Information Systems 102 (2021) 101826

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Blockchain-based Privacy-Preserving Record Linkage: enhancing data privacy in an untrusted environment

Thiago Nóbrega [a,*], Carlos Eduardo S. Pires [a], Dimas Cassimiro Nascimento [b]

[a] Federal University of Campina Grande (UFCG), Aprígio Veloso - Campina Grande - PB, Brazil
[b] Universidade Federal do Agreste de Pernambuco/Federal Rural University of Pernambuco, Brazil

ARTICLE INFO

Article history:
Received 24 September 2020
Received in revised form 18 May 2021
Accepted 2 June 2021
Available online 8 June 2021
Recommended by Alysson Neves Bessani

Keywords:
Entity resolution
Privacy preserving entity resolution
Data privacy
Bloom Filter
Blockchain

ABSTRACT

Privacy-Preserving Record Linkage (PPRL) intends to integrate private data from several data sources held by different parties. Due to recent laws and regulations (e.g, General Data Protection Regulation), PPRL approaches are increasingly demanded in real-world application areas such as health-care, credit analysis, public policy evaluation, and national security. However, the majority of the PPRL approaches consider an unrealistic adversary model, particularly the Honest but Curious (HBC) model, which assumes that all PPRL parties will follow a pre-agreed data integration protocol, and will not try to break the confidentiality of the data handled during the process. The HBC model is hard to employ in real-world applications, mainly because of the need to trust other parties fully. To overcome the limitations associated with the majority of the adversary models considered by PPRL approaches, we propose a protocol that considers covert adversaries, i.e., adversaries that may deviate arbitrarily from the protocol specification in an attempt to cheat. In such protocol, however, the honest parties are able to detect this misbehavior with a high probability. To provide a proof-of-concept implementation of this protocol, we employ the Blockchain technology and propose an improvement in the most used anonymization technique for PPRL, the Bloom Filter. The evaluation carried out using several real-world data sources has demonstrated the effectiveness (linkage quality) obtained by our contributions, as well as the ability to detect the misbehavior of a malicious adversary during the PPRL execution.

1. Introduction

Governments and private organizations collect and process large data sources to extract knowledge and assist in decision-making, data mining, and data analytics tasks. These organizations usually need to integrate and link matching records (entities) that correspond to the same real-world entity from various sources (held by different parties) with the purpose of increasing quality or enriching information [1]. In this scenario, the Record Linkage (RL) task is commonly employed to identify matching entities across multiple data sources [2]. It is important to remark that the data sources used as input in RL are usually independently projected using different data models (with distinct representations for the same real-world concepts) and technologies (i.e., relational databases, graph-oriented databases, and so on).

In general, RL needs to be applied in scenarios that hold sensitive personal information about the entities to be linked. However, sharing or exchanging such values among different organizations is often prohibited due to privacy and confidentiality concerns [1,3,4]. Thus, in scenarios where the privacy of the entities matters, Privacy-Preserving Record Linkage is employed. PPRL is the task of performing RL over data sources that belong to different parties (suppliers) without revealing any information that could compromise the privacy of the parties' data [1].

As examples of private data usage, we have banks that seek to reduce credit fraud, national security applications that aim to identify terrorists, and medical data that need to be integrated for a variety of purposes such as patient profiles and disease outbreak prediction, public health policy evaluation, new drug studies, to name a few [1,5].

According to recent PPRL surveys [1,3,4,6], most of the state-of-the-art PPRL techniques assume an Honest But Curious (HBC) security model during the matching process. This model considers that the parties will follow a pre-agreed protocol, but will attempt to learn all possible information from legitimately received messages [4]. In contrast to the HBC model, we have the malicious security model, which assumes that the parties will not follow the pre-agreed protocol by refusing to participate, performing arbitrary computation in the input data, or aborting the protocol at any time [3].

* Corresponding author.
E-mail addresses: thiagopereira@copin.ufcg.edu.br (T. Nóbrega), cesp@dsc.ufcg.edu.br (C.E.S. Pires), dimas.cassimiro@ufape.edu.br (D.C. Nascimento).
URL: http://github.com/thiagonobrega (T. Nóbrega).

https://doi.org/10.1016/j.is.2021.101826
0306-4379/

Since the HBC model requires the parties to fully trust each other, which is not realistic for real-world applications [1,4], and the malicious model is computationally expensive, due to the anonymization techniques (i.e., homomorphic encryption) and communication cost [1]. The need for new security models that enable an auditability of the similarity computations (i.e., models that lie in between the HBC and the malicious models) is reported as an open problem [1,3,4,6,7].

In this context, this paper introduces a novel security protocol that enables the auditability of the computations, also named as covert adversaries [8], performed during a PPRL process. In other words, the proposed protocol allows the parties to detect, with high probability, the misbehave of a malicious party during the entities' similarity computation. To implement the protocol, we employ a decentralized environment where untrusted (or semi-trusted) parties perform the computations required by the PPRL.

As the decentralized computing environment, we use (public and private) Blockchain networks in order to provide auditability to PPRL. The Blockchain technology aims to provide shared data access and computation for parties that do not trust each other [9]. The computations and the data sharing are possible due to an immutable cryptographically append-only log, which is replicated and managed in a decentralized environment composed by untrusted parties [10].

Blockchain platforms are being used by applications of different domains [9–12], including government, healthcare, and IoT. In such applications, the Blockchain is treated as a shared database or data processing platform which is employed when the participants do not trust each other. For this reason, we use the Blockchain (Smart Contract) as a Semi-Trusted Third Party (STTP) in PPRL.

However, the Blockchain does not provide a mechanism to preserve the privacy of the entities during the PPRL process. Actually, the Blockchain reduces the privacy of the PPRL by replicating the entire data amongst the untrusted parties. To overcome this limitation, we also propose an improvement for the most prevalent anonymization technique used in PPRL applications: Bloom Filter [13]. Such improvement, named Splitting Bloom Filter (SBF), offers a lower risk of privacy leakage and, additionally, reduces the possibility of collusion between malicious parties.

Contributions. We summarize the major contributions of this work as follows:

- We propose a novel privacy-preserving protocol that considers a covert security model, which lies between the HBC and Malicious models;
- We present a Proof-of-Concept implementation of our privacy-preserving protocol using the Blockchain technology;
- We propose an improvement to the Bloom Filter (SBF) that increases the privacy guarantees of the entity comparisons performed in the PPRL, by reducing the amount of information shared amongst the PPRL parties; and
- We perform theoretical and empirical evaluations using real-world data (18 data sources from nine domains) to assess the efficacy, efficiency, and privacy capabilities of our contributions.

Outline. The remaining of this paper is organized as follows. In Section 2, we present the required concepts for a better understating of our work. In Section 3, we formalize the problem tackled in this paper. Related work is discussed in Section 4. In Sections 5, 6 and 7, we describe the SBF, the proposed Privacy-Preserving Protocol and its Proof-of-Concept implementation, respectively. In Section 8, we evaluate (theoretically and empirically) the efficacy, efficiency, and privacy capabilities of our contributions using real-world data sources. Finally, in Section 9, we conclude the paper with an outlook on future research directions.

2. Preliminaries and background

For a better understanding of our approach, this section provides an overview of the techniques and concepts commonly used in PPRL and Blockchain.

2.1. PPRL

PPRL is a solution to the problem of recognizing entities (e.g., persons, objects, or events) available at different data sources, which represent the same real-world object, ensuring that the privacy and confidentiality of the data are preserved during the linkage process [14]. In this sense, PPRL needs to be executed in anonymized data, by perturbing the original data with the use of encryption, hash functions, and/or noise additions. At the end of PPRL, only a limited amount of information is revealed. For instance, a party only knows which of its own entities exist in the data source of the other parties [15]. The overall PPRL process is divided into several steps:

- *Data pre-processing*: converts the original data into a standard format, previously agreed between the involved parties;
- *Data anonymization*: anonymizes the original data;
- *Blocking* (or filtering): reduces the number of comparisons that need to be conducted in the following step (comparison) by pruning the entities pairs that unlikely correspond to matches. For instance, the first letter of a patient's name can be used as a blocking key to group entities (patients) into a block of candidate pairs;
- *Comparison*: applies similarity (or comparison) functions to the anonymized data in order to calculate the similarity between candidate pairs; and
- *Classification*: receives the similarity values calculated for the candidate pairs from the previous step and classifies the entities pairs into matches or non-matches.

Usually, PPRL needs to deal with two problems inherited from RL: the lack of unique identifiers (e.g., Social Security Number) and dirty data. The former makes the execution of PPRL even more challenging because the identification of similar entities is made by comparing their attributes rather than a unique identifier. For instance, consider a study involving patients that were assisted and treated in medical emergencies, e.g., heart attacks and car accidents. In this scenario, since patients can be admitted into the hospital without document (i.e., Social Security Number), PPRL needs to consider the attributes (i.e., *name* and *address*) to identify matching patients across different hospitals.

The challenge of recognizing matching entities also gets hampered by the presence of dirty data. In other terms, the comparison step must consider missing data, typos, fused or split words, and abbreviations [4,16]. For instance, one operator could misspell the name of the patient (e.g., 'Ana' turns 'Ane'), a person can change his/her name (e.g., 'Ana Silva' marries and her name is changed to 'Ana Santos'), which makes the linkage task even more challenging [2,17]. It is worthwhile to mention that, in the context of PPRL, both of the previously mentioned problems (lack of unique identifiers and dirty data) must be solved, taking into account the anonymized representation of the entities.

In this sense, consider p participants $(\mathbb{P}_1, \ldots, \mathbb{P}_p)$ with their respective data sources $(\mathbb{D}_1, \ldots, \mathbb{D}_p)$, where each data source is composed by a set of distinct entities $\mathbb{D}_p = \{e_1, \ldots, e_n\}_{\neq}$. Considering that the data sources are relational databases, then

$\mathbb{D}_p$ refers to the table (relation) of participant p and e_n represents one row of table $\mathbb{D}_p$.

Data source anonymization. *To ensure the privacy and confidentiality of the entities, PPRL must consider that the entities of a data source ($\mathbb{D}_p$) are anonymized ($e_n^\tau = anonymize(e_n)$), such $\mathbb{D}_p^\tau = \{anonymize(e) | \forall e \in \mathbb{D}_p\}$.*

To identify the entities that represent the same real-world object, PPRL needs to execute a decision model $\Xi(\cdot)$ considering all anonymized entities $(e_1^\tau, \ldots, e_p^\tau)$ of all participating data sources, as presented in Eq. (1).

$$\Xi(e_1^\tau, \ldots, e_i^\tau) \mid \{e_1^\tau, \ldots, e_i^\tau\} \in \prod_{i \in \{1,\ldots,p\}} \mathbb{D}_i^\tau, \tag{1}$$

where $\prod_{i\in\{1,\ldots,p\}} \mathbb{D}_i^\tau = \mathbb{D}_1^\tau \times \cdots \times \mathbb{D}_p^\tau$.

The decision model $\Xi(\cdot)$ compares and classifies (these tasks refer to the comparison and classification steps of PPRL) the entity pairs as matches (M) if the entities truly correspond to the same real-world object (denoted by $e_1 = e_2$); and as non-matches (U), otherwise ($e_1 \neq e_2$).

Decision Model. *Knowing that the decision model (Ξ) may consider multiple participants ($p \geq 2$), an entity pair is classified as match (M) if all entities provided as input (to the model) represent the same real-world object. On the other hand, if one (or more) entities do not represent the same real-world object, the decision model classifies the input as non-match (U). We formalize the decision model in Eq. (2):*

$$\Xi(e_1^\tau, \ldots, e_i^\tau) = \begin{cases} M \iff \forall (e_j^\tau, e_{j+1}^\tau) : (e_1^\tau, \ldots, e_i^\tau), e_j^\tau = e_{j+1}^\tau \\ U \iff \exists (e_j^\tau, e_{j+1}^\tau) : (e_1^\tau, \ldots, e_i^\tau), e_j^\tau \neq e_{j+1}^\tau \end{cases}, \tag{2}$$

where (e_j^τ, e_{j+1}^τ) is a set of anonymized entities $(e_1^\tau, \ldots, e_i^\tau)$ marked to be compared by the blocking stage of the PPRL.

In order to address privacy issues in PPRL, an adversary (or security) model must be considered. In general, PPRL approaches use two adversary models [4,18–21]: Malicious and Honest But Curious — HBC (also known as semi-honest). The HBC model assumes that the parties follow a pre-agreed protocol, yet the adversaries may attempt to learn additional information by analyzing the messages (data) received during the execution. In the Malicious model, the adversaries are not bound in any way to follow the protocol, i.e., they behave arbitrarily.

The HBC security model (also named as adversary model) can be carried out very efficiently and with good linkage quality. However, it provides weak security capabilities due to limitations of the anonymization and the employed comparison techniques [18,22]. Another aspect of HBC is that it is hard to fully trust that the PPRL parties will follow the protocol, turning HBC a non-realistic model. Regarding the Malicious model, it provides a robust security guarantee which typically does not assume that the pre-agreed protocol is followed. Nevertheless, the Malicious model is computationally expensive, and it usually generates low linkage quality due to the complexity of the similarity and comparison operations performed over the anonymized data — making it troublesome to be implemented and used in practice [1].

Considering the limitations of the PPRL approaches that use the HBC and Malicious models, the need for a more realistic model has been highlighted as an open problem in recent surveys [1,3,4,20,21]. The Accountable computing and the Covert adversary models lie between the HBC and Malicious models. The Covert model allows the honest parties to detect the misbehavior of an adversary with high probability [23] whilst the Accountable computing model provides accountability for privacy compromises without the excessive complexity and cost of the Malicious model [24]. To the best of our knowledge, the Covert model has not been investigated in PPRL yet. Thus, this paper proposes the first PPRL approach that considers such adversary model.

2.1.1. Vulnerabilities and privacy attacks in PPRL

PPRL participants need to share their anonymized entities (data) in order to identify duplicated entities. Such data may be used by an adversary as input to a privacy attack, which breaks the confidentiality and privacy of the entities. Several privacy attacks can been employed in the PPRL context. The most common ones include [7,25–28]:

1. **Collusion**: two or more parties collude to learn another party data, i.e., the malicious parties publish the anonymized data of an honest party [26,28];
2. **Frequency**: consists in observing how often a given value occurs in an anonymized dataset and comparing it to how often this value occurs in a known dataset [27];
3. **Dictionary**: consists in assembling a word list (Dictionary) using several anonymization techniques until the dictionary values match the values in the attacked dataset [28]; and
4. **Cryptoanalysis attacks**: several cryptoanalysis attacks have been developed against Bloom filters. These attacks allow the iterative mapping of anonymized values back to their original values (more detailed in Section 8) [7,25].

It is worthwhile to mention that the classes of attacks discussed in this section present one point in common: all of them need a significant amount of information to succeed. In the PPRL context, it means that a malicious party needs to have a meaningful number of anonymized entities to reidentify them.

2.2. Bloom filters

The most common anonymization technique employed in PPRL is Bloom Filter (BF) [13]. A BF consists of an array of l-bits (filter length), with all bits set to '*0*', initially. The BF can be formalized as $[b_0, \ldots, b_l]$, where b_m represents the bit of position m. To insert a set of elements ($S = \{s_1, \ldots, s_n\}$) in a BF, k independent *hash functions*,[1] $H(x) = \{h_1(x), \ldots, h_k(x)\}$, are employed to map the elements $s_i \in S$ to the l-bits array. Furthermore, the output of $H(x)$ indicates the bits (b) that need to be set to '*1*' in the BF.

The quality of the anonymization depends on the BF parametrization, number of hash functions (k) and filter length (l) [29]. For a given number of elements (n) to be inserted into the BF, the probability of a specific bit still '*0*' is $p = e^{-\frac{k \times n}{l}}$. We can choose a k to minimize the probability of two different elements being mapped to the same bit position (f) by setting $p = 0.5$ [30]. In other words, the probability of a bit in the BF still 0 (or flipped to 1) should be 0.5 to reduce the false-positive rate. For PPRL, this is relevant because the bit patterns and their frequencies in a set of BF can be exploited by frequency [26] and cryptonalysis [7] attacks. Such attacks exploit the fact that BFs that are almost empty can provide information about rare elements. As a result, the re-identification of the entities is facilitated [15].

Fig. 1 illustrates the insertion of the names **ANA** and **ANE** into the BFs a and b, respectively. First, each name is transformed into bigrams; then, each bigram is mapped by a hash function into a BF position. Finally, the positions are changed to '*1*' in the BF.

[1] A hash function is an algorithm that takes messages and maps them to a value of a certain length, called a hash value or hash.

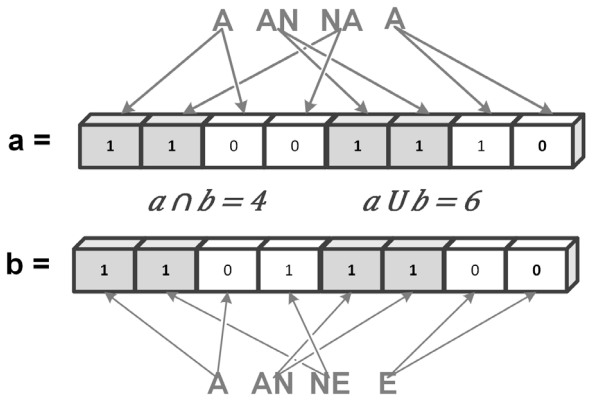

Fig. 1. Inserting the names ANA and ANE into 8-bits Bloom filters (l = 8 and k= 2).

The BF technique also enables the similarity calculation of two filters (a and b) through token distance functions, such as $Jaccard = \frac{|a \cap b|}{|a \cup b|}$, where $|a \cap b|$ is the number of positions with the value *1* that coincide in both filters; and $|a \cup b|$ represents the number of positions with the value *1* in the union of the filters a and b. Regarding the example illustrated in Fig. 1, the *Jaccard* similarity between filter a (representing the name ANA) and filter **b** (ANE) is equal to $\frac{4}{6} = 0.66$.

2.3. Blockchain

Blockchain, also called distributed ledger, is essentially an append-only DataBase Management System (DBMS) maintained by a set of nodes that do not fully trust each other [9,10]. In other words, Blockchain is a technology that maintains the states and the historical transactions, using a peer-to-peer network, without any central node to enforce compliance of the rules. Blockchain provides immutable storage (temper evidence guarantee) of transactions in a chain of blocks, by storing data (records) in blocks that are linked using cryptography tools, i.e., the previous block hash, the transaction owner signature, and the identifier (id) of the machine (miner) that executed the transaction.

To effectively store a record in the distributed ledger, the record needs to be replicated in every node of the Blockchain. In order to guarantee the consistency of the ledger, a party sends the record within a transaction with a cryptographic puzzle to be mined by the Blockchain nodes. After the first miner solves the cryptographic puzzle (embedded with the transaction), the other nodes will verify the transaction execution and store it in its local ledger. In other words, every node in the network verifies all transactions, replicating the data and the computation in all nodes of the network. Therefore, all inputs and outputs of the transaction are publicly available from every Blockchain network member.

Considering the access (publicity) to the transaction data in the Blockchain, we can classify the Blockchain into two environments: (i) public (or permissionless) Blockchains, where anyone can join the network (e.g., Ethereum and Bitcoin); and (ii) private (or permissioned) Blockchains, where there is an access control to manage who can join the network (e.g., Hyperledger, Ethereum, and Microsoft Blockchain Confidential Consortium Framework) [31].

In both scenarios, public and private Blockchains, it is expected that some of the nodes present a malicious (Byzantine) behavior. In other words, some nodes will collude to change the transactions stored in the distributed ledger or try to prevent an honest party from storing its transaction in the distributed ledger. For instance, if a set of malicious nodes claims that a legitimate transaction performed wrong computations, this transaction will not be stored in the distributed ledger [31]. To prevent this, the Blockchain employs the consensus mechanism. The examples of the consensus mechanisms employed in Blockchain are presented as follows.

The most common consensus mechanism employed in public Blockchains is the Proof-of-Work (PoW). In this mechanism, only a miner which has successfully solved a computationally laborious puzzle can append data to the Blockchain. The PoW is very effective in preventing the misbehave of a malicious party. However, it has a high computational cost — imposing a delay in the transaction execution [32]. In private Blockchains, we remark the usage of Proof-of-Authority (PoA) in order to add a block to the chain. This mechanism invokes the nodes to vote if the block can be appended to the Blockchain. If the nodes reach a consensus about the block, it will be added permanently to the Blockchain; otherwise, the block will be rejected. Thus, to employ the PoA, every node needs to be identified and authenticated.

The PoA is faster than PoW since reaching a consensus is simpler and faster than solving a cryptographic puzzle. However, PoA can be employed in a scenario where the parties have some level of trust in others, in contrast to PoW, where the parties do not need to trust others [33].

One interesting aspect of the Blockchain is the fact that we can upload applications written in a Turing complete programming language to be executed by the nodes. This application, also known as Smart contracts, allows trustworthy code execution in an untrusted environment. In other words, Smart contracts allow the nodes of a Blockchain to perform computation (or transactions) without a trusty-third party.

A smart contract can be executed when a transaction is performed in the Blockchain network. This can be compared to a stored procedure invoked by a transaction in a DBMS. It is worthwhile to mention that the smart contract execution inherits all characteristics provided by the Blockchain distributed ledger: immutable (temper evident), public available (auditable) and verified by all nodes (trustable computation) [31].

3. Problem formulation

With the basic concepts of PPRL, anonymization and Blockchain defined in Section 2, we can now formalize the problem tackled in this paper. To address the privacy-guarantees of the PPRL problem, we assume that multiple parties and an STTP engage in a PPRL process that considers an accountable security model (covert adversaries).

Problem Statement 1 (*Decentralized Execution of the Decision Model*)**.** *Given multiple participants ($p \geq 2$) with their respective anonymized data sources ($\mathbb{D}^{\tau}$), and a Semi-Trusted Third-Party (STTP) implemented as a Blockchain Smart Contract (Transaction), how to distribute the decision model execution among the parties and the STTP? Eq. (3) formalizes the decentralized execution model ($\hat{\Xi}$).*

$$\hat{\Xi} = \left(\Xi_1(e_1^{\tau}, \ldots, e_p^{\tau}) \cup \cdots \cup \Xi_p(e_1^{\tau}, \ldots, e_p^{\tau})\right) \cup \Xi_{sttp}(e_1^{\tau}, \ldots, e_p^{\tau}), \tag{3}$$

where Ξ_j and Ξ_{sttp} represent the output of the decision model generated by a party p and STTP, respectively.

Since we consider an accountable security model, our approach needs to audit all computations performed by the decision model (Ξ), regardless of who executed them (the parties or STTP).

Problem Statement 2 (*Auditability of the Decision Model*)**.** *Given the decision model execution Ξ, with p participants, the audit function (audit($\hat{\Xi}$)) should detect when the parties or the STTP tries to deviate from the protocol by sending (or computing) wrong similarity values of the entities. This misbehavior is detected by comparing the decision model output of each participant. Eq. (4) formalizes the audit function.*

$$audit(\hat{\Xi}) = \begin{cases} FP \iff \exists(\Xi_p, \Xi_{p+1}, \Xi_{sttp}) \in \hat{\Xi}, \Xi_p \\ \approx \Xi_{p+1} \approx \Xi_{sttp} \\ MB \iff \exists(\Xi_p, \Xi_{p+1}, \Xi_{sttp}) \in \hat{\Xi}, \Xi_p \\ \not\approx \Xi_{p+1} \not\approx \Xi_{sttp} \end{cases}, \tag{4}$$

where MB means that a party has misbehaved (i.e., did not follow the protocol) and FP means that all parties followed the protocol.

Knowing that a Blockchain stores all information in an unalterable and public readable database (Ψ), we also have to address the following problem.

Problem Statement 3 (*Privacy-Preserving Guarantees of the Decision Model*)**.** *Given an anonymized entity ($e^\tau \in \langle e_1^\tau, \ldots, e_p^\tau \rangle$), how to maximize the privacy capability (i.e., avoid the reidentification of entities) during the execution of the decision model in a Blockchain environment?*

Given the aforementioned problems, the goals of this paper can be summarized as:

Given : $\{\mathbb{D}_1, \ldots, \mathbb{D}_n\}, \Xi, \Psi$
Find : $M \in \Xi(\mathbb{D}_1, \ldots, \mathbb{D}_n)$
Uncover : MB
Preserving the privacy of : $e^\tau \in \mathbb{D}_i \forall i \in \{1 \cdots n\}$

A summary of the notation employed in this paper is presented in Appendix A.

4. Splitting bloom filter

As presented in Section 2, Bloom Filters may fail to preserve the privacy of the data in two cases. The first one occurs when an adversary has a list of possible values assumed by the entities [7,25–27]. The second case happens when an adversary is able to access several anonymized entities and execute a pattern mining attack [7,25]. Both vulnerabilities may endanger the privacy of the entities employed in PPRL. Considering that the entities involved in PPRL belong to a unique domain (e.g., patients with the same pathology), the participants may have a list of possible values assumed by the entities. This facilitates cryptoanalysis attacks as reported in [27,34]. Also, in order to perform the comparison step of PPRL, the parties need to share their entire anonymized entities, which favors a sophisticated cryptoanalysis attack (e.g., pattern mining attack) [7,25].

In order to mitigate the majority of the attacks to the BF and enable auditable privacy-preserving protocols to PPRL, we explore a novel aspect of the BF, named as Splitting Bloom Filter (SBF). SBF's basic idea is to perform iterative similarity computation using only a small piece (split) of the original BF (instead of the entire BF) to reduce the amount of information shared during the comparison step of PPRL. In other words, the SBF intends to split the original BF in s splits $[\phi^0, \ldots, \phi^{s-1}]$, where each split will have a fraction of the original BF length. The SBF is formalized as follows.

Definition 1 (*Splitting Bloom Filter*)**.** The SBF approach divides the l-bits of a BF into s splits ϕ. Considering that the $BF(e) = e^\tau$, we can express the anonymized entity as $e^\tau = [b_0, \ldots, b_l]$. The SBF can be formalized as:

$SBF(e^\tau, s) = [\phi^0, \ldots, \phi^{s-1}]$, such that $\phi^i = [b_j, \ldots, b_{j+(\frac{l}{s}-1)}]$ and $j = i \times \frac{l}{s}, \forall(i) | 0 \leq i \leq s-1$.

Based on the SBF, the similarity computation between two different entities, e.g., $Jaccard(e_a^\tau, e_b^\tau)$, can be executed independently by the parties. The similarity computation[2] is detailed in the Eq. (5):

$$Jaccard\_SBF(e_a^\tau, e_b^\tau, s) = \frac{1}{s} \sum_{i=0}^{s} \frac{|\phi_a^i \bigcap \phi_b^i|}{|\phi_a^i \bigcup \phi_b^i|}, \tag{5}$$

where s is the number of splits. Also, ϕ_a^i and ϕ_b^i are the ith splits of $SBF(e_a^\tau, s)$ and $SBF(e_b^\tau, s)$, respectively.

The distributed similarity computation characteristic of the BF has been exploited in previous works [15,19] to enable secure multi-party computation (SMC) in the PPRL context. However, in this work, we assume that the SBF splits (ϕ) can be employed to verify the computation performed by a PPRL participant, reducing the information available to execute attacks by malicious parties. Moreover, SBF allows the identification of malicious participants that eventually did not follow the agreed protocol and consequently enables an auditable security model in the PPRL context. Finally, the splits also reduce the amount of shared information during the linkage process, maximizing privacy-guarantees of PPRL.

In order to estimate the mean similarity between the BF splits, we use the following assumptions: (i) the SBF does not change the probability of the elements being flipped to '1'; (ii) the SBF does not alter the false-positive rate; and (iii) in the PPRL context, the false-positive rate of BF should be near 50% ($p \approx 0.5$) to reduce the probability of two different elements being mapped to the same BF position [30].

Considering the previous statements, it is unlikely that every split of a BF is identical in terms of number and position of '1'; in fact, it is expected that every split of BF, with a $p \approx 0.5$, presents variations in quantity and positions of '1'. This fact can be partially explained by the distribution of '1' in the original BF. In Appendix B, we formally acknowledge the existence of the '1' variation in the SBF splits.

Acknowledging that the splits of SBF are different, we also assume that the similarity of the splits is slightly different from the complete BF similarity, i.e., $Jaccard(e_a^\tau, e_b^\tau) \neq Jaccard(\phi_a^i, \phi_b^i)$, such that $\phi_a^i \in SBF(e_a^\tau, s)$ and $\phi_b^i \in SBF(e_b^\tau, s)$. The rationale of this assumption is justified in Appendix B. Therefore, the similarity comparison between any pair of splits is related to the similarity between the BFs plus a certain *error* (ϵ) and can be represented by Eq. (6).

$$Jaccard(e_a^\tau, e_b^\tau) = Jaccard(\phi_a^i, \phi_b^i) + \epsilon, \tag{6}$$

such that $0 \leq \epsilon \leq 1$.

In order to illustrate the similarity difference between the split and complete BF, Fig. 2 depicts this difference using the words ANA and ANE. In Fig. 2(a), the similarity of the words produced by a regular BF is 0.66. In Fig. 2(b), we apply SBF to divide the original BF into two splits. Notice that, for each split, we introduce an error (ϵ) of 0.14 and 0.16 during the comparison (Eq. (6)), respectively.

It is worthwhile to mention that the relation between the number of splits and the error is examined through experiments in Section 7.1. The experiments demonstrate that the number of splits is important to predict the *error*. Moreover, we prove that the error is crucial to determine the privacy and the quality of the PPRL linkage.

[2] Other similarity functions available for regular BF can also be used in the SBF approach, e.g., token distance.

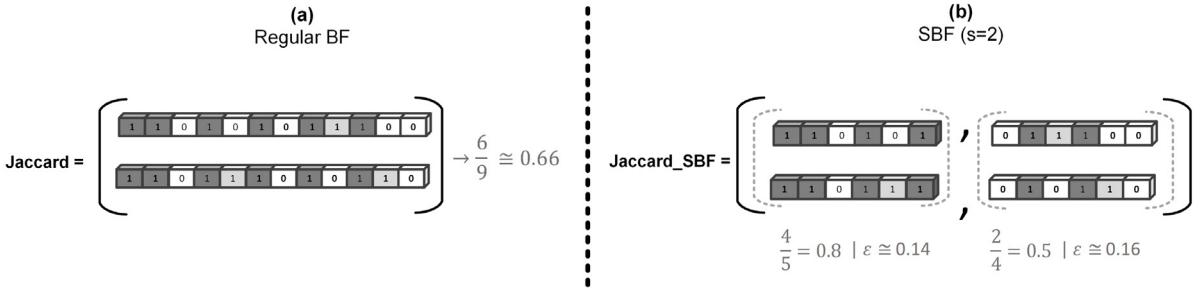

Fig. 2. An example of SBF using two splits, (a) illustrates the similarity calculation over anonymized values of the words ANA and ANE (Fig. 1). In (b), the BFs are split in two. The similarities between the splits are presented as well as the resulting *error*.

5. An auditable protocol for PPRL

In this section, we present a novel privacy-preserving linkage protocol, which considers covert adversaries to audit the comparison and classification steps of PPRL. This protocol uses SBF to reduce the amount of information shared during the PPRL execution. Moreover, this protocol uses the decentralized SBF characteristic (presented in Eq. (5)) to provide audibility in the similarity computations performed by the PPRL parties. The protocol also considers a semi-trusted third party (STTP), where the parties may audit all their actions.

Definition 2 (*Semi-trusted Third Party*)**.** The STTP is a participant that is partially trusted to perform a specific computation. Although STTP may misbehave, it will not conspire with none of the parties. In other words, it can deviate from the protocol by:

1. Leaking the information sent by the participants; and/or
2. Executing wrong computations over the information shared by the participants.

The intuition behind the protocol is to perform the comparison step in two stages in order to improve the privacy-preserving capabilities of the protocol. In the first stage, STTP performs the similarity computation according to Eq. (6) using only one split of the original BF. In this stage, the single split's similarity is used to eliminate entity pairs that present similarity values that differ from the designated threshold, β-threshold. In summary, the first stage selects entities that present a high probability of being similar using a small fraction of the parties' anonymized data source. As a result, a list of entities is disclosed to the other participants.

In the second stage of the protocol, the parties perform similarity computations (interactively) in the remaining splits of the entities disclosed by STTP. To audit the protocol execution, the parties may compare the similarity calculated in the second stage with the similarity exposed by STTP. If the difference between these similarities is higher than *error*, it means that one party (or the STTP) did not follow the protocol. The protocol is detailed in Protocol 1.

In the Setup, the parties agree among themselves on the anonymization parameters (step 1), execute the data anonymization (step 2) and send the splits to STTP (step 3). Notice that STTP is not aware of the data anonymization parameters or which split was chosen to be used. In other words, STTP receives only a small part of the original BF, which makes it difficult to execute cryptoanalysis attacks [7,25]. Two important parameters of our protocol are chosen during Setup: the β and α thresholds. These thresholds are employed in Stage 1 and 2, where β is employed in both stages, and α is employed in the final step of Stage 2. These thresholds must consider the SBF error and should be calculated as $\beta = \alpha - error$.

In Stage 1, STTP computes the similarity of all pairs of entities received during the Setup, using only one split to represent each entity. At the end of the stage, STTP publishes a list (ζ) of all entities pairs with a similarity value greater than β. The β threshold must be chosen carefully to reduce the amount of information shared in the subsequent stages. In other words, by choosing a lower value of β, the number of entities forwarded to Stage 2 is increased, which enhances the probability of successful cryptoanalysis attacks and the computation cost of the protocol. The reduction of shared information and computational costs are detailed in Section 7.

In Stage 2, the parties distribute, among each other, the remaining splits to perform the similarity computation of the entities disclosed in ζ. Each party is responsible for performing the similarity computation in φ splits, where $|\varphi| = \frac{s-1}{|P|}$, using Eq. (5) for each pair of entities (step 6a). In other words, the splits of the entities (such that $e^{\tau} \in \zeta$) are distributed among p participants.

Afterwards, in step 6a, the similarity computation is performed over the splits (φ) of the entities. In step 6ai, each party compares the similarity value (calculated in step 6a) to the value stored in ζ (step 4). Then, if the similarity value calculated in step 6a is different from the similarity present in ζ plus *error*, the party assumes that a malicious party did not follow the agreed protocol and aborts the protocol execution.

The entities' overall similarity (Eq. (5)) is interactively calculated by the parties in step 6b. The parties check the difference between the exchanged similarity value against the calculated similarity, and the value stored in ζ. If the difference is higher than *error*, a party aborts the protocol execution (step 6bi) because a malicious party seems to have sent bogus similarity values. Finally, the parties learn which of their entities exist in all datasets (step 7).

Notice that our protocol is intended to execute the Comparison and Classification steps of PPRL in a covert adversary model. In this sense, the SBF splits provide auditability in the comparison step of PPRL. It is worth mentioning that: (i) no participant has access to the complete anonymized representation of an entity; and (ii) the similarity computation is performed using entities (splits) that have a high probability of being similar. In other words, the protocol also reduces the amount of information shared by participants, decreasing the probability of a successful cryptoanalysis attack.

6. Auditable blockchain-based PPRL

In this section, we describe the **A**uditable **B**lockchain-based Privacy-preserving R**e**cord **L**inkage (ABEL), a Proof-of-Concept implementation of the 3PAC protocol presented in Section 5.

Different from the traditional HBC protocols, which consider an external (neutral) party in which all participants trust [4,14,35], ABEL considers a semi-trusted third party (STTP), which parties do not fully trust. Because they do not rely on STTP,

Protocol 1. 3-Party Comparison and Classification Auditable Protocol (3PAC)

Setup

1. The parties agree on the BF and SBF parameters; the number of splits (s), the splits that will be employed in the first stage (ϕ_i), the *error* and the thresholds α and β;
2. The parties anonymize the entities and randomly generate a unique ID for each entity;
3. The parties send the SBF splits to STTP $send(\phi_i, STTP) \mid \forall SBF(e^r, s) \in \mathbb{D}_p^r$.

Stage 01

4. STTP computes the similarities between the splits;
5. STTP reveals (to all parties) a list (ζ) comprising the IDs and the similarity values between the entities.

Stage 02

6. For each split (s) of the entities stored in ζ, the parties alternately exchange among themselves the splits, one at time. At the end, each participant receives φ splits, where $|\varphi| = \frac{s-1}{|P|}$;
   (a) The splits sent by the parties are employed as input to calculate the similarity of the entities using Eq. (5);
      i. If the difference between this similarity and the value calculated by STTP is greater than the *error*, the participant detects the misbehave and aborts the protocol execution.
   (b) The parties exchange the splits similarity calculated in the previous step in order to update the entities' overall similarity;
      i. The parties check if the difference between exchanged similarity and the value stored in ζ. If this difference if higher than *error*, the parties detect a misbehave and abort the protocol execution;
      ii. At the end of this step, the parties update the entities' similarity value in ζ.
7. Finally, the parties select entities that have a similarity value higher than α-threshold.

the parties need to audit STTP computations. In order to enable the auditability of the computations performed by the STTP, we implement it as a Smart Contract hosted in a Blockchain. In other words, the use of a Smart Contract transforms the STTP into a piece of code that executes in a Blockchain environment.

By considering the STTP as a Smart Contract, we offer three important characteristics inherited from the Blockchain. The first is the tamper-evident characteristic of the STTP. In other words, once that the STTP Smart Contract is deployed to the Blockchain, it cannot be modified [36]; in other terms, a malicious party unlikely will change the Smart Contract code.

The second characteristic is the decentralized execution model. This characteristic enables the STTP to be executed in different machines eliminating the need for a centralized computing environment. In other words, the PPRL parties could create a private network with its own machines, and the computations will be executed in all machines of the network, without a central server.

The final characteristic is the auditability (transparency) of the Blockchain. Once that STTP is stored in the Blockchain, the STTP code, inputs, and outputs can be read (audited) by the Blockchain members. This advantage provides the ABEL auditability capability while posing as a privacy disadvantage by making available private data to every Blockchain member. As presented in Section 2, every Smart Contract invocation (transaction) is stored in the Blockchain with all inputs and outputs. In other words, by default, all transaction attributes (input and outputs) are readable by the Blockchain members. It is worth mentioning the existence of Blockchain technologies that keep the transaction attributes private, such as the Microsoft Confidential Consortium Framework.[3]

The aforementioned disadvantage can be minimized by building a private Blockchain network amongst the PPRL parties and using an adequated SBF configuration during the ABEL execution. The use of a private Blockchain network limits the access to the shared data only to the PPRL parties and hide the data from the external malicious parties. In order to preserve the entities' privacy from an adversary among the PPRL parties, we employ the SBF and send a split configuration (number and length of the splits) that diminishes the chance of success of attacks performed by a malicious party. Furthermore, the computations performed at 3PAC Stage 2 are executed by the parties outside of the Blockchain, hiding the data from the PPRL parties. In Stage 2, the Blockchain is employed by the parties only to verify the computations and update the similarity value stored in ζ.

Another drawback of the Blockchain usage happens if one adversary controls at least 51% of the nodes of a Blockchain network. This adversary can take control over the Blockchain and change the ledger's states [32,33,36], i.e., switch the output of a transaction and modify the values stored in ζ. However, we will not address these attacks in this paper. For further details, we refer the reader to the related work [35]. It is worthwhile to mention that our research assumes that any Blockchain owner has control of 51% of the Blockchain nodes. For instance, if two parties want to execute ABEL, they will set up a private blockchain network where each one will have one node. Thus, each party will control 50% of the blockchain network.

Fig. 3 illustrates the ABEL execution of two participants (Alice and Bob). In this example, we assume that the parties have already deployed the STTP (Smart Contract) in the Blockchain, agreed on optimal anonymization parameters, executed all entities' anonymization in their datasets, and agreed on the number of splits ($s = 3$) and the thresholds ($\alpha = 0.7$ and $\beta = 0.4$). In summary, Alice and Bob have executed steps 1 to 3 of the 3PAC protocol. The example starts in step 4 of 3PAC. Steps 1a and 1b of Fig. 3 illustrate the parties sending the splits to STTP.

The similarity computation is performed in all splits (Cartesian product) received by STTP. If the calculated similarity value is higher than threshold β, STTP saves the IDs of the splits as well as their similarity values in the Blockchain (step 6 of 3PAC). In our example, the restriction of Stage 1 is satisfied only by the splits A1 and B9 ($similarity(A1, B9) > \beta$); this fact is illustrated in steps 2 and 3 of Fig. 3.

[3] https://microsoft.github.io/CCF/.

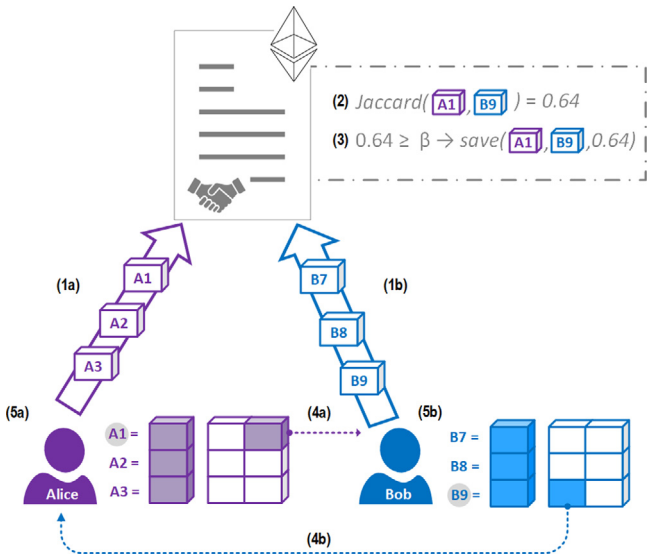

Fig. 3. Outline of ABEL execution.

It is worthwhile to mention that Stage 1 acts as a filtering process, removing from the subsequent steps entities pairs that present values smaller than the β threshold. In other words, Stage 1 reduces the number of entities (information) shared during the ABEL execution, and this reduction has an impact on the comparison reduction rate (detailed in Section 7.1) as well as on the privacy of our approach. For instance, notice that the ζ list was assembled using a minimal amount of information (i.e., only one split of each entity), reducing the success rate of cryptoanalysis attacks; the privacy impact is presented in Section 7.

Steps 4a and 4b of Fig. 3 illustrate the parties exchanging, among each other, the remaining splits of entities stored in the ζ list. Notice that, in our example, entities A1 and B9 presented a similarity value greater than β. Therefore, only A1 and B9 splits will be exchanged by the parties. Also notice that, by alternating the distribution of the split, Alice and Bob only have access to two splits as well as their similarity values, making it challenging to successfully execute a cryptoanalysis attack.

For each split, ABEL alternately distributed among the parties the splits of entities stored in ζ. In our example, Alice sent the third split of A1 to Bob, and Bob sent the second split of B9 to Alice. Once the splits are exchanged, the parties calculate the splits similarity, exchange the calculated similarity among themselves, audit the computation performed by other parties (steps 6.a and 6.b.i of 3PAC), and update the similarity value in ζ (step 6.b.ii of 3PAC).

The final stage is illustrated in Figs. 3, 5a, and 5b, where the duplicate entities are marked (with a gray circle). Thus, at the end of the process, a party only learns which of its entities are present in the other parties' datasets. A more in-depth discussion of the privacy attacks performed during the ABEL execution is presented in Sections 7 and 6.1.

6.1. Privacy sketch

This section evaluates the privacy capabilities of our approach in an auditable security model using the simulation paradigm. The paradigm assesses the approach through the messages (information) exchanged by the parties and STTP during a simulated PPRL execution. The simulation was executed according to the guidelines proposed by Lindell et al. [37]: the messages granted in the simulation must be the same as an adversary would have access to use in a real attack. The simulation fails if an adversary learns anything different from the expected output.

Considering that STTP is a Smart Contract, we can assume that the STTP code (protocol) can be read and audited by the parties. Moreover, after STTP is deployed in a Blockchain (i) the code cannot be changed (tamper evident); (ii) STTP cannot deviate from the protocol (collude); and (iii) all computations (Transactions) performed by STTP will be stored in the Blockchain. We also consider that each party performs the anonymization of its respective data source. Thus, in order to provide a better explanation, the simulation paradigm was executed over Stages 1 and 2 of the 3PAC Protocol.

Before we detail the action performed in Stage 1, we present the inputs used in this stage: one split (ϕ_1) of each entity stored in the parties' data source ($\mathbb{D}_p^\tau$) and the β threshold. To the best of our knowledge, the β threshold is neither susceptible to any vulnerability nor privacy attack.

[3PAC] Stage 1 simulation

Inputs. $\phi_i \forall i \in \mathbb{D}_p^\tau$, β

1. *Input Validation*. STTP validates the parties' inputs.
2. *Single Split Comparison & Classification*. STTP performs the similarity computations and publishes (steps 4 and 5) the entities pairs that have a high probability of being similar in ζ.

Output. ζ

STTP should implement a method to validate the splits (Input validation). For instance, the validation method verifies if all parties sent their splits, or if all splits have different values. Thus, if one (or more) party tries to deviate from the protocol by sending wrong inputs, STTP detects the misbehave and may abort the protocol execution. After the Input validation, STTP engages in the classification of the entities (Single Split Similarity Computation) and stores the ζ list (the output) in the Blockchain.

The ζ list contains pairs of entity ids with their similarity values (Eq. (6)), such that these values are higher than the β threshold. Thus, since the single splits (ϕ_1) were the inputs of a Blockchain transaction (STTP execution), all the ϕ_1 are available in the Blockchain. Notice that, until this point, an adversary has access to one split of each entity, with its estimated similarity.

Regarding the privacy issues related to the usage of single splits (ϕ_1) as input to a privacy attack by a malicious party, as demonstrated in Section 7.1.2, it is a hard task to re-identify an entity using information from a set of single splits.

The information reduction (reduction rate, detailed in Section 7.1) provided by Stage 1 also helps to improve the privacy capabilities of the SBF. Due to the reduction of the number of entities employed in Stage 2, where only the entities stored in ζ will be utilized as Stage 2 inputs, the chance of successfully executing a frequency or cryptoanalysis attacks decreases.

The input of Stage 2 (i.e., α, β, and ϕ_1) is the same for every party. However, each party receives a different split set φ, at each iteration. Fig. 4 illustrates the split exchange at each iteration. For instance, after the parties executed Stage 1, Charlie shared ϕ_4, with splits of entities C7 and C8 in the first iterations. Due to the filtering processing (i.e., similarity smaller than β threshold), Charlie excludes entity C8 in iteration 2. In the split exchange process, the parties alternate the splits in order to reduce the amount of shared information and distribute the split computations.

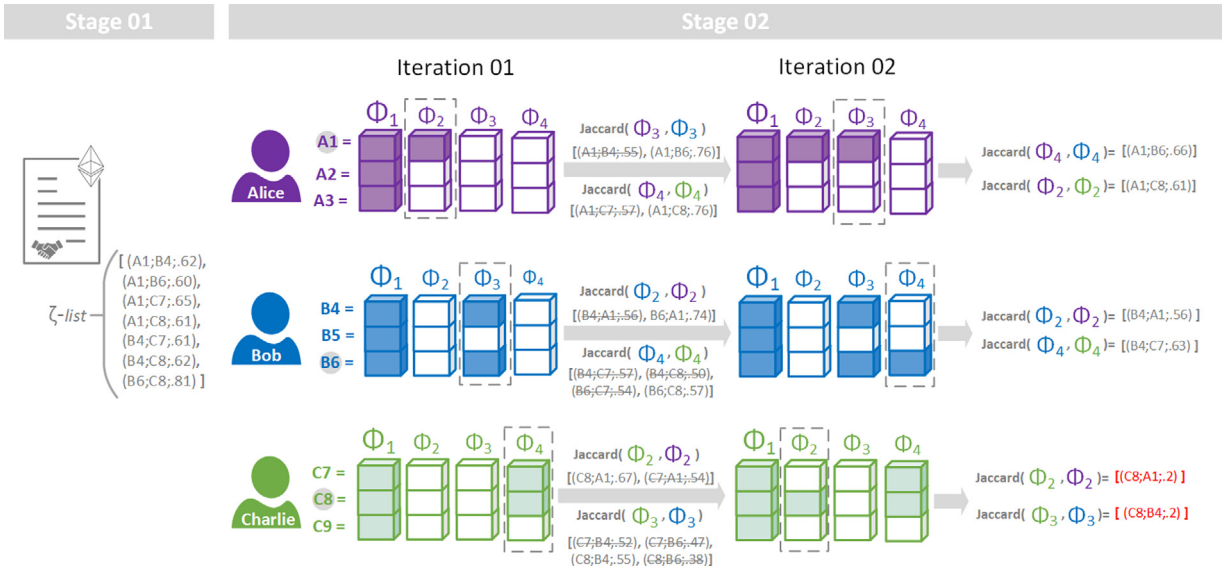

Fig. 4. Simulation of the ABEL execution. The parties' exchanged messages (i.e., splits and similarity values of entity pairs) are designated at each iteration. The misbehavior (attack) of Charlie is illustrated in the final iteration and highlighted in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

[3PAC] Stage 2 simulation

Inputs (for each party) . ϕ_1, φ_p , β, α , ζ
For each $\varrho \in \zeta \cap \mathbb{D}_p$ (step 6 of 3PAC):
3. Exchange the φ_p splits, such that $\varphi_p \in \varrho$.
4. Compute the similarity of φ_p (step 6a).
5. Exchange and audit the calculated similarity in the previous step (step 6b of 3PAC)).
6. Conjointly classify the entities listed in ζ (step 7 of 3PAC).

Output. Each party learns which of its entities is also present in other parties' data sources.

After exchanging the first split, the parties compute the similarity of the received splits' against their owns splits (step 6a of 3PAC). Then, the parties exchange among themselves the calculated split similarity values and compute the entities' overall similarity (step 6b). Notice that, at this point, an adversary may try to force the honest parties to send more information (splits), or hide information, by executing the following attacks:

- Send bogus split (e.g., with random data);
- Send wrong similarity values.

The honest parties could detect both attacks by auditing the other parties' inputs. If some adversary attempts to deviate from the protocol, it will be detected with a high probability — the misbehave identification is detailed as follows.

In order to detect the bogus split sent by an adversary, the honest parties will compare the similarity computed by themselves with the value stored in ζ. If the computed value diverges from ζ (considering the *error*), an honest party will detect the misbehave and abort the protocol execution. In other words, if an adversary sends a bogus split, e.g, a split with zero in every position, in Stage 2 step 6b, the parties will calculate the similarity and compare the similarity value against the value stored in ζ. Thus, if the calculated similarity presents a value different than the expected SBF error plus the value stored in ζ, the party will detect the adversary misbehave.

Regarding the second attack, where an adversary sends wrong similarity values, the honest parties will confront the wrong similarity values with their calculated split similarity and the value stored in ζ. And abort if the exchanged similarity diverges from the calculated values or the value stored in ζ.

Fig. 4 illustrates an attack in iteration 2 when Charlie tries to hide entity C8 by sending a lower similarity value. The similarity value sent by Charlie (0.02) is not tolerated because the overall similarity is 0.65 and the similarity of ϕ_3 is 0.61. Thus, the other parties identify the attack and abort the protocol execution.

Regarding the usage of the exchanged splits in the re-identification attack, we need to remark that our approach performs an iterative filtering process where, at each iteration, fewer splits are shared amongst the parties. In other words, only entities that will be marked as a match are shared in the last iterations, making it hard to successfully employ re-identification attacks.

In the attack illustrated in Fig. 4, Charlie deviates from the protocol in the final iteration. Otherwise stated, Charlie could have access to the maximum number of splits as possible, meaning that the attack was executed in the worst scenario. However, even if the parties collude, Charlie has access only to the colored split designated in iteration 2, reducing the chance of success during a re-identification attack.

It is worthwhile to mention that the parties could collude and share the split among each other to gather more information (splits) for a re-identification attack. In other words, the collusion among multiple parties could reveal more information to be employed in an attack, increasing the chance of success in a re-identification attack. However, the consequences of collusion among the parties could be minimized if the SBF parameters are properly chosen. Furthermore, we demonstrate in Section 7.2.4 that our approach could preserve the privacy of entities against the BF state-of-the-art privacy attacks until a certain limit. We also remark that collusion is considered in a malicious adversary model, and our approach considers a covert adversary model.

7. Evaluation

The main goal of this section is to evaluate the efficacy (quality), efficiency, and the privacy capabilities introduced by our contributions, the SBF and ABEL approaches. To this end, we present an empirical and theoretical evaluation of them.

Table 1
SBF experimented data sources characteristics and anonymization parameters.

Domain	|A|	|B|	Attributes	l	k
Bike	2689	3721	3	920	7
Beer	4330	2999	3	920	7
Books	1190	1106	4	2456	8
Electronics	2769	4007	4	9816	10
Movies	7029	6063	4	7360	9
Music	20,062	3805	7	9816	10
Restaurants	4655	8400	5	1224	8

7.1. SBF evaluation

To assess the accuracy of the similarity computation and investigate the privacy capabilities provided by SBF, we need to answer the following Research Questions (RQs):

- RQ.1 Does the similarity between two entities can be accurately calculated by SBF, Eq. (5)?
- RQ.2 What is the influence of split length (number of bits) over the similarity calculated using the SBF (Eq. (6))?
- RQ.3 Does SBF improve the privacy capabilities of PPRL processes?

To answer these questions, we employed seven pairs of real-world data sources[4] from the following domains: bikes, beers, books, electronic goods, movies, music, and restaurants. Each pair of data sources presents different characteristics, such as the number, values, length, and the level of dirtiness of attributes. A summary of the data sources characteristics is shown in Table 1.

Table 1 also presents the anonymization parameters employed in the BFs: l (number of bits of the filter) and k (number of hash functions). These parameter values were chosen to achieve an optimal privacy ($p = 0.5$) of the filters [38].

7.1.1. Accuracy of the similarity computation

To evaluate the accuracy of the similarity calculation produced by SBF, we employ three different BF strategies [13,38,39]: (i) Balanced Bloom Filter (BBF); (ii) Xor-Folding Bloom Filter (XBF); and (iii) Bloom-and-Fllp (BLIP). We vary the BF implementation to ensure that SBF provides accurate results using a regular implementation (BBF) or privacy-enhancing implementations such as BLIP and XBF.

BBF relies on the choice of the parameters in order to achieve a uniform distribution of elements in the filters and, consequently, maximizes privacy [13]. To prevent bit pattern attacks, XBF folds (in half) the bit array, and employs the XOR logical operator in each half of the filter in order to assemble a new filter [39]. In turn, BLIP [38] applies the concept of randomized responses in order to flip the values of random bit positions and, consequently, reduce the probability of entity re-identification during the PPRL.

The *error* produced by SBF (tackled by RQ 1 and RQ 2) is illustrated in Fig. 5. This *error* was calculated as the difference between the modulus of the similarity produced by the regular BF (Jaccard) and SBF (Eqs. (5) and (6)). The horizontal and vertical axis show the number of splits and the error of the similarity computation, respectively. The dashed lines indicate the results achieved by SBF using only one split (Eq. (6)), while the continuous line shows the use of all SBF splits in the similarity computation (Eq. (5)).

As illustrated in Fig. 5, regardless of the BF anonymization strategy, the similarity calculation produced by Eq. (5) presents the same similarity values as the traditional BF. In other words, SBF is able to calculate the similarity of two entities accurately when all splits are employed, answering RQ. 1.

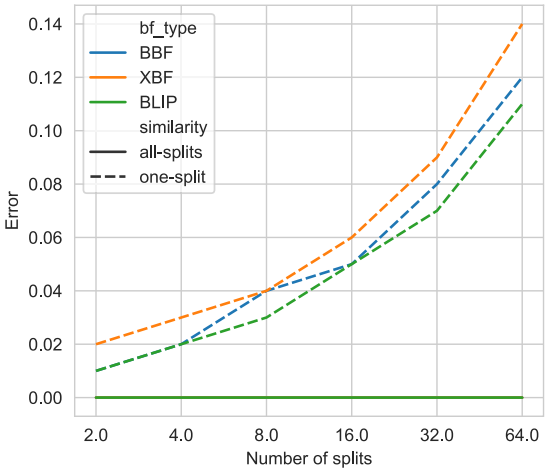

Fig. 5. Error obtained using different BF hardening strategies.

Fig. 5 also exposes that the error (dashed lines) of utilizing a single split for calculating the similarity of two entities increases as the number of splits grows. In other words, the error grows as we increase the number of splits, regardless of the BF hardening strategies employed. The SBF error disagreements of the BF hardening strategies are not statistically different from the mean error of the strategies examined using the Wilcox test with 95% confidence.

According to Fig. 5, it is possible to observe by the dotted lines a relation between the *error* (presented in Eq. (6)) and the number of splits. In order to confirm this relation, we applied a Statistical correlation test (Pearson R test) over the calculated similarities. We refuted the null hypothesis with a correlation factor of 0.82 and $p\text{-}value < 0.005$. In other words, we confirm that there is a correlation between the number of splits and the *error* presented in Eq. (6).

For a better understanding of the relation between the number of splits and the *error*, we plotted Fig. 6 considering the BBF strategy. The horizontal axis illustrates the length of the split divided by the original filter length (l/s), — e.g., if a BF originally has 1024 bits and we split this BF into four parts, each split will represent 25% of the original split. The vertical axis describes the error of comparing two entities using only one split. The colored lines exhibit the mean error produced by the use of the individual splits for each data source.

Fig. 6 shows an increase of the error while we decrease the split length. The raise of the error can be explained by the fact that, in smaller splits, every bit that diverges in Eq. (6) has a higher impact over the calculated similarity.

As shown in Fig. 6, a logarithmic function describes the error. To confirm the previous statement, we build a logarithmic regression model ($f(x) = a * ln(b * x) + c$) and use the result of the experiments to train and test (with cross-validation) this model. Fig. 7 illustrates the fitting of the model for each data source.

Fig. 7 presents the same axis of Fig. 6. The line represents the model output, and red x mark represents the model error. As illustrated in Fig. 7 and confirmed by the Root Mean Square Error (RMSE) result (rmse = 0.027), we demonstrate that a logarithmic function fits well with the error growth in our experiments. This is relevant because (i) we provide an estimation of the impact of the split length in Eq. (6) error (RQ. 2), and (ii) the fact that a logarithmic function can describe the error is relevant to some privacy aspects, detailed in Section 7.1.2.

[4] Available at https://sites.google.com/site/anhaidgroup/useful-stuff/data.

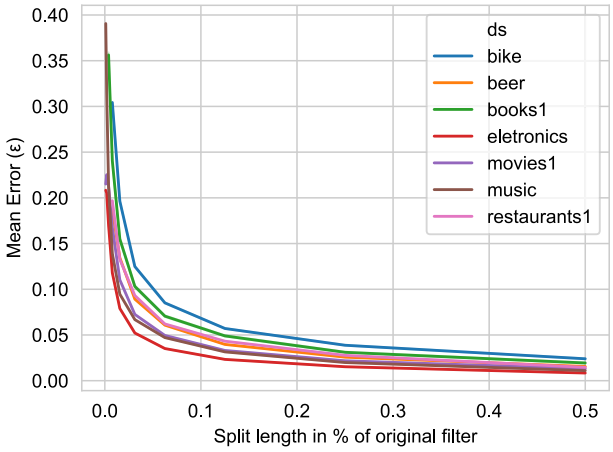

Fig. 6. Error associated to split length (percentage) by data source, employing the BBF. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

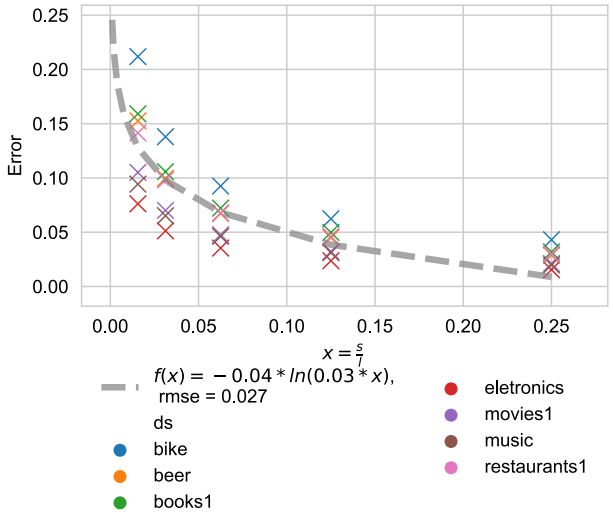

Fig. 7. Accuracy of the estimated error in SBF. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7.1.2. Privacy

In order to investigate the privacy capabilities provided by SBF (RQ. 3), we assume that an attacker needs to have access to a significant amount of information to execute a successful privacy attack [7,25]. Such attacks are more likely to succeed if the n-grams (n) are hashed into the shared splits in SBF. It is possible to execute frequency-based and cryptoanalysis attack successfully in one split. However, if we consider the usage of a small split in SBF (e.g., split with 16 or 32 bits), such attacks need to possess more information than the regular BF to re-identify a single split's information based on few bits of information. In other words, we believe that the re-identification attack is more prone to succeed when the k parts of the n-gram are stored in the shared split. Thus, we estimate the SBF's privacy by the probability of n-grams being mapped in the SBF split.

We can estimated the probability of an individual position of the filter (b_j) be used to map a particular n-gram by $P(b_j) = 1 - e^{\left(-\frac{k \times n}{l}\right)}$ [30]. Consider that SBF divides the BF in s splits, where each split has $\frac{l}{s}$ bits, and each n-gram of the plain data is mapped to k positions by different hash functions. Eq. (7) estimates the likelihood of an n-gram (ng) being mapped into a split $P(ng \in \phi)$ by calculating the probability of k bits of the splits be used to represent the n-gram.

$$P(ng \in \phi) = P(b_1 \in \phi) \cap \cdots \cap P(b_k \in \phi) \tag{7}$$

Knowing that the BF hash function provides a uniform random distribution of the n-grams inside the filter [13,30], we can assume that $P(b_k \in \phi) = P(b_j)$. Thus, it is possible to modify Eq. (7) as follows.

$$\begin{aligned}
&P(ng \in \phi) = P(b_1 \in \phi) \cap \cdots \cap P(b_k \in \phi) \\
&\textit{considering that}, P(b_k \in \phi) = P(b_j), \ \textit{and} \ P(b_j) = P(b_{j+1}) \\
&P(ng \in \phi) = P(b_1) \cap \cdots \cap P(b_k) \\
&\Rightarrow P(ng \in \phi) = P(b_1) \times \cdots \times P(b_k) \\
&\Rightarrow P(ng \in \phi) = (P(b_k))^k \\
&\Rightarrow P(ng \in \phi) = \left(1 - e^{-\frac{k \times n}{l}}\right)^k \\
&\textit{considering that we are calculating the probability over} \\
&\textit{the split } \phi, \textit{ l represents the split length, such that } l = \frac{l}{s} \\
&\Rightarrow P(ng \in \phi) = \left(1 - e^{-\frac{s \times k \times n}{l}}\right)^k \quad \blacksquare
\end{aligned} \tag{8}$$

It is worthwhile to mention that the hash functions used (k), number of n-grams (n), and the length of the regular Bloom Filter (l) provide a trade-off between the linkage quality and privacy [30]. This trade-off is given by the false-positive rate ($fpr = 1 - e^{-\frac{k \times n}{l}}$) of the BF. Therefore, the higher the value for fpr, the higher the privacy and the lower the quality of linkage, because the number of q-grams mapped to a single bit (and therefore the number of resulting collisions) rises, which leads to lower linkage quality but makes it more challenging for an adversary to learn the q-gram combinations in BF [40]. Thus, a value used to provide a suitable comprising between quality and privacy is $fpr = .5$ [1,13,40].

In order to investigate the influence of the SBF parameters over the privacy guarantees, we plotted the graph depicted in Fig. 8 varying the false-positive rate and the split length from 50% to less than 0.01% of the original BF ($0 < fpr < 1$). The original BF's false-positive rate is depicted by the lines, while the BF length (in percentual of the original length) is illustrated in the horizontal axis. The $P(ng \in \phi)$ is depicted in the vertical axis.

Fig. 8 shows the influence of the BF false-positive rate and the split configuration over the SBF privacy. Before explaining the result, we have to remember that the false-positive rate is proportional to the number of '1' in the BF. Lower f implies in a low populated filter while higher fpr results in a filter with more '1', due to the relation between the number of k and the l.

Another remark about the splits is that each split can store $\frac{l}{sk}$ n-grams. Thus, we can intuitively state that smaller splits increase SBF's privacy guarantees, illustrated in Fig. 8 and can be calculated by Eq (8).

To investigate if these properties are enough to keep the entities' privacy during a real-world PPRL process, we employ a state-of-the-art Bloom Filter cryptoanalysis attack [7], presented in the related work section, against the NCVR dataset[5] with 224 thousand and 6 millions records. In this experiment, we employ the most used configuration of the BF [13], when $fpr = .5$, over five attributes (firstname, lastname, age, gender, and city) of the NCVR dataset. The attacks were executed over the SBF split, varying l/s from 0.01 to 0.25.

It is worth mentioning that the attacks were unable to re-identify any entities in the experimented data source. This result

[5] Available at the authors website: https://dmm.anu.edu.au/pprlattack/.

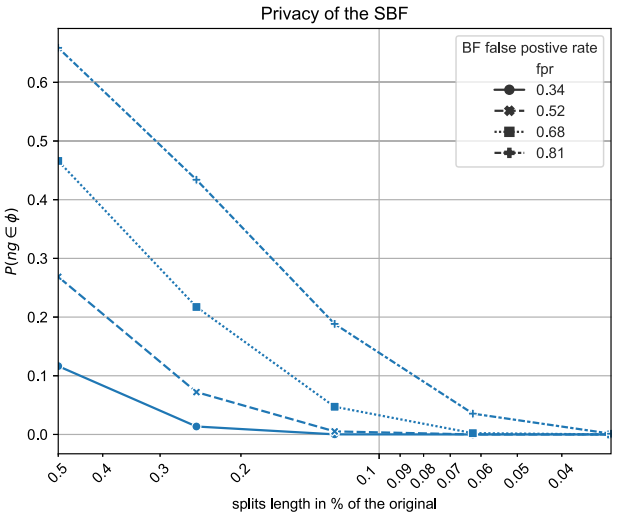

Fig. 8. Probability of an n-gram being stored in a split considering different BF parameters.

can be explained by the fact that most BF attacks need to have access to a considerable amount of information to re-identify the anonymized entities [1,3,7,25,29]. Therefore, since we share only a fraction of the original data, making it hard for state-of-the-art cryptoanalysis attacks to re-identify the entities using their strategies (e.g., pattern mining and dictionary attacks).

7.1.3. Summary

The experiments presented in this section demonstrate that SBF provides: (i) an accurate calculation of the overall similarity between two entities using all splits of a BF (including BFs with hardening techniques); and (ii) the similarity of the splits (plus an error) is related to the overall similarity. These two facts are important to configure the thresholds α (overall similarity) and β (split similarity) of the 3PAC protocol. The parties can choose the threshold α as usual, i.e., without considering any errors. However, the error introduced by the split length must be considered in order to configure threshold β.

We also evaluate the privacy guarantees provided by SBF and conclude that our approach is resilient enough to preserve the privacy of entities against state-of-the-art privacy attacks, considering that the appropriate parameters and BF techniques are chosen. Moreover, we demonstrate that the SBF optimal parameter is aligned with the BF.

7.2. Evaluation of ABEL

In this section, we evaluate the ABEL approach and present a discussion about the experimental results. The main goal of the evaluation is to assess the effectiveness, efficiency, and privacy capabilities of ABEL. To this end, we formulate the following research questions:

- RQ.4 Does the ABEL approach provide an effective linkage of entities?
- RQ.5 What is the computational cost of ABEL?
- RQ.6 Is the ABEL approach capable of preserving the privacy of entities considering a covert-security model and state-of-the-art attacks?

To answer these questions, we use real-world personal data sources detailed in Table 2, which are widely used by the research community.

Table 2
ABEL experimented data sources.

Data source	|A| x |B|	Attributes	l	k
ncvr	2,890,000	8	1144	7
mvr	2,250,000	9	1008	5
yv-er	90,231,001	6	976	7

Considering the relevance of personal data sources, we selected three data sources from distinct countries with different linguistic and structural characteristics to provide an extensive evaluation of our approach. The YV-ER[6] data source contains information about Italian Holocaust victims, while NCVR and MVR comprise information about voter registration from North Carolina and Michigan, respectively. We used the methodology presented in [7] to build the NCVR and MVR data sources. The former contains data collected from October 2016 and June 2020 while the latter includes data obtained in September 2014 and March 2017.

In our experiments, we compare ABEL against the regular BF representation of the entities. We also test our approach using Standard blocking [41] as a blocking stage of the PPRL. Thus, the comparison stage calculates the similarity for all entity pairs marked by the blocking stage. We also consider that the regular BF will be handled by a trusty third party.

7.2.1. Effectiveness

To evaluate the effectiveness of our approach, we compare the results of the SBF execution, varying the number of parties as well as the α and β thresholds against the competitor (regular BF). Briefly, we compare our approach (which considers a covert adversary) against an approach that considers an HBC security model. In this experiment, we vary the split length as well as the α and β thresholds.

Three well-known quality measures (Precision, Recall, and F-measure) from Information Retrieval are employed to evaluate the effectiveness of our approach. These measures are defined as $Precision = \frac{|\Gamma_e \cap \Gamma_r|}{|\Gamma_r|}$, $Recall = \frac{|\Gamma_e \cap \Gamma_r|}{|\Gamma_e|}$, and $F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$, where Γ_e represents the set of all existing duplicated entities between the data sources (true matches), and Γ_r is the set of entities identified as duplicate by an approach. Intuitively, low precision indicates that false correspondences were made, while low recall implies that the approach missed true matches. In turn, F-measure ($F1$) is the harmonic mean between recall and precision.

We also use the Reduction Ratio (RR) to estimates the relative decreasing in the number of comparisons carried by our approach. The RR is formally defined as $RR = 1 - \frac{|D'|}{|D|}$, where $|D|$ is the number of comparisons needed to execute the linkage of the datasets and $|D'|$ is the number of comparisons performed by our approach.

Fig. 9 illustrates the comparison between the results achieved by the competitor and ABEL. The vertical axis represents the quality measure results, while the horizontal axis refers to the α threshold. Different colors represent each data source, while the colored dotted line delineates the competitor results, and the colored bars represent ABEL's results.

The results produced by ABEL are similar to the ones of the BF. However, the F1 results also show that BF overcomes SBF for some threshold values, for instance, $\alpha = 0.9$ for the ncvr and mvr data sources. Two factors can explain this fact: (i) the precision and recall results; and (ii) the fact that we consider a combination of different α (from 0.1 to 0.9 threshold) and β (from

[6] Available at https://github.com/tomersagi/yv-er.

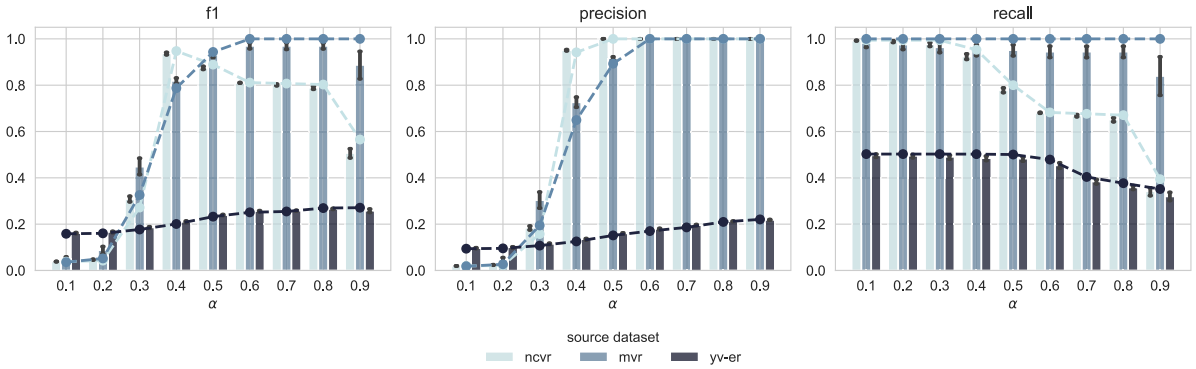

Fig. 9. Effectiveness results of ABEL for different α values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

0.01 to 0.4) thresholds to build the figure; in other words, we use non-optimum parameter combinations.

By observing the precision and recall results, notice that the precision value is more closer to the competitor than the recall value. These results are expected and can be explained by the fact that the ABEL approach only performs the full comparison of entities if the first split has a similarity value higher than β threshold (comparison filtering), reducing the recall. Otherwise, ABEL would miss true match entities pairs by eliminating comparisons.

In order to investigate the impact of the ABEL parameters (split length, α, and β), we chose the α threshold that presented the best result (in terms of F-1). With a fixed α, we varied each parameter combination exhibited in the experimental design. The result is depicted in Fig. 10. The vertical and horizontal axes represent the quality measure and β threshold ($\beta = \alpha - error$), respectively. The continuous lines represent the split length (in percentage) while the red dotted line expresses the competitor result.

Due to the error introduced by SBF, smaller splits need to consider a bigger *error* value to achieve better quality results. This error can be estimated by providing the split length as input to an exponential function, as illustrated in Fig. 7. The vertical colored areas representing F-1 in Fig. 10 highlight the calculated *error* for each split length. Notice that the maximum F-1 value starts near the designated colored area — calculated using the optimum-β threshold for the split length.

It is important to remark that the ABEL results depicted in Fig. 10 overcome or deliver the same result of the competitor when we employ an SBF error calculated by the exponential function presented in Section 8.1. The comparison filtering performed by ABEL can explain this result. For instance, consider that $\alpha = 0.65$ and the entities pairs ('Ana','Ane') and ('Ana','Lana') have similarity values equal to 0.66 and 0.60, respectively. If we define that $\beta = 0.55$ ($error = .1$), both pairs will be tagged as similar, reducing the precision. However, if $\beta = 0.61$ ($error = 0.05$), only the first pair will be tagged as similar, improving the precision.

Fig. 10 also demonstrates that ABEL was unable to overcome the competitor in only one scenario when we employ a small split length (16-bits split) for the MVR data source. We believe that this result is explained by the fact that a split containing 16 bits is two small, considering that the MVR was encoded with less than 1000 bits (Table 2), requiring a more significant SBF error than the one we considered in our experiments. This result also demonstrates the tradeoff between privacy and quality; a more in-depth discussion about this tradeoff is presented in Section 7.2.5.

The experiments presented in this section demonstrate that ABEL reaches a similar result when compared to the competitor. Furthermore, the quality metrics achieve their maximum values when we employ the optimum-β threshold. Thus, the answer to the first research question (RQ 1) of this section is: the ABEL approach provides a linkage result as effective as the approaches that utilize BF and consider an HBC security model.

7.2.2. Efficiency

In this section, we evaluate the efficiency of ABEL by analyzing its computational cost. As detailed in Section 5, ABEL computes the similarity between entities at two distinct moments. First, STTP performs the computations using only one split. Then, the parties use the output of STTP to perform the remaining similarity computations. Thus, the total computational cost is expressed as, $Cost(ABEL) = Cost(STTP) + Cost(Parties)$.

STTP performs its computation over the splits of all entities (m) provided by the parties (P). In our experiments, m represents all entities that belong to a participant. However, if a blocking or filtering technique is employed, the available entities m are determined by these techniques. Thus, $Cost(STTP)$ is related to m and the number of participants (P), such that, $Cost(STTP) = \mathcal{O}(m_{max} \times |P|)^2$, where $m_{max} = max([m_1, \ldots, m_{|P|}])$. It is worth noticing that STTP performs the comparison only in one split(ϕ) of each entity.

It is worth mentioning that the β threshold, which is defined by the parties and employed in the First Stage of the 3PAC protocol, presents an influence on the number of pairs in ζ-list. In other words, the β threshold has an impact on the number of computations performed by ABEL in the Comparison step. The impact is illustrated in Fig. 11, where the vertical axis represents the comparison reduction and the horizontal axis refers to the error value considered to set up β.

Notice that the reduction rate (rr) depicted in Fig. 11 was achieved over a blocked data source. Fig. 11 reveals that ABEL provides an extra filtering process to the comparisons. Furthermore, the experimental results indicate that the usage of smaller split lengths in SBF, with less than 5% of the original BF length, granted the highest rr. For instance, we were able to reduce the number of comparisons by 25% when we executed ABEL with 16 bits splits.

Regarding the split length influence over the rr, it is possible to observe that the use of smaller splits resulted in higher rr. Fig. 11 also evidences that, when β considers a small *error* in SBF, the number of comparisons reaches the maximum reduction rate, regardless of the split size. These results occur due to the rigorous filtering process performed by STTP, which selects only the most similar pairs of entities to be compared by the parties.

The gray line in Fig. 11 represents (with 95% of confidence) an exponential function that describes the reduction rate in relation to the split length and error considered in β, observed in our experiments. Thus, we can estimate the ζ-list's length by $|\zeta| = (1-rr) \times (m \times p)$, where $0 \leq rr \leq 1$. Notice that we employ $1-rr$

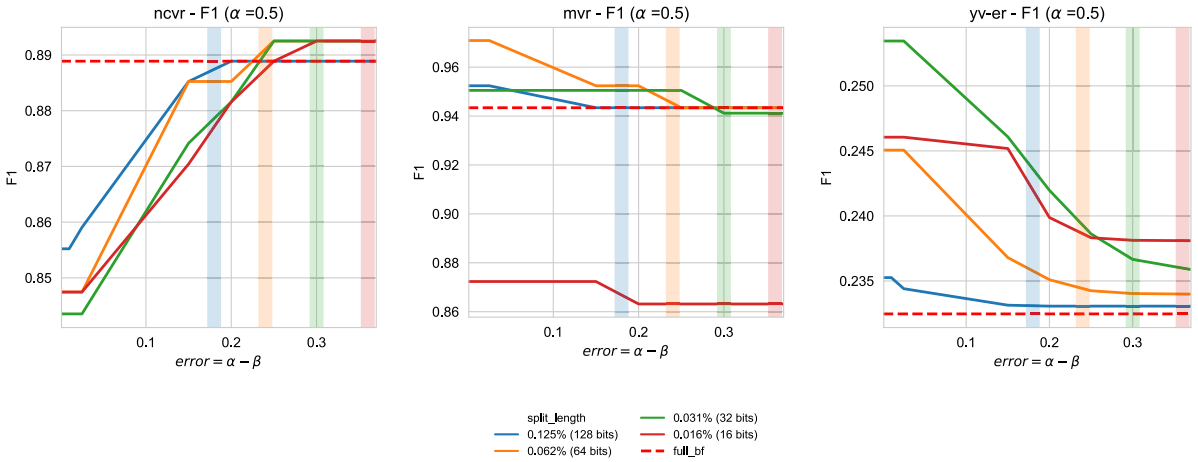

Fig. 10. Quality results achieved by ABEL for different β-thresholds. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

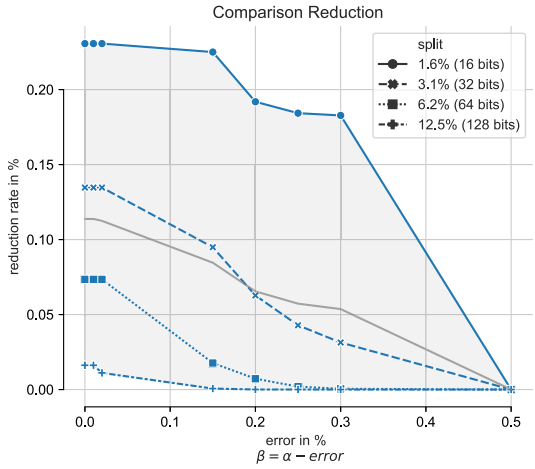

Fig. 11. Comparison reduction provided by ABEL.

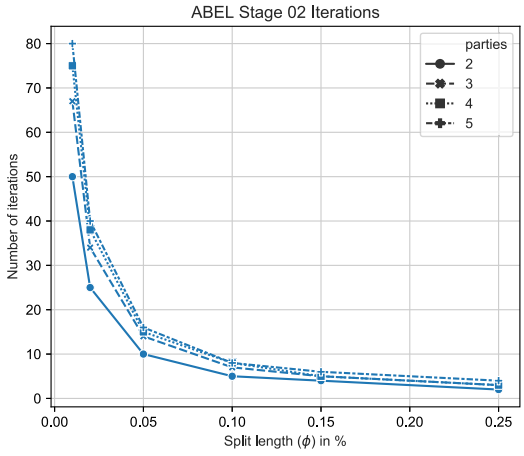

Fig. 12. Comparison reduction provided by ABEL. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to estimate the impact of the filtering process over the number of comparisons. For instance, if we have a $rr = .25$ our approach will consider the 75% $(1 - .25)$ of the original comparisons.

The number of iterations is calculated as a ratio between the number of remains parties $(|P| - 1)$ and the number of exchangeable splits (φ). As φ is proportional to the split length (in percentage), we can define the number of iterations by Eq. (9).

$$\begin{aligned} &iterations = \lceil (|P| - 1) \times |\varphi| \rceil \\ &replacing\ \varphi, where\ \varphi = \frac{(s-1)}{|P|} \\ &iterations = \left\lceil \frac{(|P| - 1)(s-1)}{|P|} \right\rceil \end{aligned} \tag{9}$$

Fig. 12 depicts the relationship between the number of iterations, split length, and number of parties. The vertical axis refers to the number of iterations performed by ABEL whilst the horizontal axis represents the split length as a percentage of the original length. The colored lines (generated using Eq. (9)) delineate the number of iterations for different parties (from two to five parties) for each split length.

According to Fig. 12, if the parties employ smaller splits to improve the privacy guarantees of SBF, then the number of iterations increases, raising the computational cost of ABEL. It is worthwhile to mention that, as more parties engage in ABEL, the number of iterations slightly increases, especially for smaller split lengths. For instance, for two parties considering splits with 25% and 1% of the original size ABEL will iterate 2 and 50 times, respectively. When we employ the same split length (25% and 1%) for 5 five parties, ABEL will iterate 4 and 80 times.

Finally, it is possible to estimate the parties' computational cost. As shown in Eq. (10), the cost is related to the ζ list and the number of ABEL iterations.

$$Cost(parties) = iterations \times \mathcal{O}(\zeta)^2 \tag{10}$$

The overall cost of ABEL is defined by Eq. (11).

$$\begin{aligned} Cost(ABEL) &= Cost(STTP) + Cost(parties) \\ &= \mathcal{O}(m_{max} \times |P|)^2 + iterations \times \mathcal{O}(\zeta)^2 \end{aligned} \tag{11}$$

7.2.3. Blockchain efficiency evaluation

Since ABEL considers STTP as a Blockchain Smart Contract, we provide a Proof-of-Concept implementation using the Ethereum DL technology. In order to investigate the impact of BC, we use three different BC setups to measure the comparison time (Λ_{sttp}) between STTP and the standard BF.

We execute the experiments using private (consortium) and public Ethereum networks. For the private network (Private-PoW), we build a three-nodes network using the Prof-of-Work (PoW) consensus mechanism — this network was accessible only

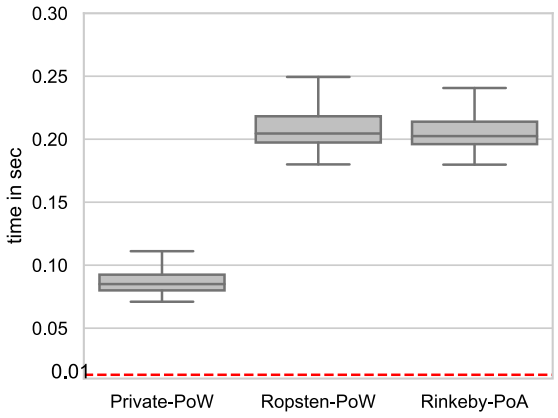

Fig. 13. Comparison time of a pair of entities over different Blockchain solutions.

by the parties. Concerning the public networks, we use the Ethereum Roposten network (Ropsten-PoW) that employs the PoW consensus mechanism, and the Rinkbey network (Rinkbey-PoA) that provides the Proof-of-Authority consensus mechanism — these networks are publicly accessible.

Fig. 13 illustrates the execution time of the individual comparisons. The vertical axis represents the execution time in seconds. The red dotted line outlines the result of the regular BF.

The results shown in Fig. 13 highlight that the mean execution time of the public networks were equivalent (with 95% confidence), regardless of the consensus mechanism. Another highlight is the fact that the private network outperformed the public network, by reaching half of the execution time of the public network. However, when we compare the STTP comparison time against the regular BF, the private network was ten times slower.

These outcomes were expected and can be explained by the mining schedule of BC, the consensus mechanism, and network delays. We detail these reasons as follows. As presented in Section 2, each transaction needs to pay a fee to the miner, setting this fee as a critical feature in the transaction scheduling [36], with an important impact on the transaction mining scheduler. For instance, if a transaction A sets its fee to 0.1 ether and transaction B sets its fee to 0.2, probably transaction B will be executed prior to A.

In order to store and execute a transaction in BC, one miner needs to break a cryptographic enigma (in the PoW) or a miner that has the execution token needs to schedule the transaction (in the PoA). In bigger networks, such as Roposten and Rinkbey, the transaction needs to be delivered to the miner node, and the network delay has an important influence on the BL transaction scheduler. Otherwise stated, delays in the transaction propagation among nodes of the networks hold back the transaction execution.

7.2.4. Privacy

In Section 6.1 we provided a privacy sketch evaluation; in this section, we evaluate if our approach is able to preserve the entity's privacy against attacks. To this end, we employ the state-of-the-art BF cryptoanalysis attacks [7,25] using all splits available in Stages 1 and 2 for the NCVR data source and considering two and three parties. We varied the split length for each different number of parties and executed the re-identification attack over the entity splits available at the end of ABLE. In other words, we tested our approach considering that we have from 10% to 50% of the original amount of information of the entities.

Table 3
Effectiveness, efficiency, and privacy trade-off.

Parameters		Influence		
Splits	Error	Privacy	Efficiency	Effectiveness
Bigger	Bigger	↓↓ Deacrease	↑↑ Increase	↑ Increase
Bigger	Smaller	↓ Deacrease	↑ Increase	↓ Deacrease
Smaller	Bigger	↑ Increase	↓ Deacrease	↑ Increase
Smaller	Smaller	↑↑ Increase	↓↓ Deacrease	↓ Deacrease

As a result, we were unable to re-identify none of the entities from the data source. We attribute the privacy-preserving capability of ABEL to the: (i) reduction rate (detailed in Section 7.2.2), that prevents the parties to share entities (splits) that are not possible match, and (ii) the use of the SBF in ABEL, which add an error (exposed in Section 7.2.1) to the entities that are possible matches. However, we believe that our approach is unable to preserve the privacy of the entities when the parties choose inappropriate parameters. For instance, if the parties choose a low α threshold (e.g., 0.1), STTP will return almost every entity of the original data source, increasing the success rate of any privacy attack.

In this Section, we tested our approach against privacy attacks, and as a result, we demonstrate that ABEL is capable of preserving the privacy of the entities during the PPRL execution. This results answers RQ.6, formulated at the beginning of Section 7.2.

7.2.5. ABEL evaluation summary

The evaluations presented in this section detail the results of the ABEL approach in terms of effectiveness, efficiency, and privacy. We tested the ABEL implementations with several data sources and Splitting Bloom Filter parameters (number of splits and thresholds) over different Blockchain configurations (PoW, PoA, public and privates networks). All experiments conducted in public Blockchain are available at our wallet public address[7] whilst the source code is available at the main author website.[8]

Regarding the results, our experiments demonstrate a relation between privacy capabilities, computational cost, and linkage quality. This relation is influenced by the BF parameters (k,l,n), SBF split length (s), ABEL thresholds error (α and β), and the number of parties (P).

The values of the aforementioned parameters should be carefully chosen. If one of the aspects is prioritized (quality, efficiency, or privacy), the remaining ones will be negatively impacted. Roughly speaking, if we use small splits we increase the privacy at the cost of minimizing quality and efficiency. Similarly, if bigger splits are preferred, we increase efficiency at the cost of lower privacy capabilities. The selection of ABEL thresholds (α and β) follows the same idea: if we consider a high error, we prioritize the quality over privacy and efficiency. The trade-off between effectiveness (quality), efficiency, and privacy is summarized in Table 3.

As shown in Table 3, if we opt to maximize privacy by employing small splits and considering a small error in the thresholds, we prejudice the effectiveness and efficiency of the approach. In turn, if we employ bigger splits and a bigger error in the thresholds, we prioritize the efficiency and effectiveness, minimizing privacy. Thus, thresholds error and split length should be carefully selected to avoid impacting the effectiveness, efficiency, or privacy of the linkage process negatively.

[7] https://ropsten.etherscan.io/address/0x99429f64cf4d5837620dcc293c1a537d58729b68.

[8] https://github.com/thiagonobrega/auditable_pprl.

8. Related work

This research is closely related to PPRL and decentralized data management. In this sense, we can list the following works as related to the problem tackled in this work.

Vatsalan and Christen [15] proposed a protocol to identify matching sets of entities (records) held by multiple parties that have a similarity above a certain threshold under the semi-honest adversary model. The protocol divides and distributes the BF chunks amongst the parties. To calculate the entities' overall similarity, the parties employ the BF chunks in a Secure Multi-party Computation (SMC). Their work is related to ours since it uses BF chunks to calculate the similarity between entities. However, their work considers an HBC model and does not explore neither the relationship between the individual similarity of the chunks nor the overall similarity of the entities.

Berman's [42] proposes a protocol to perform an exact comparisons using a unique identifier in medical records under an HBC security model. Notice that Berman's work employs exact comparisons under HBC while our approach considers approximated comparisons of the entities under the covert security model.

The idea of splitting the Bloom Filter and perform the comparisons iteratively is not new. Vatsalan and Christen's [43] presented a two-party protocol that eliminates the need of a third party by iteratively revealing selected bits in the Bloom filters between two parties. The aforementioned work presents some differences from ours such as the number of PPRL parties and the amount of information required to begin the protocol — our work needs less than 1% of the original work while Vatsalan and Christen's work needs more than 40%. However, the novelty of our work is that we are able to audit the comparison performed during the PPRL which eliminates the need to fully trusty the other parties. This auditability is only possible by the usage of Blockchain and the proposal of the SBF to overcome the privacy limitation of the Blockchain. Furthermore, to the best of our knowledge, this work is the first to consider a covert adversary model in PPRL context.

BFs are widely used in Internet of Things (IoT), wireless and wired networks, as well as network security applications. In such applications, BF generation's performance is an issue due to the hardware limitation and volume of data. These applications prefer to use as few hash functions (k) and as small filter length (l) as possible for performance and implementation issues. However, these designs lead to higher rates of false-positive values. Thus, Chum, Wei, and Zhang [44] associate the performance issue to the length (l) and the number of hash functions (k) employed in BF. Their work differs from ours because the splitting process aims to speed up the BF generation in an IoT context, while our work aims to enhance the privacy of a PPRL process. Moreover, the work of Chum, Wei, and Zhang proposes a method to generate a BF with twice the original length [44], splitting the hash space into two-parts to reduce the computational cost and the false-positive rate. On the other hand, our work splits the BF bit array, regardless of the generation algorithm, to be employed in the comparison step of the PPRL and reduce the amount of shared information.

The work of Almeida [45] demonstrates that the BF false-positive rate formula is smaller than thought. Moreover, the work demonstrates that the standard BF is prone to exhibit a high collision of '1' (named as weak spots by the authors) if naive double hashing is used. This is relevant in IoT scenarios where an element is tested against many filters, e.g., in packet forwarding. Almeida's weak spots are more evident in small BF (e.g., 8 or 32 bits BF); however, a PPRL scenario typically deals with bigger BFs when compared to network and IoT applications, e.g., the length of BF in PPRL usually varies between 1024 and 4096 bits. For instance, the difference between the approximated and the exact false-positive rate calculation, considering a BF with $l = 512$ and $k = 8$, is approximately 1×10^{-4}. This difference represents 0.01% and cannot be considered a significant threat to SBF, which usually employs BFs greater than 512 bits. However, if PPRL is conducted utilizing small BFs, this difference must be taken into account to avoid additional risk to the entity's privacy and linkage quality.

Hybrid PPRL protocols that combine differential privacy techniques with SMC techniques have been proposed to reduce the computational cost of PPRL [46,47]. However, such protocols need to disclose all entities stored in their data sources amongst the parties compromising the privacy capabilities of the anonymization. To address this issue, Rao et al. [48] propose a framework under an HBC security model that employs a trusty-third party to coordinate the record matching between the parties. The parties send to the trusty-third party synopses (with Differential Privacy guarantees) of the data. Based on the received synopses, the trusty-third party matches the entities with a distance beyond a threshold specified by the parties.

Since BF is the most common anonymization technique employed in PPRL, several works investigate the BF's ability to maintain the privacy of anonymized data. In this context, the work of Christen et al. [25] show that, to reidentify the entities, an adversary only needs to have access to a public database of attribute values. The work of Vidanage et al. [7] applies maximal frequent itemset mining (using a language model) on a BF database to identify sets of frequently co-occurring bit positions that correspond to encoded frequent sub-strings values. However, to succeed, both works [7,25] need a significant number of anonymized entities as input; the authors tested their approaches with more than 1 million and 200 thousand anonymized entities, respectively.

Considering the limitations of the HBC and Malicious security models, Aumann and Lindell [8] introduces the notion of covert adversaries. The author presents a cryptographic protocol that enables honest parties to detect the misbehavior of an adversary with high probability — this protocol applies Homomorphic Encryption (HE) to detect an adversary misbehave. Jiang et al. [24] provides accountability for privacy-preserving computation without the excessive complexity and computational cost of the Malicious model.

The Blockchain technology has gained much attention in the data management community due to the trust achieved in not fully trusted computation environments. Tuan et al. [31] survey the state-of-the-art Blockchain technologies. They propose a benchmarking framework for understanding the performance of private Blockchains considering data processing workloads. Nathan et al. [10] design and implement a decentralized replicated relational database with Blockchain properties, named as Blockchain Relational Database. The previously mentioned works confirm the need for privacy-preserving data processing in a Blockchain context.

9. Conclusion and future work

In this paper, we have presented a PPRL protocol that considers a covert adversary model. This protocol considers a decentralized computing environment (Blockchain) and uses a novel implementation of the Bloom Filter (SBF), which reduces the information available to the PPRL parties. Specifically, SBF decreases the available information in case of collusion amongst malicious parties. The Blockchain technology was employed to implement a semi-trusted party and, consequently, provides auditability to the PPRL process.

Our contributions were evaluated theoretically and experimentally (using several real-world data sources), regarding efficiency, effectiveness (quality), and privacy. The privacy evaluation showed that our contributions are capable of preserving the privacy of the entities in the PPRL execution. Regarding effectiveness (quality), the evaluation has shown that our approach achieves results similar to a traditional PPRL approach that considers HBC adversaries. Finally, the efficiency results demonstrated that our approach (ABEL) is slower than the PPRL approaches based on HBC. In summary, findings from these evaluations demonstrated that there is no optimal trade-off between efficiency, utility, and privacy.

In future work, we aim to investigate the nature of SBF error and the possibility of adding a Difference Privacy mechanism in the SBF. Moreover, we intend to employ advanced cryptographic tools (i.e., Fully Homomorphic Encryption and Proxy Re-encryption) can improve our contributions' privacy-preserving capabilities. We also plan to study the impact of the new confidential Blockchain technology from Microsoft (MCC). Another direction is to investigate how the similarity computation (performed by the Blockchain) can be improved (e.g., parallelized) to increase ABEL's efficiency. Investigating our contributions using a Malicious adversary model is also an interesting course for future work.

CRediT authorship contribution statement

Thiago Nóbrega: Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing. **Carlos Eduardo S. Pires:** Methodology, Writing - original draft, Writing - review & editing. **Dimas Cassimiro Nascimento:** Methodology, Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the Brazilian National Council of Technological and Scientific Development (CNPq).

Appendix A. Notation summary

Symbol	Description
$\mathbb{P}$	Participant of PPRL
e	Entity
e^{τ}	Anonymized entity
$\mathbb{D}_p$	Dataset of participant p
$\mathbb{D}_p^{\tau}$	Anonymized dataset of participant p
Ξ	Decision model
Bloom Filter	
l	Bloom filter length
n	Set of elements (q-grams) to be inserted into BF
k	Bloom filter hash functions
Splitting Bloom Filter	
s	Number of splits
ϕ	Split
3PAC	
α	Alpha threshold
β	Beta threshold ($\beta = \alpha - error$)
ζ	List of entities (ids) pairs with their similarity values
φ	A set of splits (ϕ)

Appendix B. SBF splits differences

A BF stores m elements with k hash functions into l-bits, such that $e^{\tau} = [b_0, \ldots, b_l]$. SBF divides the original BF into s splits with $\frac{l}{s}$ bits, such that $SBF(e^{\tau}, s) = [\phi^0, \ldots, \phi^{s-1}]$, and $\phi^i = \left[b_i, \ldots, b_{i+(\frac{l}{s}-1)}\right]$. In this context, consider that $\Pr(u) = p$ represents that a bit position stores '1',[9] and the probability of a bit position storing the value '0' can be estimated by $\Pr(z) = 1 - \Pr(u)$; thus, $\Pr(z) = 1 - p$.

The probability of a split (ϕ^i) storing a specific distribution of bits ($\Pr(\phi^i = x)$) can be estimated by the number of '0' and '1' bits stored in the split [30,45]. Assuming that the BF length (l) is big enough, and the BF parameters are correctly configured, it is possible to consider that the BF bits' values are independent events [49]. Therefore, we can use a Binomial distribution, with $\frac{l}{s}$ positions (or trials) to estimate the $\Pr(\phi^i = x)$, i.e., $\Pr(\phi^i = x) \sim B\left(\frac{l}{s}, p\right)$, as presented in Eq. (12)

$$B\left(\frac{l}{s}, p\right) = \binom{\frac{l}{s}}{x} p^x (1-p)^{\frac{l}{s}-x} \tag{12}$$

Assuming $\Pr(\phi^i = x)$ as a Binomial distribution, we are able to estimate the variation of the bits within the splits using the binomial distribution mean (μ), variance (σ^2) and standard deviation (σ). Consider that $\mathcal{X}$ is a binomially distributed random variable, such as $\mathcal{X} \sim B(n, p)$, we can estimate the mean number of '1' in each split using Eq. (13).

$$\mu = \frac{l}{s} \cdot p \tag{13}$$

Knowing that the variance of a binomial distribution can be calculated by $\sigma^2(\mathcal{X}) = n \cdot p \cdot (1-p)$, and the standard deviation is the square root of the variance, we can use σ^2 and σ to estimate the variation of the bits within the splits applying Eqs. (14) and (15).

$$\sigma^2 = \frac{l}{s} \cdot p \cdot (1-p) \tag{14}$$

$$\sigma = \sqrt{\frac{l}{s} \cdot p \cdot (1-p)} \tag{15}$$

Moreover, we employ the coefficient of variation (CV), defined as standard deviation (σ) divided by the mean (μ) to describe the variability of '1' within a split relative to the mean number of '1' in other splits. It is worthwhile to mention that the CV represents the variability in percentage. Eq. (16) calculates the CV of the SBF splits.

$$CV = \frac{\mu}{\sigma} \Rightarrow \frac{\frac{l}{s} \cdot p}{\sqrt{\frac{l}{s} \cdot p \cdot (1-p)}} \Rightarrow \frac{\frac{l}{s} \cdot p}{\sqrt{\frac{l}{s}} \cdot \sqrt{p \cdot (1-p)}},$$

[9] The $\Pr(u)$ and $\Pr(z)$ of a BF can be estimated using BF setup parameters, as presented in Section 2, such that $\Pr(z) = e^{-\frac{k \times n}{l}}$ and $\Pr(u) = 1 - e^{-\frac{k \times n}{l}}$.

considering that $\frac{l}{s} = \sqrt{\frac{l}{s}} \cdot \sqrt{\frac{l}{s}}$

$$\Rightarrow \frac{\cancel{\sqrt{\frac{l}{s}}} \cdot \sqrt{\frac{l}{s}} \cdot p}{\cancel{\sqrt{\frac{l}{s}}} \cdot \sqrt{p \cdot (1-p)}},$$

considering that $p = \sqrt{p} \cdot \sqrt{p}$, and $\sqrt{a \cdot b} \Rightarrow \sqrt{a} \cdot \sqrt{b}$, (16)

such that $a \geq 0$ and $b \geq 0$.

$$\Rightarrow \frac{\sqrt{\frac{l}{s}} \cdot \cancel{\sqrt{p}} \cdot \sqrt{p}}{\cancel{\sqrt{p}} \cdot \sqrt{(1-p)}} \Rightarrow \frac{\sqrt{\frac{l}{s} \cdot p}}{\sqrt{(1-p)}} \Rightarrow \frac{\sqrt{l} \cdot \sqrt{p}}{\sqrt{s} \cdot \sqrt{(1-p)}}$$

$$\Rightarrow \frac{\sqrt{l \cdot p}}{\sqrt{s \cdot (1-p)}} \quad \blacksquare$$

Eq. (16) represents the percentual variation of '1' in terms of l, s, and p. Thus, considering that in a PPRL context $p \approx 0.5$ and l is bigger than s, it is intuitive that Eq. (16) will produce $CV > 0$. For instance, if we consider l = 1024, s = 10 and p = 0.5, the expect variability of '1' is approximated 31%. In other words, it is possible to configure a SBF to present, with a high probability, a slightly variation in the number of '1'. Therefore, assuming that the BF and SBF parameters are properly configured, the variation in the number of '1' is reflected in the Jaccard similarity metric, such that $Jaccard(\phi_a^i, \phi_b^i) \neq Jaccard(\phi_a^{i+1}, \phi_b^{i+1})$, where $\phi_a^i \in SBF(e_a^\tau, s)$ and $\phi_b^i \in SBF(e_b^\tau, s)$.

Furthermore, the Jaccard similarity of two BFs can be expressed as the mean similarity of their splits, as expressed in Eq. (5), $Jaccard\_SBF(e_a^\tau, e_b^\tau, s) = \frac{1}{s}\sum_{i=0}^{s} \frac{|\phi_a^i \bigcap \phi_b^i|}{|\phi_a^i \bigcup \phi_b^i|}$. Thus, considering the typical parameters of BF employed in a PPRL context, it is unlikely that every split of a BF is identical. This statement is straightforward to demonstrate by using the BF and SBF parameters in Eq. (12), and can be estimated by $\sum_{i=1}^{s} \Pr(\phi^i = x)$. For instance, considering the previous example (l = 1024, p = 0.5, and s = 10), the probability of every split being identical is $\Pr(\phi^i = x) \approx 2.52 \times 10^{-133}$.

Finally, consider that the SBF can be configured to vary the number '1' in its splits, and the remote probability of every split presents the same bit distribution. We assume that there is a difference between Jaccard similarity of the original BF and the similarity of one SBF split $Jaccard(e_a^\tau, e_b^\tau) \neq Jaccard(\phi_a^i, \phi_b^i)$, such that $\phi_a^i \in SBF(e_a^\tau, s)$ and $\phi_b^i \in SBF(e_b^\tau, s)$. Moreover, we associate this difference to an error ϵ, and use Eq. (6) to represent this intuition.

$$Jaccard(e_a^\tau, e_b^\tau) = Jaccard(\phi_a^i, \phi_b^i) + \epsilon$$

, *such that* $0 \leq \epsilon \leq 1$.

In this appendix, we intend to acknowledge the existence of differences between the similarity of one split and the BF that originated it. Appendix C demonstrates the relation between our experimental results (Section 7.1) and the content of this Appendix.

Appendix C. Experimental analysis of the relation between ϵ-error and σ

In order to illustrate the alignment of the arguments presented in Appendix B with the results described in Section 7.1, we selected five datasets (bike, beer, NCVR, MVR and yv-er) that were encoded with the same filter length ($l = 1024 - bits$) to show that the ϵ error is proportional to the error presented in Eq. (15).

Therefore, knowing the BF setup for these datasets and considering $p \approx 0.53$, we calculated the standard deviation (SD) for every used split length in the experiments and compared the results against the mean measured error (ϵ). The results are shown in Fig. 14; notice that we convert the error as a percentage of the bit length.

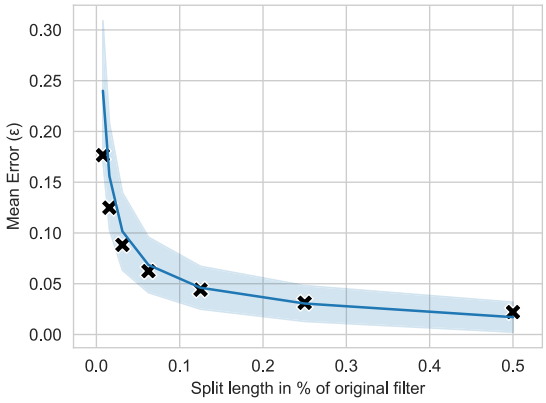

Fig. 14. Comparison of the measured and calculated errors associated with split length, considering a BBF.

Notice that the calculated SD (x marks) is proportional to and near the line that represents the measured mean error of the experiments. This result corroborates with the experimental results presented in Sections 7.1 and 7.1.2 , illustrated in Figs. 6, 7, and 8, showing that the smaller the length of the SBF splits, the bigger the error originated from Eq. (6).

We would also like to mention that, until this point, we can acknowledge the existence of ϵ, and this error is related to the error measured in our experiments. In future work, we intend to investigate the particular characteristics of ϵ.

References

[1] D. Vatsalan, P. Christen, V.S. Verykios, A taxonomy of privacy-preserving record linkage techniques, Inf. Syst. 38 (6) (2013) 946–969, http://dx.doi.org/10.1016/j.is.2012.11.005.

[2] P. Christen, D. Vatsalan, A flexible data generator for privacy-preserving data mining and record linkage, 2012.

[3] P. Christen, D. Vatsalan, V.S. Verykios, Challenges for privacy preservation in data integration, J. Data Inf. Qual. 5 (1–2) (2014) 1–3, http://dx.doi.org/10.1145/2629604, URL: http://dl.acm.org/citation.cfm?doid=2667565.2629604.

[4] D. Vatsalan, D.K. B, A. Gkoulalas-divanis, An Overview of Big Data Issues in Privacy-Preserving Record Linkage, Vol. 2, Springer International Publishing, 2019, pp. 118–136, http://dx.doi.org/10.1007/978-3-030-19759-9.

[5] R. Pita, C. Pinto, P. Melo, M. Silva, M. Barreto, D. Rasella, A spark-based workflow for probabilistic record linkage of healthcare data, in: CEUR Workshop Proceedings, Vol. 1330, 2015, pp. 17–26.

[6] G.A. Papadakis, D. Skoutas, E. Thanos, A survey of blocking and filtering techniques for entity resolution, 2019.

[7] A. Vidanage, T. Ranbaduge, P. Christen, R. Schnell, Efficient pattern mining based cryptanalysis for privacy-preserving record linkage, in: Proceedings - International Conference on Data Engineering, 2019, pp. 1698–1701, http://dx.doi.org/10.1109/ICDE.2019.00176.

[8] Y. Aumann, Y. Lindell, Security against covert adversaries: Efficient protocols for realistic adversaries, J. Cryptol. 23 (2) (2010) 281–343, http://dx.doi.org/10.1007/s00145-009-9040-7, URL: http://link.springer.com/10.1007/s00145-009-9040-7.

[9] M. El-Hindi, M. Heyden, C. Binnig, R. Ramamurthy, A. Arasu, D. Kossmann, Blockchaindb - towards a shared database on blockchains, in: Proceedings of the 2019 International Conference on Management of Data - SIGMOD '19, Vol. 12, ACM Press, New York, New York, USA, 2019, pp. 1905–1908, http://dx.doi.org/10.1145/3299869.3320237, URL: http://dl.acm.org/citation.cfm?doid=3299869.3320237.

[10] S. Nathan, C. Govindarajan, A. Saraf, M. Sethi, P. Jayachandran, Blockchain meets database, Proc. VLDB Endow. 12 (11) (2019) 1539–1552, http://dx.doi.org/10.14778/3342263.3342632, URL http://arxiv.org/abs/1903.01919http://dl.acm.org/citation.cfm?doid=3342263.3360362.

[11] T. Dasu, Y. Kanza, D. Srivastava, Unchain your blockchain, in: Foundations and Applications of Blockchain (FAB), 2018, pp. 16–23.

[12] T.T.A. Dinh, J. Wang, G. Chen, R. Liu, B.C. Ooi, K.-L. Tan, Blockbench: A framework for analyzing private blockchains, in: Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17, ACM Press, New York, New York, USA, 2017, pp. 1085–1100, http://dx.doi.org/10.1145/3035918.3064033, URL: http://dl.acm.org/citation.cfm?doid=3035918.3064033.

[13] R. Schnell, T. Bachteler, J. Reiher, Privacy-preserving record linkage using bloom filters, BMC Med. Inform. Decis. Mak. 9 (1) (2009) 41, http://dx.doi.org/10.1186/1472-6947-9-41.

[14] T.P. Nóbrega, C.E.S. Pires, T.B. Araújo, D.G. Mestre, Blind attribute pairing for privacy-preserving record linkage, in: Proceedings of the 33rd Annual ACM Symposium on Applied Computing - SAC '18, ACM Press, New York, New York, USA, 2018, pp. 557–564, http://dx.doi.org/10.1145/3167132.3167193, URL: http://dl.acm.org/citation.cfm?doid=3167132.3167193.

[15] D. Vatsalan, P. Christen, Multi-party privacy-preserving record linkage using bloom filters, 2016, URL: http://arxiv.org/abs/1612.08835.

[16] T.B. Araujo, C.E.S. Pires, T.P. da Nobrega, Spark-based streamlined metablocking, in: 2017 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2017, pp. 844–850, http://dx.doi.org/10.1109/ISCC.2017.8024632, URL: http://ieeexplore.ieee.org/document/8024632/.

[17] C. Batini, M. Scannapieco, Data and Information Quality, first ed., in: Data-Centric Systems and Applications, Springer International Publishing, 2016, http://dx.doi.org/10.1007/978-3-319-24106-7, URL: http://link.springer.com/10.1007/978-3-319-24106-7.

[18] T. Ranbaduge, P. Christen, Privacy-preserving temporal record linkage, in: 2018 IEEE International Conference on Data Mining (ICDM), 2018, pp. 377–386, http://dx.doi.org/10.1109/ICDM.2018.00053, URL: https://ieeexplore.ieee.org/document/8594862/.

[19] D. Vatsalan, P. Christen, E. Rahm, Scalable multi-database privacy-preserving record linkage using counting bloom filters, 2017, URL: http://arxiv.org/abs/1701.01232.

[20] D. Vatsalan, D. Karapiperis, V.S. Verykios, Privacy-preserving record linkage, 2018, http://dx.doi.org/10.1007/978-3-319-63962-8.

[21] D. Vatsalan, Z. Sehili, P. Christen, E. Rahm, Privacy-preserving record linkage for big data : Current approaches and research challenges, in: Big Data Handbook, Springer, 2016.

[22] X. Ding, W. Yang, K.K. Raymond Choo, X. Wang, H. Jin, Privacy preserving similarity joins using mapreduce, Inform. Sci. 493 (2019) 20–33, http://dx.doi.org/10.1016/j.ins.2019.03.035.

[23] Y. Aumann, Y. Lindell, Security against covert adversaries: Efficient protocols for realistic adversaries, J. Cryptol. 23 (2) (2010) 281–343, http://dx.doi.org/10.1007/s00145-009-9040-7.

[24] W. Jiang, C. Clifton, M. Kantarcioğlu, Transforming semi-honest protocols to ensure accountability, Data Knowl. Eng. 65 (1) (2008) 57–74, http://dx.doi.org/10.1016/j.datak.2007.06.014.

[25] P. Christen, R. Schnell, D. Vatsalan, T. Ranbaduge, Efficient cryptanalysis of bloom filters for privacy-preserving record linkage peter, in: J. Kim, K. Shim, L. Cao, J.-G. Lee, X. Lin, Y.-S. Moon (Eds.), Advances in Knowledge Discovery and Data Mining, in: Lecture Notes in Computer Science, vol. 10235, Springer International Publishing, Cham, 2017, pp. 628–640, http://dx.doi.org/10.1007/978-3-319-57529-2, URL: http://link.springer.com/10.1007/978-3-319-57529-2.

[26] G. Miklau, D. Suciu, A formal analysis of information disclosure in data exchange, J. Comput. System Sci. 73 (3) (2007) 507–534, http://dx.doi.org/10.1016/j.jcss.2006.10.004.

[27] F. Niedermeyer, S. Steinmetzer, M. Kroll, R. Schnell, M. Kuzu, M. Kantarcioglu, E. Durham, B. Malin, Cryptanalysis of basic bloom filters used for privacy preserving record linkage, J. Priv. Confidentiality 6 (2) (2014) 59–79.

[28] D. Vatsalan, P. Christen, V.S. Verykios, Ef fi cient two-party private blocking based on sorted nearest neighborhood clustering, 2013, pp. 1949–1958.

[29] D. Vatsalan, P. Christen, C.M. O'Keefe, V.S. Verykios, An evaluation framework for privacy-preserving record linkage, J. Priv. Confidentiality 6 (1) (2014) 75, URL: http://repository.cmu.edu/jpc/vol6/iss1/3.

[30] A. Broder, M. Mitzenmacher, Network applications of bloom filters: A survey, Internet Math. 1 (4) (2004) 485–509, http://dx.doi.org/10.1080/15427951.2004.10129096.

[31] T. Tuan, A. Dinh, R. Liu, M. Zhang, G. Chen, T.T.A. Dinh, R. Liu, M. Zhang, G. Chen, B.C. Ooi, J. Wang, Untangling blockchain: A data processing view of blockchain systems, IEEE Trans. Knowl. Data Eng. 30 (7) (2018) 1366–1385, http://dx.doi.org/10.1109/TKDE.2017.2781227.

[32] N. Atzei, M. Bartoletti, T. Cimoli, A survey of attacks on ethereum smart contracts (sok), in: M. Maffei, M. Ryan (Eds.), Kyoto Daigaku Kokukagaku Kiyo. Bulletin of Stomatology, Kyoto University, in: Lecture Notes in Computer Science, vol. 10204, (8) Springer Berlin Heidelberg, Berlin, Heidelberg, 2017, pp. 164–186, http://dx.doi.org/10.1007/978-3-662-54455-6{_}8, URL: http://link.springer.com/10.1007/978-3-662-54455-6http://link.springer.com/10.1007/978-3-662-54455-6_8.

[33] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The blockchain model of cryptography and privacy-preserving smart contracts, in: 2016 IEEE Symposium on Security and Privacy (SP), IEEE, 2016, pp. 839–858, http://dx.doi.org/10.1109/SP.2016.55, URL: http://ieeexplore.ieee.org/document/7546538/.

[34] R. Schnell, Privacy-preserving record linkage, 2016, pp. 201–225, http://dx.doi.org/10.1002/9781119072454.ch9, URL: http://doi.wiley.com/10.1002/9781119072454.ch9.

[35] C. Dwork, Theory and Applications of Models of Computation, Vol. 4978, 2008, pp. 1–19, http://dx.doi.org/10.1007/978-3-540-79228-4, URL: http://link.springer.com/10.1007/978-3-540-79228-4.

[36] G. Wood, Ethereum: a Secure Decentralised Generalised Transaction Ledger, Ethereum Project Yellow Paper 32, 2018, pp. 1365–1367, http://dx.doi.org/10.1017/CBO9781107415324.004, URL: https://gavwood.com/paper.pdf%0Ahttp://gavwood.com/Paper.pdf%0Ahttp://yellowpaper.io%0Ahttps://ethereum.github.io/yellowpaper/paper.pdf.

[37] Y. Lindell, in: Y. Lindell (Ed.), Tutorials on the Foundations of Cryptography, in: Information Security and Cryptography, Springer International Publishing, Cham, 2017, http://dx.doi.org/10.1007/978-3-319-57048-8, URL: http://link.springer.com/10.1007/978-3-319-57048-8.

[38] R. Schnell, C. Borgs, Randomized response and balanced bloom filters for privacy preserving record linkage, in: IEEE International Conference on Data Mining Workshops, ICDMW, 2017, pp. 218–224, http://dx.doi.org/10.1109/ICDMW.2016.0038.

[39] R. Schnell, C. Borgs, F.-b. Encryptions, XOR-Folding for bloom encryptions for record linkage, 2016.

[40] P. Christen, T. Ranbaduge, R. Schnell, Linking Sensitive Data, Springer International Publishing, Cham, 2020, http://dx.doi.org/10.1007/978-3-030-59706-1, URL: http://link.springer.com/10.1007/978-3-030-59706-1.

[41] D.G. Mestre, Uma Abordagem para Aprimoramento do Balanceamento de Carga do Método de Resolução de Entidades Standard Blocking baseado em MapReduce (Ph.D. thesis), Universidade Federal de Campina Grande, 2013.

[42] J.J. Berman, Zero-check: a zero-knowledge protocol for reconciling patient identities across institutions, Arch. Pathol. Lab. Med. 128 (3) (2004) 344–346.

[43] D. Vatsalan, P. Christen, et al., An Iterative Two-Party Protocol for Scalable Privacy-Preserving Record Linkage, Australian Computer Society Inc., 2012.

[44] C.S. Chum, X. Wei, X. Zhang, A split bloom filter for better performance, J. Appl. Secur. Res. 15 (2) (2020) 147–160, http://dx.doi.org/10.1080/19361610.2019.1613090, URL: https://doi.org/10.1080/19361610.2019.1613090https://www.tandfonline.com/doi/full/10.1080/19361610.2019.1613090.

[45] P.S. Almeida, A case for partitioned bloom filters, 2020, arXiv:2009.11789.

[46] X. He, A. Machanavajjhala, C. Flynn, D. Srivastava, Composing differential privacy and secure computation, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS '17, ACM Press, New York, New York, USA, 2017, pp. 1389–1406, http://dx.doi.org/10.1145/3133956.3134030, URL: http://arxiv.org/abs/1702.00535http://dl.acm.org/citation.cfm?doid=3133956.3134030.

[47] A. Inan, M. Kantarcioglu, G. Ghinita, E. Bertino, Private record matching using differential privacy, in: Proceedings of the 13th International Conference on Extending Database Technology - EDBT '10, ACM Press, New York, New York, USA, 2010, p. 123, http://dx.doi.org/10.1145/1739041.1739059, URL: http://portal.acm.org/citation.cfm?doid=1739041.1739059.

[48] F.Y. Rao, J. Cao, E. Bertino, M. Kantarcioglu, Hybrid private record linkage: Separating differentially private synopses from matching records, ACM Trans. Priv. Secur. 22 (3) (2019) http://dx.doi.org/10.1145/3318462.

[49] D. Kleyko, A. Rahimi, R.W. Gayler, E. Osipov, Autoscaling bloom filter: controlling trade-off between true and false positives, Neural Comput. Appl. 32 (8) (2020) 3675–3684, http://dx.doi.org/10.1007/s00521-019-04397-1, arXiv:1705.03934.