EsPRESSo: Efficient Privacy-Preserving Evaluation of Sample Set Similarity*

Carlo Blundo[1], Emiliano De Cristofaro[2], Paolo Gasti[3]

[1] Università di Salerno – Fisciano (SA), Italy I-84084, *cblundo@unisa.it*

[2] PARC – 3333 Coyote Hill Road, Palo Alto, CA 94304, *me@emilianodc.com*

[3] New York Institute of Technology – 1855 Broadway, New York, NY 10023, *pgasti@nyit.edu*

Abstract

Electronic information is increasingly often shared among entities without complete mutual trust. To address related security and privacy issues, a few cryptographic techniques have emerged that support privacy-preserving information sharing and retrieval. One interesting open problem in this context involves two parties that need to assess the *similarity* of their datasets, but are reluctant to disclose their actual content. This paper presents an efficient and provably-secure construction supporting the privacy-preserving evaluation of sample set similarity, where similarity is measured as the *Jaccard* index. We present two protocols: the first securely computes the (Jaccard) similarity of two sets, and the second approximates it, using MinHash techniques, with lower complexities. We show that our novel protocols are attractive in many compelling applications, including document/multimedia similarity, biometric authentication, and genetic tests. In the process, we demonstrate that our constructions are appreciably more efficient than prior work.

Keywords: Secure computation, privacy-preserving protocols, privacy-enhancing technologies

1 Introduction

The availability of electronic information is essential to the functioning of our communities. Increasingly often, data needs to be shared between parties without complete mutual trust. Naturally, this raises important privacy concerns with respect to the disclosure and the long-term safety of sensitive content. One interesting problem occurs

*A preliminary version of this paper was published in the Proceedings of the 7th ESORICS International Workshop on Digital Privacy Management (DPM 2012). This is the full version.

whenever two or more entities need to evaluate the similarity of their datasets, but are reluctant to openly disclose their data.

This task faces three important technical challenges: (1) how to identify a meaningful metric to estimate similarity, (2) how to compute a measure thereof such that no private information is revealed during the process, and (3) how to do so efficiently. We address such challenges by introducing a cryptographic primitive called ***EsPRESSo – Privacy-Preserving Evaluation of Sample Set Similarity***. Among others, this construction is appealing in a few relevant applications, presented below.

Document similarity: Two parties need to estimate the similarity of their documents, or collections thereof. In many settings, documents contain sensitive information and parties may be unwilling, or simply forbidden, to reveal their content. For instance, program chairs of a conference may want to verify that none of submitted papers is also under review in other conferences or journals, but, obviously, they are not allowed to disclose papers in submission. Likewise, two law enforcement authorities (e.g., the FBI and local police), or two investigation teams with different clearance levels, might need to share documents pertaining suspect terrorists, but they can do so only conditioned upon a clear indication that content is relevant to the same investigation.

Iris Matching: Biometric identification and authentication are increasingly used due to fast and inexpensive devices that can extract biometric information from a multitude of sources, e.g., voice, fingerprints, iris, and so on. Clearly, given its utmost sensitivity, biometric data must be protected from arbitrary disclosure. Consider, for instance, an agency that needs to determine whether a given biometric appears on a government watch-list. As agencies may have different clearance levels, privacy of biometric's owner needs to be preserved if no matches are found, but, at the same time, unrestricted access to the watch-list cannot be granted.

Multimedia File Similarity: Digital media, e.g., images, audio, video, are increasingly relevant in today's computing ecosystems. Consider two parties that wish to evaluate similarity of their media files, e.g., for plagiarism detection: sensitivity of possibly unreleased material (or copyright issues) may prevent parties from revealing actual content.

EsPRESSO does not only appeal to examples above, but are also relevant to a wide spectrum of applications, for instance, in the context of privacy-preserving sharing of information and/or recommender systems, e.g., to privately assess similarity of genomic information [5], social network profiles [4], attackers' information [30], etc.

1.1 Technical Roadmap & Contributions

Our first step is to identify a *metric* for effectively evaluating similarity of sample sets. Several similarity measures are available and commonly used in different contexts, such as Cosine, Euclidean, Manhattan, Minkowski similar-

ity, or Hamming and Levenshtein distances. In this paper, we focus on a well-known metric, namely, the *Jaccard Similarity Index* [24], which quantifies the similarity of *any* two sets A and B. It is expressed as a rational number between 0 and 1, and, as showed in [9], it effectively captures the informal notion of "roughly the same". The Jaccard Index can be used, e.g., to find near duplicate records [55] and similar documents [9], for web-page clustering [51], data mining [52], and genetic tests [16, 18, 44]. Also note that, as sample sets can be relatively large, in distributed settings an approximation of the index is oftentimes preferred to its exact calculation. To this end, *MinHash* techniques [9] are often used to estimate the Jaccard index, with remarkably lower computation and communication costs (see Section 2.1).

We define and instantiate a cryptographic primitive for efficient privacy-preserving evaluation of sample set similarity (or EsPRESSo, for short). We present two instantiations, that allow two interacting parties to compute and/or approximate the Jaccard similarity of their private sets, without reciprocally disclosing any information about their content (or, at most, their size). Our main cryptographic building block is *Private Set Intersection Cardinality* (PSI-CA) [21], which we review in Section 2.2. Specifically, we use PSI-CA to privately compute the magnitude of set intersection and union, and we then derive the value of the Jaccard index. As fast (linear-complexity) PSI-CA protocols become available (e.g., [17]), this can be done efficiently, even on large sets. Nonetheless, our work shows that, using MinHash approximations, one can obtain an estimate of the Jaccard index with remarkably increased efficiency, by reducing the size of input sets (thus, the number of underlying cryptographic operations).

Privacy-preserving evaluation of sample set similarity is appealing in many scenarios. We focus on document and multimedia similarity as well as iris matching, and show that privacy is attainable with low overhead. Experiments demonstrate that our generic technique – while not bounded to any specific application – is appreciably more efficient than state-of-the-art protocols that only focus on one specific scenario, while maintaing comparable accuracy. Finally, in the process of reviewing related work, we identify limits and flaws of some prior results.

Organization. The rest of this paper is organized as follows. Next section introduces building blocks, then Section 3 presents our construction for secure computation of Jaccard index and an even more efficient technique to (privately) approximate it. Then, Sections 4, 5, and 6 present our constructions for privacy-preserving similarity evaluation of, respectively, documents, irises, and multimedia content. Finally, Section 7 sketches a very efficient protocol that privately approximates set intersection cardinality, additionally hiding input set sizes, while the paper concludes in Section 8. Appendix A presents some more details on MinHash, and Appendix B shows a flaw in the protocol for secure document similarity in [28].

2 Preliminaries

This section provides some relevant background information on Jaccard index, MinHash techniques, and our main cryptographic building blocks.

2.1 Jaccard Similarity Index and MinHash Techniques

Jaccard Index. One of the most common metrics for assessing the similarity of two sets A and B (hence, of data they represent) is the Jaccard index [24], defined as $J(A, B) = |A \cap B|/|A \cup B|$. Values close to 1 suggest that two sets are very similar, whereas, those closer to 0 indicate that A and B are almost disjoint. Note that the Jaccard index of A and B can be rewritten as a mere function of the *intersection*: $J(A, B) = |A \cap B|/(|A| + |B| - |A \cap B|)$.

MinHash Techniques. Clearly, computing the Jaccard index incurs a complexity linear in set sizes. Thus, in the context of a large number of big sets, its computation might be relatively expensive. In fact, for each pair of sets, the Jaccard index must be computed from scratch, i.e., no information used to calculate $J(A, B)$ can be re-used for $J(A, C)$. (That is, similarity is not a transitive measure.) As a result, an *approximation* of the Jaccard index is often preferred, as it can be obtained at a significantly lower cost, e.g., using MinHash [9]. Informally, MinHash techniques extract a small representation $h_k(S)$ of a set S through deterministic (salted) sampling. This representation has a constant size k, independent from $|S|$, and can be used to compute an approximation of the Jaccard index. The parameter k also defines the expected error with respect to the exact Jaccard index. Intuitively, larger values of k yield smaller approximation errors. The computation of $h_k(S)$ also incurs a complexity linear in set sizes, however, it must be performed *only once* per set, for *any* number of comparisons. Thus, with MinHash techniques, evaluating the similarity of any two sets requires only a constant number of comparisons. Similarly, the bandwidth used by two interacting parties to approximate the Jaccard index of their respective sets is also constant (i.e., $O(k)$).

There are two strategies to realize MinHashes: one employs multiple hash functions, while the other is built from a single hash function. This paper focuses on the former technique. Thus, we defer the description of the latter to Appendix A, which also overviews possible MinHash instantiations.

MinHash with many hash functions. Let $\mathcal{F}$ be a family of hash functions that map items from set U to distinct τ-bit integers. Select k different functions $h^{(1)}(\cdot), \ldots, h^{(k)}(\cdot)$ from $\mathcal{F}$; for any set $S \subseteq U$, let $h^{(i)}_{min}(S)$ be the item $s \in S$ with the smallest value $h^{(i)}(s)$. The MinHash representation $h_k(S)$ of set S is a vector $h_k(S) = \{h^{(i)}_{min}(S)\}^k_{i=1}$. The Jaccard index $J(A, B)$ is estimated by counting the number of indexes i-s, such that that $h^{(i)}_{min}(A) = h^{(i)}_{min}(B)$, and this value is then divided by k. Observe that it holds that $h^{(i)}_{min}(A) = h^{(i)}_{min}(B)$ iff the minimum hash value of $A \cup B$ lies in $A \cap B$.

This measure can be obtained by computing the cardinality of the intersection of $h_k(A)$ and $h_k(B)$ as follows. Each element a_i of the vector $h_k(A)$ is encoded as $\langle a_i, i\rangle$. Similarly, $\langle b_i, i\rangle$ represents the i-th element of vector $h_k(B)$. An unbiased estimate of the Jaccard index of A and B is given by:

$$sim(A,B) = \frac{\left|\{\langle a_i, i\rangle\}_{i=1}^{k} \cap \{\langle b_i, i\rangle\}_{i=1}^{k}\right|}{k} \tag{1}$$

As discussed in [10], if $\mathcal{F}$ is a family of min-wise independent hash functions, then each value of a fixed set A has the same probability to be the element with the smallest hash value, for all functions in $\mathcal{F}$. Specifically, for each min-wise independent hash function $h^{(i)}(\cdot)$ and for any set S, we have that, for any $s_j, s_l \in S$, $\Pr[s_j = h_{min}^{(i)}(S)] = \Pr[s_l = h_{min}^{(i)}(S)]$. Thus, we also obtain that $Pr[h_{min}^{(i)}(A) = h_{min}^{(i)}(B)] = J(A,B)$. In other words, if r is a random variable that is 1 when $h_{min}^{(i)}(A) = h_{min}^{(i)}(B)$ and 0 otherwise, then r is an unbiased estimator of $J(A,B)$; however, in order to reduce its variance, such random variable must be sampled several times, i.e., $k \gg 1$ hash values must be used. In particular, by Chernoff bounds [12], the expected error of this estimate is $O(1/\sqrt{k})$ [9].

Approximating (Jaccard) Similarity of Vectors without MinHash. If one needs to approximate the Jaccard index of two fixed-length *vectors* (rather than sets), one could use other (more efficient) techniques similar to MinHash. Observe that a vector $\overrightarrow{S}$ can be represented as a set $S = \{\langle s_i, i\rangle\}$, where s_i is simply the i-th element of $\overrightarrow{S}$. We now discuss a new efficient strategy to approximate the Jaccard index of two vectors $A = \{\langle a_i, i\rangle\}_{i=1}^{n}$, $B = \{\langle b_i, i\rangle\}_{i=1}^{n}$ of length n, without using MinHash. Our approach incurs constant ($O(k)$) computational and communication complexity, i.e., it is independent from the length of the vectors being compared.

First, select k random values $(r_1, \ldots, r_k)$, for r_i uniformly distributed in $[1, n]$, and compute $A_k = \{\langle a_{r_i}, r_i\rangle\}_{i=1}^{k}$ and $B_k = \{\langle b_{r_i}, r_i\rangle\}_{i=1}^{k}$, respectively. The value $\delta = |A_k \cap B_k|/k$ can then be used to assess the similarity of A and B. We argue that δ is an unbiased estimate of $J(A,B)$: for each $\alpha \in (A_k \cup B_k)$ we have that $\Pr[\alpha \in (A_k \cap B_k)] = \Pr[\alpha \in (A \cap B)]$ since $\alpha \in (A \cap B) \wedge \alpha \in (A_k \cup B_k) \Leftrightarrow \alpha \in (A_k \cap B_k)$. We also have $\Pr[\alpha \in (A \cap B)] = J(A,B)$, thus, δ is indeed an unbiased estimate of $J(A,B)$.

The above algorithm implements a perfect min-wise permutation for this setting: since elements $(r_1, \ldots, r_k)$ are uniformly distributed, for each $i \in [1, k]$ any element in A and B has the same probability of being selected. As such, similar to MinHash with many hash functions, the expected error is also $O(1/\sqrt{k})$.

2.2 Cryptography Background

Private Set Intersection Cardinality (PSI-CA). PSI-CA is a cryptographic protocol involving two parties: Alice, on input $A = \{a_1, \ldots, a_w\}$, and Bob, on input $B = \{b_1, \ldots, b_v\}$, such that Alice outputs $|A \cap B|$, while Bob has no output. In the last few years, several PSI-CA protocols have been proposed, including [17, 21, 32, 54].

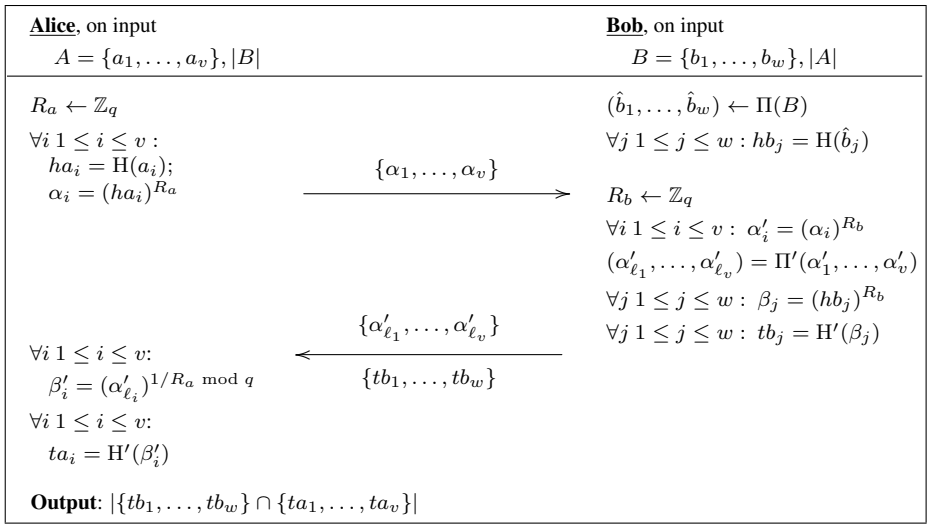

Figure 1: PSI-CA protocol from [17], denoted as DGT12, executed on common input of two primes p and q (such that $q|p-1$), a generator g of a subgroup of size q and two hash functions, H and H′, modeled as random oracles. $\Pi(\cdot)$ and $\Pi'(\cdot)$ denote random permutations. *All computation is mod p.*

De Cristofaro et al.'s PSI-CA [17]. Throughout this paper, we will use the PSI-CA construction presented by De Cristofaro, Gasti, and Tsudik in [17], denoted as DGT12 in the rest of the paper, as it offers the best communication and computation complexities (linear in set sizes). DGT12 is secure, in the presence of semi-honest adversaries, under the Decisional Diffie-Hellman assumption (DDH) in the Random Oracle Model (ROM).

The DGT12 protocol is illustrated in Fig. 1. First, Alice hashes and masks its set items (a_i-s) with a random exponent (R_a) and sends resulting values (α_i-s) to Bob, which "blindly" exponentiates them with its own random value R_b. Bob then shuffles the resulting values (α'_i-s) and sends them to Alice. Then, Bob sends Alice the output of a one-way function, $\mathrm{H}'(\cdot)$, computed over the exponentiations of Bob's (hashed and shuffled) items (b_j-s) to randomness R_b. Finally, Alice tries to match one-way function outputs received from Bob with one-way function outputs computed over the shuffled (α'_i-s) values, stripped of the initial randomness R_a. Alice learns the set intersection cardinality (and nothing else) by counting the number of such matches. Unless they correspond to items in the intersection, one-way function outputs received from Bob cannot be used by Alice to learn related items in Bob's set (under the DDH assumption). Also, Alice does not learn *which* items are in the intersection as the matching occurs using *shuffled* α_i values.

DGT12 requires $O(|A| + |B|)$ *offline* and $O(|A|)$ *online* modular exponentiations in $\mathbb{Z}_p$ with exponents from subgroup $\mathbb{Z}_q$. (Offline operations are computed only once, for any number of interactions and any number of interacting parties). Communication overhead amounts to $O(|A|)$ elements in $\mathbb{Z}_p$ and $O(|B|)$ – in $\mathbb{Z}_q$. (Assuming 80-bit security parameter, $|q| = 160$ bits and $|p| = 1024$ bits.)

Protocol correctness is easily verifiable: for any a_i held by Alice and b_j held by Bob, if $a_i = b_j$ (hence, $ha_i = hb_j$), we obtain:

$$ta_{\ell_i} = \mathrm{H}'(\beta'_{\ell_i}) = \mathrm{H}'({\alpha_{\ell_i}}^{(1/R_a)}) = \mathrm{H}'({ha_i}^{R_b}) = \mathrm{H}'({hb_j}^{R_b}) = \mathrm{H}'(\beta_j) = tb_j$$

Hence, Alice learns set intersection cardinality by counting the number of matching pairs (ta_{ℓ_i}, tb_j).

Adversarial Model. We use standard security models for secure two-party computation, which assume adversaries to be either semi-honest or malicious.[1] As per definitions in [22], protocols secure in the presence of *semi-honest adversaries* assume that parties faithfully follow all protocol specifications and do not misrepresent any information related to their inputs, e.g., size and content. However, during or after protocol execution, any party might (passively) attempt to infer additional information about other party's input.

Whereas, security in the presence of *malicious parties* allows arbitrary deviations from the protocol. Security arguments in this paper are made with respect to ***semi-honest*** participants.

3 Privacy-preserving Sample Set Similarity

This section presents and analyzes our protocols for privacy-preserving computation of sample set similarity, via secure computation of the Jaccard Similarity index.

3.1 Protocols Description

We propose two protocols – both based on Private Set Intersection Cardinality (PSI-CA). The first protocol provides secure and exact computation of the Jaccard index, whereas, the other securely approximates it, using MinHash techniques, incurring significantly lower communication and computation overhead.

Privacy-Preserving Computation of Jaccard Index. Fig. 2 illustrates our first protocol construction for securely computing the Jaccard index. It involves two parties, Alice and Bob, on input sets A and B, respectively, that wish to compute $J(A, B)$ in a privacy-preserving manner, i.e., in such a way that nothing is revealed about their input sets – besides their size and joint Jaccard index.

Given $|A|$, $|B|$ and $J(A, B)$, the size of the intersection between A and B can be easily computed as $|A \cap B| = J(A, B) \cdot (|A| + |B|)/(J(A, B) + 1)$. In other words, knowledge of $(|A|, |B|, J(A, B))$ is equivalent to knowledge of $(|A|, |B|, |A \cap B|)$. Therefore, in our protocol Alice computes $|A \cap B|$ and uses it – together with her input – to derive $J(A, B)$. As it is customary in secure-computation protocols, the size of parties' input is available to the

[1] Hereafter, the term *adversary* refers to protocol participants. Outside adversaries are not considered, since their actions can be mitigated via standard network security techniques.

Privacy-preserving computation of $J(A, B)$
(Run by Alice and Bob, on input, resp., $(A, |B|), (B, |A|)$)

1. Alice and Bob execute PSI-CA on input, respectively, $(A, |B|)$ and $(B, |A|)$
2. Alice learns $c = |A \cap B|$
3. Alice computes $u = |A \cup B| = |A| + |B| - c$
4. Alice outputs $J(A, B) = c/u$

Figure 2: Proposed protocol for privacy-preserving computation of set similarity.

counterpart, thus, it is included as part of the protocol input. The protocol does not assume any specific secure PSI-CA instantiation.

Privacy-Preserving Approximation of Jaccard Index using MinHash. The computation of the Jaccard index, with or without privacy, can be relatively expensive when (1) sample sets are very large, or (2) each set must be compared with a large number of other sets – since for each comparison, all computation must be re-executed *from scratch*. Thus, MinHash is often used to estimate the Jaccard index, trading (bounded) error with appreciably faster computation. In order to jointly and privately approximate $J(A, B)$ as $sim(A, B)$, Alice and Bob agree on k and select a random subset of their sets using the MinHash technique in Section 2.1. In particular, Alice computes $\{\langle a_i, i\rangle\}_{i=1}^{k}$ where $a_i = h_{min}^{(i)}(A)$, and Bob computes $\{\langle b_i, i\rangle\}_{i=1}^{k}$ for $b_i = h_{min}^{(i)}(B)$. Similarity of two sample sets is then computed as $sim(A, B) = |\{\langle a_i, i\rangle\}_{i=1}^{k} \cap \{\langle b_i, i\rangle\}_{i=1}^{k}|/k$ using any secure instantiation of PSI-CA. Therefore, privacy-preserving estimation of the Jaccard index, using multi-hash MinHash, can be reduced to securely computing cardinality of set intersection. The resulting protocol is presented in Fig. 3 below.

Private Jaccard index estimation $sim(A, B)$
Run by Alice and Bob, on common input k and private input $\{\langle a_i, i\rangle\}_{i=1}^{k}$ (for Alice) and $\{\langle b_i, i\rangle\}_{i=1}^{k}$ (for Bob)

1. Alice and Bob execute PSI-CA on input, resp., $(\{\langle a_i, i\rangle\}_{i=1}^{k}, k)$ and $(\{\langle b_i, i\rangle\}_{i=1}^{k}, k)$
2. Alice learns $\delta = |\{\langle a_i, i\rangle\}_{i=1}^{k} \cap \{\langle b_i, i\rangle\}_{i=1}^{k}|$
3. Alice outputs $sim(A, B) = \delta/k$

Figure 3: Proposed protocol for privacy-preserving approximation of set similarity.

It is easy to observe that, compared to the Jaccard index computation (Fig. 2), the use of MinHash leads to executing PSI-CA on smaller sets, as $k \ll \text{Min}(|A|, |B|)$. Communication and computation overhead only depend on k, since inputs to PSI-CA are now sets of k items.

3.2 Security Analysis

Security of Privacy-Preserving Computation of Jaccard Index. Informally, by running the protocol in Fig. 2, parties do not reciprocally disclose the content of their private sets. Alice learns similarity computed as the Jaccard index, while both parties learn the size of the other party's input.

The security of the protocol in Fig. 2 relies on the security of the instantiation of the underlying PSI-CA protocol. In particular we assume that, in the semi- honest model, PSI-CA only reveals $|A \cap B|$ to Alice, while Bob learns nothing besides $|A|$.

Alice and Bob do not exchange any information besides messages related to the PSI-CA protocol. For this reason, a secure implementation of the underlying PSI-CA guarantees that neither Alice nor Bob learn additional information about the other party's set. Since knowledge of $(|A|, |B|, J(A, B))$ is equivalent to knowledge of $(|A|, |B|, |A \cap B|)$, the protocol in Fig. 2 is secure in the semi-honest setting.

Security of Privacy-Preserving Approximation of Jaccard Index. Similar to the protocol in Fig. 2, the security of protocol in Fig. 3 relies on the security of the underlying PSI-CA construction. In particular, $sim(A, B)$ is defined as the size of the intersection between $\{\langle a_i, i\rangle\}_{i=1}^{k}$ and $\{\langle b_i, i\rangle\}_{i=1}^{k}$, divided by a (public) constant k. Therefore, any information Alice and Bob learn about the other party's input can also be used to break the underlying PSI-CA protocol. Since the PSI-CA protocol is assumed to be secure, Alice and Bob do not learn additional information.

k is selected independently from $|A|$ and $|B|$, therefore it does not reveal any information about the two sets. PSI-CA, together with the way input is constructed, conceals the relationship between k and $|A|, |B|$ by not disclosing how many elements $a_i = a_j$ and $b_i = b_j$ for $i \neq j$ on the parties' inputs. Therefore, the protocol does not disclose the size of Alice and Bob's inputs.

Extension of Privacy-Preserving Approximation of Jaccard Index. In the previous protocol, Alice learns some additional information compared to the protocol in Fig. 2. In particular, rather than computing the similarity – and therefore the size of the intersection – of sets A and B, she determines how many elements from a particular subset of A (constructed using MinHash) also appear in the subset selected from B. We now provide a brief overview of how this issue can be fixed efficiently.

Alice and Bob can construct their input sets (i.e., $\{\langle a_i, i\rangle\}_{i=1}^{k}$ and $\{\langle b_i, i\rangle\}_{i=1}^{k}$) using a set of Oblivious Pseudo Random Function (OPRF) evaluations[2] rather than a set of hash functions: Alice and Bob engage in a multi-party protocol where Alice inputs her set $A = \{a_1, \ldots, a_v\}$ and learns a random permutation of $\text{OPRF}_{key_j}(a_1), \ldots, \text{OPRF}_{key_j}(a_v)$ for $1 \leq j \leq k$. Alice constructs her input selecting the smallest value $\text{OPRF}_{key_j}(a_i)$ for each j. Bob constructs his input without interacting with Alice. While the cost of this protocol is linear in the size of the input sets, it is significantly higher than that of protocol Fig. 3.

[2] An Oblivious Pseudo Random Function (OPRF) is a two-party protocol, involving one party, on input key, and another, on input s. At the end of the interaction the former learns nothing, while the latter obtains $f_{key}(s)$, where f is a pseudo-random function.

3.3 Performance Evaluation

3.3.1 Privacy-Preserving Computation of Jaccard Index.

Cost of protocol in Fig. 2 is dominated by that incurred by the underlying PSI-CA protocol. While it could be instantiated using *any* PSI-CA construction, we choose DGT12 in order to maximize efficiency. This protocol, reviewed in Section 2.2, incurs linear communication and computational complexities, thus, overall complexities of protocol in Fig. 2 are also linear in the size of sets. If we were to compute the Jaccard index without privacy, asymptotic complexities would be same as our privacy-preserving protocol – i.e., linear. However, given the lack of cryptographic operations, constants hidden by the big $O(\cdot)$ notation would be much smaller.

To assess the real-world practicality of resulting construction, protocol in Fig. 2 has been implemented in C (with OpenSSL and GMP libraries), using 160-bit random exponents and 1024-bit moduli to obtain 80-bit security. We run experiments on sets such that $|A| = |B| = 1000$. Recall that, in DGT12 items are always hashed (DGT12 assumes ROM), so their size is non-influent. We use select SHA-1 as the hash function, thus, hashed items are 160-bit.

In this setting, protocol in Fig. 2 incurs (i) $\mathbf{0.5s}$ total computation time on a single Intel Xeon E5420 core running at 2.50GHz and (ii) $\mathbf{276}$KB in bandwidth. We omit running times for larger sets since, as complexities are linear, one can easily derive a meaningful estimate of time/bandwidth for virtually any size.

We also implement an optimized prototype that further improves total running time by (1) pipelining computation and transmission and (2) parallelizing computation on two cores. We test the prototype by running Alice and Bob on two PCs equipped with 2 quad-core Intel Xeon E5420 processors running at 2.50GHz, however, we always use (at most) 2 cores. On a conservative stance, we do not allow parties to perform any pre-computation offline. We simulate a 9Mbps link, since, according to [41], it represents the current average Internet bandwidth in US and Canada. In this setting, and again considering $|A| = |B| = 1000$, total running time of protocol in Fig. 2 amounts to $\mathbf{0.23s}$. Whereas, the computation of Jaccard index *without privacy* takes $\mathbf{0.018s}$. Therefore, we conclude that privacy protection, in our experiments, only introduces a (roughly) **12-fold slowdown**, independently from set sizes.

Comparison to prior work. Performance evaluation above does not include any prior solutions, since, to the best of our knowledge, there is no comparable cryptographic primitive for privacy-preserving computation of the Jaccard index. The work in [48] is somewhat related: it targets private computation of the Jaccard index using Private Equality Testing (PET) [34] and deterministic encryption. However, it introduces the need for a non-colluding semi-honest *third party*, which violates our design model. Also, it incurs an impractical number of public-key operations, i.e., *quadratic* in the size of sample sets (as opposed to linear in our case). Finally, additional (only vaguely) related techniques include: (i) work on privately approximating dot product of two vectors, such as, [29, 45], and (ii) probabilistic/approximated private set operations based on Bloom filters, such as, [29, 31]. (None of these techniques,

however, can be used to solve problems considered in this paper.)

3.3.2 Privacy-Preserving Estimation of Jaccard Index with MinHash

We also tested the performance of our construction for privacy-preserving *approximation* of Jaccard similarity, again, using DGT12, i.e., the PSI-CA from [17]. Once again, we select sets with 1000 items, 1024-bit moduli and 160-bit random exponents, and run experiments on two PCs with 2.5GHz CPU and a 9Mbps link. We select $\mathbf{k = 400}$ in order to have an estimated error of about 5%. (Recall that, as mentioned in Section 2.1 the error is approximated as $1/\sqrt{k}$).

In this setting, the total running time of protocol in Fig. 3 amounts to $\mathbf{0.09s}$ – less than half compared to the one in Fig. 2. Whereas, in the same setting, the approximation of Jaccard index *without privacy* takes $\mathbf{0.007s}$. Thus, the slow-down factor introduced by the privacy-protecting layer (similar to the protocol proposed in Section 3.1) is **12-fold**. Again, note that times for different values of k can be easily estimated since the complexity of the protocol is linear.

Comparison to Prior Work. The estimation of set similarity through MinHash – whether privacy-preserving or not – requires counting the number of times for which it holds that $h^{(i)}_{min}(A) = h^{(i)}_{min}(B)$, with $i = 1, \ldots, k$. We have denoted this number as δ. Protocol in Fig. 3 above attains secure computation of δ through privacy-preserving set intersection cardinality. However, it appears somewhat more intuitive to do so by using the approach proposed by [4] in the context of social-network friends discovery. Specifically, in [4], Alice and Bob compute, resp., $\{a_i\}_{i=1}^k$ and $\{b_i\}_{i=1}^k$, just like in our protocol. Then, Alice generates a public-private keypair (pk, sk) for Paillier's additively homomorphic encryption cryptosystem [43] and sends Bob $\{z_i = Enc_{pk}(a_i)\}_{i=1}^k$. Bob computes $\{(z_i \cdot Enc_{pk}(-b_i))^{r_i}\}_{i=1}^k$ for random r_i's and returns the resulting vector of ciphertexts after shuffling it. Upon decryption, Alice learns δ by counting the number of 0's. Nonetheless, the technique proposed by [4] actually incurs an increased complexity, compared to our protocol in Fig. 3 (instantiated with DGT12). Assuming 80-bit security parameters, thus, 1024-bit moduli and 160-bit subgroups, and 2048-bit Paillier moduli, and using m to denote a multiplication of 1024-bit numbers, multiplications of 2048-bit numbers count for $4m$. Using square-and-multiply, exponentiations with q-bit exponents modulo 1024-bit count for $(1.5|q|)m$. In [4], Alice performs k Paillier encryptions (i.e., $2k$ exponentiations and k multiplications) and k decryptions (i.e., k exponentiations and multiplications), while Bob computes k exponentiations and multiplications. Therefore, the total computation complexity amounts to $(6 \cdot 4 \cdot 1.5 \cdot 1024 + 4 \cdot 4)mk = 36,880m$. Whereas, our approach (even without pre-computation) requires both Alice and Bob to perform $4k$ exponentiations of 160-bit numbers modulo 1024-moduli and $2k$ multiplications, i.e., $(4 \cdot 1.5 \cdot 160 + 2)km = 962mk$, thus, our protocol achieves a 38-fold efficiency improvement. Communication overhead is also higher in [4]: it amounts to $(2 \cdot 2048)k$ bits; whereas, using PSI-CA, we need to transfer $(2 \cdot 1024 + 160)k$ bits, i.e., slightly more than half the traffic.

4 Privacy-Preserving Document Similarity

After building efficient (linear-complexity) primitives for privacy-preserving computation/approximation of Jaccard index, we now explore their applications to a few compelling problems. We start with evaluating the similarity of two documents, which is relevant in many common applications, including copyright protection, file management, plagiarism prevention, duplicate submission detection, law enforcement. In last few years, the security community has started investigating privacy-preserving techniques to enable detection of similar documents without disclosing documents' actual content. Below, we first review prior work and, then, present our technique for efficient privacy-preserving document similarity.

4.1 Prior Work

The work in [27] (later extended in [39]) is the first to realize *privacy-preserving document similarity*. It realizes secure computation of the *cosine similarity* of vectors representing the documents, i.e., each document is represented as the list of words appearing in it, along with the normalized number of occurrences. Recently, Jiang and Samanthula [28] have proposed a novel technique relying on the Jaccard index and N-gram based document representation [38]. (Given any string, an N-gram is a substring of size N). According to [28], the N-gram based technique presents several advantages over cosine similarity: (1) it improves on finding *local similarity*, e.g., overlapping of pieces of texts, (2) it is language-independent, (3) it requires a much simpler representation, and (4) it is less sensitive to document modification. We overview it below.

Documents as sets of *N*-grams. A document can be represented as a set of N-grams contained in it. To obtain such a representation, one needs to remove spaces and punctuation and build the set of successive N-grams in the document. An example of a sentence, along with its N-gram representation (for $N = 3$), is illustrated in Fig. 4. The similarity of two documents can then be estimated as the *Jaccard index* of the two corresponding sets of N-grams. In the context of document similarity, experts point out that 3 results as a good choice of N [9].

the quick brown fox jumps over the lazy dog
↓
{azy, bro, ckb, dog, ela, equ, ert, fox, hel, heq, ick, jum, kbr, laz, mps, nfo,
ove, own, oxj, pso, qui, row, rth, sov, the, uic, ump, ver, wnf, xju, ydo, zyd}

Figure 4: Tri-gram representation.

To enable privacy-preserving computation of Jaccard index, and therefore estimation of document similarity, Jiang and Samanthula [28] propose a two-stage protocol based on *Paillier*'s additively homomorphic encryption [43]. Suppose Alice wants to privately evaluate the similarity of her document D_A against a list of n documents held by Bob, i.e., $D_{B:1}, \ldots, D_{B:n}$. First, Bob generates a global space, $|S|$, of tri-grams based on his document collection. This

way, D_A as well as each of Bob's document, $D_{B:i}$, for $i = 1, \ldots, n$, can be represented as binary vectors in the global space of tri-grams: each component is 1 if the corresponding tri-gram is included in the document and 0 otherwise. In the following, we denote with A the representation of D_A and with B_i that of $D_{B:i}$.

Then, Alice and Bob respectively compute random shares a and b_i such that $a + b_i = |A \cap B_i|$. Next, they set $c = |A| - a$ and $d_i = |B| - b_i$. Finally, Alice and Bob, on input (a, c) and (b_i, d_i), resp., execute a Secure Division protocol (e.g., [11, 1]) to obtain $(a + b_i)/(c + d_i) = |A \cap B_i|/|A \cup B_i| = J(A, B_i)$.

The computational complexity of the protocol in [28] amounts to $O(|S|)$ Paillier encryptions performed by Alice, and $O(n \cdot |S|)$ modular multiplications – by Bob. Whereas, communication overhead amounts to $O(n \cdot |S|)$ Paillier ciphertexts.

Flaw in [28]. Unfortunately, protocol in [28] *is not secure*, since Bob has to disclose his global space of tri-grams (i.e., the set of all tri-grams appearing in his document collection). Therefore, Alice can passively check whether or not a word appears in Bob's document collection. Actually, Alice can learn much more, as we show in Appendix B. We argue that this flaw could be fixed by considering the global space of tri-grams as the set of all possible tri-grams, thus, avoiding the disclosure of Bob's tri-grams set. Assuming that documents are stripped of any symbol and contain only lower-cased letters and digits, we obtain $S = \{a, b, \ldots, z, 0, 1, \ldots, 9\}^3$. Unfortunately, this modification would tremendously increase computation and communication overhead.

4.2 Our Construction

As discussed in Section 3, we can realize privacy-preserving computation of the Jaccard index using PSI-CA. To privately evaluate the similarity of documents D_A and any document $D_{B:i}$, Alice and Bob execute protocol in Fig. 5. Function Tri-Gram$(\cdot)$ denotes the representation of a document as the set of tri-grams appearing in it.

Alice (D_A)		Bob $(D_{B:i})$
$A \leftarrow$ Tri-Gram(D_A)		$B_i \leftarrow$ Tri-Gram$(D_{B:i})$
	$\{\longrightarrow, \longleftarrow\}$	
$\lvert A \cap B_i \rvert \leftarrow$ PSI-CA(A, B_i)		
Output Similarity as $J(A, B_i) = \dfrac{\lvert A \cap B_i \rvert}{\lvert A \rvert + \lvert B_i \rvert - \lvert A \cap B_i \rvert}$		

Figure 5: Privacy-preserving *evaluation* of document similarity of documents D_A and $D_{B:i}$.

Complexity. Complexity of protocol in Fig. 5 is bounded by that of the underlying PSI-CA construction. Using DGT12, computational complexity amounts to $O(|A| + |B_i|)$ modular exponentiations, whereas, communication overhead – to $O(|A| + |B_i|)$. Observe that, in the setting where Alice holds one documents and Bob a collection of

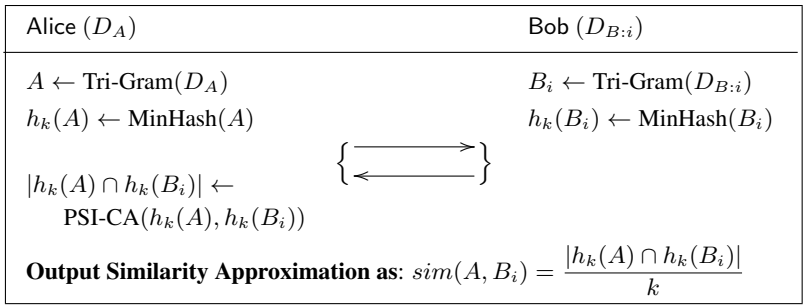

Figure 6: Privacy-preserving *approximation* of document similarity of documents D_A and $D_{B:i}$.

n documents, complexities should be amended to $O(n|A| + \sum_{i=1}^{n} |B_i|)$. However, due to the nature of DGT12, Bob can perform $O(\sum_{i=1}^{n} |B_i|)$ computation *off-line*, ahead of time. Hence, total *online* computation amounts to $O(n|A|)$.

More efficient computation using MinHash. As discussed in Section 2.1, one can approximate the Jaccard index by using MinHash techniques, thus, trading off accuracy with significant improvement in protocol complexity. The resulting construction is similar to the one presented above and is illustrated in Fig. 6. It adds an intermediate step between the tri-gram representation and the execution of PSI-CA: Alice and Bob apply MinHash to sets A and B_i, respectively, and obtain $h_k(A)$ and $h_k(B_i)$. The main advantage results from the fact that PSI-CA is now executed on smaller sets, of constant size k, thus, achieving significantly improved communication and computational complexities. Again, note that the error is bounded by $O(1/\sqrt{k})$.

Performance Evaluation. We now compare the performance of our constructions to the most efficient prior technique, i.e., the protocol in [28] (that, unfortunately, is insecure). We consider the setting of [28], where Bob maintains a collection of n documents. Recall that our constructions use DGT12, i.e., the PSI-CA in [17]. Assuming 80-bit security parameters, we select 1024-bit moduli and 160-bit random exponents. As [28] relies on Paillier encryption, it uses 2048-bit moduli and 1024-bit exponents. In the following, let m denote a multiplication of 1024-bit numbers. Multiplications of 2048-bit numbers count for $4m$. Modular exponentiations with q-bit exponents modulo 1024-bit count for $(1.5|q|)m$. The protocol in [28] requires $O(|S|)$ Paillier encryptions and $O(n \cdot |S|)$ modular multiplications. We need $|S| = 36^3 = 46,656$ as we consider 3-grams and 26 alphabet letters plus [0–9] digits. Therefore, the total complexity amounts to $(4 \cdot 1.5 \cdot 1024 + 4n)|S|m = (6144 + 4n)|S|m \approx (2.9 \cdot 10^8 + 1.9 \cdot 10^5 n)m$.

n	[28]	Fig. 5	Fig. 6	
			$k = 100$	$k = 40$
10	9.5 mins	6.3 secs	0.05 secs	0.05 secs
10^2	9.9 mins	63 secs	1.9 secs	1.9 secs
10^3	12.7 mins	10.4 mins	48 secs	19.2 secs
10^4	40.7 mins	1.74 hours	8 mins	3.2 mins
10^5	5.3 hours	17.4 hours	1.2 hours	32 mins

Table 1: Computation time of privacy-preserving document similarity. n denotes the number of documents.

Our construction above requires $(2 \cdot 1.5 \cdot 160n(max(|A|, \{B_i\}_{i=1}^{n}))\mathsf{m}$ for the computation of Jaccard index similarity and $(1.5 \cdot 160nk)\mathsf{m}$ for its approximation. Thus, to compare performance of our protocol to that of [28], we need to take into account the dimensions of A, B_i, as well as n and k. To this end, we collected 393 scientific papers from the KDDcup dataset of scientific papers published in ArXiv between 1996 and 2003 [15]. The average number of different tri-grams appearing in each paper is 1307. Therefore, cost of our two techniques can be estimated as $(2 \cdot 1.5 \cdot 160 \cdot 1307n)\mathsf{m}$ and $(1.5 \cdot 160 \cdot nk)\mathsf{m}$, respectively. Thus, our technique for privacy-preserving document similarity is faster than [28] for $n < 2000$. Furthermore, using MinHash techniques, complexity is always faster (and of at least one order of magnitude), using both $k = 40$ and $k = 100$. Also, recall that, as opposed to ours, the protocol in [28] is ***not*** secure.

Assuming that it takes about $1\mu s$ to perform modular multiplications of 1024-bit integers (as per our experiments on a single Intel Xeon E5420 core running at 2.50GHz), we report estimated running times in Table 1 for increasing values of n (i.e., the number of Bob's documents).

We performed some statistical analysis to determine the real magnitude of the error introduced by MinHash, when compared to the Jaccard index without MinHash. Our analysis is based on the trigrams from documents in the KDDcup dataset [15], and confirms that the average error is within the expected bounds: for $k = 40$, we obtained an average error of 14%, while for $k = 100$ the average error was 9%. This is acceptable, considering that the Jaccard index actually provides a normalized *estimate* of the similarity between two sets, not a definite metric.

5 Privacy-Preserving Iris Matching

Advances in biometric recognition enable the use of biometric data as a practical mean of authentication and identification. Today, several governmental agencies around the world perform large-scale collections of different biometric features. As an example, the US Department of Homeland Security (DHS) collects face, fingerprint and iris images, from visitors within its US-VISIT program [53]. Iris images are also collected from all foreigners, by the United Arab Emirates (UAE) Ministry of Interior, as well as fingerprints and photographs from certain types of travelers [23].

While biometry serves as an excellent mechanism for identification of individuals, biometric data is, undeniably, extremely sensitive and must be subject to minimal exposure. As a result, such data cannot be disclosed arbitrarily. Nonetheless, there are many legitimate scenarios where biometric data should be shared, in a controlled way, between different entities. For instance, an agency may need to determine whether a given biometric appears on a government watch-list. As agencies may have different clearance levels, privacy of biometric's owner should be preserved if no matches are found, but, at the same time, unrestricted access to the watch-list cannot be granted.

5.1 Prior Work

As biometric identification techniques are increasingly employed, related privacy concerns have been investigated by the research community [13]. A number of recent results address the problem of privacy-preserving face recognition. The work in [19] is the first to present a secure protocol, based on Eigenfaces, later improved by [46]. Next, [42] designs a new privacy-preserving face recognition algorithm, called SCiFI. Furthermore, the protocol in [6] realizes privacy-preserving fingerprint identification, using FingerCodes [25]. FingerCodes use texture information from a fingerprint to compare two biometrics. The algorithm is not as discriminative as traditional fingerprint matching techniques based on location of minutiae points, but it is adopted in [6] given its suitability for efficient *privacy-preserving* realization. Among all biometric techniques, this paper focuses on *iris-based* identification. The problem of privacy-preserving iris matching has been introduced by Blanton and Gasti in [7], who propose an approach based on a combination of garbled circuits [56] and homomorphic encryption.

5.2 Our Construction

A (human) iris can be digitalized as an n-bit string $S = s_1 s_2 \cdots s_n$ with an n-bit mask $M_S = ms_1 ms_2 \cdots ms_n$. The mask indicates which bits of S have been read reliably. In particular, the i-th bit of S should be used for matching only if the i-th bit of M_S is set to 1. A common value for n, which we use in our experiments, is 2048. As, during an iris scan, the subject may rotate its head, a right or left shift can occur in the iris representation, depending on the direction of the rotation. Therefore, the distance between two irises A and B is computed as the minimum distance between all rotations of A and the representation of B. In practice, it is reasonable to assume that the rotation is limited to a shift of at most 5 positions towards left/right [7].

The matching between two irises, A and B, is computed via the *Weighted Hamming Distance* (WHD) of the samples. Let $M = (M_A \wedge M_B)$; WHD is computed as:

$$\mathrm{WHD}(A, B, M) = \frac{HD(A \wedge M, B \wedge M)}{\|M\|} \tag{2}$$

where $\|\cdot\|$ denotes hamming weight, i.e. the number of string bits set to 1. Given a threshold t, if $\mathrm{WHD}(A, B, M) < t$, we say that irises A and B are *matching*. (Assuming a maximum rotation of 5 positions, the distance must be computed 11 times.)

In the following, we propose a probabilistic technique for privately estimating of $\mathrm{WHD}(A, B, M)$, that relies on the construction for privacy-preserving estimation of Jaccard index based on MinHash (introduced in Section 3.1). The error on the approximation is bounded by the MinHash parameter k.

Proposed protocol is illustrated in Fig. 7. Given any two n-bit strings $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ and a list of k values $R = (r_1, \ldots, r_k)$, with $r_i \in [1, n]$, $r_i \neq r_j$ for all $i \neq j$, we define probabilistic function

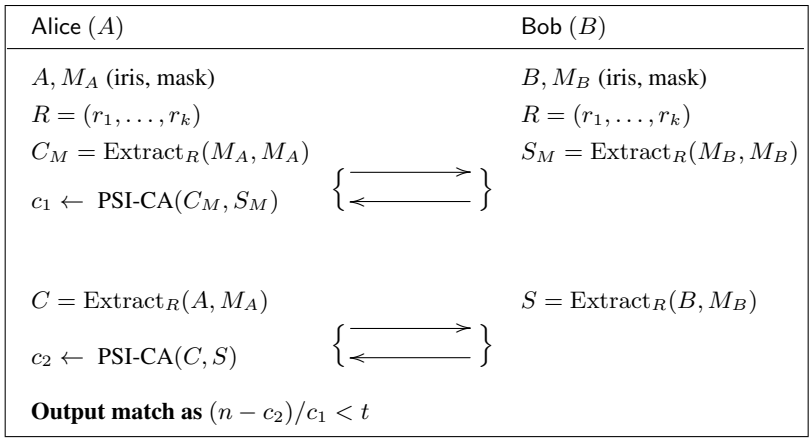

Figure 7: Privacy-preserving iris matching of biometric A and B.

$\text{Extract}_R : \{0,1\}^n \times \{0,1\}^n \to (\{0,1\} \times \{1,\ldots,n\})^k$ as:

$$\text{Extract}_R(X, Y) = \{w_{r_1}, \ldots, w_{r_k}\}, \text{where } w_{r_i} = \begin{cases} \langle x_{r_i}, r_i \rangle & \text{if } y_{r_i} = 1 \\ \langle r, r_i \rangle \text{ with } r \leftarrow \{0,1\}^\tau & \text{otherwise} \end{cases}$$

where $x \leftarrow Y$ represents uniform random sampling of element x from set Y.

Intuitively, for each value r_i in R, $\text{Extract}_R(X, Y)$ selects the r_i-th bit of X and encodes it with r_i (e.g., concatenates the two), if the r_i-th bit of Y is 1. If the r_i-th bit of Y is 0, the function selects a random value end encodes it with r_i.

Given A, M_A, B, M_B, Alice and Bob privately determine $\text{WHD}(A, B, M_A \wedge M_B)$:

- Alice and Bob negotiate R.
- Alice computes $C_M = \text{Extract}_R(M_A, M_A)$; Bob computes $S_M = \text{Extract}_R(M_B, M_B)$.
- Alice and Bob engage in a PSI-CA protocol where their inputs are C_M and S_M and Alice learns the output c_1 of PSI-CA. c_1 corresponds to the number of bits set to 1 in both M_A and M_B at indices specified by R.
- Alice computes $C = \text{Extract}_R(A, M_A)$; similarly, Bob computes $S = \text{Extract}_R(B, M_B)$.
- Alice and Bob interact in a PSI-CA protocol with input C and S respectively; at the end of the protocol, Alice learns c_2, i.e., the size of the intersection of the subsets of A and B defined by R.
- Biometric A matches B iff $(n - c_2)/c_1 < t$.

5.3 Comparison to prior work

We now compare our technique for privacy-preserving iris matching to prior work, namely the technique in [7]. We compare our approach with the protocol in [7] because, at the time of writing, it provides the best performance for privacy-preserving comparison of iris codes. First, observe that protocol in Fig. 7 estimates the Weighted Hamming Distance with bounded error, whereas, construction in [7] yields its exact computation. However, as we discuss below, the error incurred by our technique is low enough to be used in practice and achieves reduced computational complexity. In fact, our probabilistic protocol could be used to perform a fast, preliminary test: if differences between two irises are significant, then there is no need for further tests. Otherwise, the two parties can engage in the protocol in [7] to obtain (in a privacy-preserving way) a precise result. Next, as opposed to the technique in [7], Alice also learns an estimate on the number of bits set to 1 in the combined mask $M_A \wedge M_B$, but not their position. However, this information is not sensitive, thus, it does not leak any information about the iris sampled by Alice or Bob.

Optimization. As discussed above, it is reasonable to assume that Bob (e.g., the Department of Homeland Security) holds a database with a large number of biometric samples, whereas, Alice (e.g., the Transportation Security Administration) has only one or few samples that she is searching in Bob's database. To this end, we now show how the protocol in Fig. 7 can be optimized, by pre-computing several expensive operations offline, for such a scenario.

Note that Bob can perform the offline phase of DGT12 (see Fig. 1) on all bits of his biometric samples: unlike the protocol in [7], this is required *only once*, independently on the number of interactions between Bob and any user.

After negotiating with Bob the values $R = (r_1, \ldots, r_k)$, and before receiving her input, Alice pre-computes k pairs $\langle \alpha_{0,i} = H(\langle 0, r_i \rangle)^{R'_c}, \alpha_{1,i} = H(\langle 1, r_i \rangle)^{R'_c} \rangle$. (This assumes the use of DGT12.) Once Alice's mask has been revealed, she constructs the corresponding encrypted representation by simply selecting the appropriate element from each pair. Similarly, she computes k triples $(\alpha_{0,i}, \alpha_{1,i}, \alpha_{\rho,i})$ where $\alpha_{0,i}, \alpha_{1,i}, \alpha_{\rho,i}$ represent 0, 1 and a random element in $\{0,1\}^\tau$, respectively. Alice later uses such triples to represent each bit β_i of her iris sample as $\alpha_i = \alpha_{\beta,i}$ if the corresponding bit in the mask is 1, else, as $\alpha_i = \alpha_{\rho,i}$. During the online phase, Alice selects the appropriate pre-computed values to match the mask and the iris bits. Similarly, Bob inputs the selected bits of each record's mask and iris into the PSI-CA protocol.

Performance Comparison. In Table 2, we report running times from implementations of, respectively, our protocol in Fig. 7 and technique in [7]. We assume that about 75% of the bits in the mask are set to 1 (like in [7]). We set the length of each iris and mask to 2048 bits and the database size to 320 irises, which is the number used in prior work. All tests are run on a single Intel Xeon E5420 core running at 2.50GHz. We set $k = 25$, thus, obtaining an expected error in the order of $1/\sqrt{25}$, i.e., 20%.

Observe that *online* cost incurred by Bob with our technique is significantly lower compared to that of protocol

Protocol in Fig. 7				*Protocol in* [7]	
		Offline	**Online**	**Offline**	**Online**
Bob	± 5-bit rot.	0.13 ms + 5.8 s/rec	71.5 ms/rec	2.8 s + 71.55 ms/rec	97.2 ms + 134.28 ms/rec
	no rot.	0.13 ms + 530 ms/rec	6.5 ms/rec	2.6 s + 6.48 ms/rec	97.2 ms + 12.33 ms/rec
Alice	± 5-bit rot.	71.63 ms	71.5 ms/rec	12.2 s + 3 ms/rec	20.34 ms/rec
	no rot.	6.63 ms	6.5 ms/rec	11.7 s + 0.27 ms/rec	1.8 ms/rec

Table 2: Computation overhead of our randomized iris matching technique in Fig. 7 and that of [7]. Experiments are performed with 5-bit left/right rotation and with no rotation of the iris sample. "rot" abbreviates "rotation" and "rec" – "record".

in [7]. Whereas, it is higher for Alice. Nonetheless, summing up the computation overhead incurred by both Alice and Bob, our protocol always results faster that the one in [7] for the online computation.

The *offline* cost imposed on Bob is about twice as high as its counterpart in protocol from [7]. However, in our protocol, the offline part is done once, for all possible interactions, independently from their number. Whereas, in [7], the offline computation needs to be performed anew, for *every* interaction. In settings where Bob interacts frequently with multiple entities, this may significantly effect protocol's overall efficiency. Furthermore, the offline cost imposed on Alice is markedly lower (several orders of magnitudes) using our technique.

We conclude that the protocol in Fig. 7 improves, in many settings, overall efficiency compared to state of the art. However, it introduces a maximum error of about 20%, whereas, the scheme in [7] compute the exact – rather than approximate – outcome of an iris comparison. Thus, a good practice is to use the scheme in Fig. 7 to perform an initial selection of relevant biometric samples, using a threshold $t' > t$, in order to compensate for the error. The final matching on selected samples can then be done, in a privacy-preserving manner, using the protocol in [7].

6 Privacy-Preserving Multimedia File Similarity

Amid widespread availability of digital cameras, digital audio recorders, and media-enabled smartphones, users generate a staggering amount of *multimedia* content. As a result, secure online storage (and management) of large volumes of multimedia data becomes increasingly desirable. According to [57], YouTube received more than 13 million hours of video in 2010, and 48 hours of new content are uploaded *every minute* (i.e., 8 years of video each day). Similarly, Flickr users upload about 60 photos every second.

Such an enormous amount of multimedia data prevents manual content curation – e.g., manually *tagging* all uploaded content to allow textual searches. For this reason, in recent years research has focused on automated tools for feature extraction and analysis on multimedia content. A prominent example is Content-Based Image Retrieval (CBIR) [49]. It allows automatic extraction of features from an image, a video, an audio file or any other multimedia content. These features can then be compared across different files, establishing for example *how similar* two documents are. There are several available techniques to implement CBIR, including search techniques based on color histograms [50], bin similarity coefficients [40], texture for image characterization [37], shape features [47], edge

directions [26], and matching of shape components such as corners, line segments or circular arcs [14].

In this section we design a generic privacy-preserving technique for comparing multimedia documents by comparing their features. Our technique is based on Jaccard and MinHash, and is oblivious to the specific type of features used to perform comparison.We implement a small prototype, which we use to evaluate the performance of our approach.

Prior Work. Motivated by the potential sensitivity of multimedia data, the research community has begun to develop mechanisms for secure signal processing. For instance, authors in [20] are the first to investigate secure signal processing related to multimedia documents. Then, the work in [35, 36] introduces two protocols to search over encrypted multimedia databases. Specifically, it extracts 256 visual features from each image. Then, files are encrypted in a distance-preserving fashion, so that encrypted features can be directly compared for similarity evaluation. Similarity is computed using the Jaccard index between the visual features of searched image and those of images in a database. However, the security of the scheme relies on order-preserving encryption (used to mask frequencies of recurring visual features), which is known to provide only a limited level of security [8].

Our Approach. We use the Jaccard index to assess the similarity of multimedia files. As showed in Section 3, we can do so, in a privacy-preserving way, using protocol in Fig. 2. Our Privacy-preserving evaluation of multimedia file similarity protocol is presented in Fig. 8. We denote a multimedia file owned by Alice as F_A, and a file owned by Bob as $F_{B:i}$.

Alice (F_A)		Bob ($F_{B:i}$)
$A \leftarrow \text{Extract}(F_A)$		$B_i \leftarrow \text{Extract}(F_{B:i})$
	$\{ \longrightarrow \}$	
	$\{ \longleftarrow \}$	
$\lvert A \cap B_i \rvert \leftarrow \text{PSI-CA}(A, B_i)$		
Output Similarity as $J(A, B_i) = \dfrac{\lvert A \cap B_i \rvert}{\lvert A \rvert + \lvert B_i \rvert - \lvert A \cap B_i \rvert}$		

Figure 8: Privacy-preserving evaluation of multimedia file similarity.

Our approach is independent of the underlying feature extraction algorithm, even though protocol accuracy naturally relies on the quality of the feature extraction phase. Once features have been extracted, our privacy-preserving protocols only reveal their similarity, thus, without disclosing the features themselves. As an example, we instantiate our techniques for privacy-preserving *image* similarity. Our approach for feature extraction is based on [36], since its accuracy is reasonable enough for real-world use, using color histograms in the color space of Hue, Saturation and Value (HSV). Thus, our scheme achieves the same accuracy of [36], in terms of precision and recall.

Once again, to obtain improved efficiency, similarity can be approximated using MinHash techniques, as per protocol in Fig. 9. In this case, protocol performance and accuracy depend on the MinHash parameter k.

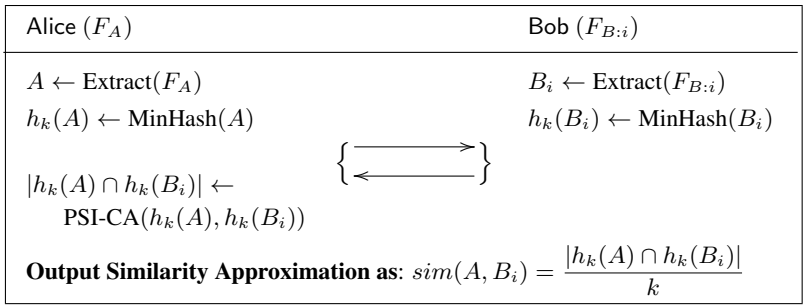

Figure 9: Privacy-preserving *approximation* of multimedia file similarity.

		Offline	Online
Bob	Exact	0.13 ms + 33.28 ms/record	33.28 ms/record
	Approximate	0.13 ms + 13 ms/record	13 ms/rec
Alice	Exact	66.69 ms	33.28 ms/record
	Approximate	26.13 ms	13 ms/record

Table 3: Computation cost of our multimedia documents similarity protocol.

Performance Evaluation. We test our technique with the same dataset used by [36], i.e., 1000 images from the standard Corel dataset. We extract 256 features from each image, for a total of 256,000 features for the whole database. We envision a user, Alice, willing to assess similarity of an image against an image database, held by Bob. We run our protocol for privately computing the Jaccard index ("Exact" rows in Table 3) and for estimated similarity, using MinHash with $k = 100$ ("Approximate" row). Table 3 summarizes our experiments. All tests are run on a single Intel Xeon E5420 core running at 2.50GHz and show that privacy protection is attainable at a very limited cost.

Remark that a thorough performance comparison between our protocol and related work is out of the scope of this paper, since the main effort of prior work has been achieving high accuracy in similarity detection, rather improving efficiency. Thus, we defer it to future work. Nonetheless, the authors of [36] report that the running time of their protocol is in the order of 1 second per image, on a hardware comparable to our testbed (a dual-core 3GHz PC with 4GB of RAM). Therefore, it is safe to assume that our protocol for privacy-preserving multimedia file similarity is about one order of magnitude faster than available techniques, even without considering pre-computation.

7 Faster and Size-Hiding (Approximated) PSI-CA

Privacy-preserving computation of set intersection cardinality has been investigated quite extensively by the research community [17, 21, 32, 54], motivated by several interesting applications, including: privacy-preserving authentication and key exchange protocols [2], data and association rule mining [54], genomic applications [5], healthcare [32], policy-based information sharing [17], anonymous routing [58], and – as argued by this paper – sample set similarity.

In many of the application scenarios, however, it may be enough to obtain an estimation, rather than the exact measure, of set intersection cardinality. For instance, if PSI-CA is used to privately quantify the number of common

social-network friends (e.g., to assess profile similarity) [33], then one may want to trade off a bounded accuracy loss with a significant improvement in protocol overhead (and without sacrificing the level of attained privacy protection). Clearly, such an improved construction is particularly appealing whenever participants' input sets are very large.

Using MinHash techniques, we can construct a protocol for privacy-preserving estimation of set intersection cardinality with (constant) computation and communication complexities, that only depend on the MinHash parameter – i.e., $O(k)$. Proposed construction is illustrated in Fig. 10. While we tolerate a bounded accuracy loss depending on MinHash's parameter k, i.e., $O(1/\sqrt{k})$, observe that our protocol achieves the same, provably-secure, privacy guarantees as if we ran PSI-CA on whole sets.

Privacy-preserving approximation of $|A \cap B|$

Run by Alice and Bob, on input, resp., A and B

1. Alice and Bob compute, $\{\langle a_i, i\rangle\}_{i=1}^{k}$ and $\{\langle b_i, i\rangle\}_{i=1}^{k}$, resp., using multi-hash MinHash where: $a_i \stackrel{\text{def}}{=} h_{min}^{(i)}(A)$ and $b_i \stackrel{\text{def}}{=} h_{min}^{(i)}(B)$
2. Alice and Bob execute PSI-CA on input, resp., $(\{\langle a_i, i\rangle\}_{i=1}^{k}, k)$ and $(\{\langle b_i, i\rangle\}_{i=1}^{k}, k)$
3. Alice learns $\delta = |\{\langle a_i, i\rangle\}_{i=1}^{k} \cap \{\langle b_i, i\rangle\}_{i=1}^{k}|$
4. Bob sends w to Alice
5. Alice outputs $\delta \cdot (v + w)/(1 + \delta)$

Figure 10: Our technique for Approximated Private Set Intersection Cardinality.

Size-Hiding. Another factor motivating the use of MinHash techniques for PSI-CA is related to input size secrecy. Available PSI-CA protocols always disclose, from the execution, at least an upper bound on input set sizes. Whereas, protocol in Fig. 10 conceals – unconditionally – Alice's set size, thus, achieving *Size-Hiding* Private Set Intersection Cardinality. With this protocol, Alice and Bob do not need to disclose $|A|$ and $|B|$. Rather, public protocol input includes k, which is ***independent from*** $|A|$ ***and*** $|B|$. Because secure PSI-CA, used as building block for the protocol in Fig. 10, does not leak information about the input sets, neither party learns information about the ratio between k and $|A|$, $|B|$. Considering recent results motivating the need for size-hiding features in private set operations (see [3]), this additional feature is particularly valuable.

Note: While we leave as part of future work a thorough experimental performance evaluation, observe that PSI-CA and the approximated and size-hiding protocol (using MinHash) are essentially the same protocols. The latter runs on smaller, constant-size input (k): since the protocols have linear complexities, it is straightforward to assess the performance spread.

8 Conclusion

This paper introduced the first efficient construction for privacy-preserving evaluation of sample set similarity, relying on the Jaccard index measure. We also presented an efficient randomized protocol that approximates, with bounded error, this similarity index. Our techniques are generic and practical enough to be used as a basic building block for a wide array of different privacy-preserving functionalities, including document and multimedia file similarity, biometric matching, genomic testing, similarity of social profiles, and so on. Experimental analyses support our efficiency claims and demonstrate improvements over prior results.

As part of future work, we plan to study privacy-preserving computation of other similarity measures, as well as to further investigate additional applications and extensions.

Acknowledgments. The work of Carlo Blundo has been supported, in part, by the Italian Ministry of Research (MIUR), under project n. 2010RTFWBH "Data-Driven Genomic Computing (GenData 2020)." We gratefully acknowledge Elaine Shi for providing us with the idea of genetic paternity testing as one of our motivating examples. Finally, we thank the Journal of Computer Security's (anonymous) reviewers, whose comments helped us improve papers' content and presentation.

References

[1] M. J. Atallah, M. Bykova, J. Li, K. B. Frikken, and M. Topkara. Private collaborative forecasting and benchmarking. In *WPES*, 2004.

[2] G. Ateniese, M. Blanton, and J. Kirsch. Secret handshakes with dynamic and fuzzy matching. In *NDSS*, 2007.

[3] G. Ateniese, E. De Cristofaro, and G. Tsudik. (If) size matters: Size-Hiding Private Set Intersection. In *PKC*, 2011.

[4] E. Baglioni, L. Becchetti, L. Bergamini, U. Colesanti, L. Filipponi, A. Vitaletti, and G. Persiano. A lightweight privacy preserving sms-based recommendation system for mobile users. In *RecSys*, 2010.

[5] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik. Countering gattaca: efficient and secure testing of fully-sequenced human genomes. In *CCS*, 2011.

[6] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva. Privacy-preserving fingercode authentication. In *MM&Sec*, 2010.

[7] M. Blanton and P. Gasti. Secure and Efficient Protocols for Iris and Fingerprint Identification. In *ESORICS*, 2011.

[8] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, 2009.

[9] A. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, 1997.

[10] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *STOC*, 1998.

[11] P. Bunn and R. Ostrovsky. Secure two-party k-means clustering. In *CCS*, 2007.

[12] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 1952.

[13] S. Cimato, M. Gamassi, V. Piuri, R. Sassi, and F. Scotti. *Privacy in Biometrics*. Wiley-IEEE Press, 2009.

[14] S. D. Cohen and L. J. Guibas. Shape-based image retrieval using geometric hashing. In *ARPA Image Understanding Workshop*, 1997.

[15] Cornell Univ. KDDCUP Dataset. http://www.cs.cornell.edu/projects/kddcup/datasets.html.

[16] E. D. Cristofaro, S. Faber, P. Gasti, and G. Tsudik. Genodroid: are privacy-preserving genomic tests ready for prime time? In *WPES*, 2012.

[17] E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and Private Computation of Cardinality of Set Intersection and Union. In *CANS*, 2012.

[18] P. Dombek, L. Johnson, S. Zimmerley, and M. Sadowsky. Use of repetitive DNA sequences and the PCR to differentiate Escherichia coli isolates from human and animal sources. *Applied and Environmental Microbiology*, 66(6), 2000.

[19] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *PETS*, 2009.

[20] Z. Erkin, A. Piva, S. Katzenbeisser, R. L. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni. Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing. *EURASIP J. Information Security*, 2007, 2007.

[21] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.

[22] O. Goldreich. *Foundations of cryptography*. Cambridge Univ Press, 2004.

[23] IrisGuard Press Release. http://cl.ly/3KIB.

[24] P. Jaccard. Etude comparative de la distribution florale dans une portion des Alpes et du Jura, 1901.

[25] A. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Filterbank-based fingerprint matching. *IEEE Transactions on Image Processing*, 9(5), 2000.

[26] A. K. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 1996.

[27] W. Jiang, M. Murugesan, C. Clifton, and L. Si. Similar document detection with limited information disclosure. In *ICDE*, 2008.

[28] W. Jiang and B. K. Samanthula. N-gram based secure similar document detection. In *DBSec 2011*, Lecture Notes in Computer Science. Springer, 2011.

[29] M. Kantarcioglu, R. Nix, and J. Vaidya. An efficient approximate protocol for privacy-preserving association rule mining. In *KDD*, 2009.

[30] S. Katti, B. Krishnamurthy, and D. Katabi. Collaborating against common enemies. In *IMC*, 2005.

[31] F. Kerschbaum. Outsourced Private Set Intersection Using Homomorphic Encryption. In *AsiaCCS*, 2012.

[32] L. Kissner and D. Song. Privacy-preserving set operations. In *CRYPTO*, 2005.

[33] M. Li, N. Cao, S. Yu, and W. Lou. Findu: Privacy-preserving personal profile matching in mobile social networks. In *INFOCOM*, 2011.

[34] H. Lipmaa. Verifiable Homomorphic Oblivious Transfer and Private Equality Test. In *ASIACRYPT*, 2003.

[35] W. Lu, A. Swaminathan, A. L. Varna, and M. Wu. Enabling search over encrypted multimedia databases. In *Media Forensics and Security*, 2009.

[36] W. Lu, A. L. Varna, A. Swaminathan, and M. Wu. Secure image retrieval through feature protection. In *ICASSP*, 2009.

[37] W.-Y. Ma and B. S. Manjunath. Texture features and learning similarity. In *CVPR*, 1996.

[38] U. Manber. Finding similar files in a large file system. In *USENIX*, 1994.

[39] M. Murugesan, W. Jiang, C. Clifton, L. Si, and J. Vaidya. Efficient privacy-preserving similar document detection. *The VLDB Journal*, 19, August 2010.

[40] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC Project: Querying Images by Content, Using Color, Texture, and Shape. In *SPIE*, 1993.

[41] Ookla Net Metrics. Canada and US Source Data. http://www.netindex.com/source-data/, 2011.

[42] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. SCiFI – A system for secure face identification. In *IEEE S&P*, 2010.

[43] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999.

[44] M. Popescu, J. Keller, and J. Mitchell. Fuzzy measures on the gene ontology for gene product similarity. *Transactions on Computational Biology and Bioinformatics*, 2006.

[45] P. Ravikumar, W. Cohen, and S. Fienberg. A secure protocol for computing string distance metrics. In *PSDM*, 2004.

[46] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *ICISC*, 2009.

[47] S. Sclaroff. Deformable prototypes for encoding shape categories in image databases. *Pattern Recognition*, 30(4), 1997.

[48] M. Singh, P. Krishna, and A. Saxena. A privacy preserving jaccard similarity function for mining encrypted data. In *TENCON*, 2009.

[49] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12), 2000.

[50] J. R. Smith and S.-F. Chang. Tools and techniques for color image retrieval. In *SPIE*, 1996.

[51] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *AAAI*, 2000.

[52] P. Tan, M. Steinbach, V. Kumar, et al. *Introduction to data mining*. Pearson, 2006.

[53] U.S. Department of Homeland Security. http://www.dhs.gov/files/programs/usv.shtm.

[54] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4), 2005.

[55] C. Xiao, W. Wang, X. Lin, and J. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, 2008.

[56] A. Yao. How to generate and exchange secrets. In *FOCS*, 1986.

[57] Youtube press release. http://www.youtube.com/t/press_statistics.

[58] Y. Zhang, W. Liu, and W. Lou. Anonymous communications in mobile ad hoc networks. In *INFOCOM*, volume 3, 2005.

A Additional Details on MinHash Techniques

Single-Hash MinHashes. Besides the multiple-hash technique presented in Section 2, another approach for approximating the Jaccard index using MinHash employs a single hash function. In this case, rather than selecting one value per hash function, one selects the k values from set A that hash the to smallest integers. Specifically, let $h(\cdot)$ be a hash function, and k a fixed parameter; for any set S, define $h_k(S) \subset S$ as the set of the pre-images of the k smallest hash values of elements of S. Consider:

$$sim(A,B) = \frac{|(h_k(h_k(A) \cup h_k(B))) \cap (h_k(A) \cap h_k(B))|}{|h_k(h_k(A) \cup h_k(B))|} \tag{3}$$

It holds that $sim(A,B)$ is an unbiased estimate of the Jaccard index of A and B. Again, by standard Chernoff bounds [12], the expected error is $O(1/\sqrt{k})$.

MinHash Instantiations. In order to implement the MinHash schemes described in this paper, the hash function should be defined by a random permutation over the set $A \cup B$. Assuming $m = |A \cup B|$, then one would need $\Omega(m \log m)$ bits to specify a truly random permutation, thus, yielding an infeasible overhead even for small values of m. Broder, et al. [10] point out that one can obviate to this problem by using *Min-wise Independent Permutation* (MWIP) families rather than random permutations. Using MWIPs, for any subset of the domain, any element is equally likely to be the minimum, but the number of bits to specify such a permutation is showed to be still be relatively large, i.e., $\Omega(m)$. In practice, however, one can allow certain relaxations. To this end, [10] introduces *approximate* MWIPs, by accepting a small error ε. The authors require all items in a set S to have only a (almost equal) chance to become the minimum element of A's image under the permutation. Thus, for any approximate MWIP, implemented using $h^{(i)}_{min}(\cdot)$ as defined above, for an expected relative error ε, it holds:

$$\left| \Pr\left[h^{(i)}_{min}(S) = s\right] - \frac{1}{|S|} \right| \leq \frac{\varepsilon}{|S|}$$

A class of permutations often used in practice is one based on *linear transformations*. It is assumed that the universe is $\mathbb{Z}_p$ for some prime p and the family of permutation is constructed using a hash function computed as $h(x) = ax + b \mod p$, where $(a,b) \in (\mathbb{Z}^*_p, \mathbb{Z}_p)$. Such a linear transformations are easy to represent and efficiently calculable.

B Flaw in Private Document Similarity in [28]

In this section, we show that the protocol in [28] is not privacy-preserving (even in semi-honest model). In fact, Bob, in order to participate in the protocol, must disclose his global space of tri-grams. Given this information, Alice can efficiently check whether a word, e.g., w, appears in Bob's document collection. Indeed, Alice computes w's tri-gram based representation, then she checks whether all such tri-grams appear in Bob's public global space. If so, Alice learns that w appears in a document held by Bob with some non-zero probability. Technically, this probability is not 1 because Alice could have a *false positive*, i.e. w may not be in Bob's documents even though w's trigrams are in Bob's public global space. On the other hand, if at least one of the tri-grams of w is not in Bob's public global space, Alice learns that Bob's documents do not contain w. This, obviously, violates privacy requirements. If Alice and Bob include punctation and spaces in their tri-grams representation of their documents, the probability of *false positive* becomes negligible. We do not exploit "relations" between consecutive meaningful words in the sentence, which could potentially (further) aggravate information leakage about Bob's documents.

We now show yet another attack that lets Alice learn even more, since the N-grams representation *embeds* document's structure. From the global space of tri-grams $\mathcal{GS}$, we can construct a directed graph $G(V, E)$ representing relations between tri-grams in Bob's document collection. Any path in such a graph will lead to a textual fragment contained in some document held by Bob. A vertex in the graph represents a tri-gram; whereas, an edge between two vertices implies that the two corresponding tri-grams are consecutive tri-grams in a word. Given a trigram $x \in \mathcal{GS}$, with $x^{(i)}$ we denote the i-th letter in x. The directed graph $G(V, E)$ is constructed as follows. The vertex set is $V = \{V_x \mid x \in \mathcal{GS}\}$ and the edge set is $E = \{\langle V_x, V_y\rangle \mid x^{(2)} = y^{(1)} \wedge\ x^{(3)} = y^{(2)}\}$. A path $V_{x_1}, \ldots, V_{x_n}$ in G, will correspond to the string $x_1^{(1)}x_1^{(2)}x_2^{(3)}x_3^{(3)}\cdots x_n^{(3)}$. Such a string (or some of its substring) appears in some document in Bob's collection. By using algorithms based on Deep First Search visit of a graph, a vocabulary, and syntactic rules, we could extract large document's chunks. We did not explore further other techniques to extract "information" from the global space of tri-grams as we consider them to be out of the scope of this paper.