Privacy-Preserving Record Linkage over Big Data Platforms

Elias Dritsas, Maria Trigka, and Phivos Mylonas
University of West Attica, Egaleo 12243, Greece
{idritsas,mtrigka,mylonasf}@uniwa.gr

***Abstract*—Privacy-Preserving Record Linkage (PPRL) integrates sensitive datasets from independent parties without exposing personal identifiers. While secure multiparty computation (SMC) and homomorphic encryption ensure strong privacy, they suffer from high computational costs and poor scalability. Encoding-based methods like Bloom filters are lightweight but face quality issues at scale due to saturation and blocking inefficiencies. This paper proposes a scalable, modular PPRL framework over distributed platforms. It combines Bloom filter encoding, Hamming-based locality-sensitive hashing (LSH), and Dice similarity within a MapReduce pipeline on Hadoop distributed file system (HDFS). The system supports decentralized, end-to-end linkage under semi-honest or covert adversarial models. Experiments on datasets of 100,000–500,000 records show linear scalability, 7.2× speedup over cryptographic baselines, and recall degradation linked to filter saturation. A regression model captures the execution–candidate volume relationship, aiding system tuning. The framework supports high-throughput, regulation-compliant linkage for healthcare, finance, and public sector use.**

***Index Terms*—Privacy-Preserving, Big Data, MapReduce, Bloom Filter, Distributed Systems**

I. Introduction

Record linkage is a fundamental data-integration task that aims to identify semantically equivalent entities across disparate datasets. In modern distributed systems, such as those operated by hospitals, financial institutions, or government agencies, this task becomes increasingly complex due to privacy constraints and the lack of centralized data access. The growing prevalence of privacy regulations, including the General Data Protection Regulation (GDPR), the Health Insurance Portability and Accountability Act (HIPAA), and other regional frameworks, has made PPRL a critical requirement for data interoperability [1].

Conventional solutions to PPRL frequently employ SMC, homomorphic encryption, or trusted third parties to protect sensitive identifiers during the linkage. Although these approaches offer strong theoretical privacy guarantees, they are often computationally prohibitive and impractical for large-scale deployments. In response, encoding-based techniques, particularly Bloom filter transformations, have gained traction for enabling approximate matching under weaker privacy assumptions. These methods reduce the computational overhead but introduce new challenges, such as filter saturation, hash collisions, and inefficiencies in blocking strategies [2], [3].

To address these issues on a scale, PPRL systems must be confronted with algorithmic and architectural challenges. Encoding schemes must strike a balance between recall, privacy, and resistance to frequency-based inferences, particularly in the sparse or saturated filter regimes. Blocking techniques are necessary to reduce candidate volume without compromising sensitivity, even in the presence of heterogeneous or skewed data distributions. Furthermore, the underlying execution infrastructure must ensure scalability, memory efficiency, and minimal privacy leakage during distributed processing [4], [5].

A. Motivation and Contribution

The need for PPRL arises in various real-world contexts such as inter-hospital data exchange, cross-border compliance, and multi-institutional fraud detection, where sensitive data must be integrated across decentralized databases without compromising confidentiality. These scenarios impose strict requirements on efficiency, scalability, and robustness against semi-honest or covert adversaries, often within heterogeneous or regulation-bound infrastructures. Existing solutions frequently struggle to balance formal privacy guarantees with practical performance, particularly when operating at scale or across institutional boundaries.

To address these challenges, this study introduces a scalable and modular framework for PPRL over distributed Big Data platforms. It employs Bloom filter encoding to pseudonymize quasi-identifiers across parties, combined with a Hamming-based LSH scheme for efficient blocking within a MapReduce environment. Similarity evaluation is performed using Dice coefficients directly over the HDFS, enabling high-throughput linkage without centralized coordination.

The system was empirically evaluated across deployment settings and benchmarked against cryptographic and centralized baselines. Results indicate linear scalability, measurable recall degradation tied to Bloom filter saturation, and significant improvements in latency and resource efficiency.

The remainder of this paper is organized as follows: Section II reviews related works, Section III describes the system architecture, Section IV presents the experimental results, and Section V concludes with future research directions.

II. Related Works

PPRL techniques span a range of trade-offs between privacy, scalability, and efficiency. Early solutions based on SMC provided strong cryptographic guarantees but incurred high computational costs. For example, the general-purpose framework in [6] uses secret sharing with constant-round protocols but is not tailored to record linkage. A mainzelliste-based (MainSEL) system [7] addresses PPRL specifically via exhaustive encrypted comparisons without relying on trusted third parties, yet omits blocking and cannot scale to large datasets.

To mitigate computational overhead, approximate methods employing Bloom filters have gained traction. The protocol in [8] combines Bloom filters and Dice similarity for secure two-party linkage, but lacks distributed execution and does not model the impact of filter parameters. A more scalable design is presented in [9], which implements four Bloom filter workflows in Hadoop using MapReduce and Hamming LSH (HLSH) or frequent pattern-based signatures (FPS) blocking. However, the architecture is monolithic and lacks analytic tools for modeling recall degradation or parameter tuning.

Empirical studies further validate PPRL's potential. The blinded evaluation in [10] demonstrates a 99.3% match rate using Bloom filters over real-world health data, though in a static, non-distributed setup. In [11], Bloom filters were used to accelerate structured query language (SQL) queries, illustrating their scalability in large workloads, albeit outside privacy contexts.

Alternative approaches include private set intersection (PSI) and differential privacy (DP). In [12], PSI protocols are combined with Bloom filters and masking for secure linkage under a semi-honest model. However, such methods typically operate pairwise, lack blocking, and exhibit quadratic complexity. DP-based systems like [13] inject noise into Bloom encodings or similarity scores, enhancing privacy but degrading utility, particularly in sparse or high-dimensional data.

In contrast to the aforementioned studies (summarized in Table I), the proposed system unifies Bloom filter encoding, LSH-based blocking, and MapReduce-based similarity comparison into a fully modular, end-to-end PPRL pipeline. It supports scalable, high-throughput linkage within Hadoop and is the first to incorporate analytic models that relate execution time to candidate volume and quantify recall degradation due to Bloom filter saturation. These predictive capabilities provide actionable tuning guidance, effectively bridging the gap between theoretical scalability and practical deployment in privacy-aware Big Data environments.

III. Proposed System

This section presents the design of a scalable PPRL system based on Bloom filter encoding and MapReduce execution over the HDFS (Figure 1). The system consists of four core stages: modeling the linkage problem, secure Bloom-based encoding, blocking, similarity computation using MapReduce, and complexity analysis.

A. Problem Statement and Adversarial Model

Let the two data custodians A and B hold disjoint private datasets $R_A = \{r_1^{(A)}, \ldots, r_n^{(A)}\}$ and $R_B = \{r_1^{(B)}, \ldots, r_m^{(B)}\}$, respectively. Each record consists of k quasi-identifying string

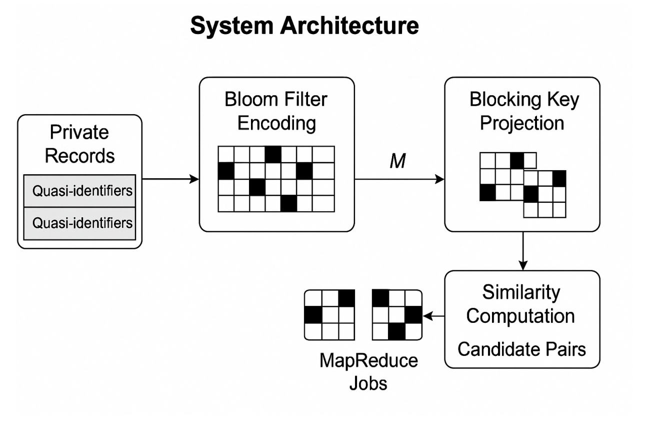

Fig. 1. System architecture of the proposed PPRL framework. Each data custodian encodes local records into Bloom filters, stores them in HDFS, and initiates two MapReduce stages: blocking and pairwise similarity comparison.

attributes, $\mathbf{x} = (x_1, \ldots, x_k)$. The objective was to identify all matching pairs $(r_i^{(A)}, r_j^{(B)})$ such that both refer to the same real-world entity, without exposing raw attribute values between custodians.

We adopted a *semi-honest* adversarial model in which both parties follow the protocol, but may attempt to infer information from intermediate data. The only shared information consists of Bloom filter representations, which are lossy, non-invertible, and obfuscated via multiple hash functions. Under practical parameterization, Bloom filters are computationally difficult to invert and exhibit low information leakage.

From a formal perspective, the leakage function $\mathcal{L}(R)$ is limited to a set of encoded Bloom filter vectors $b(r) \in \{0,1\}^l$, where $r \in R_A \cup R_B$. Assuming uniform hash functions and sufficient sparsity (i.e., expected density $\delta < 0.5$), such representations are indistinguishable across inputs with equal Bloom parameters. Hence, the system satisfies the bounded leakage property, with no reconstruction guarantees available to the adversary under standard cryptographic assumptions.

B. Secure Record Encoding

Each record r is transformed into a Bloom filter vector $b(r) \in \{0,1\}^l$ using a q-gram decomposition and p independent hash functions. For each string attribute x_j, the set of q-grams is defined as

$$Q(x_j) = \{x_j[t : t+q-1] \mid 1 \le t \le |x_j| - q + 1\}. \tag{1}$$

The variable t indicates the position at which each q-gram starts in the input string. By incrementing t, all the overlapping q-grams of the string are generated.

The full set for a record is $Q(r) = \bigcup_{j=1}^{k} Q(x_j)$. Each q-gram $g \in Q(r)$ is hashed using functions $H = \{h_1, \ldots, h_p\}$, setting p positions in a binary vector of length l. The expected number of 1-bits in the resulting Bloom filter is

$$\mathbb{E}[s] = l \cdot \left(1 - \left(1 - \frac{1}{l}\right)^{p \cdot |Q(r)|}\right). \tag{2}$$

TABLE I
SUMMARY OF RELATED WORKS.

Work	Technique	Encoding	Blocking	Distribution	Scalability	Differentiation
[8]	Secure multi-party PPRL	Bloom Filter + Dice	×	×	✓	No MapReduce or modular blocking
[9]	Four MapReduce workflows	Bloom Filter	HLSH / FPS	✓	Partial	No saturation model or end-to-end matching
[7]	Encrypted exhaustive SMC	Encrypted Bloom Filters	×	×	×	Strong privacy, not scalable to large N
[6]	General-purpose SMC	Secret Sharing	×	×	✓	Cryptographic MPC only, no linkage logic
[10]	Blinded PPRL evaluation	Bloom Filter	Manual blocking	×	✓	Operational validation only, no pipeline
[11]	Query acceleration in Relational Database Management System	Bloom Filter Indexes	×	×	×	Not privacy-oriented, no linkage pipeline
[12]	PSI with masked Bloom filters	Bloom Filter + Masking	×	×	×	Formal PSI under semi-honest model, no blocking or distributed support
[13]	DP for PPRL	Noisy Bloom Filters	×	×	×	DP-based protection with accuracy degradation in sparse domains
This Study	MapReduce-based modular PPRL	Bloom Filter	LSH + FPS	✓	✓	Full pipeline, scalable, saturation-aware, regression-based tuning, leakage-bounded

The expected bit density in equation (2) reflects the probabilistic nature of the Bloom filter construction, which is implemented in practice as a distributed MapReduce pipeline: each mapper emits encoded vectors, and reducers validate them and write the output to the HDFS. In practice, records with missing or invalid attribute values were discarded before encoding, and local duplicates may be filtered to improve the blocking efficiency and avoid self-matches.

C. Blocking and Similarity Matching

To avoid a full cross-product comparison $(n \cdot m)$, the system uses a projection-based blocking scheme. Let $\mathcal{M} = \{t_1, \ldots, t_s\}$ be a projection mask that selects s bit positions, shared by both parties prior to execution. For any Bloom filter b, the blocking key is

$$\psi(b) = (b[t_1], b[t_2], \ldots, b[t_s]). \tag{3}$$

Records with identical blocking keys were grouped together. Within each block, candidate pairs (b_i, b_j) were compared using *normalized Hamming similarity*

$$\text{sim}(b_i, b_j) = 1 - \frac{\text{popcount}(b_i \oplus b_j)}{l}. \tag{4}$$

Pairs with sim $\geq \theta$ are retained. This logic is implemented in two MapReduce jobs: the first assigns blocking keys, and the second executes comparison and filtering.

As illustrated in Figure 1, the system executes each phase in a distributed manner, with all intermediate outputs stored in the HDFS between the stages.

Figure 2 illustrates the modular execution flow of the proposed PPRL pipeline over a distributed Hadoop infrastructure. Custodian A and Custodian B independently encode their datasets into Bloom filter representations and store them in the HDFS. The pipeline is composed of two sequential MapReduce jobs: the first performs LSH–based blocking to limit the number of candidate comparisons, and the second executes a Dice similarity evaluation within each candidate block. Intermediate results between stages are stored in the HDFS, enabling scalable, fault-tolerant execution without centralized coordination. This design facilitates efficient, high-throughput linkage across distributed nodes while preserving privacy guarantees.

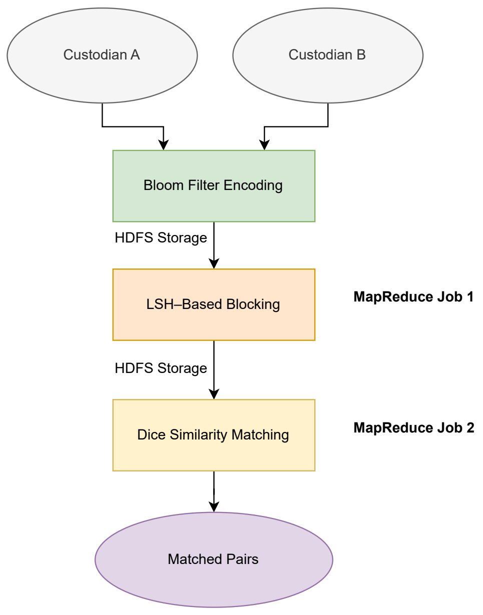

Fig. 2. Modular data flow in the proposed PPRL pipeline.

D. Complexity Analysis

The computational complexity of the proposed system arises from three principal stages: record encoding, blocking, candidate generation, and similarity comparisons. Each phase contributes to runtime and resource utilization, depending on the dataset size and Bloom filter parameters [14].

During encoding, each record underwent q-gram extraction across k string attributes, followed by hashing using h independent functions. This process incurred a cost of $O(qkh)$ per record. For datasets R_A and R_B of sizes n and m, respectively, the total encoding complexity is given by:

$$T_{\text{encode}} = O\left((n+m) \cdot qkh\right). \tag{5}$$

To reduce the number of pairwise comparisons, the system applies a projection-based blocking scheme using a mask $\mathcal{M}$ of s bit positions. Each Bloom filter is mapped to a blocking key in $O(s)$ time. Records with identical keys were grouped into blocks, within which all pairs were compared. Let K denote the number of resulting blocks, and let n_i, m_i be the number of records in block i from R_A and R_B, respectively. The total number of candidate comparisons is then

$$B = \sum_{i=1}^{K} n_i \cdot m_i. \tag{6}$$

For each candidate pair, the system computes Hamming similarity by performing an XOR operation and counting the set bits. This operation has cost $O(l)$, where l is the length of the Bloom filter. Therefore, the total cost of similarity comparisons across all candidate pairs becomes

$$T_{\text{match}} = O(B \cdot l). \tag{7}$$

By combining the encoding and comparison stages, the end-to-end complexity of the system can be expressed as

$$T_{\text{total}} = O\left((n+m) \cdot qkh + B \cdot l\right). \tag{8}$$

This expression highlights the sensitivity of the system to the number of candidate comparisons B. In practice, B is orders of magnitude smaller than $n \cdot m$ due to effective blocking; however, it remains the dominant factor in the runtime. As such, Bloom filter sparsity and blocking selectivity play a crucial role in ensuring scalability.

IV. Experimental Analysis

This section presents the experimental evaluation of the proposed system in terms of its performance, scalability, and linkage quality. All measurements were obtained through controlled deployments in a distributed Hadoop environment using datasets of increasing size and known ground-truth correspondence.

A. Experimental Setup

The proposed system was deployed on a homogeneous Hadoop cluster consisting of five nodes. Each node was equipped with two Intel Xeon E5-2640 processors operating at 2.4 GHz, 32 GB RAM, and 1 TB local SSD storage. The nodes were interconnected through a dedicated 1 Gbps Ethernet switch. Apache Hadoop version 2.7.4 was used, with YARN as the resource manager and HDFS configured with a replication factor of three. All MapReduce jobs were executed using the default fair scheduler, without custom parameter tuning or resource allocation adjustments.

Three dataset configurations were prepared to evaluate the scalability and linkage accuracy of the system. Each configuration involved disjoint datasets from two custodians, denoted R_A and R_B, with a known set of overlapping records exclusively used for evaluation. Configuration E1 contained 100,000 records in R_A and 10,000 in R_B; E2 consisted of 200,000 and 100,000 records respectively; and E3 used 500,000 and 100,000 records. All records were encoded using Bloom filters with the following fixed parameters: q-gram length $q = 2$, filter length $l = 512$, number of hash functions $p = 4$, and the Hamming similarity threshold $\theta = 0.80$. These values were selected following established configurations in the literature [8], [10], and were empirically validated to preserve recall above 90% while maintaining acceptable saturation ratios across all dataset sizes. Each experiment was executed five times to assess variability and ensure statistical significance. The reported values correspond to the arithmetic means.

B. Evaluation

The performance evaluation focused on execution time (T), communication overhead, and linkage quality. The execution time was measured as the wall-clock duration from data ingestion to the output of the matched pairs, including all stages of the pipeline. Communication overhead was quantified as shuffle volume (S), extracted from Hadoop's `reduce_shuffle_bytes` counter and reported in megabytes [15].

The linkage quality was assessed using standard classification metrics. Let true positive (TP) denote the number of correctly matched pairs, false positive (FP) the number of incorrect matches, and false negative (FN) the number of missed true matches [16]. The precision, recall, and F1-score are defined as follows:

$$\text{Precision} = \frac{TP}{TP+FP}, \quad \text{Recall} = \frac{TP}{TP+FN},$$
$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{9}$$

The Bloom filter saturation ratio δ was used to estimate the encoding efficiency, which was computed as the average fraction of bits set per filter:

$$\delta = \frac{1}{|R|} \sum_{r \in R} \frac{\|b(r)\|_1}{l}. \tag{10}$$

Let K be the number of blocking buckets. If each block i contains n_i and m_i records from R_A and R_B, respectively, the total number of candidate comparisons is:

$$B = \sum_{i=1}^{K} n_i \cdot m_i. \tag{11}$$

C. Results and Discussion

Table II reports the performance and linkage quality across three configurations. As the dataset size increases from E1 to E3, the execution time T grows from 158 s to 958 s, while the number of candidate comparisons B increases from 2.7×10^6 to 53.2×10^6. This nonlinear growth in B arises from the increased block density and q-gram redundancy, leading to more intra-block comparisons. The shuffle volume

TABLE II
PERFORMANCE AND LINKAGE METRICS.

ID	$\|R_A\|$	$\|R_B\|$	T (s)	S (MB)	B (M)	Precision (%)	Recall (%)	F1 (%)
E1	100k	10k	158	21.4	2.7	99.12	96.80	97.94
E2	200k	100k	426	84.7	13.4	98.77	95.26	96.98
E3	500k	100k	958	197.3	53.2	98.20	92.11	95.05

S also increases tenfold, reflecting the increased I/O and communication overhead.

Despite this scaling, the system maintained a strong linkage quality. The precision remained above 98% in all settings, and the F1-score only declines slightly from 97.94% to 95.05%. The recall drop in E3 (to 92.11%) is attributed to Bloom filter saturation: as the record size increases, more q-grams are hashed into each filter, increasing bit collisions and reducing the matching selectivity.

These metrics also provide insights into the blocking performance of the LSH scheme. The number of candidate comparisons B is several orders of magnitude smaller than the Cartesian product $|R_A| \cdot |R_B|$, demonstrating that the LSH scheme achieves substantial pruning without compromising the sensitivity. Across all three experimental setups, recall remained consistently high, above 92%, indicating that very few true matches were lost due to blocking-induced false negatives. Furthermore, the recall remains robust even when the ratio $\frac{|R_A|}{|R_B|}$ varies from 10:1 in E1 to 2:1 in E2 and 5:1 in E3, suggesting that the blocking mechanism scales well under moderate class imbalance and skewed distributions.

To assess the runtime complexity theoretically derived in Section III.D, we performed a linear regression between T and B, resulting in:

$$T = 15.17 \cdot B + 163.64 \quad \text{with } R^2 = 0.982. \tag{12}$$

This model was fitted using the three experimental configurations (E1–E3) listed in Table II. A linear least-squares regression yielded the slope and intercept, reflecting the per-comparison cost and fixed system overhead, respectively. The results, especially the R^2 value of 0.982, empirically confirm that execution time is dominated by the $O(B \cdot l)$ term, which is consistent with the theoretical complexity discussed in Section III.D. As shown in Figure 3, the runtime scales linearly with the number of candidate comparisons, with no significant deviation from linearity observed across the tested range. This regression model could enable practitioners to anticipate execution time under different blocking granularities and data volumes, supporting informed system tuning and deployment planning.

However, for larger datasets, reducing the memory pressure or skewed key distribution may introduce bottlenecks. This highlights the need for load-balancing mechanisms on a higher scale. In our experimental runs, we observed a moderate variance in reducer execution times, with the slowest reducer lagging behind the average in configuration E3. This imbalance is attributed to the key skew introduced by the frequent q-gram patterns, which can result in uneven candidate group sizes. Although no explicit mitigation was applied in the current pipeline, preliminary experiments with hash-based partitioning and sampling-based pre-balancing showed a slight improvement in tail reducer time.

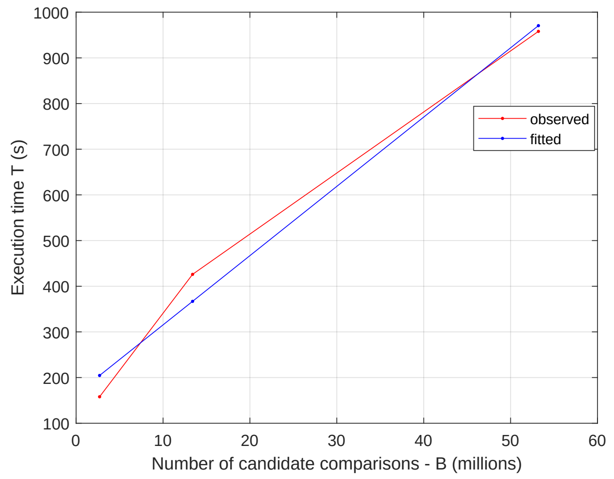

Fig. 3. Execution time T versus number of candidate comparisons B.

Figure 4 shows that the recall degrades as the Bloom filter saturation ratio δ increases. From E1 to E3, δ increased from 0.1875 to 0.2055, correlating with a 4.69% absolute drop in recall. To further quantify this relationship, we fitted a linear model of the form recall $\approx 1 - \alpha \cdot \delta$, yielding $\alpha = 2.543$ with $R^2 = 0.993$. The fitted line closely approximates the observed trend, indicating that a 0.01 increase in δ resulted in a recall loss of approximately 0.025 within the tested interval. Although the model captured the overall degradation pattern well, adding a 95% confidence interval and residual plots could reveal local variations and potential heteroskedasticity, especially in larger or more diverse datasets. These findings confirm that maintaining $\delta < 0.22$ is essential for preserving linkage sensitivity.

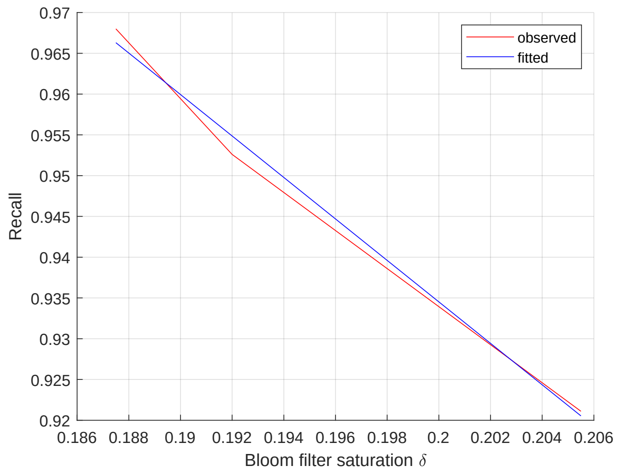

Fig. 4. Recall as a function of Bloom filter saturation ratio δ.

Table III compares the proposed system with two baseline approaches: a centralized record linkage (CRL) and an SMC-based method. CRL, lacking distributed execution, required approximately 3.4× more time than the proposed pipeline. Despite achieving a slightly higher F1-score (98.12%), it offers no scalability and imposes significantly higher CPU utilization (98%), compared to 76% of the proposed method. The SMC-based solution, which provides formal cryptographic guarantees, suffered from 7.2× slower execution, 1.65× higher shuffle volume, and 1.17× higher CPU load. Although it is appropriate for scenarios requiring strong formal privacy (e.g., GDPR level-3 settings in inter-institutional data sharing), SMC remains impractical for large-scale deployment. By contrast, the proposed framework achieves near-interactive performance with pseudonymization-grade privacy and substantial efficiency in both runtime and resource usage. In particular, the Bloom filter encodes the conceals of attribute values through non-invertible hashing with randomized collisions, which mitigates direct re-identification under typical linkage attacks. Although this approach does not provide formal privacy guarantees such as DP or provable leakage bounds, it offers practical protection against passive adversaries without auxiliary background knowledge, which is consistent with the pseudonymization requirements defined in GDPR Recital 26 [17].

In conclusion, the results confirm that the system meets its design objectives: runtime complexity scales linearly with candidate volume, filter saturation impacts recall in a quantifiable manner, and privacy-performance trade-offs remain controllable under realistic configurations. Although the tested dataset configurations were carefully designed to reflect practical deployment scenarios with controlled variability in size and overlap, we acknowledge that a broader generalization of the findings requires further experimentation.

TABLE III
COMPARISON WITH BASELINE METHODS (100K × 10K)

Method	T (s)	F1 (%)	S (MB)	CPU (%)
Proposed	158.3	97.94	21.4	76
CRL	532.5	98.12	0	98
SMC-based	1147.2	94.65	35.2	89

V. CONCLUSION AND FUTURE WORK

This paper introduced a scalable, privacy-preserving framework for approximate record linkage over distributed Big Data infrastructures. By combining Bloom filter encoding with MapReduce-based blocking and matching on HDFS, the system enables efficient linkage of quasi-identifiers without exposing raw data. Empirical evaluation demonstrated linear runtime scalability, precision above 98.2%, and recall above 92.1%, with a 7.2× speedup over SMC-based methods. A regression model confirmed the theoretical complexity and quantified recall degradation as a function of Bloom filter saturation, offering practical tuning guidance.

Future work will focus on extending the system's robustness under heterogeneous data, skewed distributions, and adversarial perturbations. Emphasis will be placed on modeling blocking-induced false negatives, formally bounding leakage, and integrating DP mechanisms. Further optimization will target reducer load balancing and adaptive blocking. Finally, domain-specific extensions for regulated data-sharing environments and federated registries will be explored.

REFERENCES

[1] A. Gkoulalas-Divanis, D. Vatsalan, D. Karapiperis, and M. Kantarcioglu, "Modern privacy-preserving record linkage techniques: An overview," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4966–4987, 2021.

[2] I. Gamiz, C. Regueiro, O. Lage, E. Jacob, and J. Astorga, "Challenges and future research directions in secure multi-party computation for resource-constrained devices and large-scale computations," *International Journal of Information Security*, vol. 24, no. 1, pp. 1–29, 2025.

[3] W. Yin, L. Yuan, Y. Ren, W. Meng, D. Wang, and Q. W. W. Yin, "Differential cryptanalysis of bloom filters for privacy-preserving record linkage," *IEEE Transactions on Information Forensics and Security*, 2024.

[4] F. Armknecht, Y. Heng, and R. Schnell, "Strengthening privacy-preserving record linkage using diffusion," *Proceedings on Privacy Enhancing Technologies*, vol. 2023, pp. 298–311, 2023.

[5] A. Karakasidis and G. Koloniari, "More sparking soundex-based privacy-preserving record linkage," in *International Symposium on Algorithmic Aspects of Cloud Computing*. Springer, 2022, pp. 73–93.

[6] I. Damgård and Y. Ishai, "Scalable secure multiparty computation," in *Annual International Cryptology Conference*. Springer, 2016, pp. 501–520.

[7] S. Stammler, T. Kussel, P. Schoppmann, F. Stampe, G. Tremper, S. Katzenbeisser, K. Hamacher, and M. Lablans, "Mainzelliste secureepilinker (mainsel): privacy-preserving record linkage using secure multi-party computation," *Bioinformatics*, vol. 38, no. 6, pp. 1657–1668, 2022.

[8] D. Vatsalan and P. Christen, "Scalable privacy-preserving record linkage for multiple databases," in *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, 2014, pp. 1795–1798.

[9] D. Boussis, E. Dritsas, A. Kanavos, S. Sioutas, G. Tzimas, and V. S. Verykios, "Mapreduce implementations for privacy preserving record linkage," in *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, 2018, pp. 1–4.

[10] S. Randall, H. Wichmann, A. Brown, J. Boyd, T. Eitelhuber, A. Merchant, and A. Ferrante, "A blinded evaluation of privacy preserving record linkage with bloom filters," *BMC medical research methodology*, vol. 22, no. 1, p. 22, 2022.

[11] E. Chioti, E. Dritsas, A. Kanavos, X. Liapakis, S. Sioutas, and A. Tsakalidis, "Bloom filters for efficient coupling between tables of a database," in *Engineering Applications of Neural Networks: 18th International Conference, EANN 2017, Athens, Greece, August 25–27, 2017, Proceedings*. Springer, 2017, pp. 596–608.

[12] A. Adir, E. Aharoni, N. Drucker, E. Kushnir, R. Masalha, M. Mirkin, and O. Soceanu, "Privacy-preserving record linkage using local sensitive hash and private set intersection," in *International Conference on Applied Cryptography and Network Security*. Springer, 2022, pp. 398–424.

[13] T. Nóbrega, C. E. S. Pires, and D. C. Nascimento, "Blockchain-based privacy-preserving record linkage: enhancing data privacy in an untrusted environment," *Information Systems*, vol. 102, p. 101826, 2021.

[14] L. Luo, D. Guo, R. T. Ma, O. Rottenstreich, and X. Luo, "Optimizing bloom filter: Challenges, solutions, and comparisons," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1912–1949, 2018.

[15] Y. Guo, J. Rao, D. Cheng, and X. Zhou, "ishuffle: Improving hadoop performance with shuffle-on-write," *IEEE transactions on parallel and distributed systems*, vol. 28, no. 6, pp. 1649–1662, 2016.

[16] G. Naidu, T. Zuva, and E. M. Sibanda, "A review of evaluation metrics in machine learning algorithms," in *Computer science on-line conference*. Springer, 2023, pp. 15–25.

[17] M. Finck and F. Pallas, "They who must not be identified—distinguishing personal from non-personal data under the gdpr," *International Data Privacy Law*, vol. 10, no. 1, pp. 11–36, 2020.