Secure and Private Record Linkage: A Survey of Methods and Applications

1st Narjiss Youyou
Universiteit Antwerpen
Antwerp, Belgium
narjiss.youyou@student.uantwerpen.be

2nd Joanna Kisaakye
Universiteit Antwerpen
Antwerp, Belgium
joanna.kisaakye@uantwerpen.be

3rd Hans De Loof
Universiteit Antwerpen
Antwerp, Belgium
hans.deloof@uantwerpen.be

4th Serge Demeyer
Universiteit Antwerpen
Antwerp, Belgium
serge.demeyer@uantwerpen.be

5th Bernard De Ruyck
Universiteit Antwerpen
Antwerp, Belgium
bpjaderuyck@icloud.com

***Abstract*—Researchers in healthcare face challenges in linking patient data from disparate sources, like the ICareData database. Due to stringent privacy regulations such as the General Data Protection Regulation (GDPR), integrating data for comprehensive analysis without violating the limitations on sharing personally identifiable information is difficult.**

Secure Multi-Party Computation (MPC) emerges as a powerful cryptographic solution, enabling privacy-preserving record linkage to integrate and analyze sensitive information across different sources securely. In this work, we explore the application of Multi-Party Computation for privacy-preserving record linkage within public healthcare. Our contribution is an analysis of the applicable assumptions, models, techniques, and a proof-of-concept prototype applicable to the iCareData use case.

***Index Terms*—component, formatting, style, styling, insert.**

I. Introduction

The increasing volume of data generated across diverse sectors such as healthcare, finance, and government presents great opportunities for comprehensive analysis and the extraction of valuable insights. Within healthcare, linking patient records from various hospitals can provide a more complete understanding of disease progression and treatment outcomes, or connect cancer registry data with treatment records to assess long-term survival. Consequently, record linkage becomes essential, particularly for datasets without common unique identifiers. However, linking records across these services presents a privacy challenge because of the regulations to which the linkage process must adhere.

We take as our use case the iCAREdata database. iCAREdata (Improving Care And Research Electronic Data Trust Antwerp) is a central research database infrastructure established in 2014 at the University of Antwerp for primary and interdisciplinary healthcare [3]. Its objective is to collect, link and integrate data from different out-of-hours (OOH) primary care sources, including General Practice Cooperatives (GPCs), hospital Emergency Departments (EDs) and pharmacies in the Antwerp region. The infrastructure is developed to be a unique, interactive and central clinical research database based on clinical data collected weekly in different OOH primary care settings. The aim is to connect and integrate data from GPCs, EDs of general and academic hospitals and pharmacies involved in OOH care in Antwerp.

iCAREdata's current approach to data linkage relies on the use of a Trusted Third Party (TTP). To ensure that data can be transferred to iCAREdata while respecting the privacy of patients and physicians, a multi-step process is used. Medical and demographic data is encrypted at the source level [3]. This encrypted data is then sent to a TTP, which is also eHealth (a Belgian government institution) in this context [3]. Personal data (such as the patient's social security number and physician identification code) is encoded by the TTP [3]. The TTP then sends all data (encrypted medical data and encoded personal data) to iCAREdata's eHealthBox [3]. Upon receiving the encrypted message, iCAREdata decrypts the medical information, keeping the personal data coded [3]. This process allows data from different individuals to be linked from various sources while complying with privacy regulations [3]. Pseudonymization is used to encode personal data, producing a coded unique identifier or pseudonym for those individuals, thus making it possible to link the data to individuals while protecting their privacy by hiding their true identity [3]. For encoding, only one key is used and the TTP is the sole owner [3].

While a centralized Trusted Third Party (TTP) is a viable solution, it also represents a single point of failure and introduces several risks. One major concern is the concentration of sensitive data, which becomes highly vulnerable in the event of a breach. If the TTP is compromised, it becomes a prime target for attacks, and the sensitive data of all participating parties is at risk of being exposed [5]. This inherent vulnerability underscores the risk of centralization. That is why iCAREdata is currently exploring alternative solutions. This effort is driven by the goal of enabling data analysis across a broader set of data holders, who have legitimate concerns about centralizing identifier processing,

Identify applicable funding agency here. If none, delete this.

while still seeking to protect patient privacy. To mitigate the single-point-of-failure and data-concentration risks inherent in a centralized TTP, iCAREdata is investigating decentralized cryptographic techniques. In particular, Secure Multi-Party Computation (MPC) offers a way for multiple, mutually distrusting parties to collaboratively compute on sensitive data without ever exposing their raw inputs to one another or to any intermediary. Sensitive information can thus be jointly matched and analyzed across sources without compromising confidentiality [10], [7], [17].

This paper focusses on iCAREdata use case, which is structured around the core research questions outlined in Section 2, demonstrating how each linkage strategy addresses specific challenges in privacy-preserving record linkage (PPRL) within the iCAREdata context. We begin by exploring record linkage methods using exact identifiers, followed by techniques that rely solely on quasi-identifiers in the absence of direct identifiers. We then provide an overview of secure multi-party computation (MPC), before examining both exact and approximate (fuzzy) matching techniques implemented within MPC settings. These are analyzed in terms of their security and correctness guarantees. Finally, we present a proof-of-concept prototype developed using the MPyC framework to demonstrate the feasibility of our approach.

The paper is structured as follows: Section 2 introduces the research questions, Section 3 outlines the literature survey we used, Section 4 provides background on the iCAREdata dataset, Section 5 discusses record linkage techniques, Section 6 details secure linkage using MPC using the same methods discussed in Section 5 only now in a secure setting, and finally Section 7 describes the MPyC-based proof-of-concept. Throughout this paper, the terms Privacy Preserving Record Linakage (PPRL) and Secure Record Linkage, will be used interchangeably.

II. Research Questions

We frame the discussion around the following research questions, which explore how privacy-preserving record linkage may be achieved in the absence of a Trusted Third Party (TTP).

1) How can records be linked using exact identifiers to ensure accurate matches?
   Exact identifiers such as patient's social security number or ID number, enable highly accurate record linkage if handled securely. We explore how secure equality testing allows matching records.
2) How can records be linked using fuzzy fields to allow for error tolerance and partial matches? Real-world data often contains inconsistencies, making exact matching insufficient. We explore fuzzy matching techniques that allow for error-prone record matching.
3) How can blocking or filtering be integrated into privacy-preserving record linkage workflows to improve scalability? We examine how standard blocking and advanced techniques like LSH can be adapted to reduce computational overhead.
4) How can secure record linkage be performed to ensure sensitive data remains protected and compliant with GDPR and HIPAA regulations? We show how MPC enables secure linkage without a trusted third party, preserving input privacy, and ensuring compliance with GDPR and HIPPA.

Proof-Of-Concept Prototype In the scope of this research, we utilize exact identifiers for the proof-of-concept implementation using MPyC framework. This decision is driven by the nature of the data provided by iCAREdata. Additionally, this paper will include discussions on alternative approaches for privacy-preserving record linkage in cases where such exact identifiers are not present.

III. Literature Survey Method

We conducted a comprehensive literature search using Google Scholar, exploring recent relevant research regarding privacy-preserving record linking (PPRL) and secure multi-party computating (MPC), in the context of healthcare applications as well as in the general context. During our search we used the keywords "privacy-preserving record linkage," "secure multiparty computation," "fuzzy matching in healthcare," and "GDPR-compliant data sharing."

A. Study Selection Criteria

The studies identified by the search strategy underwent a multi-stage selection process based on predefined inclusion and exclusion criteria: The keywords mentioned above yielded 50 studies. We screened the titles and abstracts of these studies based on the following selection criteria.

- The study had to be peer reviewed.
- It must have been published between 2016 and 2025.
- It focused on the following aspects: implementations of privacy-preserving record linkage, theoretical foundations and general concepts of Secure Multiparty Computation, frameworks and the implementation of MPC, and finally, applications and challenges within Healthcare.
- It was written in English
- Its full text was available.

After screening, 19 studies were selected for full-text review and used in the final analysis.

B. Data Extraction and Synthesis

The 20 studies selected for the full review were read carefully. Information relevant to the study was extracted, including key concepts, methids (theoretical, implementation, evaluation) [10], [16], [18], [13], the properties of schemas or protocols [17], [14], specific application areas (especially health) [3], [16], the frameworks used or analyzed [14], [2], the challenges identified [5], [16], and future research directions [17], [5], [18]

IV. iCAREDATA

The *iCAREdata* dataset represents real-world patient records collected across different stages of healthcare. It includes three key record sheets: `Encounter`, `MedicationRequest`, and `MedicationDispense`.

The dataset is partitioned into two data sources originating from separate healthcare institutions:

- **Source 1:** `Encounter` and `MedicationRequest`
- **Source 2:** `MedicationDispense`

This setup reflects a realistic scenario in which prescription information (requests) and actual medication dispensing are handled by different entities, such as hospitals and community pharmacies. The dataset contains:

- 22,267 records in the `Encounter` table,
- 20,288 records in the `MedicationRequest` table,
- 10,001 records in the `MedicationDispense` table.

The decision to use deterministic record linkage for this specific dataset is motivated by the presence of stable, pseudonymized identifiers, `request_patient_md5` in **Source 1** and `subject_id_md5` in **Source 2**. In this context, the available linking fields show minimal variability or ambiguity, making probabilistic techniques—which rely on uncertainty across fields—less suitable. However, fuzzy matching remains an important consideration. Other datasets used within the *iCAREdata* framework may lack exact identifiers, necessitating alternative strategies. Therefore, it is essential to also address approaches for handling approximate linkage in such cases.

V. RECORD LINKAGE

Record linkage, also referred to as de-duplication, entity resolution, or data matching, is the process of identifying records in different datasets that correspond to the same real-world entity [10], [11], [5], [7], [18]. This task is important to data integration efforts [10], [7], for a more holistic understanding of entities by merging information from different sources [10]. Personal information is increasingly collected and linked across disparate data sources to provide for customized, high-quality, and timely analytical services [5]. Its importance spans numerous domains [10], including healthcare for patient data integration [10], [5], national security for identifying related entities across intelligence databases [5], Other applications that include financial data [18], and recommendation systems [5].

Three main approaches have been developed to address the challenges of record linkage: deterministic, probabilistic, and machine learning-based methods [10].

- **Deterministic Record Linkage:** A method that links records based on exact matches of identifiers or fields (e.g., name, date of birth, or hashed IDs). It assumes the presence of stable, high-quality identifiers [10].
- **Probabilistic Record Linkage:** This approach calculates the likelihood that records refer to the same entity based on partial agreements across multiple fields. It accounts for data entry errors and missing values using statistical models [11], [10].
- **Machine Learning-Based Methods:** These methods use supervised or unsupervised algorithms (e.g., decision trees, neural networks, or clustering) to learn patterns in matched and unmatched record pairs and improve linkage accuracy [10], [5].
- **Blocking or Filtering Techniques:** Strategies used to reduce the number of record pairs that will be compared [11] because naively comparing all pairs is computationally prohibitive or too inefficient for large files [18], [11], [7]. Blocking partitions or groups records based on shared attributes (blocking keys), such as a postal code or the first initial of the last name, to restrict comparisons to records within the same group, thereby improving computational efficiency. Indexing techniques quickly filter out dissimilar record pairs. Filtering techniques can also be applied to eliminate pairs that cannot meet a similarity threshold, further reducing the comparison space and speeding up the process [11], [5].

Each of these approaches is suited to different use cases and offers distinct advantages depending on the quality, volume , and privacy requirements of the data to be linked. The literature emphasize that data quality (errors, variations) and the volume of data (scalability, computational efficiency) pose significant challenges that influence the choice of techniques and have stimulated the development of more efficient methods, particularly in recent generations of PPRL [5].

In this work, we focus on deterministic because it is best suited for data with the availability of exact identifiers [10], [16], [18]. And for fuzzy matching, in this scope, we only address probabilistic approaches which are commonly used to handle lower quality data when exact identifiers are unavailable, where the process then relies on quasi-identifiers [10], [11], [5].

We also discuss Blocking and Filtering techniques to make record linkage more efficient when it comes to large datasets [11], [5].

A. Deterministic Record Linkage: Using Exact Identifiers

This subsection addresses Research Question 1: *How can records be linked using exact identifiers to ensure accurate matches?*

In the ideal scenario of linking records, unique and accurate identifiers are available for each entity (such as a Social Insurance Number - SSN) [10]. Where such identifiers exist, the matching of records is theoretically trivial when excluding the possibility of making errors [10].

In a context where identifiers are accurate but sensitive (and cannot be shared in the clear text), the problem of linking records with exact identifiers boils down to a problem of database joins or intersection of sets in privacy-preserving mode [7]. The data are shared with a TTP which performs the necessary computations and shares the results with the different sources [3], [5], [7]. Match accuracy is ensured by

directly comparing the identifiers between records in different datasets. If the identifiers are identical, the records are considered to refer to the same entity [7].

B. Linking Records with Quasi-Identifiers When Exact Identifiers Are Unavailable

This subsection addresses Research Question 2: *How can records be linked using fuzzy fields to allow for error tolerance and partial matches?*

When unique identifiers are missing or error-prone, record linking relies on identifying records that reference the same entity using common attributes that may not match exactly. These attributes called quasi-identifiers or fuzzy fields typically include name, date of birth, and address [10], [5], [16]. The challenge is that these fields are often inaccurate due to human errors during manual input, such as typos, spelling errors, or inverting information (such as first and last name) [10], [5], [16], [18], record linkage has historically used techniques like comparing q-grams (consecutive characters in a string) [10]. A widely used method for this in Privacy-Preserving Record Linkage (PPRL) has been the use of Bloom filters [10], [16], [5], [13], which allow for error-tolerant linkage of encoded data [16]. However, Bloom filter-based approaches have been shown to be vulnerable to dictionary attacks and frequency analysis/cryptanalysis, which can potentially leak sensitive information [10], [16], [5], [13], [7]. A recent advancement, proposed in the sources, suggests directly computing string similarity on secret-shared bigrams, a technique that offers an advantage by eliminating the need for Bloom filters [10].

In privacy-preserving record linking (PPRL) methods, these bigrams are often securely encoded and shared to allow computation without revealing the original values [10]. The similarity between two text fields is then evaluated by calculating the Sørensen–Dice coefficient (or simply Dice) on these filters. The Dice coefficient measures the overlap between the sets of bigrams represented by the activated bits [10].

Mathematical Formulation: Let Σ be our character alphabet and Σ^2 the set of all ordered bigrams. We index $\Sigma^2 = \{b_1, b_2, \ldots, b_{|\Sigma|^2}\}$. For any string s, define the *bigram indicator map*

$$M(s) \;=\; \big(m_1(s),\, m_2(s),\, \ldots,\, m_{|\Sigma|^2}(s)\big) \;\in\; \{0,1\}^{|\Sigma|^2},$$

where

$$m_i(s) \;=\; \begin{cases} 1, & \text{if the bigram } b_i \text{ appears at least once in } s, \\ 0, & \text{otherwise.} \end{cases}$$

In particular,

$$|M(s)| \;=\; \sum_{i=1}^{|\Sigma|^2} m_i(s) \quad \text{and} |M(s) \cap M(t)| \;=\; \sum_{i=1}^{|\Sigma|^2} m_i(s)\, m_i(t).$$

The Dice coefficient between s and t is then

$$\mathrm{Dice}(s,t) \;=\; \frac{2\,|M(s) \cap M(t)|}{|M(s)| + |M(t)|},$$

which takes values in $[0,1]$. We declare s and t a *fuzzy match* precisely when

$$\mathrm{Dice}(s,t) \;\geq\; \tau,$$

for a chosen threshold $\tau \in [0,1]$ (commonly $\tau = 0.8$ in practice).

Suppose now we wish to combine both • a collection of exact-match fields F_{ex} (e.g. SSN, gender) and • a collection of fuzzy fields F_{fz} (e.g. name, address).

For each exact field $f \in F_{\mathrm{ex}}$ define the indicator

$$\delta_f(s,t) = \begin{cases} 1 & s_f = t_f, \\ 0 & s_f \neq t_f, \end{cases}$$

and for each fuzzy field $g \in F_{\mathrm{fz}}$ set

$$S_g(s,t) = \mathrm{Dice}(s_g, t_g).$$

Assign nonnegative weights w_f to each exact field f and w_g to each fuzzy field g. Then the overall composite similarity is

$$\mathrm{Score}(s,t) \;=\; \frac{\displaystyle\sum_{f \in F_{\mathrm{ex}}} w_f\, \delta_f(s,t) \;+\; \sum_{g \in F_{\mathrm{fz}}} w_g\, S_g(s,t)}{\displaystyle\sum_{h \in F_{\mathrm{ex}} \cup F_{\mathrm{fz}}} w_h} \;\in\; [0,1].$$

Finally, one declares the record pair (s,t) a *link* exactly when

$$\mathrm{Score}(s,t) \;\geq\; T,$$

for a chosen composite threshold $T \in [0,1]$. Adjusting the individual weights $\{w_h\}$ and thresholds τ, T lets practitioners trade off precision and recall in deterministic linkage.

C. Blocking/Filtering Techniques

This subsection addresses Research Question 3: *How can blocking or filtering be integrated into privacy-preserving record linkage workflows to improve scalability?*

The problem of record linakge, when working on very large datasets, is the prohibitive cost of $O(|D_1| \times |D_2|)$ of comparing all possible pairs between two datasets(D_1) and (D_2) [11], which severely limits scalbility. To overcome this limitation it is common to apply a *blocking* or *filtering* pre–processing step [11], [5]. The goal is to partition each dataset into smaller subsets (blocks) so that only records within the same block are compared in full, thereby avoiding the $O(|D_1| \times |D_2|)$ cost of an all-pairs approach [11].

1. Basic Blocking: Choose one or more inexpensive "blocking keys" (e.g. the first three characters of last name, ZIP code, birth year) [11], [5], [13]. Assign each record in D_1 and D_2 to a block labeled by its key. Then form the candidate set [11]

where $r \in D_1$ and $s \in D_2$ denote individual records from dataset 1 and dataset 2, respectively.

Let $\mathrm{blk}(\cdot)$ be a blocking function that assigns each record r to a block key $\mathrm{blk}(r)$ [11].

$$\mathcal{C} \;=\; \bigcup_k \Big\{\, (r,s) \in D_1 \times D_2 : \mathrm{blk}(r) = \mathrm{blk}(s) = k \Big\}.$$

If there are B blocks of roughly equal size $\approx n/B$, the total number of comparisons drops from n^2 to about $B \times (n/B)^2 = n^2/B$ [11], [5]. Even a modest $B = 10$ yields a 10× speed-up [11].

2. Multi-pass and Composite Blocking: To avoid missing true matches that differ on one key, perform several passes with different blocking functions (e.g. pass 1 on name, pass 2 on date of birth). The union of candidate sets from all passes increases recall at the cost of a few extra comparisons [11]

3. Locality-Sensitive Hashing (LSH): LSH offers a probabilistic way to bucket *similar* records into the same block without fixed keys [5], [18]. Define a family of hash functions $h : \mathcal{X} \to \{1, \ldots, m\}$ with the property

$$\Pr\big[h(x) = h(y)\big] \;=\; \mathrm{sim}(x, y),$$

where $\mathrm{sim}(\cdot, \cdot)$ is (for example) the Dice coefficient on bigram sets.

In practice [5], [18]: label=()

1) Generate L independent hash functions $h_1, \ldots, h_L$.
2) For each record r, compute its *signature* $\big(h_1(r), \ldots, h_L(r)\big)$.
3) Use one or more *bands* of these signatures to assign records to buckets (blocks): two records share a bucket if they agree on all hashes within at least one band.

With carefully chosen L and band sizes, LSH ensures that truly similar records collide in a bucket with high probability, while dissimilar ones almost never do. If there are M total buckets and each bucket has expected size n/M, the expected number of comparisons is on the order of

$$\sum_{\mathrm{b}=1}^{M} \binom{n/M}{2} \;\approx\; M \times \frac{(n/M)^2}{2} \;=\; \frac{n^2}{2M},$$

again yielding a speed-up linear in M [18].

4.Conclusion:

1) Blocking reduces candidates by factors of B–M. Instead of $O(n^2)$, you compare only $O(n^2/B)$ (basic) or $O(n^2/M)$ (LSH).
2) Multi-pass balances recall vs. cost. More passes $\rightarrow$ higher chance of capturing true matches, at marginally higher cost.
3) LSH adapts to fuzzy similarity. Without predefined keys, it probabilistically groups near-duplicates, ideal for high-variance fields like names or addresses.

Incorporating blocking or LSH as a first step thus transforms the linkage workflow from an infeasible global comparison to a tractable series of block-level comparisons, making record linkage practical even on millions of records [11].

VI. Secure Record Linkage Using MPC

Having addressed our first two research questions, we now turn to the third one: how to perform record linkage securely while preserving the privacy of sensitive data. One powerful approach for achieving this is Secure Multi-Party Computation (MPC). MPC enables multiple parties, who may not fully trust each other, to collaboratively compute a function over their private inputs without revealing any information beyond the final output [10], [16], [1], [14].

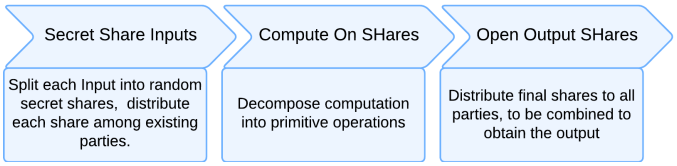

Fig. 1. Three stages of MPC

MPC is not a singular technology but rather an umbrella term encompassing several distinct cryptographic approaches, each with its own strengths and weaknesses [7], [9]. In a nutshell it encodes each party's private data within an algebraic *computational domain*, [9], most commonly rings of integers modulo M or finite fields such as $\mathbb{F}_M$ or $\mathbb{F}_{2^k}$ [9], [14]. When $M = 2$, the domain corresponds to Boolean circuits (addition as XOR, multiplication as AND), [9] whereas larger or composite M yields arithmetic circuits supporting modular addition and multiplication [9], [6]. If M is prime, $\mathbb{Z}_M$ becomes a field, enables division [9]. In these domains, linear operations on shared values (addition and scalar multiplication) can be performed locally due to the scheme's linearity, while non-linear operations (e.g. multiplication in arithmetic circuits or AND in Boolean circuits) necessitate interactive protocols to prevent leakage [8], [9], [6], [12], [14].

In the following sub-sections, we present the fundamental MPC concepts. Then we address secure computation over exact identifiers. Subsequently, we discuss secure fuzzy matching, focusing on how the Dice coefficient can be securely computed over distributed records, ensuring that no sensitive data is disclosed in the process.

A. Secret Sharing Schemes

A secret sharing scheme allows a confidential value x to be split into multiple shares distributed among n parties, such that only authorized subsets of parties can reconstruct x. In the context of MPC, secret sharing allows inputs to remain hidden in shares throughout computation, and only the agreed-upon output is revealed.

[tab:mpc-op]Table I provides a comparative overview of various MPC secret sharing schemes.

1) Linear Operations vs non-linear Operations on additive shares: There is a distinction between linear and non-linear operations on shared secret values [9], [6]. All schemes support linear operations (additions, scalar multiplications) via local computation on shares, with no communication required. By contrast, non-linear operations (secret-share multiplication or AND) introduce cross-terms that cannot be computed locally without revealing information. Thus every multiplication gate in an arithmetic or Boolean circuit triggers an interactive subprotocol among parties, often using pre-shared randomness (e.g. Beaver Triples)

2) Beaver Triples: Beaver triples provide a *preprocessing model* for efficient secure multiplication. In an offline phase,

TABLE I
SECRET SHARING SCHEMES

Scheme	Concept	Properties—	Usage	Frameworks
Additive	Split x into n random shares with $x_n = x - \sum_{i=1}^{n-1} x_i \pmod M$ [6].	- Correctness: $\sum x_i = x$ - Privacy: Any $< n$ parties learn nothing - Linearity	Used in MPC with dishonest majority [6].	GMW [15], [6], SPDZ [9], SPDZ2k [9]
Shamir's	Secret is constant term of random degree-t polynomial f over $\mathbb{F}_q$. Share: $(i, f(i))$ [9].	- Threshold $t+1$ for reconstruction [6] - Privacy with t or fewer shares [15] - Linearity [15] - Requires resharing after multiplication [6]	Used in honest-majority MPC [6].	MPyC [9], [2], SCALE-MAMBA [9], MP-SPDZ [9]
Replicated	Each party gets all but one additive share (e.g., P_1: x_1, x_2, etc.) [9].	- Linearity - Requires resharing post multiplication	Efficient communication for honest-majority MPC [9].	MP-SPDZ [9], SPDZ2k [9]
Binary	Secrets are bits; addition is XOR over $\mathrm{GF}(2)$ (Galois Field) [6], [9].	- Efficient for Boolean circuits [6]	Used for secure Boolean operations and malicious-majority MPC [6], [9].	MP-SPDZ (binary + SPDZ$^{2^k}$) , EMP-toolkit [9], PICCO [9]
Bitwise / Bit-Decomp.	Decomposes an integer a into bits a_i and shares them [9], [12].	- Enables bit-level operations [12] - Bit-decomposition is communication-heavy [12]	Used for comparisons, equality tests, hybrid arithmetic-Boolean MPC [6].	MP-SPDZ [9], EMP-toolkit [9], TinyGarble [12] , PICCO [9]

parties generate random shared values $([a], [b], [c])$ with $c = a \cdot b$. During the online phase, given shares $[x]$, $[y]$, they compute:

1) Locally subtract shares and open $d = x - a$, $e = y - b$.
2) Compute the product share as $[z] = e, [x] + d, [y] - d \cdot e + [c]$.

This reduces interaction rounds for each multiplication to a constant, at the cost of the offline triple generation. Similar correlated randomness can accelerate other operations (e.g. matrices, convolutions) in commodity-based MPC.

B. MPC Security Models: Security Guarantees

In the realm of secure multi-party computation for privacy-preserving record linkage, different security models define the assumptions about the behavior of participating parties and the level of protection offered. The semi-honest model assumes all parties follow the protocol but might passively try to learn more, making it suitable when working with trusted entities. The choice of security model in PPRL dictates the complexity and overhead of the MPC protocol employed, balancing the need for strong privacy guarantees with computational efficiency. [tab:mpc-Sec]Table II represents the different security models.

C. Secure Exact Matching

Secure exact matching is a type of comparison used in PPRL [10], [7], [16]. It involves determining whether specific records or fields of records from different databases are identical. Unlike approximate or fuzzy matching which can tolerate variations or errors in the data, exact matching requires that the values be exactly the same. For exact fields, a direct equality comparison is performed [16], [10]. This comparison produces a binary result: 1 if the fields are exactly the same, and 0 otherwise [16]. This decision making here is deterministic [7].

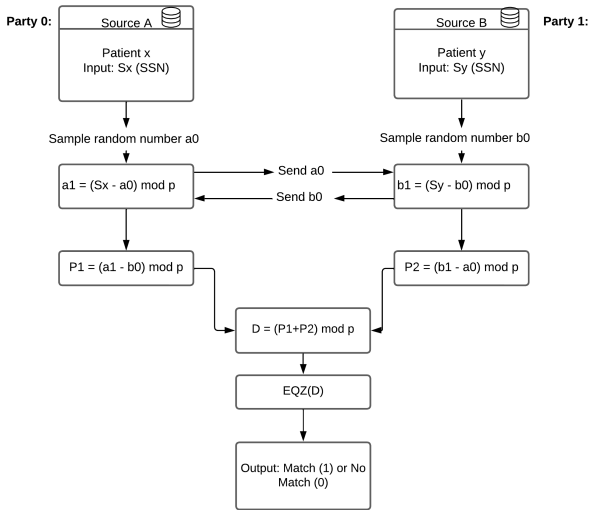

Fig. 2. Secure Record Linkage Using EQZ Over Social Security Number(SSN)

Figure 2 illustrates the flow of a secure equality check between two secret-shared values. Suppose two parties hold secret shares of values a and b, where each value is distributed among the computing parties using a secret-sharing scheme. To securely check whether $a = b$, the first step is to compute their difference $d = a - b$ using additive secret sharing. Since subtraction is a linear operation, it can be performed locally and efficiently [6].

The problem now reduces to determining whether the secret-

TABLE II
SECURITY MODELS IN SECURE MULTI-PARTY COMPUTATION

Security Model	**Adversary Behavior**	**Goal of Adversary**	**Security Guarantee**	**Number of Corrupt Parties**
Semi-honest	Follows the protocol correctly but tries to learn extra information from the data it observes [10], [14], [16], [6].	Gain more knowledge than allowed by analyzing intermediate values and communication [14], [16], [6].	Adversaries learn nothing more than what they can deduce from their input and output [6].	Any number, assuming they do not deviate from the protocol [6].
Malicious (Active)	Can arbitrarily deviate from the protocol [14], [1], [6].	Learn others' inputs, cause incorrect outputs, or make the protocol fail [14], [1].	Secure even if some parties behave incorrectly or dishonestly [14], [1].	Depends on the protocol and model (e.g., requires special techniques like zero-knowledge proofs) [6], [9].
Malicious Majority	Controls more than half the parties and may act arbitrarily [6], [1].	Break protocol correctness or learn private inputs [6], [1], [1].	Security is limited, but some guarantees may still be enforced [6].	More than 50% of particiipants [6].
Honest Majority	Controls a minority of the parties [6], [10], [9].	Disrupt the protocol or learn private information [14], [6], [1].	Robustness and privacy can be preserved against such adversaries [10], [6].	Fewer than 50% of particiipants [10], [6].

shared difference d is equal to zero, without revealing its actual value. To achieve this, a secure equality-to-zero (EQZ) test is performed [6], [14]. Which is a cryptographic procedure for determining whether a secretly shared value, denoted for example by a is equal to zero, without revealing a [6], [14].

One common approach for this is **bit decomposition**, [12], [14]: the secret-shared value d is converted into its bitwise representation, where each bit is itself secret-shared [14]. This approach is considered a "core component" of integer operations in computational frameworks [9]. It is a very useful tool [12] because its main utility is that it allows bitwise operations, even when secrets are shared arithmetically [12]. This also allows you to benefit from the advantages of Boolean and arithmetic circuits by changing the representation [12].

A secure OR operation is then applied across all these bits [12], [14]. A "secure OR" operation is a logical operation (OR/disjunction) performed on bits that are secretly shared between multiple parties, so that the result (a shared secret bit) is correctly calculated without any party learning the values of the input bits [12], [14].

- If $d = 0$, all bits are zero, and the OR of all-zero bits yields a secret-shared 0 [12], [14].
- If $d \neq 0$, at least one bit is non-zero, and the OR yields a secret-shared 1 [12], [14].

The result of this secure OR operation, a secret-shared 0 or 1, indicates whether d was zero or not, thereby securely determining whether $a = b$ [12], [14].

D. Secure Fuzzy Matching

In many real-world healthcare datasets [10], [16], fields such as patient names or addresses are inconsistently formatted or contain typographical errors [16]. To tolerate these discrepancies, we use a *bigram + Dice coefficient* approach [16], like mentioned in section 4, but performed over secret shares in an MPC setting [16], [10].

For two input records x_i and y_j, let

$$\langle M(x_i^f)\rangle,\ \langle M(y_j^f)\rangle\ \in\ \{0,1\}^{|\Sigma|^2} \tag{1}$$

be the shared bigram maps [10], and let

$$\langle\,|M(x_i^f)|\rangle,\ \langle\,|M(y_j^f)|\rangle \tag{2}$$

be the shares of their cardinalities [10]. We compute the Dice coefficient in MPC as follows:

1) **Intersection:**

$$\langle b\rangle\ =\ \langle M(x_i^f)\ \wedge\ M(y_j^f)\rangle \tag{3}$$

, [10] Bitwise AND on each coordinate, [10]

2) **Numerator:**

$$\langle n\rangle\ =\ 2\ \times\ \sum_{k=1}^{|\Sigma|^2}\langle b_k\rangle \tag{4}$$

, [10] Shared sum of bits, scaled by 2 [10]

3) **Denominator:**

$$\langle d\rangle\ =\ \langle\,|M(x_i^f)|\rangle\ +\ \langle\,|M(y_j^f)|\rangle \tag{5}$$

, [10] Shared sum of cardinalities, [10]

4) **Output Shares:**

$$\big(\langle n\rangle,\ \langle d\rangle\big)\ =\ \mathrm{DiceCircuit}\big(\langle M(x_i^f)\rangle,\ \langle M(y_j^f)\rangle, \tag{6}$$

$$\langle\,|M(x_i^f)|\rangle,\ \langle\,|M(y_j^f)|\rangle\big). \tag{7}$$

, [10]

VII. Prototype

In this section, we describe a proof-of-concept prototype demonstrating the feasibility of PPRL via MPC. Within the iCAREdata database, the goal is to match patient identifiers across two datasets, simulated using two Excel files: one containing request and encounter data, and the other containing dispense data. The prototype must securely compute the number of matching patient records without revealing the actual identifiers. Given that iCAREdata operates within a framework of trusted parties where adherence to the protocol is mutually beneficial, we adopt the semi-honest security model for our privacy-preserving record linkage approach. This assumption is reasonable in our context, aligning with the collaborative nature of iCAREdata and the shared interest in maintaining data privacy while achieving accurate record linkage. Our prototype implementation focuses only on exact matching of patient records because each patient is identified using a unique identifier within the iCAREdata database.

We adopt the semi-honest (honest-but-curious) adversarial model, as iCAREdata operates among mutually trusted parties.

To achieve this, we use the MPyC (Multi-Party Computation in Python) framework to simulate a secure protocol under the semi-honest adversarial model. This framework implements the BGW (Ben-Or, Goldwasser, Wigderson) honest majority multi-party protocol. It is secure against semi-honest adversaries. The BGW (Ben-Or, Goldwasser, Wigderson) protocol is based on Shamir's secret sharing and allows secure evaluation of arithmetic circuits, supporting both addition and multiplication operations over shared data. BGW provides information-theoretic security against up to one-third of malicious participants (i.e., t ¡ n/3) and does not rely on cryptographic assumptions, which means that the protocol is secure no matter how powerful the attacker is, even with unlimited computing resources. This is achieved using mathematical guarantees (like Shamir's Secret Sharing), which ensure that a subset of shares reveals absolutely nothing about the secret.

The simulation involves two parties:

- Party 0 holds hashed patient IDs from the request and encounter dataset.
- Party 1 holds hashed patient IDs from the dispense dataset.

A. Hashing and Input Preparation

Although the datasets already contain anonymized identifiers, we apply an additional pseudonymization step for simulation purposes. Patient identifiers are hashed using SHA-256 and truncated to 64 bits. This step, performed prior to secret sharing, reduces the input domain and mimics a realistic preprocessing stage in privacy-preserving computation pipelines

B. Secure Equality Testing in MPyC

MPyC allows developers to write expressions such as `c = a == b` over secret-shared values in standard Python syntax [9], [2]. Behind the scenes, this triggers a secure multi-party protocol [12].

For two secret-shared 64-bit integers a and b, this expression evaluates to a secret-shared boolean value c, such that $c = 1$ if and only if $a = b$, and $c = 0$ otherwise [12], [14]. No party learns the actual values of a, b, or even c, unless the result is explicitly revealed [2]. For more details refer to section 5.

The equality protocol in MPyC proceeds as follows:

1) **Compute the difference:** Each party locally computes shares of the difference $d = a - b$ over a large prime field [12]. If $a = b$, then $d = 0$; otherwise, $d \neq 0$ [12]. The value d remains secret-shared [12].
2) **Bit decomposition:** To test whether $d = 0$, MPyC invokes a secure bit-decomposition protocol, converting the secret-shared integer d into a vector of secret-shared bits $[d_0, d_1, \ldots, d_{k-1}]$ representing its binary form (least significant bit first) [9]. This step is performed securely without revealing any bit values [9], [12], [14].
3) **Equality bit computation:** MPyC then checks whether all bits of d are zero [12]. This is equivalent to computing the logical OR of all bits:

   $$z = d_0 \vee d_1 \vee \cdots \vee d_{k-1}$$

   and then negating the result to obtain:

   $$c = 1 - z$$

   If any $d_i = 1$, then $z = 1$ and $c = 0$ (i.e., $a \neq b$); if all $d_i = 0$, then $z = 0$ and $c = 1$ (i.e., $a = b$) [12].

VIII. Discussion

A. Algorithmic Feasibility vs. Computational Feasibility

a) Algorithmic Feasibility: The fundamental theory of MPC, established in seminal works such as those of Yao and of Goldreich, Micali, and Wigderson (GMW), demonstrated that any function representable as a circuit (Boolean or arithmetic) can be securely computed. This capability, often called a "general plausibility result," shows what is possible in principle from an algorithmic perspective, with proven security guarantees [4], [6], [7], [12], [16].

b) Computational Feasibility: The challenge of computational feasibility lies in inefficiencies manifested as high communication costs. A naive comparison of all pairs of records between two databases of size N requires N^2 comparisons. Hence, blocking or indexing techniques (see Section 5.3) are employed to reduce computation cost by up to 10 times [5], [7], [16], [18].

- Preprocessing Model (Offline/Online): Many modern MPC protocols, including those for PPRL, split the computation into an input-independent preprocessing phase (often expensive but performable in advance) and a highly efficient online phase that leverages the preprocessed material once inputs are known. This design moves heavy cryptographic operations off the critical path when low latency is crucial [6], [8], [17].

- Frameworks and software implementations such as MP-SPDZ, Sharemind, ABY, MPyC, and MainSEL [9], [10], [14], [16] embody these optimized protocols and provide tooling for constructing complex MPC workflows. They offer standardized platforms to avoid re-implementing basic protocols. For

example, the MainSEL project is a three-party probabilistic PPRL system targeting medical datasets; it optimizes key operations (e.g., integer division) and reports concrete execution times (seconds to minutes to link 10,000 records under realistic network conditions), providing direct evidence of practical feasibility [16].

- Efficiency Gains with Three-Party Computation: Three-party secure multi-party computation (MPC) protocols offer notable efficiency improvements over traditional two-party models [10], [8], especially when a semi-honest helper party is introduced [8]. In this setting, the third party participates in computations without learning the input or output, acting solely to facilitate the protocol. This allows for reduced communication rounds and lower computational overhead [10].

For example, secret-sharing-based protocols such as those implemented in `MP-SPDZ` or `ABY3` leverage additive sharing schemes and preprocessed multiplication triples [9], significantly accelerating operations like secure comparisons and matrix multiplications [14], [17]. These protocols can achieve near-linear scalability and performance that is suitable for real-time or large-scale applications [10], [12], [17]

B. Cloud Deployment Considerations

Deploying MPC protocols in a cloud environment further enhances their practical use [19]. Cloud platforms provide elastic computational resources, geographic distribution, and scalability [19], [10].

A typical cloud-based MPC setup involves deploying each party (including the helper) in isolated virtual environments, potentially on different cloud providers, to minimize the risk of collusion [19]. Services like Microsoft Azure Confidential Computing, Google Cloud's Confidential VMs, and AWS Nitro Enclaves offer trusted execution environments (TEEs) that can complement MPC by securing runtime computations and reducing trust assumptions [8].

IX. Conclusion

This work explored the design and implementation considerations for privacy-preserving record linkage (PPRL) using secure multi-party computation (MPC), with a particular focus on the healthcare domain. We addressed four core research questions:

- (1) How can exact and fuzzy matching be implemented securely using MPC? We discussed secure protocols for both exact and bigram-based fuzzy matching on secret-shared data, supporting both deterministic and probabilistic approximate linkage.
- (2) How can blocking or filtering be integrated into privacy- preserving record linkage workflows to improve scalability? We have discussed the use of these techniques to improve the efficiency of linking algorithms.
- (3) and (4) How can secure record linkage be performed to ensure sensitive data remains protected and compliant with GDPR and HIPAA regulations?

Future work can extend our findings in several directions. First, a full implementation and evaluation of a three-party prototype on healthcare data. Second, integrating privacy-preserving blocking techniques in our implementation for further improved scalability. Lastly, exploring cloud solutions to facilitate computation overheads.

Key Considerations:

- **Data Quality and Matching Type:** The choice between deterministic and probabilistic (fuzzy) matching depends on the availability and reliability of exact identifiers versus quasi-identifiers.
- **Security Model:** The adversarial model (semi-honest vs. malicious) affects protocol design, efficiency, and implementation trust assumptions.
- **Scalability:** Efficient record linkage on large datasets requires blocking/filtering techniques such as LSH to reduce computational overhead.
- **Privacy Compliance:** Ensuring compliance with GDPR by never revealing raw personal identifiers and performing all computations on encrypted or secret-shared data.

X. Acknowledgments

This work is supported by (a) University of Antwerp; (b) iCARE-data;

References

[1] Bar Alon, Amos Beimel, and Eran Omri. Three party secure computation with friends and foes. In *Theory of Cryptography Conference*, pages 156–185. Springer, 2023.

[2] Fatih Aykurt. Analysis of two versatile mpc frameworks mp-spdz and mpyc. Master's thesis, Middle East Technical University (Turkey), 2023.

[3] Annelies Colliers, Stefaan Bartholomeeusen, Roy Remmen, Samuel Coenen, Barbara Michiels, Hilde Bastiaens, Paul Van Royen, Veronique Verhoeven, Philip Holmgren, Bernard De Ruyck, et al. Improving care and research electronic data trust antwerp (icaredata): a research database of linked data on out-of-hours primary care. *BMC research notes*, 9:1–7, 2016.

[4] Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 473–503. Springer, 2019.

[5] Aris Gkoulalas-Divanis, Dinusha Vatsalan, Dimitrios Karapiperis, and Murat Kantarcioglu. Modern privacy-preserving record linkage techniques: An overview. *IEEE Transactions on Information Forensics and Security*, 16:4966–4987, 2021.

[6] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78(110):1–108, 1998.

[7] Rob Hall and Stephen E Fienberg. Privacy-preserving record linkage. In *Interational conference on privacy in statistical databases*, pages 269–283. Springer, 2010.

[8] Alberto Ibarrondo, Hervé Chabanne, and Melek Önen. Funshade: Function secret sharing for two-party secure thresholded distance evaluation. In *PETS 2023, 23rd Privacy Enhancing Technologies Symposium*, volume 2023, pages 21–34, 2023.

[9] Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 1575–1590, 2020.

[10] Şeyma Selcan Mağara, Noah Dietrich, Ali Burak Ünal, and Mete Akgün. Accelerating privacy-preserving medical record linkage: A three-party mpc approach. *arXiv preprint arXiv:2410.21605*, 2024.

[11] Jared S Murray. Probabilistic record linkage and deduplication after indexing, blocking, and filtering. *arXiv preprint arXiv:1603.07816*, 2016.

[12] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *International Workshop on Public Key Cryptography*, pages 343–360. Springer, 2007.

[13] Sean Randall, Helen Wichmann, Adrian Brown, James Boyd, Tom Eitelhuber, Alexandra Merchant, and Anna Ferrante. A blinded evaluation of privacy preserving record linkage with bloom filters. *BMC medical research methodology*, 22(1):22, 2022.

[14] Tord Ingolf Reistad. Multiparty comparison-an improved multiparty protocol for comparison of secret-shared values. In *International Conference on Security and Cryptography*, volume 1, pages 325–330. SCITEPRESS, 2009.

[15] Hovav Shacham and Alexandra Boldyreva. *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II*, volume 10992. Springer, 2018.

[16] Sebastian Stammler, Tobias Kussel, Phillipp Schoppmann, Florian Stampe, Galina Tremper, Stefan Katzenbeisser, Kay Hamacher, and Martin Lablans. Mainzelliste secureepilinker (mainsel): privacy-preserving record linkage using secure multi-party computation. *Bioinformatics*, 38(6):1657–1668, 2022.

[17] Shuang Sun and Eleftheria Makri. Sok: Multiparty computation in preprocessing model. *Cryptology ePrint Archive*, 2025.

[18] Ruidi Wei and Florian Kerschbaum. Cryptographically secure private record linkage using locality-sensitive hashing. *Proceedings of the VLDB Endowment*, 17(2):79–91, 2023.

[19] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu-an Tan. Secure multi-party computation: theory, practice and applications. *Information Sciences*, 476:357–372, 2019.