Privacy-Preserving Production Process Parameter Exchange

Jan Pennekamp*, Erik Buchholz*, Yannik Lockner†, Markus Dahlmanns*, Tiandong Xi‡, Marcel Fey‡, Christian Brecher‡, Christian Hopmann†, and Klaus Wehrle*
*Communication and Distributed Systems, RWTH Aachen University, Germany
†Institute of Plastics Processing, RWTH Aachen University, Germany
‡Machine Tools and Production Engineering, RWTH Aachen University, Germany
{pennekamp, buchholz, dahlmanns, wehrle}@comsys.rwth-aachen.de
{yannik.lockner, christian.hopmann}@ikv.rwth-aachen.de · {t.xi, m.fey, c.brecher}@wzl.rwth-aachen.de

ABSTRACT

Nowadays, collaborations between industrial companies always go hand in hand with trust issues, i.e., exchanging valuable production data entails the risk of improper use of potentially sensitive information. Therefore, companies hesitate to offer their production data, e.g., process parameters that would allow other companies to establish new production lines faster, against a quid pro quo. Nevertheless, the expected benefits of industrial collaboration, data exchanges, and the utilization of external knowledge are significant.

In this paper, we introduce our **B**loom filter-based **P**arameter **E**xchange (BPE), which enables companies to exchange process parameters privacy-preservingly. We demonstrate the applicability of our platform based on two distinct real-world use cases: injection molding and machine tools. We show that BPE is both scalable and deployable for different needs to foster industrial collaborations. Thereby, we reward data-providing companies with payments while preserving their valuable data and reducing the risks of data leakage.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; *Usability in security and privacy*; *Domain-specific security and privacy architectures*; • **Applied computing** → Engineering;

KEYWORDS

secure industrial collaboration; Bloom filter; oblivious transfer; Internet of Production

ACM Reference Format:
Jan Pennekamp, Erik Buchholz, Yannik Lockner, Markus Dahlmanns, Tiandong Xi, Marcel Fey, Christian Brecher, Christian Hopmann, and Klaus Wehrle. 2020. Privacy-Preserving Production Process Parameter Exchange. In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3427228.3427248

ACSAC 2020, December 7–11, 2020, Austin, USA

ACM ISBN 978-1-4503-8858-0/20/12…$15.00
https://doi.org/10.1145/3427228.3427248

1 INTRODUCTION

The impact of the Internet of Things (IoT) across various areas, e.g., the Industrial IoT (IIoT) and cyber-physical systems (CPSs), has led to a vastly increased sensor-based collection of production data that is not only used for their original purpose but also collected for further analyses [20, 40, 69]. Therefore, companies can utilize this analyzed data to improve their production processes, e.g., by feeding back results to optimize process parameters [32, 62]. In theory, sharing such data could be especially useful for quick adjustments to address customer change requests [56] or the cheaper commissioning of production lines [61]. Still, plenty of information is only retained locally [44], i.e., stored in data silos [33], hindering the entirety of companies to profit from already analyzed parameters.

Novel concepts, such as the Internet of Production (IoP) [33, 55], propose to further facilitate collaborations to utilize external knowledge. However, these advances are hindered by the lack of suitable solutions that consider industry needs [40], i.e., a prevention of sensitive information leakage on data provider as well as on the client-side and a valuable reward for providing data. Existing solutions [8, 9, 17, 21, 24, 77] either do not consider the privacy of the data-providing companies, do not take multiple data providers into account, or do not value the privacy of data requesting companies.

To counter this unsatisfying situation, we propose an exchange platform which allows companies to privacy-preservingly retrieve valuable (external) data. Motivated by a real-world use case in the domain of injection molding, we first explicitly specify the platform's general functionality. Moreover, we consider the already outlined industry needs, i.e., data provider privacy to not leak any confidential information within the shared data as well as client privacy to not reveal any future or current business plans through submitted requests, and a proper reward system to instigate companies to exchange their valuable data. Thereby, our platform facilitates the privacy-preserving exchange of production data in various domains and enables novel use cases, e.g., as introduced by the IoP, and unlocks new business models for participating data-providing companies as the value of data is widely acknowledged [33, 40, 56, 69].

However, a suitable concept for industrial data exchanges not only has to address privacy concerns but must also scale to its needs. Hence, we make use of (cryptographic) building blocks whose applicability has significantly improved [2, 57, 63, 66, 67]. In particular, we use Bloom filters and oblivious transfers to build a universal design called BPE, which we evaluate with two real-world use cases.

Contributions. Our main contributions are as follows.

- We propose a novel design that enables a privacy-preserving exchange of process parameters in industrial settings.
- Our new platform protects the data privacy of the data owner as well as the details of all potentially sensitive client queries.
- We open-source our fully-tested prototype[1], which is considered a functional artifact after an independent audit.
- We demonstrate the applicability of our approach in different scenarios and evaluate two real-world use cases: (i) a process parameter retrieval for injection molding to reduce the ramp-up phase of new production lines, and (ii) an exchange to improve the machine tool settings for individual workpieces.

Organization. In Section 2, we present our scenario illustrated with a suitable application in injection molding. Then, we introduce our design goals (Section 3) and used building blocks (Section 4). In Section 5, we detail our design of BPE and present our implementation in Section 6. We demonstrate its performance and applicability (incl. machine tool use case) in Section 7. We further discuss its security and additional design variants in Section 8. Then, in Section 9, we present related work and conclude our paper in Section 10.

2 SCENARIO

In this section, we motivate the need for a privacy-preserving parameter exchange by shorter ramp-up phases of new product lines in an injection molding use case and further introduce the benefits of such an exchange in general. Subsequently, we derive challenges that stem from the sensitivity of potentially shared information.

2.1 Transfer Learning for Injection Molding

The selection of injection molding (IM) processes as our starting point for deriving the benefits of a parameter exchange is based on their importance in thermoplastics. IM is responsible for the processing of around 55 Mio. tons of polymer materials worldwide each year, which grosses to 16.42 % of yearly polymer production [14, 58].

Production Process. The production is discontinuous. Polymer granulate is fed into a barrel in which a screw rotates, superposed with a translational drawback movement. The friction generated by motion between granulate, melt, screw, and barrel surfaces as well as heat introduced by heater bands around the barrel cause the plasticizing of the material along the axial transport to the screw tip. The screw anteroom fills during the drawback, accumulating material for the injection phase. During injection, the screw serves as a piston, injecting the polymer melt under high pressure into the cavity of an actively cooled mold. Once the volumetric fill of the cavity is concluded, the pressure-defined packing phase starts. The machine presses material into the cavity to compensate volumetric shrinkage occurring during the cooling process. Eventually, the closed mold opens and allows the ejection of the solidified part. During cooling, the machine recuperated the injected material in the screw anteroom, ready for the next production cycle [52].

Identifying Parameters. In this highly complex environment, a major challenge is to determine an optimized set of IM-machine parameters during the process setup. Suboptimal processes yield a higher scrap rate, resulting in lower part quality, or consuming more energy during (mass) production. While arbitrary optimization by expert knowledge [11, 47], i.e., by trial-and-error, or process-oriented optimization by simulation [6, 10, 34] is widespread, objective optimization can be achieved with unbiased mathematical approaches such as artificial neural networks (ANNs). ANNs are vastly audited methods to model the correspondence between IM-machine and process parameters as input and part quality parameters as output [15, 75, 83, 90, 92]. Models are used by evolutionary algorithms [47, 73, 84] or particle-swarm algorithms [1, 6, 43, 79], or to derive an optimized parameter set for (mass) production.

Transfer Learning. However, even shallow ANNs usually require an amount of training data, which is rarely obtainable during production, rendering ANNs unattractive for this use case. Transfer learning could mitigate this downside. In terms of machine learning, it defines the transfer of knowledge from a source assignment A_S, consisting of a source domain D_S and a source task T_S, to a target assignment A_T [87]. A domain D describes the data origin with a defined feature set X and a belonging probability distribution $P(X)$. A task T, on the contrary, is determined by the output parameter space Y and the mapping $f(X)$ with $f : X \rightarrow Y$.

While different transfer learning approaches have been identified [53, 87], network-based transfers have been validated when correlating IM-machine and part quality parameters. Especially the transfer between simulation and experimental data is intensively researched and also implemented as demonstrators [37, 38, 48, 82]. The network-based transfer considers a pre-trained model $f_S(Y_S)$ which has been fitted by abundant labeled samples (x_S, y_S) from a source domain D_S. Then, $f_S(Y_S)$ is transferred to a target task T_T. Here, as little as possible training data (x_T, y_T) to achieve a good model is desired. However, simulations have to be rerun if influences on the IM process, such as the material or the part change, introducing a high repetitiveness. Especially the successful transfer of knowledge between processes with varying influencing parameters make transfer learning a real alternative to expert knowledge. The transfer between processes of different molded parts indicates promising results [36], suggesting a close correlation of the transfer learning success with the similarity of source and target domain.

A Lack of Models. When preparing for process modeling, suitable data or models might only be available at other, unaffiliated companies. However, customer interests and legal boundaries as well as the desire to retain a maximum of process information as proprietary information conflict with the availability and publication of production data in general [54, 56] and also in the field of plastics processing [45]. However, advances in data acquisition and availability on the own enterprise in terms of Industry 4.0 expedite the need for new business models and value streams. A reasonable compensation as well as a proper separation of relevant process data and customer information can possibly motivate enterprises to assume a role as data provider. Hence, a privacy-preserving parameter exchange would be highly beneficial for the establishment of data-driven methods for process optimization in manufacturing.

2.2 Production Process Parameter Exchange

We use injection molding as an example showing that utilizing production data across organizational boundaries is a desirable aspect of the future production landscape. Figure 1 shows a workflow of

[1] Our Python code is available at: https://github.com/COMSYS/parameter-exchange.

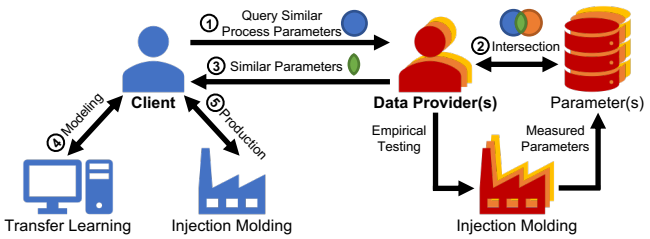


Figure 1: An exchange of process parameters between companies illustrated based on the use case of injection molding.

how companies can profit. Thereby, (1) clients query parameters of similar processes from (external) data providers, (2) data providers curate matching parameters from their own production and (3) send these results back to the client which can enhance both their (4) modeling, e.g., integrating more real-world process data, as well as (5) production, e.g., utilizing well-fitting configurations.

Need for a Suitable Approach. Today, companies already collect much process information [44], i.e., they own data that is potentially also relevant for other companies. However, the lack of suitable data security mechanisms, missing opportunities to gain benefits from sharing data, and the fear of leaking sensitive business secrets, i.e., information leaking know-how of a company, manifests the existing silo mentality [33, 72]. Simply making all information freely accessible is no option in competitive environments.

Contrary to work in the medical domain [41, 78, 93, 95], where usually a single stakeholder offloads data to an untrusted cloud (with m stakeholders querying information), we consider a setting where multiple stakeholders offload their data and multiple (other) stakeholders query information. To retain the utility of information, opposed to best practices when handling sensitive user data [81], we cannot anonymize data records when serving them in the cloud.

Hence, companies need suitable ways to ease the exchange of process parameters without leaking confidential data and to introduce a quid pro quo for data providers for motivation to share their data even with potential competitors. Instead of simply retaining their process information in local data silos, companies could sell their data, which is collected anyway, to third parties which themselves want to reduce their costs by utilizing this information, e.g., by performing process optimization using this shared information.

2.3 Scenario Challenges

Based on the need for a privacy-preserving exchange of production data, we now highlight scenario-specific challenges. While the most crucial aspects concern privacy and information security, further challenges are related to the operation of such a data exchange.

Crucial Properties. The most crucial properties directly follow from the competitive environment. As companies are notoriously cautious [72], they intend to only share specifically requested datasets [33] while preserving long-term security [13], i.e., data requests must be specified precisely. Consequentially, a global catalog of existing data or a way to browse available data items must not exist. Thereby, data providers do not to lose control of their offered data while monetizing its usage. Furthermore, requesting companies, i.e., clients, want to utilize external information for their benefits, e.g., to improve their production processes, and do not want to share their interest. To still achieve a competitive advantage, the utilization of requested and retrieved data must remain private, including the process of identifying relevant data items.

Industrial Setting. Additional challenges originate from the industrial setting. In contrast to the analysis of personal data, companies impose very strict usage rules [94], e.g., molds are regularly owned by customers of injection molding manufacturers and only loaned to the companies for production. Therefore, any unintentional disclosure of intellectual property has to be strictly avoided. For example, in injection molding, geometrical data of produced parts is needed for the calculation of similarity scores. Hence, it should remain private if not shared or sold on purpose. Establishing trust in a single third party is highly unlikely as companies strive to limit the threat of data leaks [72]. Regardless, creating new business models for companies with existing data repositories could incentivize their participation [94], i.e., a privacy-preserving exchange could open up a new stream of revenue for data-providing companies. Consequentially, a suitable billing mechanism is required.

Operational Considerations. Given that privacy-preserving designs and security building blocks usually introduce a computational overhead and possibly add communication [88], the data exchange must be executed within reasonable, use case-induced boundaries. This aspect is not limited to the exchange, but also includes a potential setup. In particular, various security and privacy guarantees might directly contradict the feasibility of a proposed concept. Furthermore, data-providing companies might not be able to participate in data exchanges and their associated protocols. Hence, flexibility is needed to also account for such situations.

Based on the example of injection molding, we discussed the value of production data and the benefits of utilizing external information, i.e., we discovered reasons for inter-organizational data exchanges. Here, the industrial setting and the need for strong security and privacy challenge the establishment of a widely-accepted exchange platform.

3 DESIGN GOALS

Based on the description of our considered scenario, we now derive a set of five distinct design goals, which must be considered by any concept that proposes an exchange of process parameters. These goals summarize the needs of the individual participants (**G1** and **G2**) as well as universal conceptual requirements (**G3**, **G4**, and **G5**).

G1: Provider Privacy. Companies offering their process parameters to other companies still have a strong desire to maintain their privacy and data secrecy as the combined information of offered data can reveal internal information. For example, in our injection molding use case, knowledge about the data provider correlated with shared geometry parameters could result in the identification of specific parts and, thereby, reveal highly sensitive information about the implemented production processes and the company's customers. Thus, data providers mandate that access to their data is only granted in parts and only to authorized parties. Furthermore, as long as data providers do not share provider-identifying information voluntarily, they must remain anonymous for all clients.

G2: Client Privacy. Protecting the client's requests is just as important for the success of a process parameter exchange. First, data providers must not be able to attribute the requested data items to the client. Otherwise, information on new developments might

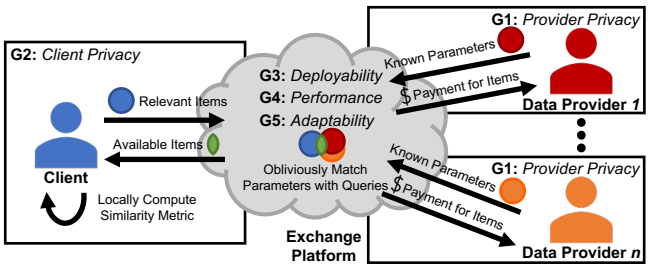


Figure 2: Apart from design-specific goals, any suitable exchange platform has to especially consider G1 and G2.

be identifiable and directly linked to a company. Second, the request generation, i.e., the metric identifying meaningful data items, must remain private. In a production landscape with ubiquitous data exchanges, such knowledge constitutes the competitive advantage as the individual parameters can be considered a common good.

G3: Deployability. In terms of realizing a real-wold exchange, two main aspects are crucial. On the one hand, requests must allow for a flexible matching, i.e., clients can use any metric they like to identify meaningful data items and must be able to request these identified data items. Hence, this metric can neither be part of the exchange, nor should it be public during the exchange (cf. **G2**). On the other hand, to incentivize data providers to offer their valuable data, a billing mechanism is required to enable new business models. Finally, providers must not be required to remain online all the time, i.e., client requests can be handled without their active involvement.

G4: Performance. As privacy-preserving designs usually incur a performance overhead, the overall performance should still be reasonable and appropriate for the respective use case, i.e., it should not outweigh the potential benefits. However, specifying concrete constraints is counterproductive since performance limits can always depend on the importance of exchanged data, i.e., very valuable data can justify significant resource needs. Similarly, introduced hardware and network requirements should be reasonable as well, i.e., ideally, commodity devices are sufficient to participate.

G5: Adaptability. Along with the previous goals of privacy (**G1** and **G2**) and performance (**G4**), adapting the trade-off between security and performance must be considered. Some data is more sensitive than others and should be treated accordingly, i.e., a concept to deal with these situations should be offered to optimally address the trade-off while minimizing the number of changes.

These design goals are critical to realize a parameter exchange. We provide an overview of our use case-independent scenario in Figure 2 along with the design goals and the exchanged messages. Any proposed design must provide a concept of how to connect clients with relevant data providers. In particular, it must realize the functionality, which we illustrate with a cloud, while addressing the presented design goals.

4 PRELIMINARIES

To establish a common background of our utilized building blocks, we briefly introduce their concepts in this section. Namely, we rely on Bloom filters and oblivious transfers (OTs) as components of our design and optionally on private set intersections (PSIs) for a variation that offers improved security guarantees (cf. Section 8.2).

Bloom Filter. A Bloom filter is a probabilistic and space-efficient data structure that allows for efficient membership tests without an efficient possibility to extract a list of all inserted elements [7]. Apart from insertions, Bloom filters support membership tests that check whether a specific element was inserted. Due to its probabilistic property, such queries can return false positives with a tunable false positive (FP) rate ε. However, false negatives cannot occur.

A Bloom filter B consist of an array with fixed length n and uses k hash functions $(h_1, \ldots, h_k)$ to map elements to the individual fields of the array. Inserting an element x works by setting $B[h_i(x)] = 1 \, \forall i \in \{1, \ldots, k\}$. Consequently, querying an element y equals a bitwise comparison of $h_i(y) \, \forall i \in \{1, \ldots, k\}$ with B. Taking the FP rate into account, y was inserted in B if all set values in $h_i(y) \, \forall i \in \{1, \ldots, k\}$ are set in B as well. The FP rate ε can be computed based on the number of stored elements m, the length n, and the number of hash functions k [70]: $\varepsilon = (1 - (1 - \frac{1}{n})^{km})^k$.

Adjusting the individual parameters (e.g., to reduce ε) influences the storage size as well as the processing of insertion and querying.

Oblivious Transfer (OT). OTs allow a client (receiver) to retrieve one of two items from a server (sender) without the server knowing which of the items has been transferred [29, 60]. After the OT, the receiver has access to a single item only and is unaware of the other. This basic form is also called 1-out-of-2 OT. Several additions enable more sophisticated scenarios: 1-out-of-n OTs or k-out-n OTs [18]. For improved performance, a few expensive base OTs can seed a large number of less expensive OT extensions [3].

To achieve the required security, i.e., hiding the contents of the data transfer, significant computational overhead and communication are introduced [51]. While the trade-off between computations and communication is adaptable, OTs are still costly, and, thus, cannot be used to efficiently transfer large amounts of data.

Private Set Intersection (PSI). PSI is a cryptographic building block that allows two parties to calculate the intersection of two confidential sets without revealing included elements [25]. Depending on the concrete implementation, only one or both parties learn the content or the size of the intersection [22]. To realize PSIs, different cryptographic concepts have been utilized. For improved security, many efficient designs utilize OTs [42, 67]. Similar to OTs, PSIs also suffer from overhead with increasing set sizes.

5 BPE: A BLOOM FILTER-BASED EXCHANGE

In this section, we propose *BPE*, a novel privacy-preserving **B**loom filter-based production process **P**arameter **E**xchange for companies.

5.1 Notation for the Exchange of Parameters

To provide a more formal understanding, we first introduce the underlying foundations of any offloaded record and potential queries. A parameter record $p = x \parallel y = x_1, \ldots, x_n, y_1, \ldots y_m$ consists of a payload y and a number of (identifying) parameters x_i. Here, x can correspond to a part that should be manufactured at a specific machine while y, for example, represents used machine settings. The respective indexing is defined by $X \rightarrow H : h_k(x'_1, ..., x'_n) = id_{x'}$ with a use case-specific rounding function $r(x_i) = x'_i$ (cf. Appendix A.1) to derive its input, i.e., we apply a binning to match related records to the same index. Both h and r are globally defined by the exchange platform. We derive $idk_{x'} \in K$ as truncation of $id_{x'} \in H$, for the indexing of AES encryption keys $k_{x'}$, i.e., the encryption key can be derived using the identifying parameters x_i only. Records

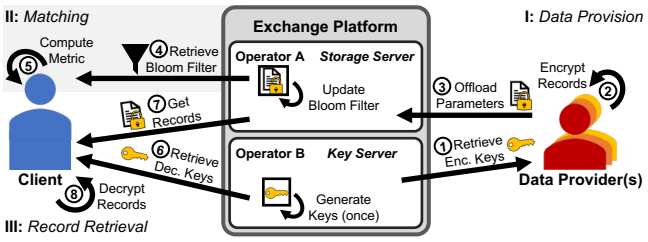


Figure 3: Our exchange platform is split into two components to separate key material from shared ciphertexts.

can share an encryption key if $K \subsetneq H$, i.e., fewer indices are available at the key server, which also handles the mapping $(idk_{x'}, k_{x'})$. To reduce the computational overhead, a smaller set size K is desirable (cf. Section 7.3). An encrypted parameter record $c_{x'}$ is further defined as $c = E_{k_{x'}}(p)$. The storage server maintains the respective pairs $(id_{x'}, c_{x'})$. A single index $id_{x'}$ can refer to the ciphertexts of multiple records due to the rounding with $r(x)$ (to put records into bins). By design, these ciphertexts also share their encryption key.

A similarity metric $s(q)$ (cf. Appendix A.2), which should be considered sensitive (**G2**), computes a candidate set S from a client-known initial record q. To compute S, a client does not necessarily require any payload data as records are indexed with their identifying parameters x_i only. The client eventually retrieves all records q' with $id_{q'} \in S$ that are available (indexed) at the exchange platform.

In the following, we provide a high-level design overview before focusing on the entities. Subsequently, we detail BPE's protocol.

5.2 Design Overview

We realize a privacy-preserving exchange platform (cf. Figure 3) by splitting ciphertexts and key material over two independent operators. To achieve privacy, our platform must be built up with carefully selected operators (cf. Section 6.1) who may not collude (to ensure **G1**). Apart from this aspect, both clients and data providers do not have to trust any other entity in our proposed architecture.

I: *Data Provision.* Data providers retrieve encryption keys $k_{x'}$ from the key server via oblivious transfer, encrypt their records p, and then offload (cf. **G3**) their encrypted records c, annotated with the indices $id_{x'}$ to the storage server, which inserts the indices of received records (from all data providers) into a single Bloom filter. OTs hide the data providers' access patterns from the key server.

II: *Matching.* Upon request, the client receives the Bloom filter from the storage server. Then, starting with a known record q, the client locally computes all indices that she is interested in (her candidate set S) based on a similarity metric s (her intellectual property) and subsequently tests these indices $id_{q'}$ for membership in the Bloom filter. The local matching provides client privacy (**G2**) as the query content is not shared with another entity. Using a Bloom filter, the storage server only shares a probabilistic data structures of all inserted hashes and not the values or full indices.

III: *Record Retrieval.* If the client found an index that was inserted into the Bloom filter, she retrieves the corresponding decryption key $k_{x'}$ from the key server via OT. She further purchases the respective ciphertext from the storage server, which also triggers the billing (out of scope for this paper) for this retrieval. Finally, she decrypts the ciphertext $c_{x'}$ and gets access to the wanted parameter record. Again, OTs hide the (clients') access patterns from the key server.

After these three phases, the client is oblivious of data-sharing providers (cf. **G1**), and assuming a proper billing mechanism, the selling provider cannot identify the purchaser either (cf. **G2**). Further, the client's valuable similarity metric is kept private as the client locally computes the matching (cf. **G2**). As all items are encrypted, the storage server is unaware of the mediated records (cf. **G1** and **G2**). Moreover, the key server is oblivious of requested and transferred keys given that the respective communication places via OTs (cf. **G1** and **G2**). Finally, computationally expensive tasks are mostly run at the client or data providers keeping the total utilization of our platform providers comparatively low (cf. **G4**).

5.3 Entities and Trust Assumptions

Next, we detail all four involved entities to clearly understand their individual responsibilities, interactions, and trust relationships as well as how our platform incorporates their individual interests.

Data Provider(s). Given that potential providers invest resources when collecting parameter records [1, 30, 38] and possibly share their know-how with business partners or competitors, they are only willing to contribute against compensation [94], e.g., payments, and despite a required participation overhead. Furthermore, the data provider's identity and valuable provided data must be protected, i.e., no third party may get access to all records. To this end, in our platform, we separate key material and ciphertexts by relying on two non-colluding operators. To reward the provider, our platform bills clients, i.e., data providers receive payments for their records if clients retrieved them. Finally, to ease the participation, our platform allows providers to offload data once, which is not time-critical, and supports adding additional records at a later time.

Client(s). The privacy interests of clients are twofold. First, similarity metrics are potentially valuable as they originate from ongoing research [38, 82] and, thus, must be protected accordingly. Second, the initial input for the metric (i.e., a known record) is sensitive as well since it might reveal internal information [16], e.g., production plans. Apart from privacy interests, clients should only have to pay for retrieved data records to compensate providers.

While our design requires the client to reveal certain parts of her candidate set S, i.e., the matched (requested) indices to the storage operator (e.g., to realize the billing), it completely relies on a local matching, i.e., the metric as well as the initial input remain at the client. Further, as the storage server is unable to decrypt or identify the requested records, it cannot draw conclusions about this sensitive information from the transmitted indices. Moreover, client and key server only interact via OTs for potentially leaking requests, i.e., the key server never learns anything about the client's query. Although, depending on the number of input parameters and the used similarity metric, the matching can become time-consuming, it is usually not very time-critical. For instance, injection molding productions are planned weeks in advance [23] and, thus, a processing of multiple days for the matching and retrieval is feasible.

Key Server. The interests of the key server operator are limited to an ideally low computational and storage overhead. While the generation of the key material for every possible index in a preliminary phase temporarily generates a high workload and forces the server to store all generated keys, the number of keys is limited by the used OT set size. Thus, the key generation neither produces

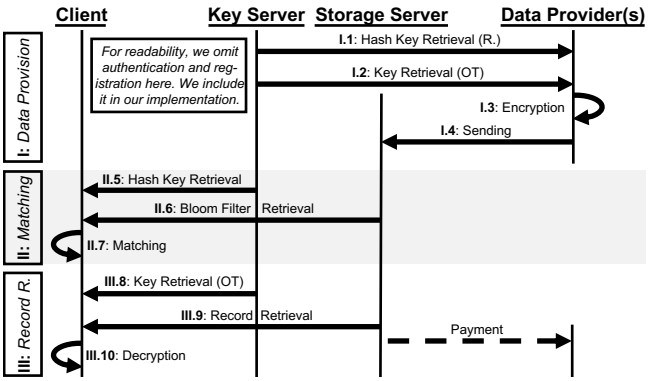


Figure 4: Sequence chart detailing the messages of BPE

significant overhead nor requires excessive storage. Although data transfers via OTs are known to be computationally expensive and time-consuming [2], a fundamental requirement is to meet the interests of key-retrieving providers and clients, i.e., OTs prevent any information leakage from these entities [29, 60], including the number of transferred keys [18]. Hence, except for non-collusion with the storage server, no trust in the key server is required.

Storage Server. In terms of computation, the interests of the storage key server operators are aligned, i.e., low overhead is desirable. Since the storage server only maintains the Bloom filter (i.e., inserts new records) as well as shares it and requested data with interested clients, no expensive computations are performed.

However, in our design the storage server operator has a very sensitive task, i.e., the storage server operator must not collude with any of the data providers and the key server, and therefore must be chosen carefully (cf. Section 6.1). While the operator must observe the indices of offloaded and requested records to enable billing, this knowledge does not allow for conclusions on any of the sensitive information, e.g., the client's similarity metric $s(q)$. Only if the storage server operator colludes with the key server operator or the offloading provider and therefore gains insight into requested data, conclusions about the client's candidate set are possible.

Overall, BPE requires no trust between clients and data providers and relies on the key server and storage server operators to not collude, i.e., to ensure provider privacy, and the storage server operator to not collude with data providers to realize client privacy. We propose suitable server operators in Section 6.1, and further consider more complex variants, offering additional security guarantees in Section 8.2.

5.4 Protocol Sequence

Figure 4 outlines all exchanged messages of our three-phased BPE design in more detail whose meanings we break down now.

I: Data Provision. Initially, (I.1) data providers request a hash key from the key server to compute all needed indices $id_{x'}$, which (a) prevents the storage server operator from concluding the stored data by brute-forcing the indices and (b) increases the variability of indices allowing to reduce their size. Subsequently, (I.2) the provider requests key material $idk_{x'}$ from the key server via OT to (I.3) then encrypt the records. Finally, (I.4) the provider sends the encrypted records and their indices to the storage server.

II: Matching. In advance of the actual matching process, the client requests (II.5) the hash key from the key server and (II.6) the Bloom filter from the storage server. The hash key enables the client to (II.7) derive the indices of candidates, i.e., her candidate set S, computed by her metric $s(q)$ based on input q by checking whether the received Bloom filter contains the respective indices.

III: Record Retrieval. After matches have been determined, the client (III.8) retrieves the required decryption keys $idk_{x'}$ via OTs from the key server and (III.9) requests the encrypted records $E_{k_{x'}}(p)$ from the storage server using the matching indices $id_{x'}$ which consequently triggers a payment process. Finally, (III.10) the client decrypts the retrieved ciphertexts to obtain the records.

Afterward, the parameter exchange with this client is concluded.

6 REAL-WORLD REALIZATION

For our injection molding scenario, we give an overview of suitable platform operators that underline its real-world deployability (**G3**). Then, we detail the libraries of our fully-tested implementation.

6.1 Exchange Platform Operators

To realize the claimed privacy guarantees, our design requires non-colluding platform components. Consequentially, key and storage server must be hosted by different stakeholders. As described in Section 5.3, both servers require different levels of trust. The key server is oblivious of key retrievals. Hence, no trust is required. Accordingly, it can be operated by an untrusted third party. Here, startups that charge a small fee for each key retrieval come to mind. When using a trusted third party, the key retrieval (during data provision and record retrieval) could be implemented without oblivious transfers. However, as we detail in Section 7, the matching phase is responsible for most of the runtime in real-world settings. Thus, we prefer an OT-based retrieval without any trust assumptions.

The storage server is more critical for both provider and client privacy. On the one hand, this server learns the ciphertexts of stored records and the associated data providers. On the other hand, the storage server is aware of the clients' matches. Therefore, public organizations or the government are well-suited for hosting the storage server. Concerning our injection molding scenario (cf. Section 2.1), suitable organization are the *Association of German Engineers (VDI)* [85] or the *Mechanical Engineering Industry Association (VDMA)* [86]. These organizations are already semi-trusted by injection molding businesses and usually funded through membership fees. Hence, they are more appropriate to operate the storage server than a (random) untrusted, potentially unreliable third party.

Using our design, we do not require any trust between (all) clients and (all) data providers, facilitating the parameter exchange, as each of them only interacts with the (semi-trusted) storage server.

The costs of both entities could be covered by a participation fee paid by all participants of the exchange platform. Another possibility is a per operation fee, e.g., for each key and record retrieval.

6.2 Implementation

We implemented a client and a data provider application, as well as the exchange platform in Python 3. For OTs, we utilize libOTe [64] and select the semi-honest 1-out-of-n OT algorithm KKRT16 [42]. For PSIs (cf. Section 8.2), we rely on libPSI [65] and choose the semi-honest PSI algorithm KKRT16 [42]. We call them using Cython [4].

Pybloomfiltermmap3 [59] serves as our underlying Bloom filter. For its transmission, we utilize the library's base64 export. We realize both servers as Flask [68] applications with Celery [76] as a task queue utilizing a Redis [71] broker. Celery workers handle OT and PSI endpoints as well as Bloom filter updates at the storage.

The server APIs are realized as RESTful JSON APIs that require HTTP basic authentication using the authorization header field. All communication between applications and the server APIs are protected by TLS 1.2 [27]. We decided to calculate the TLS overhead for OTs and PSIs (cf. Section 7.1 for details). The storage server relies on SQLite [80] while the key server keeps all keys in memory.

7 EVALUATION OF BPE

As performance (**G4**) is a critical aspect to realize a real-world deployable solution (**G3**), we now evaluate BPE. In Section 7.1, we present our experimental setup for all measurements. Afterward, we show BPE's real-world feasibility in four individual steps. First, in Section 7.2, we investigate the scalability of our integrated building blocks. Then, in Section 7.3, keeping the previous results in mind, we analyze the performance of our three-step protocol with generated data, before evaluating it using our real-world use case and realistic queries in the domain of injection molding in Section 7.4. Finally, we demonstrate our design's universality based on a second real-world use case dealing with machine tools in Section 7.5.

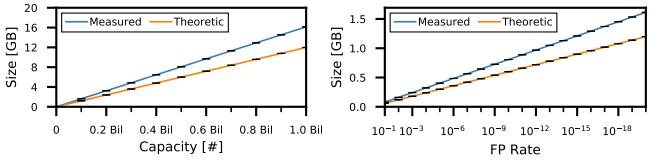
7.1 Experimental Setup

For all measurements, we utilized a single server (2x Intel XeonSilver 4116 and 196 GB RAM) and performed 10 runs each. All entities ran on the same machine and communicated over the loopback interface. We measured the data volume with tcpdump [39] and, if applicable, configured latency and bandwidth with tcconfig [35].

We noticed an unreasonably out-of-scale overhead in the (unsupported) TLS endpoints of libOTe and libPSI forcing us to add the expected overhead arithmetically. To this end, we evaluated the TLS handshake overhead (53.94 ms) and the maximum TLS throughput (567.16 MBit/s) on our evaluation server using Flask's TLS settings (TLS 1.2, *ECDHE-RSA-AES256-GCM-SHA384*, and the elliptic curve *secp256r1*). If not stated otherwise, we included the calculated TLS overhead based on these values (hatched in our plots). The hash key and the encryption keys are 128 Bit long each. We only parallelized the Bloom filter-based matching and the OT-based key retrieval.

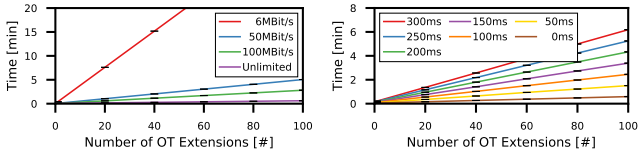
7.2 Performance of BPE's Used Building Blocks

Before evaluating the (combined) BPE design, we first investigate the performance of our building blocks regarding the influence of different parameters to show their applicability in real scenarios.



(a) Even huge capacities lead to reasonable file sizes. **(b) The FP rate only has a linear influence on the file size.**

Figure 5: A larger capacity and a lower false positive (FP) rate linearly influence the array length of the Bloom filter.



(a) A reduced bandwidth affects the large transmissions of OTs. **(b) Latency impacts the communication overhead of costly OTs.**

Figure 6: Both decreased bandwidth and increased latency negatively influence the linear coefficient when considering the number of OTs and the processing time.

We thus examine the influences on capacity and FP rate on the size of Bloom filters, the runtime of OTs given different bandwidth limits and latencies, and the candidate set size S for different metrics.

7.2.1 Bloom Filter. The matching phase of BPE relies on a Bloom filter to enable the checking for specific indices. Thus, we evaluate two parameters of relevance, which both affect the size, i.e., increase the amount of data that must be transferred to clients. First, the capacity limits the maximal number of supported indices. Second, the FP rate determines the probability of incorrect membership tests, i.e., an index is not available even though the test is positive.

We chose to evaluate capacities up to 1 Bil. elements (with a fixed FP rate of 10^{-20}) as an excessive upper border. For our injection molding scenario, a capacity of 1 Mio. is considered reasonable by domain experts. As false positives result in the retrieval of unwanted records, the FP rate must be configured as small as possible. Accordingly, we consider values up to 10^{-20} (with the capacity fixed at 100 Mio. elements) to support billions of membership tests.

Figure 5 shows the influence of these parameters on the size. Due to the used base64 encoding, we exceed the calculated theoretic optimum. Even for immense capacities, the size is reasonable due to its linear scaling with the capacity (cf. Figure 5a). Nowadays, one-time transmissions (to clients) of less than 20 GB are realistic [19]. Notably, the size increases linearly for an *exponentially* decreasing FP rate. Thus, even small FP rates (e.g., 10^{-20}) yield feasible sizes.

Further, the query and insert times are relevant for the matching as well as the provision phase. We detail in Appendix B.1 that while the performance of the query time is mostly unaffected by both capacity and FP rate, and only depends on the number of performed queries, the insert times increase approximately linear with both increased capacity and FP rate. However, the data provision is a one-time activity and occurs with time delay. In the following, we fix the capacity at 100 Mio. elements as our injection molding scenario is unlikely to exceed this value. We further set the FP rate to 10^{-20}, which results in a comparably small Bloom filter size (<2 GB).

7.2.2 Oblivious Transfers. We rely on OTs for the data provision and record retrieval. By considering legitimate businesses, which are bound to a jurisdiction, we can reasonably rely on a semi-honest OT protocol [42], and thereby avoid unnecessary protocol overhead.

The runtime is mainly influenced by the set size (total number of keys) and the number of OT extensions (number of retrieved keys). A large set size K is desirable as more distinct encryption keys can be handled by the key server, i.e., fewer records share their encryption keys. The number of retrieved keys depends (i) on the number of sharable records at the data provider and (ii) the number of matches at the client, which are both highly use case-specific.

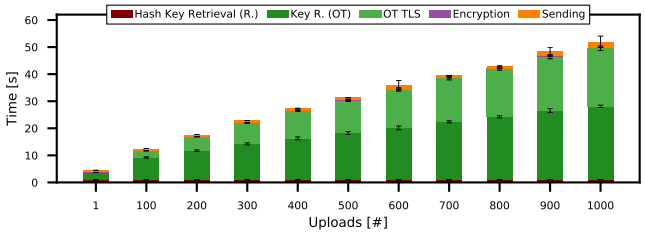


Figure 7: Obliviously obtaining the encryption keys is the most time-consuming step when offloading data records.

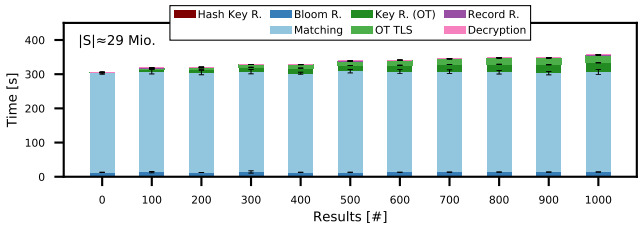


Figure 8: The local computation of complex similarity metrics by the client dominates the retrieval of data records.

As we outline in Appendix B.2, the runtime of the OTs scales both linearly with an increased set size and the number of performed OT extension. We fix the set size at 2^{20}, which allows for more than 1 Mio. distinct keys. Thus, in our injection molding use case, each record could be encrypted with an individual key. For this set size, the retrieval of 200 keys takes less than 70 s (cf. Appendix B.2).

In Figure 6, we detail the influence of realistic network conditions [5, 19, 91] on OTs in terms of latency and bandwidth. We limit the bandwidth asynchronously (labels correspond to server-client speed). The client-server link is restricted to 1/10 to mimic common broadband connections. Both latency and reduced bandwidth add a factor to the overhead of additional OTs. However, even for a large latency of 300 ms, 100 OT extensions are executed in ≈6 minutes. Similarly, a severely constrained bandwidth (6 MBit/s) adds only 30 min overhead. While the data provision is not time-critical, several days for client requests are acceptable (cf. injection molding). Thus, we can tolerate slow Internet links with significant latency.

OT protocols balance the trade-off between computation and communication overhead differently [2]. Hence, the underlying OT protocol can be selected based on use case-specific needs.

7.2.3 Similarity Metrics. The matching phase of BPE is driven by the number of elements that the similarity metric $s(q)$ returns as each element must be tested for membership in the Bloom filter. Even today, membership tests are time-consuming if the candidate set S grows very large. A usable metric for transfer learning in injection molding is able to evaluate the similarity of two processes, hence, how data from a different process can substitute missing training data of the process, which must be modeled. Today, sensible real-world metrics are still being actively researched [38, 82].

The number of elements further depends on the exact representation of parameters. For example, a fine-granular rounding results in more values for the same interval of potentially interesting records. An increase of the granularity of one parameter (x'_i) yields a linear increase of candidates while a granularity increase for all input parameters (x'_1 to x'_n) results in an exponential blow-up of the candidate set S. The number of input parameters n is further influential because S grows exponential in the number of varied parameters.

Given that the similarity metric $s(q)$ is solely computed at the client, its performance does not impact the other parts of our design.

7.3 Combined Performance of BPE

We now combine the individual building blocks and look at the overall performance of each phase. We treat the data providers (data provision) and clients (matching and record retrieval) separately.

Data Provision. Figure 7 details the runtime when offloading up to 1000 records and shows that the provision scales linearly with the number of records. Here, we used records with 100 parameters, each representing a float chosen uniform at random. Accordingly, all records have unique identifiers and require a unique key for encryption. The key retrieval dominates this phase. The length of the records only marginally influences the runtime (cf. Appendix B.3).

Matching and Record Retrieval. In contrast to the data provision, client requests can be time-critical and are use case-specific.

We offloaded records with 100 parameters and use 10 parameters as input for the indexing ($n = 10$, $m = 90$). Each input parameter is discretized to three digits. As similarity metric $s(q)$, we computed an offset of 0.5 % for each input record q. We ensured that sufficient records are matched and available that the storage server. While OTs mainly impact the provision, Figure 8 shows that the matching dominates the client queries. Despite the good performance of Bloom filters (cf. Section 7.2.1), the matching is expensive as it results in a large candidate set S of >29 Mio. elements. While we observe a runtime below 5 min, real-world metrics might produce sets that are magnitudes larger, further increasing the runtime.

Given that several days for client requests are feasible (cf. injection molding), metrics with excessive candidate sets are possible as well. Besides, testing for membership is embarrassingly parallelizable and does not depend on external entities. Accordingly, clients can scale their metrics to their constraints, i.e., time and computational resources.

7.4 Real-World Performance Measurements

To test the real-world applicability of our scenario (cf. Section 2), we now operate on a total of 4620 genuine records, consisting of 28 parameters each. They describe the production of toy bricks. For the optimization of the IM-machine settings during process setup, each toy brick is defined by $m = 21$ geometry parameters, while the remaining $n = 7$ parameters describe 6 essential machine settings (injection volume flow, melt temperature, mold temperature, packing pressure, packing pressure time, cooling time) and one quality indicator (part weight). For other use cases than the optimization of IM-machine settings during the process setup, the data and its representation may differ. Here, sensitive information is represented by the machine settings and the corresponding part quality: Only combined with the identifying parameters, i.e., geometry information, the data can be used for transfer learning. With this indexing, we have a total of 60 indices, where each index points to 77 unique records that contain varied machine parameters.

Data Provision. The entire provision takes less than 12 s (cf. Appendix B.3) and is comparable to our previous experiment (cf. Figure 7). Thus, even orders of magnitude more records are feasible.

Matching and Record Retrieval. We evaluated two potentially used metrics to look into client queries. For metric *IM-2%*, we

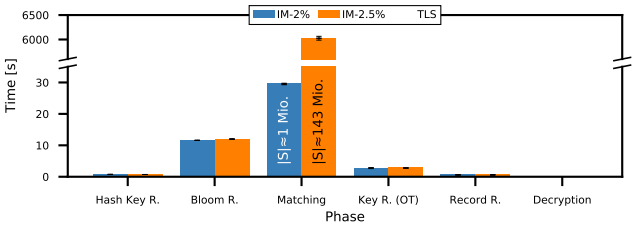


Figure 9: The larger candidate set produced by *IM-2.5%* leads to a dominating matching phase for the full client request.

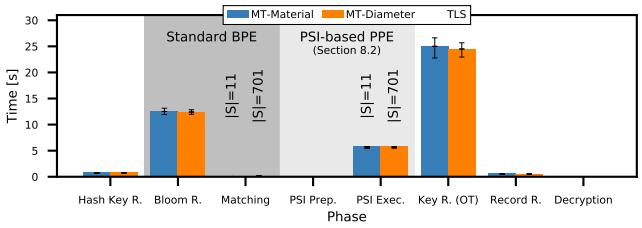


Figure 10: Both metrics produce negligible overhead for the matching. For these sizes, both BPE and PPE are feasible.

used a relative offset for all (21) input parameters of 2 % and for *IM-2.5%*, a relative offset of 2.5 %. The rounding is set to two digits for each input parameter. In our example, both metrics resulted in a single match, i.e., the client retrieves the records corresponding to a single index. As visualized in Figure 9, the matching quickly dominates client queries, rendering the remaining steps negligible.

We again underline that the locally conducted membership tests are the crucial factor. Keeping the use case-induced time constraints (of several days) in mind, even metrics with significantly larger candidate sets can be supported. In conjunction with the virtually irrelevant performance of offloading records, we thus conclude that the performance of BPE is well-suited for a privacy-preserving exchange of parameters in the domain of injection molding. Therefore, it could greatly support the application and implementation of transfer learning for the highly complex task of process setups.

To conclude, we showcased that BPE can handle client requests with a large candidate set in a real-world setting. The one-time required provision is negligible for the performance of BPE. Our client-sided computation easily enables complex similarity metrics in the future.

7.5 Universally Applied Parameter Exchange

To showcase the applicability of our proposed exchange platform, we now look into a second real-world use case, i.e., machine tools.

7.5.1 A Parameter Exchange for Machine Tools. For subtractive manufacturing (turning and milling), major factors that affect the workpiece quality and productivity are the machine tools and the choice of the cutting parameters, such as cutting speed, feed rate, and cutting depths. Traditionally, cutting parameters are determined based on experience or manufacturer-specific recommendations. In a lengthy process, the machine operator starts with a conservative value and then tunes the parameter through real tests. While this approach yields acceptable results, it is time-consuming.

Thus, optimization methods are actively being researched [12, 26, 46]. Particle swarm optimization promises to obtain optimal cutting parameters for certain requirements, such as roughness and tool lifetime [46]. A model-based approach integrating real-time process data actively combines quality measurement data [12]. Thereby, the potential for optimization of the cutting process can be estimated, resulting in improved productivity. Similarly, Denkena et al. [26] apply machine learning to determine the optimal cutting parameters under consideration of the process outcome.

However, all optimization methods require detailed modeling of the machining process and the machine tool, which is difficult and not always feasible. Meanwhile, other companies may already have optimized cutting parameters, ready for a parameter exchange.

7.5.2 Evaluation. Here, we rely on a dataset with 600 records with 19 parameters each ($n = 17$, $m = 2$). Each record has a unique index.

Data Provision. Offloading all records is completed within 30 s and, therefore, uncritical for any real setting and its providers.

Matching and Record Retrieval. For this second use case, we evaluate two different client queries. First, for *MT-Material*, we only vary the production material of a workpiece, i.e., the client wants to produce an identical workpiece with another material. Second, for *MT-Diameter*, we request parameters where the same workpiece should be produced with a different milling cutter. To this end, we iterate over the input defining the milling cutter's diameter.

We detail the processing times for both metrics in Figure 10 (incl. times of our PPE design variant, which we specifically elaborate on in Section 8.2). Even though this use case does not impose any hard time constraints, concluding the client query after less than 1 min is a fitting result. Given that we only vary a single input parameter for each metric, the resulting candidate set is tiny compared to the evaluated injection molding metrics. Thus, the (large) Bloom filter and the key retrieval dominate the runtime of these client requests.

Here, we showed the further applicability of BPE on a second real-world use-case, i.e., BPE also handles simple metrics efficiently.

8 PRACTICAL PRIVACY IMPROVEMENTS

In this section, we discuss the privacy provided by our BPE design. Based on our findings in Section 8.1, we then propose two variants that further improve provider and client privacy (**G1** and **G2**).

8.1 Security Discussion

Given that we only operate with well-known and authenticated entities, i.e., registered companies under known jurisdictions (cf. Section 6.1), we focus on a semi-honest setting, i.e., we assume that all entities follow the specified protocol as deviations are prosecuted. Consequentially, we do not have to rely on more complex building blocks, such as secret sharing [74], for our parameter exchange as suitable (semi-trusted) operators are available in industrial settings.

The security goals of the design are twofold. First, provider privacy (**G1**), i.e., protecting uploaded data records, and second, client privacy (**G2**), i.e., hiding all queries, need to be considered.

Key Server. As all sensitive key retrievals are handled via OT, the key server cannot harm provider and client privacy. While colluding with data providers does not harm the client privacy, colluding with the client could harm the provider privacy if ciphertexts are retrieved illegitimately. A collusion of the operators of key and storage servers is the main threat in our design as it can result in plaintext access, violating both provider and client privacy. However, we envision a storage server operator with a significant

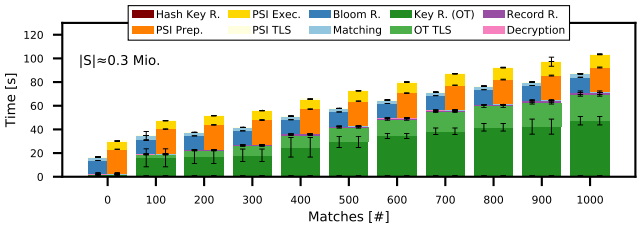


Figure 11: In settings where our more secure PSI variant PPE is applicable, it achieves comparable performance to BPE.

reputation (cf. Section 6.1) not willing to risk legal punishment. Thus, we expect that this kind of misbehavior is unlikely.

Storage Server. As discussed in Section 5.3, BPE does not hide the indices of client-requested records. Thus, it allows the storage server to partially reconstruct the client's candidate set, slightly violating **G2**. However, inferring the similarity metric is infeasible as neither the metric's input nor the unmatched indices are known to the server. Moreover, handing out the Bloom filter is a problematic step for provider privacy as the client obtains an encoded representation of the available records. While Bloom filters do not allow the retrieval of all stored items directly, brute-force attacks could provide rough estimates, especially with a low FP rate. We tolerate this slight violation of **G1** to enable the local computation of client metrics even in huge settings (up to billions of elements).

By using a hash key for indices computation, which is unknown to the storage server, we increase both provider and client privacy as the storage server cannot compute any index itself even if it is aware of suitable input parameters. We achieve provider privacy as requested records are only shared without their origins against a payment. Similarly, providers are unaware of who paid for a record, satisfying client privacy. In the case of unintended data leaks, we protect records by utilizing different encryption keys to render brute-force attacks infeasible. Other misbehavior can by retraced through access logs at both key and storage servers. We leave an analysis of the implications of joining these logs for future work.

To summarize, the security foundations of BPE build on the separation of key material and ciphertexts. To ensure client privacy, the storage server may not collude with data providers either. To further improve provider and client privacy at the expense of additional overhead, we take a look at possible design variants in the following.

8.2 Design Variants

To achieve adaptability (**G5**), we propose two variants to improve the privacy for settings where BPE is currently insufficient. We evaluate a variant with PSIs, which improves provider privacy. We further design a fully OT-based concept for enhanced client privacy.

8.2.1 PPE: A PSI-based Approach. To prevent potential information leaks through the Bloom filter, i.e., a list indicating all available indicies shared with every client, we also propose a design variant that replaces the Bloom filter-based matching with a PSI (cf. Appendix C.1). By using PSIs, clients only learn the matching elements and cannot brute force the complete set of all available records.

However, due to the limited supported size of the candidate set S in PPE, we favor BPE over PPE despite its weaker provider privacy. In settings, with specific privacy needs and comparable small candidate sets, PPE can be a suitable, more secure alternative.

PPE Performance. We repeated the setting from Section 7.3 with a PPE-feasible sized candidate set S through a relative offset of 0.3 % with only 0.3 Mio. elements (compared to 29 Mio. elements). Figure 11 visualizes the performance results and compares them to BPE. By design, only the matching phases differ and in this setting, the PSI introduces slight overhead when compared to BPE. In Appendix C.2, we detail the linear influences of set size, latency, and bandwidth on the PSI performance in a building block analysis.

We also evaluate *IM-2%* in Appendix C.3 showing a larger PSI overhead. Moreover, we measured our second use case (cf. Section 7.5.1) with PPE as its small candidate sets are well-suited. Figure 10 shows a comparison of BPE and PPE for both metrics. In this use case, the download of the large Bloom filter even outweighs the PSI execution time, such that PPE results in a shorter runtime.

PPE can provide improved provider privacy. However, small candidate sets are essential for the use of PSIs. Hence, in general, it is infeasible for injection molding with potentially large candidate sets.

8.2.2 OPE: Fully OT-Powered Approach. Given that the storage server learns the identifiers of retrieved records, client privacy is impaired. To mitigate this effect, the record retrieval could be realized over OTs as well (similar to the key retrieval). The resulting approach, OPE, is conceptually similar to work by Dahlmanns et al. [21]: First, the matching is computed via a PSI (cf. Section 8.2.1) and then both keys and ciphertexts (records) are retrieved via OTs.

However, relying on OTs for the data retrieval introduces significant limitations. LibOTe can only transmit 128 Bit per OT because OTs are mainly designed for the transmission of key material and not the payload itself [2]. However, our ciphertexts are significantly larger, ultimately depending on the use case. Accordingly, t subsequent OTs are needed to retrieve a single ciphertext, which increases the overhead by factor t. More importantly, the OT set size defines the number of supported indices. Consequentially, only a low number of records can be handled by the exchange platform.

These limitations highlight that OPE is only applicable to small scenarios with strong privacy needs. We expect that it is not applicable to most use cases and thus refrain from further evaluation.

9 RELATED WORK

Next, we present related work dealing with privacy-preserving information retrieval and discuss to which extent they are applicable to our scenario. In Table 1, we give an overview of our findings.

Private information retrieval (PIR) [17] protocols deal with privacy-preserving data retrieval from a database. However, PIR protocols only consider the client's privacy, i.e., the query is hidden from the database server, while the server's privacy (**G1**) is not protected. Accordingly, this class of protocols [17, 49] is not applicable to our scenario, as the client is not allowed to learn anything beyond the matching records. *Oblivious transfer (OT)* [60], which is used as a building block of our design, represents symmetric PIR. While it can provide a high level of privacy, OTs alone are not feasible for transmitting large amounts of data, as explained in Section 8.2.2.

Other primitives for secure computations, such as *secure multi-party computation (SMC)* [89] and *homomorphic encryption (HE)* [31], can be used for privacy-preserving information retrieval as well [95]. However, SMC comes with high overhead (**G4**) and does not reach the efficiency of purpose-driven protocols for private information

Table 1: A classification of related work and their properties.

Approach	Client Privacy	Server Privacy	Feasibility	Trust Assumptions
PIR [17]	●	○	◕	●
RKS [21]	●	●	◑	●
SSE [77]	●	○	●	○
PKSE [8]	●	○	◕	◔
PPSSI [24]	◕	◑	◕	◑
PDBQ [9]	●	◑	◑	◑
BPE / PPE	◕	◕/●	◕/◑	◕

retrieval [24]. HE approaches that mimic such protocols suffer from the same inefficiency [50]. In addition, supporting arbitrary similarity metrics (**G3**) with an HE scheme is infeasible as it either offers only a restricted set of operations or becomes overly complex [28].

The *privacy-preserving remote knowledge system (RKS)* [21] tackles the feasibility of data retrievals via OT. A PSI determines matching elements, such that only matched elements induce an expensive OT. We base PPE on this approach. However, limitations of the PSI restrict the size of the candidate set, as discussed in Section 8.2.

Both *symmetric searchable encryption (SSE)* [77] and *public-key searchable encryption (PKSE)* [8] allow the delegation of a search operation to an untrusted third party, e.g., a cloud service. These approaches encrypt data and search queries. The third party returns matched elements to the client without learning the plaintexts. Applied to our scenario, data providers could upload their (encrypted) data. Then, clients could send a search query. However, both approaches assume that the party delegating the search, i.e., the client, is allowed to freely access *all* stored data without restrictions. Accordingly, they cannot satisfy the required server privacy (**G1**).

Privacy-preserving sharing of sensitive information (PPSSI) [24] considers related design goals, i.e., demanding both client and server privacy (**G1** and **G2**). This approach introduces a semi-trusted third party, called isolated box (IB), that must be non-colluding with client and server. It cannot access plaintext information on its own. The data represents database records with multiple attributes that allow the client to pose disjunctive queries over multiple attributes. However, conjunctive queries are not supported. Disjunctive queries are not useful in our scenario as all input parameters have to match the client's candidate. Additionally, PPSSI only considers a single data source (the server), while we have to support multiple data providers. The encryption process, which is offloaded to the IB, requires knowledge on how many records with a certain attribute-value pair exist. Accordingly, the encryption cannot independently be outsourced from the server to the data providers (**G3**). Therefore, adapting this approach to our scenario is far from trivial.

The approach of *private database queries using SWHE (PDBQ)* [9] extents the PPSSI solution by conjunctive queries. However, it also assumes that the data is provided by one server, which is actively involved in the data exchange. Moreover, PDBQ requires the computation of an inverted index by the server entity. Due to the fact that this computation needs plaintext access to the stored data, an additional semi-trusted storage server cannot perform it, i.e., computing the inverted index requires information on how many records with a certain attribute-value pair exist. Hence, the challenge of adapting it to multiple independent data providers, as required by our scenario, remains. Additionally, we expect that PDBQ does not scale to our scenario (**G4**) as it was only evaluated with up to 5 attribute-value pairs and our scenario calls for significantly more query parameters, each adding a random linear combination.

While many diverse applications in the area of private information retrieval exist, they are inapplicable to our scenario as existing work either results in reduced server privacy or requires significant adoption effort for our scenario. We bridge this gap by proposing BPE and PPE, two variants of privacy-preserving exchange platforms.

10 CONCLUSION

In this paper, we introduced a new design for the industrial setting to enable the privacy-preserving exchange of production process parameters, which is expected to significantly improve productivity and reduce costs alike. BPE is based on existing (cryptographic) building blocks, i.e., Bloom filters and OTs, and respects the privacy needs of both clients and data providers. For scenarios with fewer records and stronger privacy needs, we also propose a PSI-based variant called PPE. We showcase the applicability and relevance of our approaches based on two real-world use cases: (i) a process parameter retrieval for injection molding, which allows companies to integrate external knowledge into their transfer learning, and (ii) an exchange for machine tool parameters which enables companies to improve their machine settings even for individual workpieces.

We conducted an in-depth analysis of all aspects of our design. Our evaluation shows that BPE scales well to today's real-world needs (both in terms of privacy and processing) and is easily deployable as no specific hardware is required. Especially in scenarios where the exchanged production data is valuable and impactful for retrieving companies, the processing times for privacy preservation of our design are generally acceptable. Thus, settings with sensitive metrics, such as injection molding, are prime candidates for BPE.

Future work should look into concepts that rate the value of exchanged process data and, thereby, enable new business models for data-providing companies. Similarly, research could look into ways to transform the platform into a subscription model to ease the billing process. Measures to improve the auditability of transfered (and queried) records could be researched to address potential accountability needs of companies. By releasing BPE and PPE as open-source, we hope to contribute to realize newly envisioned industrial collaborations: We offer a ready-to-use privacy-preserving architecture to address widely-established privacy concerns.

ACKNOWLEDGMENTS

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

REFERENCES

[1] Alejandro Alvarado Iniesta, Jorge L García Alcaraz, and ManuelIván Rodríguez Borbón. 2013. Optimization of injection molding process parameters by a hybrid of artificial neural network and artificial bee colony algorithm. *Revista Facultad de Ingeniería Universidad de Antioquia* 67 (2013), 43–51.

[2] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. 2013. More Efficient Oblivious Transfer and Extensions for Faster Secure Computation. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS '13)*. ACM, 535–548. https://doi.org/10.1145/2508859.2516738

[3] Donald Beaver. 1996. Correlated Pseudorandomness and the Complexity of Private Computations. In *Proceedings of the 28th Annual ACM Symposium on*

Theory of Computing (STOC '96). ACM, 479–488. https://doi.org/10.1145/237814.237996

[4] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. 2011. Cython: The Best of Both Worlds. *Computing in Science & Engineering* 13, 2 (2011), 31–39. https://doi.org/10.1109/MCSE.2010.118

[5] David Belson. 2017. *State of the Internet Report — Q1 2017 report.* Technical Report. Akamai Technologies.

[6] R. Joseph Bensingh, Rajendra Machavaram, Sadayan Rajendra Boopathy, and Chidambaram Jebaraj. 2019. Injection molding process optimization of a bi-aspheric lens using hybrid artificial neural networks (ANNs) and particle swarm optimization (PSO). *Measurement* 134 (2019), 359–374. https://doi.org/10.1016/j.measurement.2018.10.066

[7] Burton H. Bloom. 1970. Space/Time Trade-Offs in Hash Coding with Allowable Errors. *Commun. ACM* 13, 7 (1970), 422–426. https://doi.org/10.1145/362686.362692

[8] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. 2004. Public Key Encryption with Keyword Search. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '04)*. Springer, 506–522. https://doi.org/10.1007/978-3-540-24676-3_30

[9] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and David J. Wu. 2013. Private Database Queries Using Somewhat Homomorphic Encryption. In *Proceedings of the 11th International Conference on Applied Cryptography and Network Security (ACNS '13)*. Springer, 102–118. https://doi.org/10.1007/978-3-642-38980-1_7

[10] Rainer Bourdon, Andreas Hellmann, Jan-Bernd Schreckenberg, and Ralf Schwegmann. 2010. Sind Wechselwirkungen simulierbar? Prozessoptimierung beim Spritzgießen mit statistischer Versuchsplanung. *Kunststoffe* 10 (2010), 526.

[11] Rainer Bourdon, Andreas Hellmann, Jan-Bernd Schreckenberg, and Ralf Schwegmann. 2012. Standardized optimization of process and quality by DOE methods — a short manual for injection molding in practice. *Journal of Plastics Technology* 8, 5 (2012), 525–549.

[12] Christian Brecher, Marian Wiesch, and Frederik Wellmann. 2019. Productivity Increase – Model-based optimisation of NC-controlled milling processes to reduce machining time and improve process quality. *IFAC-PapersOnLine* 52, 13 (2019), 1803–1807. https://doi.org/10.1016/j.ifacol.2019.11.463

[13] Daniele Catteddu. 2010. Cloud Computing: Benefits, Risks and Recommendations for Information Security. In *Proceedings of the Iberic Web Application Security Conference (IBWAS '10)*. Springer. https://doi.org/10.1007/978-3-642-16120-9_9

[14] Ceresana. 2016. *Plastic Injection Market Report.* Technical Report. Ceresana.

[15] Wen-Chin Chen, Min-Wen Wang, Chen-Tai Chen, and Gong-Loung Fu. 2009. An integrated parameter optimization system for MISO plastic injection molding. *The International Journal of Advanced Manufacturing Technology* 44, 5–6 (2009), 501–511. https://doi.org/10.1007/s00170-008-1843-4

[16] Sujit Rokka Chhetri, Sina Faezi, and Mohammad Abdullah Al Faruque. 2017. Fix the Leak! An Information Leakage Aware Secured Cyber-Physical Manufacturing System. In *Design, Automation & Test in Europe Conference & Exhibition (DATE '17)*. IEEE, 1408–1413. https://doi.org/10.23919/DATE.2017.7927213

[17] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. 1995. Private Information Retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science (FOCS '95)*. IEEE, 41–50. https://doi.org/10.1109/SFCS.1995.492461

[18] Cheng-Kang Chu and Wen-Guey Tzeng. 2005. Efficient k-Out-of-n Oblivious Transfer Schemes with Adaptive and Non-adaptive Queries. In *Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC '05)*. Springer, 172–183. https://doi.org/10.1007/978-3-540-30580-4_12

[19] Cisco. 2020. *Cisco Annual Internet Report (2018–2023) White Paper.* White Paper. Cisco.

[20] Li Da Xu, Wu He, and Shancang Li. 2014. Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics* 10, 4 (2014), 2233–2243. https://doi.org/10.1109/TII.2014.2300753

[21] Markus Dahlmanns, Chris Dax, Roman Matzutt, Jan Pennekamp, Jens Hiller, and Klaus Wehrle. 2019. Privacy-Preserving Remote Knowledge System. In *Proceedings of the 2019 IEEE 27th International Conference on Network Protocols (ICNP '19)*. IEEE. https://doi.org/10.1109/ICNP.2019.8888121

[22] Paolo D'Arco, María Isabel González Vasco, Angel L. Pérez del Pozo, and Claudio Soriente. 2012. Size-Hiding in Private Set Intersection: Existential Results and Constructions. In *Proceedings of the 5th International Conference on Cryptology in Africa (AFRICACRYPT '12)*. Springer, 378–394. https://doi.org/10.1007/978-3-642-31410-0_23

[23] Satyaki Ghosh Dastidar and Rakesh Nagi. 2005. Scheduling injection molding operations with multiple resource constraints and sequence dependent setup times and costs. *Computers & Operations Research* 32, 11 (2005), 2987–3005. https://doi.org/10.1016/j.cor.2004.04.012

[24] Emiliano De Cristofaro, Yanbin Lu, and Gene Tsudik. 2010. Privacy-preserving Sharing of Sensitive Information. Cryptology ePrint Archive 2010/471.

[25] Emiliano De Cristofaro and Gene Tsudik. 2010. Practical Private Set Intersection Protocols With Linear Complexity. In *Proceedings of the 14th International Conference on Financial Cryptography and Data Security (FC '10)*. Springer, 143–159. https://doi.org/10.1007/978-3-642-14577-3_13

[26] Berend Denkena, Marc-André Dittrich, and Florian Uhlich. 2016. Self-optimizing Cutting Process Using Learning Process Models. *Procedia Technology* 26 (2016), 221–226. https://doi.org/10.1016/j.protcy.2016.08.030

[27] Tim Dierks and Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.2. IETF RFC 5246.

[28] Wenxiu Ding, Zheng Yan, and Robert H. Deng. 2017. Encrypted data processing with Homomorphic Re-Encryption. *Information Sciences* 409–410 (2017), 35–55. https://doi.org/10.1016/j.ins.2017.05.004

[29] Shimon Even, Oded Goldreich, and Abraham Lempel. 1985. A Randomized Protocol for Signing Contracts. *Commun. ACM* 28, 6 (1985), 637–647. https://doi.org/10.1145/3812.3818

[30] Huang Gao, Yun Zhang, Xundao Zhou, and Dequn Li. 2018. Intelligent Methods for the Process Parameter Determination of Plastic Injection Molding. *Frontiers of Mechanical Engineering* 13, 1 (2018), 85–95. https://doi.org/10.1007/s11465-018-0491-0

[31] Craig Gentry. 2009. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09)*. ACM, 169–178. https://doi.org/10.1145/1536414.1536440

[32] René Glebke, Johannes Krude, Ike Kunze, Jan Rüth, Felix Senger, and Klaus Wehrle. 2019. Towards Executing Computer Vision Functionality on Programmable Network Devices. In *Proceedings of the 1st ACM CoNEXT Workshop on Emerging in-Network Computing Paradigms (ENCP '19)*. ACM, 15–20. https://doi.org/10.1145/3359993.3366646

[33] Lars Gleim, Jan Pennekamp, Martin Liebenberg, Melanie Buchsbaum, Philipp Niemietz, Simon Knape, Alexander Epple, Simon Storms, Daniel Trauth, Thomas Bergs, Christian Brecher, Stefan Decker, Gerhard Lakemeyer, and Klaus Wehrle. 2020. FactDAG: Formalizing Data Interoperability in an Internet of Production. *IEEE Internet of Things Journal* 7, 4 (2020), 3243–3253. https://doi.org/10.1109/JIOT.2020.2966402

[34] Fatma Hentati, Ismail Hadriche, Neila Masmoudi, and Chedly Bradai. 2019. Optimization of the injection molding process for the PC/ABS parts by integrating Taguchi approach and CAE simulation. *The International Journal of Advanced Manufacturing Technology* 104, 9–12 (2019), 4353–4363. https://doi.org/10.1007/s00170-019-04283-z

[35] Tsuyoshi Hombashi. 2016. Tcconfig. https://github.com/thombashi/tcconfig.

[36] Christian Hopmann, Pascal Bibow, Thomas Kosthorst, and Yannik Lockner. 2020. Process setup in injection moulding by Human-Machine-Interfaces and AI. In *Proceedings of the 30th International Colloquium Plastics Technology.*

[37] Christian Hopmann and Julian Heinisch. 2018. Injection Molding Setup by Means of Machine Learning Based on Simulation and Experimental Data. In *Proceedings of the 76th SPE Annual Technical Conference and Tradeshow (ANTEC '18)*. Society of Plastics Engineers, 269–274.

[38] Christian Hopmann, Sabina Jeschke, Tobias Meisen, Thomas Thiele, Hasan Tercan, Martin Liebenberg, Julian Heinisch, and Matthias Theunissen. 2019. Combined learning processes for injection moulding based on simulation and experimental data. In *Proceedings of the 33rd Polymer Processing Society Annual Meeting (PPS '17)*, Vol. 2139. AIP, 152–156. https://doi.org/10.1063/1.5121656

[39] Van Jacobson, Craig Leres, and Steven McCanne. 1988. TCPDUMP/LIBPCAP public repository. https://www.tcpdump.org/.

[40] Sabina Jeschke, Christian Brecher, Tobias Meisen, Denis Özdemir, and Tim Eschert. 2017. *Industrial Internet of Things and Cyber Manufacturing Systems.* Springer. https://doi.org/10.1007/978-3-319-42559-7_1

[41] Miran Kim and Kristin Lauter. 2015. Private genome analysis through homomorphic encryption. *BMC Medical Informatics and Decision Making* 15 (Suppl 5) (2015). https://doi.org/10.1186/1472-6947-15-S5-S3

[42] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. 2016. Efficient Batched Oblivious PRF with Applications to Private Set Intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, 818–829. https://doi.org/10.1145/2976749.2978381

[43] Davorin Kramar and Djordje Cica. 2017. Predictive model and optimization of processing parameters for plastic injection moulding. *Materials and Technology* 51, 4 (2017), 597–602. https://doi.org/10.17222/mit.2016.129

[44] Andrew Kusiak. 2017. Smart manufacturing must embrace big data. *Nature* 544, 7648 (2017), 23–25. https://doi.org/10.1038/544023a

[45] Zhi Li, Layne Liu, Ali Vatankhah Barenji, and Waiming Wang. 2018. Cloud-based Manufacturing Blockchain: Secure Knowledge Sharing for Injection Mould Redesign. *Procedia CIRP* 72, 1 (2018), 961–966. https://doi.org/10.1016/j.procir.2018.03.004

[46] Hrelja Marko, Klancnik Simon, Irgolic Tomaz, Paulic Matej, Balic Joze, and Brezocnik Miran. 2014. Turning Parameters Optimization Using Particle Swarm Optimization. *Procedia Engineering* 69 (2014), 670–677. https://doi.org/10.1016/j.proeng.2014.03.041

[47] Mohammad Saleh Meiabadi, Abbas Vafaeesefat, and Fatemeh Sharifi. 2013. Optimization of plastic injection molding process by combination of artificial neural network and genetic algorithm. *Journal of Optimization in Industrial Engineering* 6, 13 (2013), 49–54.

[48] Richard Meyes, Hasan Tercan, Thomas Thiele, Alexander Krämer, Julian Heinisch, Martin Liebenberg, Gerhard Hirt, Christian Hopmann, Gerhard Lakemeyer, Tobias Meisen, and Sabina Jeschke. 2018. Interdisciplinary Data Driven Production Process Analysis for the Internet of Production. *Procedia Manufacturing* 26 (2018), 1065–1076. https://doi.org/10.1016/j.promfg.2018.07.143

[49] Hamid Mozaffari and Amir Houmansadr. 2020. Heterogeneous Private Information Retrieval. In *Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS '20)*. Internet Society.

[50] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can Homomorphic Encryption Be Practical?. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop (CCSW '11)*. ACM, 113–124. https://doi.org/10.1145/2046660.2046682

[51] Moni Naor, Benny Pinkas, and Benny Pinkas. 2001. Efficient Oblivious Transfer Protocols. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '01)*. SIAM, 448–457.

[52] Tim A. Osswald, Lih-Sheng Turng, and Paul J. Gramann. 2007. *Injection Molding Handbook* (2nd ed.). Carl Hanser.

[53] Sinno Jialin Pan and Qiang Yang. 2009. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2009), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

[54] Jan Pennekamp, Markus Dahlmanns, Lars Gleim, Stefan Decker, and Klaus Wehrle. 2019. Security Considerations for Collaborations in an Industrial IoT-based Lab of Labs. In *Proceedings of the 3rd IEEE Global Conference on Internet of Things (GCIoT '19)*. IEEE. https://doi.org/10.1109/GCIoT47977.2019.9058413

[55] Jan Pennekamp, René Glebke, Martin Henze, Tobias Meisen, Christoph Quix, Rihan Hai, Lars Gleim, Philipp Niemietz, Maximilian Rudack, Simon Knape, Alexander Epple, Daniel Trauth, Uwe Vroomen, Thomas Bergs, Christian Brecher, Andreas Bührig-Polaczek, Matthias Jarke, and Klaus Wehrle. 2019. Towards an Infrastructure Enabling the Internet of Production. In *Proceedings of the 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS '19)*. IEEE, 31–37. https://doi.org/10.1109/ICPHYS.2019.8780276

[56] Jan Pennekamp, Martin Henze, Simo Schmidt, Philipp Niemietz, Marcel Fey, Daniel Trauth, Thomas Bergs, Christian Brecher, and Klaus Wehrle. 2019. Dataflow Challenges in an *Internet* of Production: A Security & Privacy Perspective. In *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy (CPS-SPC '19)*. ACM, 27–38. https://doi.org/10.1145/3338499.3357357

[57] Benny Pinkas, Thomas Schneider, and Michael Zohner. 2014. Faster Private Set Intersection Based on OT Extension. In *Proceedings of the 23rd USENIX Conference on Security Symposium (SEC '14)*. USENIX Association, 797–812.

[58] PlasticsEurope. 2019. *Geschäftsbericht 2018*. Technical Report. PlasticsEurope Deutschland e.V.

[59] Sinha Prashant. 2016. pybloomfiltermmap3. https://github.com/prashnts/pybloomfiltermmap3.

[60] Michael O. Rabin. 2005. How To Exchange Secrets with Oblivious Transfer. Cryptology ePrint Archive 2005/187.

[61] Fadillah Ramadhan and T. M. A. Ari Samadhi. 2016. Inter-Organizational Trust and Knowledge Sharing Model Between Manufacturer and Supplier in the Automotive Industry. In *Proceedings of the 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM '16)*. IEEE, 856–860. https://doi.org/10.1109/IEEM.2016.7797998

[62] Shan Ren, Yingfeng Zhang, Yang Liu, Tomohiko Sakao, Donald Huisingh, and Cecilia MVB Almeida. 2019. A comprehensive review of big data analytics throughout product lifecycle to support sustainable smart manufacturing: A framework, challenges and future research directions. *Journal of Cleaner Production* 210 (2019), 1343–1365. https://doi.org/10.1016/j.jclepro.2018.11.025

[63] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. IETF RFC 8446.

[64] Peter Rindal. 2016. libOTe: an efficient, portable, and easy to use Oblivious Transfer Library. https://github.com/osu-crypto/libOTe.

[65] Peter Rindal. 2016. libPSI: A Private Set Intersection Library. https://github.com/osu-crypto/libPSI.

[66] Peter Rindal and Mike Rosulek. 2017. Improved Private Set Intersection against Malicious Adversaries. In *Proceedings of the 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '17)*. Springer, 235–259. https://doi.org/10.1007/978-3-319-56620-7_9

[67] Peter Rindal and Mike Rosulek. 2017. Malicious-Secure Private Set Intersection via Dual Execution. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, 1229–1242. https://doi.org/10.1145/3133956.3134044

[68] Armin Ronacher. 2010. Flask. https://palletsprojects.com/p/flask/.

[69] Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. 2015. Security and Privacy Challenges in Industrial Internet of Things. In *Proceedings of the 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC '15)*. ACM. https://doi.org/10.1145/2744769.2747942

[70] Kamil Salikhov, Gustavo Sacomoto, and Gregory Kucherov. 2014. Using cascading Bloom filters to improve the memory usage for de Brujin graphs. *Algorithms for Molecular Biology* 9, 1 (2014). https://doi.org/10.1186/1748-7188-9-2

[71] Salvatore Sanfilippo. 2009. Redis. https://redis.io/.

[72] Christian Schröder. 2016. *The Challenges of Industry 4.0 for Small and Medium-sized Enterprises*. Technical Report. Friedrich-Ebert-Stiftung.

[73] Roholamin Sedighi, Mohammad Saleh Meiabadi, and Mohammadreza Sedighi. 2017. Optimisation of gate location based on weld line in plastic injection moulding using computer-aided engineering, artificial neural network, and genetic algorithm. *International Journal of Automotive and Mechanical Engineering* 14, 3 (2017), 4419–4431. https://doi.org/10.15282/ijame.14.3.2017.3.0350

[74] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (1979), 612–613. https://doi.org/10.1145/359168.359176

[75] F. Shi, Z. L. Lou, Y. Q. Zhang, and J. G. Lu. 2003. Optimisation of Plastic Injection Moulding Process with Soft Computing. *The International Journal of Advanced Manufacturing Technology* 21, 9 (2003), 656–661. https://doi.org/10.1007/s00170-002-1374-3

[76] Ask Solem. 2009. Celery: Distributed Task Queue. http://www.celeryproject.org/.

[77] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. 2000. Practical Techniques For Searches On Encrypted Data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy (SP '00)*. IEEE, 44–55. https://doi.org/10.1109/SECPRI.2000.848445

[78] João Sá Sousa, Cédric Lefebvre, Zhicong Huang, Jean Louis Raisaro, Carlos Aguilar-Melchor, Marc-Olivier Killijian, and Jean-Pierre Hubaux. 2017. Efficient and secure outsourcing of genomic data storage. *BMC Medical Genomics* 10 (Suppl 2) (2017). https://doi.org/10.1186/s12920-017-0275-0

[79] Roberto Spina. 2006. Optimisation of injection moulded parts by using ANN-PSO approach. *Journal of Achievements in Materials and Manufacturing Engineering* 15, 1–2 (2006), 146–152.

[80] SQLite. 2000. SQLite. https://www.sqlite.org/.

[81] Wencheng Sun, Zhiping Cai, Yangyang Li, Fang Liu, Shengqun Fang, and Guoyan Wang. 2018. Security and Privacy in the Medical Internet of Things: A Review. *Security and Communication Networks* 2018 (2018). https://doi.org/10.1155/2018/5978636

[82] Hasan Tercan, Alexandro Guajardo, Julian Heinisch, Thomas Thiele, Christian Hopmann, and Tobias Meisen. 2018. Transfer-Learning: Bridging the Gap between Real and Simulation Data for Machine Learning in Injection Molding. *Procedia CIRP* 72 (2018), 185–190. https://doi.org/10.1016/j.procir.2018.03.087

[83] Kuo-Ming Tsai and Hao-Jhih Luo. 2015. Comparison of injection molding process windows for plastic lens established by artificial neural network and response surface methodology. *The International Journal of Advanced Manufacturing Technology* 77, 9-12 (2015), 1599–1611. https://doi.org/10.1007/s00170-014-6366-6

[84] Kuo-Ming Tsai and Hao-Jhih Luo. 2017. An inverse model for injection molding of optical lens using artificial neural network coupled with genetic algorithm. *Journal of Intelligent Manufacturing* 28, 2 (2017), 473–487. https://doi.org/10.1007/s10845-014-0999-z

[85] VDI Verein Deutscher Ingenieure e.V. 2020. VDI – The Association of German Engineers. https://www.vdi.de/en/home.

[86] VDMA e. V. (Mechanical Engineering Industry Association). 2015. The VDMA – VDMA. https://www.vdma.org/en/.

[87] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big Data* 3, 1 (2016), 9. https://doi.org/10.1186/s40537-016-0043-6

[88] Katinka Wolter and Philipp Reinecke. 2010. Performance and Security Tradeoff. *Proceedings of the 10th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM '10)* 6154 (2010), 135–167. https://doi.org/10.1007/978-3-642-13678-8_4

[89] Andrew C. Yao. 1982. Protocols For Secure Computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS '82)*. IEEE, 160–164. https://doi.org/10.1109/SFCS.1982.38

[90] Prasad K. D. V. Yarlagadda. 2001. Prediction of processing parameters for injection moulding by using a hybrid neural network. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 215, 10 (2001), 1465–1470. https://doi.org/10.1243/0954405011519097

[91] zafaco GmbH. 2020. *Jahresbericht 2018/19*. Technical Report. Breitbandmessung.

[92] Junhong Zhang, Jian Wang, Jiewei Lin, Qian Guo, Kongwu Chen, and Liang Ma. 2016. Multiobjective optimization of injection molding process parameters based on Opt LHD, EBFNN, and MOPSO. *The International Journal of Advanced Manufacturing Technology* 85, 9–12 (2016), 2857–2872. https://doi.org/10.1007/s00170-015-8100-4

[93] Yuchen Zhang, Wenrui Dai, Xiaoqian Jiang, Hongkai Xiong, and Shuang Wang. 2015. FORESEE: Fully Outsourced secuRe gEnome Study basEd on homomorphic Encryption. *BMC Medical Informatics and Decision Making* 15 (Suppl 5) (2015). https://doi.org/10.1186/1472-6947-15-S5-S3

[94] Xu Zheng and Zhipeng Cai. 2020. Privacy-Preserved Data Sharing Towards Multiple Parties in Industrial IoTs. *IEEE Journal on Selected Areas in Communications* 38, 5 (2020), 968–979. https://doi.org/10.1109/JSAC.2020.2980802

[95] Jan Henrik Ziegeldorf, Jan Pennekamp, David Hellmanns, Felix Schwinger, Ike Kunze, Martin Henze, Jens Hiller, Roman Matzutt, and Klaus Wehrle. 2017. BLOOM: BLoom filter based Oblivious Outsourced Matchings. *BMC Medical Genomics* 10 (Suppl 2) (2017). https://doi.org/10.1186/s12920-017-0277-y

A NOTATION DETAILS

In this section, we extend the description of our notation introduced in Section 5.1. First, we elaborate on our implemented rounding function in Section A.1. Second, we present a basic example of a similarity metric used as part of our evaluations in Section A.2.

A.1 Use Case-Specific Rounding

As described in Section 5.1, we apply a use case-specific rounding function $r(x_i) = x_i'$ to the input parameters x_i before computing the index $id_{x'}$. Our rounding approach resembles a binning of related records, i.e., related records are rounded to same value such that their indices are identical. The rounding depends on the absolute value of the input parameter to achieve a granularity adaption because smaller changes in smaller values are expected to have a larger influence than the same absolute change on a larger value. For example, cooling times can easily be as low as 5 s, while common melt temperatures for regularly-used polypropylene polymer are above 200 °C. Here, the absolute difference between the values is far more than a magnitude, demanding a use case-specific rounding. In theory, any rounding could be implemented and used as well.

We demonstrate our used rounding approach based on such input parameters (1.21, 22.22, 333.33). Our defined function r rounds each input parameter x_i to a certain number of digits, here exemplified with 2 for all inputs, starting from the digit with the highest potency. This rounding yields the identifying parameters (1.2, 22.0, 330.0). This example demonstrates that the rounding of x_1 uses a finer granularity than $r(x_3)$ due to its smaller absolute value.

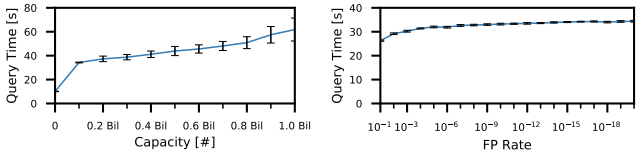
A.2 A Basic Similarity Metric

To illustrate the computation of the candidate set $S = s(q)$, we consider the target record $(1.2, 22.0, 330.0)$, a rounding to two digits, and a metric s that computes the 10 % offset for each identifying parameter. This metric yields 1.1, 1.2, and 1.3 as possible values for the first parameter because the absolute offset is 0.12 such that 1.0 and 1.4 are not covered. For the second parameter, the metric computes 5 possible values and 7 for the third parameter. Hence, the candidate set consists of $3 \cdot 5 \cdot 7 = 105$ candidates in total.

B SUPPLEMENTAL PERFORMANCE RESULTS

We include additional supplemental evaluations to substantiate the made feasibility claims and to justify our parameter choices. We now present the results of theses supporting measurements.

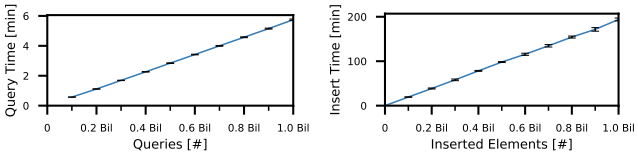
B.1 Bloom Filter



(a) The capacity of the Bloom filter has only a limited impact on the measured query time. **(b) The FP rate only has a negligible influence on the time required to query the Bloom filter.**

Figure 12: The capacity has only a small influence on the query time, while the FP rate has nearly no influence at all.

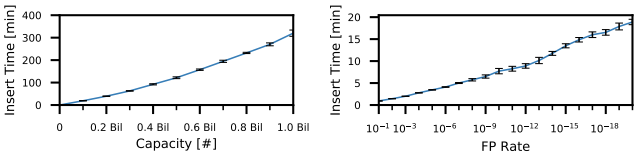
Apart from the Bloom filter size, we also measured the influence of the capacity and configured FP rate on the insertion or query



(a) The query time depends linearly on the number of performed queries. **(b) The insertion time depends linearly on the number of inserted elements.**

Figure 13: Both the number of inserted and queried elements have a linear influence on the corresponding time.

times for a certain number of elements. Furthermore, we measured the influence of the number of queried elements on the query time. If not stated otherwise, we set a FP rate of 2^{20}, a capacity of 100 Mio. elements, inserted as many elements as configured by the capacity and queried a total of 100 Mio. elements. While Figure 12 illustrates the influence of capacity and FP rate on the query time, Figure 13a details the influence of the number of queried elements. Both capacity and FP rate only have a limited influence on the time needed to query 100 Mio. elements. As expected, for a fixed capacity and FP rate, the number of queried elements influences the processing linearly (cf. Figure 13a). By design, querying is embarrassingly parallel as the individual queries are independent of each other.



(a) Even huge sets with a billion elements can be inserted into a Bloom filter in under 6 h. **(b) An exponential decrease of the FP rate yields an approx. linearly increased insertion time.**

Figure 14: By design, a larger capacity and a lower false positive (FP) rate influence the insertion time of a Bloom filter.

The time required for the insertion scales linearly with the number of inserted elements, for a fixed capacity and FP rate (cf. Figure 13a). In Figure 14, we observe that the insertion of millions of elements takes several hours with our experimental setup. However, concerning our design, this time is tolerable because insertions are one-time activities, which are not time-critical. The insertion likely occurs with delay as data providers do not offload their records simultaneously. Moreover, our measurements indicate that the insert time increases approximately linearly with an exponentially decreasing FP rate. Accordingly, very low false positive rates can be configured while maintaining reasonable insertion times.

B.2 Oblivious Transfer

We also conducted additional measurements for our second building block. As detailed in Section 7.2.2, we rely on the semi-honest OT protocol KKRT16 [42] for all measurements, which is the fastest semi-honest OT protocol supported by libOTe [64]. Two parameters influence the OT runtime and memory usage by design: Both the set size (cf. Figure 15) and the number of OT extensions (cf. Figure 16) have a linear correlation. All measured runtime stay below 2 min in these measurements, however, as discussed in Section 7.2.2, network conditions have a major influence on the runtime as well. With decent network conditions, our measurements indicate that even

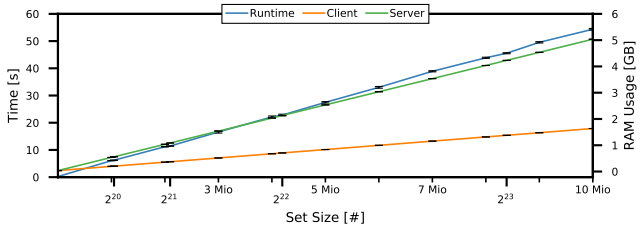


Figure 15: The OT set size linearly influences the runtime and memory usage. We measure 10 OT extensions each.

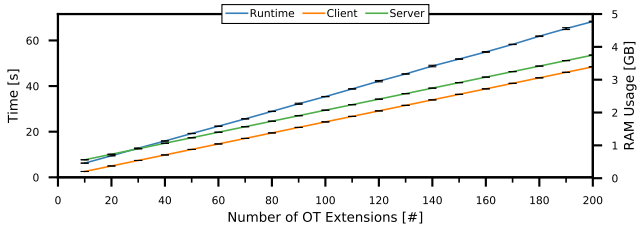


Figure 16: With a set size of 2^{20}, the number of performed OT extensions linearly influences the runtime and RAM usage.

200 key retrievals via OT are feasible for a set size of 2^{20}. These numbers are relevant when offloading millions of records overall and when retrieving tens or even hundreds of keys per client.

B.3 Data Provision

In addition to the building block evaluations, we performed measurements of the data provision phase for each setting. Figure 17 illustrates the influence of the record length. For this measurement, we uploaded 100 random records while varying the number of included parameters m. The key retrieval remains constant as the same encryption keys are retrieved for each measurement. The length of the records does not have a significant influence on the runtime as the key retrieval dominates the data provision.

In Figure 18, we include the data provision phase of our injection molding evaluation (cf. Section 7.4). We ran two evaluations with a varying share of uploaded parameters. For the first measurement, shown by the left-sided bars, we chose to upload all parameters with the same identifier before considering the data belonging to a different geometry. Here, the key retrieval overhead increases with a larger number of uploaded records because 77 records have the same index and therefore need the same encryption key. Therefore, the number of retrieved keys only increases when parameters with distinct identifiers are offloaded. For the second measurement, we selected the records uniform at random for each share. Here, already the first upload (10 %), equaling 462 records, has a high probability of containing one record of each of the 60 groups, such that all keys are required. Thus, the key retrieval times remain nearly constant.

Moreover, the figure shows that the key retrieval dominates the data provision as for our setting with random records in Section 7.3. The entire provision phase takes less than 12 s, even if all records are uploaded at once. Consequentially, the data provision is feasible even in settings where providers upload large amounts of records.

C PPE: PSI-BASED PARAMETER EXCHANGE

We proposed PPE (cf. Section 8.2.1) for settings with metrics that yield small candidate sets and require increased provider privacy.

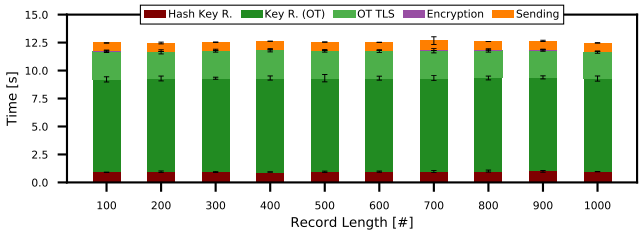


Figure 17: The key retrieval dominates the runtime of the data provision, and the record size has a negligible impact.

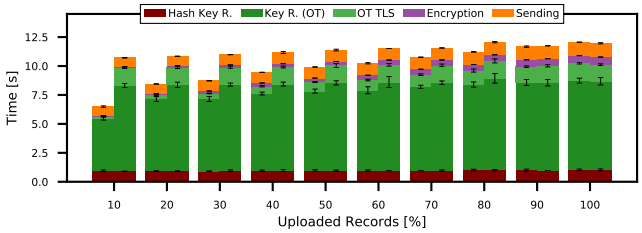


Figure 18: Our real-world injection molding use case also shows that the key retrieval dominates the data provision.

C.1 PPE Protocol Differences

PPE provides improved provider privacy (**G1**) by replacing the Bloom filter-based matching in the matching phase with a PSI (cf. Section 4). As shown in Figure 19, the other parts of the BPE protocol remain unchanged. Due to the PSI, the client cannot gain any knowledge about the server's set except for the matching elements. The client utilizes the computed candidate set S, and the server takes the indices of all stored records as their *sets* for the PSI. Since a PSI further requires indices to perform the intersection on (2^{128} in our case and larger than the chosen OT set size of $K = 2^{20}$), we introduce a third indexing L with $K \subset L \subset H$, and calculate the respective truncation for L using the values inserted in the sets.

Although, in theory, the PSI would support the intersection of sets with a size of 2^{128}, to achieve computational feasibility, the number of elements in the set must be reduced. Notably, in contrast to OT, the input indices in L are not limited by the PSI set size reducing the chance of clients guessing matching indices and further, due to computational effort, preventing clients from performing PSI operations with an extensive number of elements in their candidate set (e.g., to request all records from the server). Even though the size of S depends on the client and its chosen metric, we expect that the server set is unlikely to exceed 100 Mio. elements (cf. Section 7.2.1), and thus, we fix the PSI set size to 2^{20}.

C.2 Private Set Intersection Performance

As for the oblivious transfers (cf. Section 7.2.2), we rely on the semi-honest PSI protocol KKRT16 [42], which is the fastest protocol supported by libPSI [65]. The main influence on the runtime of a PSI protocol is the used PSI set size, which scales linearly with the runtime and memory usage (cf. Figure 20). Due to the linear influence on the runtime, the maximal supportable PSI set size for PPE depends on the available memory at the storage server. Furthermore, the storage server must potentially serve multiple clients at once, i.e., offer a PSI sender instance for each client. Hence, defining the maximum supported PSI set size in accordance to the available memory of the storage server is unreasonable. For the

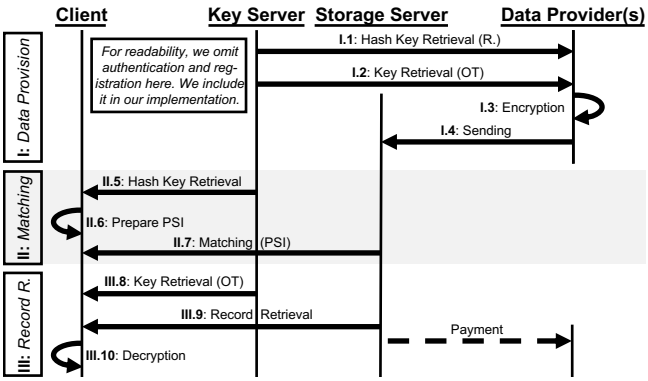


Figure 19: For PPE, we replace the Bloom filter-based matching with a PSI, while the remaining protocol is unchanged.

chosen set size of 2^{20} (cf. Appendix C.1), approximately 0.6 GB are utilized at maximum, so that the server can interact with multiple clients simultaneously. Moreover, the PSI protocol runtime is also influenced by network conditions, i.e., both latency and bandwidth have a major impact (cf. Figure 21). We limited the bandwidth asynchronously with a ratio of 1/10 to mimic common broadband connections. The labels in Figure 21a again refer to the server-client direction. The figure highlights that, especially, a low bandwidth has a major impact on the performance. However, even for a restricted bandwidth with 6 MBit/s, the execution time for a PSI with a set size of 1 Mio. elements stays around 45 min, which is still acceptable as our scenario tolerates the combined matching and record retrieval to take several days. As stated for OTs, PSI protocols also exhibit a trade-off between communication and computation overhead [42]. Accordingly, the used protocol can be adapted to specific needs.

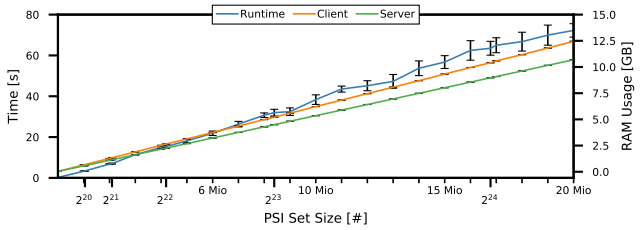
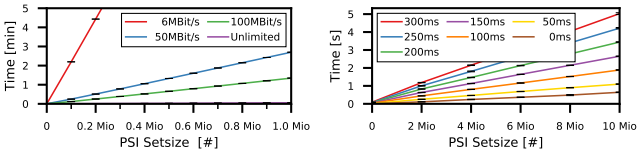


Figure 20: Both runtime and required memory increase approximately linearly with an increasing PSI set size.



(a) A reduced bandwidth affects the large transmissions of PSIs. **(b) Latency also impacts the communication overhead of PSIs.**

Figure 21: Both a decreased bandwidth and an increased latency negatively influence the linear coefficient when considering the PSI set size and the overall processing time.

C.3 PPE Full Application Performance

In addition to the building blocks, we also measured the performance of PPE. Given that the data provision and record retrieval remain unchanged, the same considerations as described in Section 7.3 apply. Figures 11 and 22 compare the matching and record

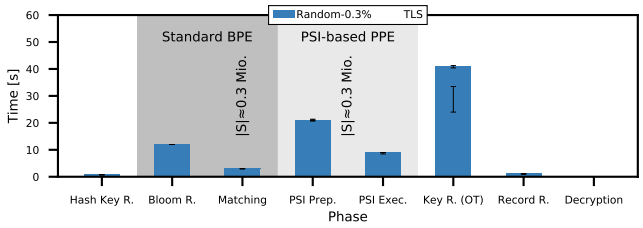


Figure 22: For small candidate sets, both BPE's and PPE's matching phases yield similar runtimes.

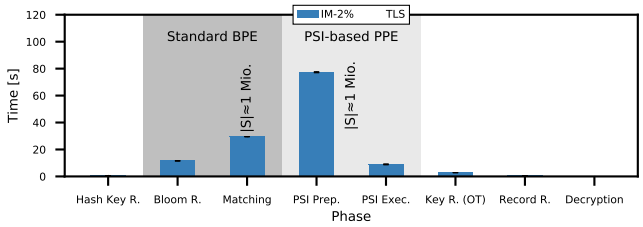


Figure 23: While the PSI execution time stays constant (cf. Figure 22), the preparation time increases with the size of S.

retrieval phases to BPE. The two figures underline that the performance of BPE and PPE is comparable. In the given setting, the main overhead is caused by the preparation of the PSI. This phase is responsible for creating a duplicate free candidate set, which is used as the receiver set for the PSI. However, as shown in Appendix C.2, the PSI execution time increases if larger set sizes are used.

We could only perform an evaluation of PPE with *IM-2%* from Section 7.4 because the utilized similarity metric for *IM-2.5%* yields a candidate set (143 Mio. elements) that is not feasible for PPE as it exceeds our maximally supported set size of 2^{20}. This situation illustrates the main drawback of the PPE design and explains why we favor BPE. While PPE can offer an increased level of provider privacy for privacy cautious providers, it is not applicable in general.

The evaluation of *IM-2%* with the PPE design (cf. Figure 23) shows that the runtime of the PSI preparation phase increases for larger candidate sets, while the PSI execution takes approximately the same time. Moreover, the measurement implies that for larger candidate sets, the PSI-based matching produces significantly more overhead than the Bloom filter-based matching of the BPE variant.

In contrast, our second use case, as described in Section 7.5, results in significantly smaller candidate sets such that PPE is applicable for both our evaluated metrics. Due to the small candidate sets (cf. Figure 10), i.e., $|S| = 11$ for *MT-Material*, and $|S| = 701$ for *MT-Diameter*, the PSI preparation that dominated our first evaluations (cf. Figure 11 and 23) only produces negligible overhead. Instead, the PSI execution accounts for the main runtime in the PSI-based matching. Here, PPE even outperforms our BPE design. The key retrieval times differ due to the number of matched and subsequently retrieved records (10 for *MT-Material* vs. 6 for *MT-Diameter*).

In conclusion, the evaluation of the PPE design variant shows that the variant is not universally applicable due to the restriction of the candidate set by the PSI set size and the overhead produced by the PSI preparation phase. However, for use cases that yield small candidate sets, such as our considered machine tool setting from Section 7.5.1, PPE can even outperform BPE while providing increased provider privacy. Accordingly, the choice of which design variant is best-suited depends on the given use case.