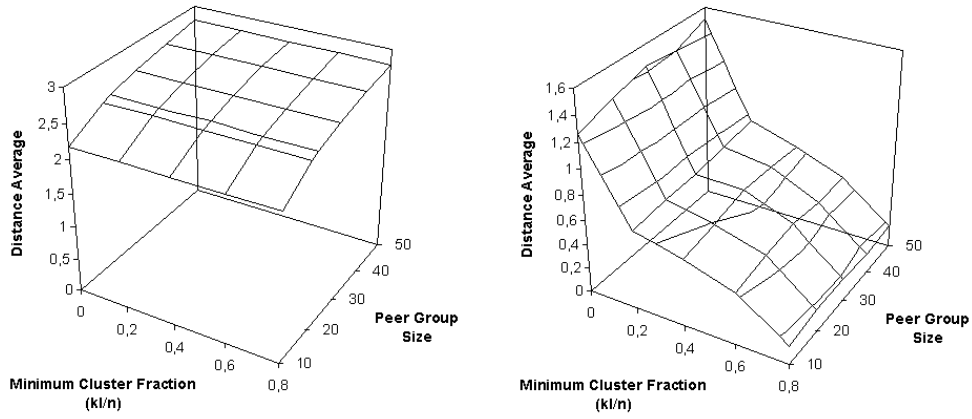


Figure 5. Average distance from cluster center

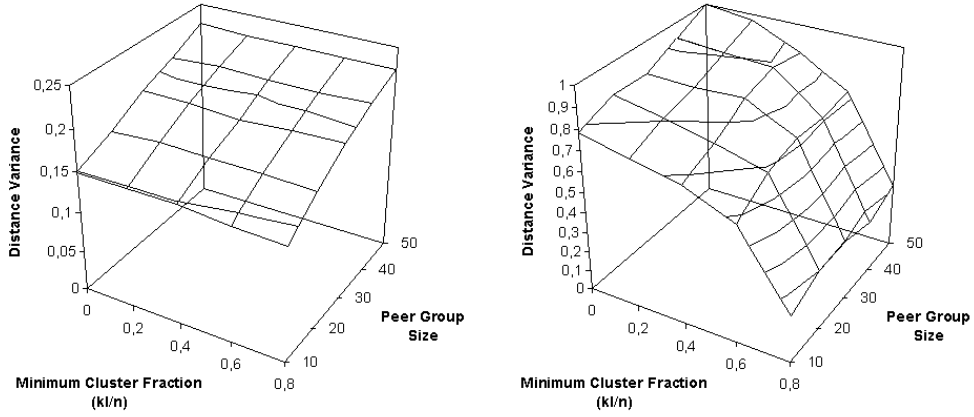


Figure 6. Variance of distance from cluster center

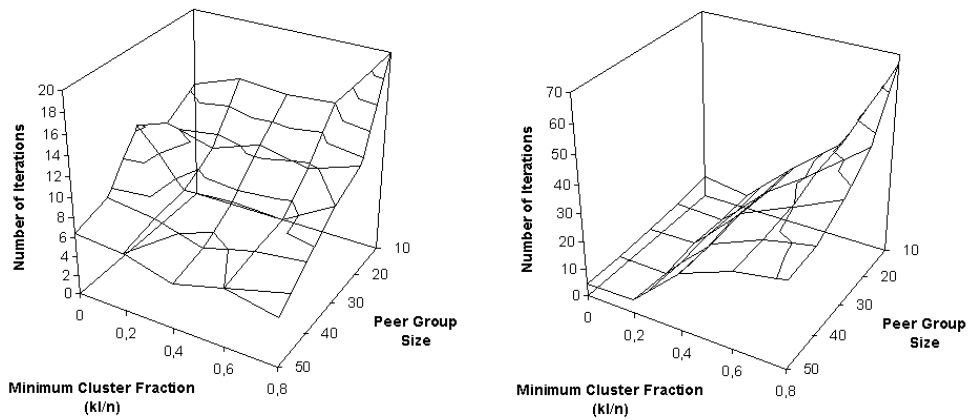


Figure 7. Number of iterations of algorithm

The results of our experiments for 10000 data points are depicted in figures 5,6 and 7. The result of the uniform distribution are shown on the left side and the result with predefined clusters on the right. In figure 5 the average distance from the cluster center is depicted. This is our main quality measure, since the lower the distance, the stronger the coherence of the group and we can assume the higher the competition. Figure 6 shows the variance of the distance from the cluster center and figure 7 shows the number of iterations until the cluster centers stabilized. We experimented with an increasing threshold for the stabilizing criterion when the number of iterations increases, since in some situations the algorithm does not converge otherwise. E.g. when l is chosen to be equal to $\mu = \frac{n}{k}$ then the algorithm does not converge for large data point sets. For each graph we show l as a fraction of μ on the x-axis from 0 to 0.8 in 0.2 steps. On the y-axis the average cluster size μ is arranged from 10 to 50 in steps of 10. E.g. then a graph point at 0.6, 20 means a value of $k = 500$ and $l = 12$.

From the graphs we can conclude, that given an uniform distribution, the l parameter has almost no impact on the quality of the result. On the other hand, given a clustered distribution it provides a significant advantage in performance even at low factors at the cost of slightly destabilizing the algorithm (i.e. increasing the number of iterations). Summarizing, the modified k,l-means algorithm performs at least as well as k-means algorithm and may even perform better for benchmarking applications.

4.5 Incremental Peer Group Formation

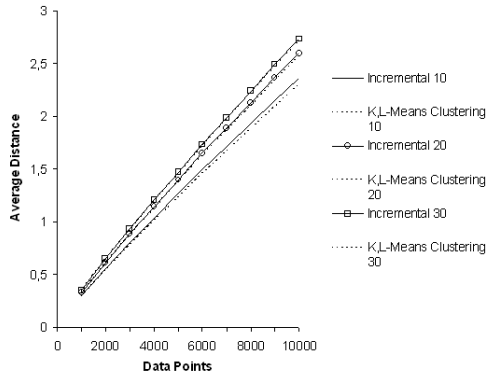


Figure 8. Incremental vs. regular k,l-means clustering

K,l-means clustering provides sufficient quality to form peer groups, but it operates on all data points potentially reassigning all subscribers. This would put a huge recom-

putation burden on the platform, since after a clustering algorithm run all peer groups would need to be recomputed. Instead it is advisable to limit the recomputation to the affected subscribers only. Furthermore a customer wants to retrieve a benchmarking result right after he becomes a subscriber and not wait for a clustering operation.

The solution to both problems is to compute the peer groups incrementally. When a new subscriber arrives, he is assigned to an existing peer group (and can therefore immediately retrieve statistics). We set an upper limit on the peer group size, and when it has been reached the peer group is split into two. The k,l-means clustering algorithm is used with $k = 2, l = \frac{\text{upper limit}}{2}$ on the previous peer group. Note that, the k,l-means algorithm converges for small data sets, even if the average cluster size μ is equal to $l: \mu = l$. Then at most two peer groups need to be recomputed when a new customer subscribes. The previous cluster mean is removed and the two new ones are added to the set of cluster means, incrementing the number of clusters by one. We also set a lower time limit for the interval between recomputations, such that one can assume the KPIs have changed (and are now independent from the previous values), and therefore the multiple peer groups do not affect privacy.

We compared the performance of our incremental algorithm with the regular k,l-means algorithm. We increased data sets from 0 to 10000 and measured the average distance of both algorithms in intervals of 1000. We did this for low limits of 10, 20, and 30 which corresponds to average cluster sizes of 13, 27, and 40, and high limits of 20, 40, and 60 respectively. The results are shown in figure 8 and we conclude that the performance difference is negligible. Incremental peer group formation is favorable to regular k,l-means clustering due to the limited recomputation effort when a new customer subscribes.

5 Protocol Orchestration

Our SMC protocol operates in a central communication model [9], but it also operates as every other SMC protocol in several (4) rounds. Each subscriber needs to execute a sub-protocol with the platform provider in each round. The order of the subscribers in each round is arbitrary, but each subscriber's turn needs to happen before the next round can be started. The open questions is how do subscribers know that it is their turn, since the SMC protocol is started by the platform provider (or some other party).

There are two options: notification and polling. In the notification model, each client runs a little server that when notified (contacted) triggers the execution of the sub-protocol for that step. The platform can fully coordinate the notifications and the execution time of the entire protocol is limited only by communication and computation cost (i.e. there is no idle time). Our preliminary implementation re-

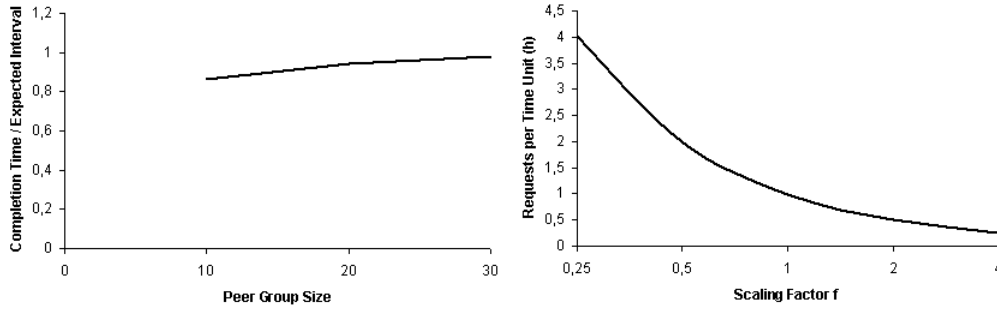


Figure 10. Completion time as a fraction of average arrival time and average number of requests per participant per arrival time

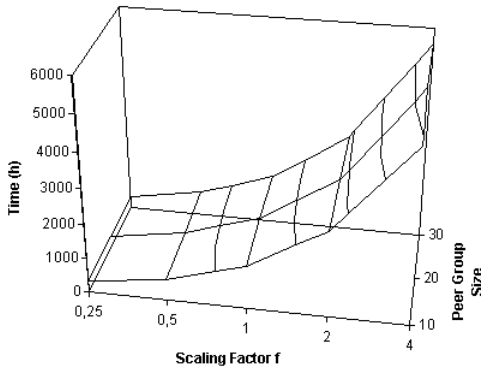


Figure 9. Average completion time in polling mode

sults indicate a execution time per participant in the very low seconds.

In the polling model, the subscribers poll the platform at intervals and if the subscribers can execute a sub-protocol, they do so. This model carries with it some idle time before the protocol can be completed, since the subscribers are only loosely synchronized. The great advantage of this model is that it does not require a server on the subscriber's side which eases deployment. The subscriber software can be a client deployed behind a corporate firewall that does not allow inbound connections. We therefore prefer this model.

The next sections describe a randomized algorithm to determine the interval between polls by the subscriber to the platform. We model the arrival of new subscribers with an exponential distribution with a fixed arrival rate λ . The average interval between two arrivals $t_\lambda = \frac{1}{\lambda}$. We use the incremental peer group formation algorithm, such that a

new arrival is mapped to an existing peer group and if its size reaches a high limit, the peer group is split into two equal-sized groups that are then recomputed. Every time the subscriber polls the platform it gets two values: the current round $s \in \{0, 1, 2, 3\}$ and the number of subscribers that have completed that round q (including the subscriber itself). She gets the pair $\langle s = 0, q = 1 \rangle$ if the protocol has not been started or is about to finish (i.e. round 4 has been completed by this subscriber). Let l be the number of members of the peer group (i.e. l in the k,l-means algorithm) and f be a scaling factor. We then set the time t_s for the subscriber to sleep until the next poll to a random number in the interval between 1 and $f \cdot \frac{t_\lambda}{2^s} \cdot \frac{l-q}{l-1}$. The rationale behind this formula is that, if the subscriber is the first to poll in a round his expected waiting time until all subscribers have polled in that round is half their interval. If he is the last in the round, he can execute the next round right away. Between those two events the interval is linearly decreasing.

We simulated the polling algorithm and evaluated its expected waiting time. The time to execute the protocol was neglected. We assumed 1 new customer per day, such that fixing a number for λ determines the granularity of our simulation. We used 1440 fixing the granularity at one hour (recall the protocol execution time is on the order of seconds). The average time to completion time for protocol (which is the waiting time in our simulation) is depicted in figure 9. The scaling factor f is depicted on the x-axis and the minimum peer group size l on the y-axis. The average time to completion is linear in f (note the logarithmic scale) and the effect of the minimum peer group size is shown in greater detail on the left side of figure 10. The graph is the average of the sections parallel to the y-axis where each have been divided by the scaling factor f . The completion time as a fraction of t_λ slowly increases with the minimum peer group size l , but generally stays below 1.0, i.e. the protocol can be completed within one polling interval. On the other

hand the number of polling requests per time unit (one hour with our assumptions) linearly decreases with the scaling factor f as shown on the right side of figure 10. One has to pick an appropriate scaling factor f given the requirements and capabilities of a real platform balancing the load by the requests against the time to complete the protocol.

6 Related Work

Several benchmarking system have been proposed and implemented. In [5] the emphasis is on the interactivity of the approach as the differentiating feature. Anonymity is mentioned as a potential obstacle. In [6] a practical system has been realized for the hospitality industry (hotels). They report customer acceptance problems as a major obstacle for the technology. We hope that privacy protection will help alleviate these problems. None of these systems use privacy enhancing or protecting technologies. Both systems ignore the peer group formation issue, since they deal with a homogenous customer basis (1 peer group).

In the project report [4] building a secure auction system is being described. This system uses SMC protocols and the authors intend to extend it to benchmarking.

Secure computation for benchmarking has been suggested in [1], but we prefer our protocols from [9], since they provide anonymity of the participants and have a central communication pattern corresponding to the service provider model. Also [1] focuses on theoretical aspects and ignores system architecture issues of building an enterprise system out of such a protocol. They therefore have not encountered the problem of peer group formation yet. We are not aware of any e-commerce system that has been brought to the system architecture phase that uses secure computation.

The survey performed in [13] identifies privacy as the major obstacle for e-commerce. Although no direct results for benchmarking or secure computation are reported, it supports our focus on privacy protection as the differentiating feature. In particular, since our system protects against the service provider and does not rely on a trusted third party, it is intended to increase customer acceptance.

[7] provides a good survey of data clustering techniques. K-means clustering has been introduced in [11]. Constrained k-means clustering [3] allows k-means clustering, such that a minimum cluster size is guaranteed. Nevertheless the resulting linear programming problem is infeasible for our requirements and different solution was needed. Other approaches have modified k-means clustering to adhere to constraints, e.g. [14], but different type of constraints. Another approach is to balance the cluster sizes in an optimistic fashion [2], but this may still violate the privacy of some customers and is therefore unacceptable. Furthermore, these customers would receive no or inferior

service, due to a significantly smaller peer group.

7 Conclusions

We have shown the important aspects when building a privacy-preserving benchmarking enterprise system besides the SMC protocols: peer group formation and protocol orchestration. Our conclusions are based on analysis of the end user centric use cases for such an enterprise system. We analyzed the impact of privacy on the peer group model and derive the first automatic peer group formation algorithm. The underlying k,l-means clustering algorithm might be of independent interest. Finally, we compared notification-based and polling-based computation of the protocols.

In summary, this paper gives several scientific recommendations to builders of privacy-preserving benchmarking enterprise systems:

- Under current network conditions the single peer group model is the only feasible.
- Automatic incremental peer group formation is possible using a specialized data clustering approach.
- Given a constant number of rounds of the protocol, polling-based computation can be expected to complete within one polling interval.

We would like to also stress the fact that is the first architecture for an enterprise system that uses SMC. We therefore anticipate the results to be useful outside the application domain of benchmarking and provide guidance to developers of other SMC-based enterprise systems.

Future work would be to apply benchmarking on a project level. We anticipate that the classification and clustering approach works well for KPIs solely dependent on the project, but KPIs that correlate over multiple project provide an additional challenge for privacy. A more thorough evaluation of the convergence pattern of the k,l-means clustering algorithm is outside of the scope of this work and a valuable extension in future work.

8 Acknowledgements

We would like to thank Heiko Krumm for insightful discussions on the paper. We would also like to thank the reviewers for useful comments to improve the paper.

References

- [1] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private Collaborative Forecasting and Benchmarking. *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, 2004.

- [2] A. Banerjee, and J. Ghosh. Frequency Sensitive Competitive Learning for Clustering on High-Dimensional Hyperspheres. *Proceedings International Joint Conference on Neural Networks*, 2002.
- [3] K. Bennett, P. Bradley, and A. Demiriz. Constrained K-Means Clustering. *Microsoft Technical Report*, 2000.
- [4] P. Bogetoft, I. Damgard, T. Jakobsen, K. Nielsen, J. Pagter, and T. Toft Secure Computing, Economy, and Trust: A Generic Solution for Secure Auctions with Real-World Applications. *BRICS Report Series RS-05-18*, 2005.
- [5] P. Bogetoft, and K. Nielsen. Internet Based Benchmarking. *Group Decision and Negotiation* 14(3), 2005.
- [6] J. Crotts, B. Pan, and C. Dimitry. Hospitality Performance Index: A Case Study of Developing an Internet-based Competitive Analysis and Benchmarking Tool for Hospitality Industry. *Proceedings of Conference of Travel and Tourism Research Association*, 2006.
- [7] A. Jain, M. Murty, and P. Flynn. Data Clustering: A Review. *ACM Computing Surveys* 31(3), 1999.
- [8] J. Jonas. Advanced Analytics in the Anonymized Data Space. Available at <http://jeffjonas.typepad.com/jeff-jonas/2006/03/index.html>, 2006.
- [9] F. Kerschbaum. A Privacy-Preserving Benchmarking Platform. *Technical Report*, 2006.
- [10] F. Kerschbaum, and O. Terzidis. Filtering for Private Collaborative Benchmarking. *Proceedings of the International Conference on Emerging Trends in Information and Communication Security*, 2006.
- [11] J. MacQueen. Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.
- [12] T. O'Reilly. What is Web 2.0. Available at <http://tim.oreilly.com/news/2005/09/30/what-is-web-20.html>, 2005.
- [13] G. Udo. Privacy and security concerns as major barriers for e-commerce: a survey study. *Information Management & Computer Security* 9(4), 2001.
- [14] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-means Clustering with Background Knowledge. *Proceedings of the 18th International Conference on Machine Learning*, 2001.