Privacy-Preserving Matching of Community-Contributed Content

Mishari Almishari[‡][⋆], Paolo Gasti[‡‡], Gene Tsudik[†], Ekin Oguz[†]

[‡] King Saud University
[‡‡]New York Institute of Technology
[†]University of California, Irvine

Abstract. Popular consumer review sites, such as Yelp and Tripadvisor, are based upon massive amounts of voluntarily contributed content. Sharing of data among different review sites can offer certain benefits, such as more customized service and better-targeted advertisements. However, business, legal and ethical issues prevent review site providers from sharing data in bulk.
This paper investigates how two parties can *privately compare* their review datasets. In presents a technique for two parties to determine which (or how many) users have contributed to both review sites. This is achieved based only upon review content, rather than personally identifying information (PII). The proposed technique relies on extracting certain key features from textual reviews, while the privacy-preserving user matching protocol is built using additively homomorphic encryption and garbled circuit evaluation. Experimental evaluation shows that the proposed technique offers highly accurate results with reasonable performance.

1 Introduction

On-line social networks (OSNs) are a valuable resource for untold masses who rely on them in both personal or professional aspects of everyday life, including: sharing personal content [2,4], broadcasting pithy "news" messages to others with similar interests [7], finding jobs or identifying job candidates [5], planning travel [6], and assessing businesses (stores, restaurants, services) [8] or products [1]. A typical OSN provides the framework wherein volunteers contribute virtually all available content. Within this framework, users generally reveal – often unwittingly – tremendous amounts of personal information, including habits and tastes. This information is very valuable for quickly detecting trends and serving timely targeted advertisements [3].

Community-based review sites form a specific class of OSNs. Well-known examples are `yelp.com`, `tripadvisor.com` and `amazon.com`. On these sites, users read and contribute reviews expressing their opinions on different products, services and businesses. Users can also discover other groups or individuals with similar interests. In recent years, such sites have become very popular. For example, `yelp.com` received 78 million unique monthly visitors, on average, in the second quarter of 2012 [9]. Also, in 2012, `yelp.com` users have contributed more than 30 million reviews [9].

The most valuable asset of community-based review sites is user-generated content. It is perceived to be unbiased and represents the main reason for attracting multitudes of people

[⋆] Work performed while at University of California, Irvine.

to sites like `yelp.com` or `tripadvisor.com`. While review sites are happy to let anyone – including casual users without accounts – read individual reviews, they zealously guard *bulk* user content. To this end, they usually employ both technical and legal (e.g., terms of service [10]) measures to prevent bulk access and large-scale content harvesting.

We believe that sharing user-related information across sites could be beneficial to review sites themselves as well as their users. Various sites have access to information concerning different aspects of public and private lives of their users. Knowing which users belong to multiple sites would allow the latter to provide better service or better-targeted ads, e.g., a travel site could highlight gastronomic destinations for users who contributed to a restaurant review site, or a product-oriented site might advertise ski gear for users who reviewed mountain resorts on a travel site.

This paper makes the following contributions:

1. We construct a technique that identifies common users across two review sites by comparing user-generated content, rather than user names or IDs. (In general, IDs are problematic because users tend not to use consistent identifiers across sites. Furthermore, imposing, incentivizing or enforcing consistent identification/naming is very difficult.)
2. We show how to efficiently implement the proposed technique *with privacy*, such that one of the two sites learns *only* which (or how many) users belong to both, while the other learns nothing. Furthermore, this is achieved with a high degree of accuracy.

Previous literature [12,43] shows that sets of anonymous reviews can be linked by merely relying on simple textual features. However, these prior techniques require at least one of the parties to reveal all of its reviews or their compact representation. Our work explores new and more sophisticated textual features, and provides the first privacy-preserving approach for efficiently computing user similarity.

Our work also helps mitigate so-called *review spam* [35], which involves creating fake reviews, with the intent to over-promote or defame a product or a service. Fake reviews appear as if generated by legitimate users and are therefore hard to identify. We anticipate that detection of suspected spammers' accounts would be a useful service. One way to implement this service is as follows: one site with the expertise in detecting spammer accounts accumulates a set of confirmed spammers along with their content. It then runs our protocols with any other site that has a set of its own suspected spammers. As a result, the latter obtains a list of confirmed spammers.

Organization: Related work is summarized in Section 2. Our review matching algorithm (without privacy) is introduced in Section 3. Next, cryptographic preliminaries are discussed in Section 4, followed by our privacy-preserving matching protocols in Sections 5 and 6. Then, protocol performance is assessed in Section 7 and Section 8 concludes the paper. Security analysis can be found in Appendix A.

2 Related work

Most related work falls into two categories: (1) authorship identification and (2) privacy-preserving protocols. The former offers a number of results showing that authorship linkage based on textual (stylometric) features is feasible and sometimes very effective. The latter yields numerous cryptographic techniques for privately computing certain set operations and similarity measures.

2.1 Authorship Identification and Linkage

Most prior work on this topic deals with free-style text, such as news reports, scripts, novels, essays and diaries. This is motivated by the recent increase in scholastic, academic and regular literary plagiarism.

A number of techniques have been explored to identify common authorship. For example, Narayanan et al. [43] conducted a large-scale author identification study of anonymous blogs using stylometric features. A number of features were extracted and used in training classifiers that recognized authors based on their writing style. A set of $100,000$ blog authors was used to evaluate proposed techniques. Accuracy of up to 80% was obtained.

A more recent result [12] shows how to link reviews authored by the same person. One de-anonymization technique was based on constructing a Naïve Bayesian (NB) model [39] for every user and then mapping each set of anonymous reviews to the corresponding user with the highest probability. The second technique was based on the symmetric Kullback-Leibler (KL) divergence distance function [14]. With KL, the user whose reviews have the shortest distance to anonymous reviews is labeled as the original author. This demonstrates that anonymous review sets (at least, for prolific reviewers) by same author can be linked with very high probability. Moreover, distribution of digram (two-letter) tokens is very effective in determining similarity among review sets.

There have been other interesting authorship analysis studies. Notably, [32] proposed a technique that extracts frequent pattern write-prints that distinguish an author. Accuracy reached 88% using a single anonymous message. The study in [11] explored author identification and similarity detection by using stylistic features, based on Karhunen-Loeve transform to obtain write-prints. Accuracy reached 91% in identifying the author of anonymous text from a set of 100 authors. Results indicate the feasibility of linking bodies of text authored by the same person. A comprehensive survey of authorship identification and attribution studies can be found in [48].

2.2 Privacy-Preserving Protocols

There is extensive literature on secure multi-party computation. Starting from the seminal work on garbled circuit evaluation [51,28], it has been shown that any function can be securely evaluated by representing it as a boolean circuit. Similar results exist for secure evaluation of any function using secret sharing techniques, e.g., [46], or homomorphic encryption, e.g., [20].

Recent results on garbled circuits provide optimizations that reduce computation and communication overheads associated with circuit construction and evaluation. Kolesnikov and Schneider [38] described an optimization that permits XOR gates to be evaluated for free, i.e., without communication normally associated with such gates and without involving any cryptographic functions. This optimization is possible when the hash function used for creating garbled gates is correlation-robust under the definition in [19]. Under similar assumptions, Pinkas et al. [45] provided a mechanism for reducing communication complexity of binary gates by 25%: each gate can be specified by encoding only three outcomes of the gate instead of four. Finally, [37] improved complexity of certain common operations, such as addition, multiplication, and comparison, by reducing the number of non-XOR gates.

In recent years, a number of tools have been developed for automatically creating a secure protocol from its function description written in a high-level language. Examples include Fairplay [41], VIFF [24] and TASTY [31]. However, “custom” optimized protocols for specific applications are often much more efficient than such general techniques.

There are also a number of results in privacy-preserving set operations, e.g., private set intersection (PSI) [27,36,29,30,34,25] and cardinality-only PSI (PSI-CA) [21]. The work in [34] introduced a PSI protocol based on oblivious pseudorandom functions (OPRFs) secure in the malicious model. This protocol incurs linear complexity in size of combined client/server inputs and it is secure under the One-More Gap Diffie-Hellman assumption. In [21], a very efficient, also OPRF-based, PSI-CA protocol is constructed offering linear complexity in the size of server and client inputs. Its security, in the semi-honest model, is based on the DDH assumption.

As mentioned in Section 1, although PSI and PSI-CA offer functionalities similar to those required to determine common authors across multiple review sites, *noisy* nature of features extracted from reviews prevents the use of such tools. Whereas, privacy-preserving protocols in [16] are more relevant to our context of review matching. In particular, [16] shows how to efficiently and privately approximate the computation of Jaccard index [49] using minhash techniques [17]. This approach is effective to compare text in order to detect plagiarism or enforce copyright.

3 Review Matching

Contributors to a review site are referred to, and are known by, their user-names, unique per site. As mentioned in Section 1, relying solely on user-names to determine common users across sites is problematic, since identical user-names on different sites may not correspond to the same user. Conversely, the same person may select distinct user-names on different sites. Similarly, relying on the user's real identity for matching may not be viable, since users may not be willing to disclose any personal information.

Let C (client) and S (server) denote two mutually suspicious review sites. Each site has access to a collection of reviews, partitioned by user. Let $U_C = \{C_1, C_2, \ldots, C_v\}$ denote the set of users that of C, and $U_S = \{S_1, S_2, \ldots, S_w\}$ – the set of users of S. R_{C_i} and R_{S_i} refer to the set of reviews authored by C_i and S_i, respectively. $C's$ goal is to learn privately (i.e., without disclosing the content of reviews associated with its users) one of the following: **Common Users**, denoted as $\Psi = U_C \cap U_S$, or ***Number* of Common Users** ($|\Psi|$). Notation is summarized in Table 1.

v	Number of users at C	$\widehat{\Psi}$	Common users computed by the matching algorithm
w	Number of users at S	er	Error rate
U_C	Users of C	rr	Recall rate
U_S	Users of S	ε	Matching threshold
Ψ	Common users ($U_C \cap U_S$)	mr	matching user approximation error
X_i	Feature vector computed from C_i's reviews	Y_j	feature vector computed from S_j' reviews
C_i	user at C	S_j	user at S
R_{C_i}	set of reviews authored by C_i	R_{S_j}	set of reviews authored by S_j

Table 1: Notation

In this section, we construct a technique for computing Ψ and $|\Psi|$ *without privacy.* We then add privacy features in Sections 5 and 6.

3.1 Matching Process Overview

To find common users, we need to determine similarity between two sets of reviews. We consider C_i and S_j as corresponding to the same user if their corresponding review sets are very similar. One way to assess similarity is to use a distance function. This works as follows: from each user review set, extract a number of features and represent them as a vector. Let $X = \mathit{feat}(\cdot)$ be a feature extraction function that takes as input a set of reviews and returns the associated feature vector X. Let $d = D(\cdot, \cdot)$ be a distance function that takes as input two feature vectors and outputs a value $d \geq 0$. Informally, 0 indicates that two inputs are identical, and the larger the d, the more different they are. We say that two feature vectors X, Y (and their corresponding review sets) are *similar* if $D(X, Y) \leq \varepsilon$, for some value ε.

Each protocol party computes a feature vector per user resulting in a set of feature vectors $\mathcal{X} = \{X_1, \ldots, X_v\}$ for C and $\mathcal{Y} = \{Y_1, \ldots, Y_w\}$ for S, where $X_i = \mathit{feat}(R_{C_i})$, and $Y_i = \mathit{feat}(R_{S_i})$. Let X_i and Y_j be the feature vectors corresponding to reviews of users C_i and S_j, respectively. We approximate *Common Users* and *Number of Common Users* as: **Matching Users**, defined as $\widehat{\Psi} = \{C_i \in U_C \mid \exists S_j \in U_S \text{ s.t. } D(X_i, Y_j) \leq \varepsilon\}$ and **Number of Matching Users**, defined as $|\widehat{\Psi}|$. Clearly, approximation accuracy depends on specific properties of the features being considered and on the distance function D.

There are several distance functions that have been shown to provide good results on textual documents retrieval, including Cosine, Jaccard, and Euclidean distances [13]. We rely on Euclidean distance. Our experiments (see Section 3.3) confirm that it is a sensible choice for review similarity. Euclidean distance between vectors $X = \{x_1, \ldots, x_\ell\}$ and $Y = \{y_1, \ldots, y_\ell\}$ is defined as:

$$D'(X, Y) = \sqrt{\sum_{i=1}^{\ell} (x_i - y_i)^2}$$

For technical reasons, in the rest of the paper, we consider D to be *squared euclidean distance*, i.e., $D(X, Y) = (D'(X, Y))^2$. We acknowledge that other distance functions may offer different, and possibly better, accuracy results. We leave evaluation of other distance functions to future work.

3.2 Dataset: Training and Testing Settings

To assess accuracy of our review matching technique, we rely on approximately 1 million reviews from 1,997 users of `yelp.com`. [1] We define two metrics that capture two performance aspects of review matching process:

1. Recall Ratio (rr) – measures $\widehat{\Psi}$'s coverage of Ψ: $rr = |\widehat{\Psi} \cap \Psi| / |\Psi|$.
2. Error Ratio (er) – measures how often an element not in Ψ is included in $\widehat{\Psi}$:

$$er = \frac{|\{(C_i, S_j) \text{ s.t. } C_i \neq S_j \text{ and } D(X_i, Y_j) \leq \varepsilon\}|}{|\{(C_i, S_j) \text{ s.t. } C_i \neq S_j\}|}$$

We divide users (along with their reviews) into two distinct sets of nearly the same size: Tr and Te, used for training and testing purposes, respectively. We use Tr to determine a set

[1] Experiments were performed on the same dataset used in [12]

of features and a threshold ε that maximize rr while keeping er low. We then check how these parameters perform over Te. We emphasize that no data from Te is used to select any parameters.

For every user in Tr, we randomly split its reviews into two parts. Let Tr_C and Tr_S represent first and second half of each user's reviews. Based on Tr_C and Tr_S, we build two sets of feature vectors P_C^{Tr} and P_S^{Tr}. We then select ε as follows: First, we compute the distance between all pairs of feature vectors from P_C^{Tr} and P_S^{Tr}. Then, we vary rr from (0%-100%] by selecting different values for ε. For each ε, we measure corresponding er. Finally, we select ε that yields the best trade-off between rr and er.[2]

3.3 Feature Sets

Proper selection of features is crucial for achieving high accuracy. We now assess different feature sets and determine the combination that offers the best performance.

Write-Prints and Basic-9 Features. We first examine two standard feature sets: Basic-9 [18] and Write-Print [11]. The former consists of 9 features that measure different textual characteristics, e.g., number of unique words used in a review set and its ratio to the total number of words. These features have been shown to be effective in identifying authors of anonymous texts [18]. Write-Print is a set of static and dynamic features that fall into five groups: lexical, syntactic, structural, content and idiosyncratic. It is highly effective in identifying authors, as shown in [11]. We use the implementation of both feature sets from JStylo ([42]), a stylometric java-based library.[3]

Figure 1 shows rr and er values on Tr for various ε values, as described in Section 3.2, using either Basic-9 or Write-Print. Results in Figure 1 show that, regardless of ε, features we consider do not allow us to achieve high rr and low er. Thus, we explore different feature sets.

Character n-gram. *n-grams* (n consecutive characters in a text fragment) are a well-known feature that have been extensively used in textual analysis. We experiment with n-gram feature sets for n = 2 (digrams) and n = 3 (trigrams). As shown in [12], digrams are very effective in identifying review authors. N-gram feature vectors for sets of reviews are constructed as follows: each array element labeled with a given n-gram is represents frequency of occurrence of this n-gram in a user's review set.

Figure 2 shows rr and er results using digrams and trigrams. Digrams show better performance compared to Write-Prints and Basic-9 features. For example, for $rr = 95\%$, $er = 5.11 \cdot 10^{-5}$ with digrams and $2.01 \cdot 10^{-4}$ with trigrams.

Part-Of-Speech (POS) Tagging. Part-Of-Speech (POS) tagging involves mapping words to parts of speech, e.g., noun or verb. The idea is that different individuals write using distinct grammatical structures and choose different words. We rely on digram and trigram

[2] Ideally, evaluation of our technique would be performed on two or more datasets from different sites, which share a *correctly identified* subset of users. However, we are not aware of the existence of such a dataset. Therefore, we rely on partitioning reviews from each user into two sets.

[3] JStylo implements a partial set of Write-Print features that amounts to 22 feature categories.

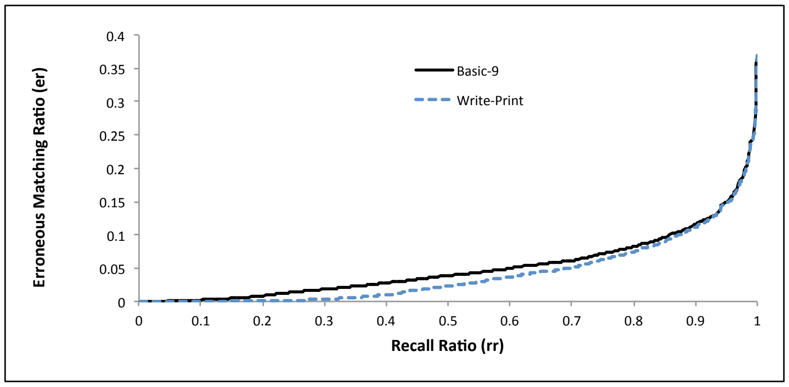

Fig. 1: Error and recall ratio of Write-Print and Basic-9

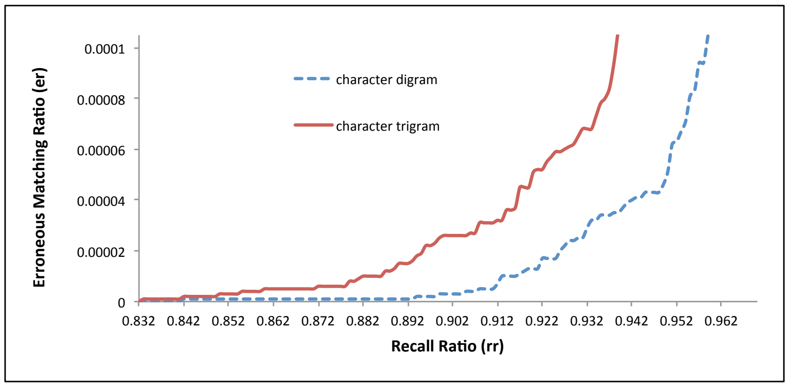

Fig. 2: Error and recall ratio of character digram and trigram

versions of POS tags (2 or 3 consecutive parts of speech tags) and use Stanford POS Maxent Tagger [50] to label each word with one of 45 possible POS tags. We assign weights to POS features similarly to character n-grams.

Figure 3 shows performance results of POS features. Clearly, digrams outperform trigrams: for example, with $rr = 95\%$, the corresponding $er = 7.01 \cdot 10^{-6}$ digrams and $6 \cdot\ 10^{-3}$ with trigrams.

Combining Character and POS n-grams. Since character and POS digram feature sets offer good performance, we explore ways to combine them to further improve matching accuracy. In particular, we use a simple weighted average technique, i.e.:

$$D_{combined}(X, Y) = (a) \times D_{character\_digram}(X, Y) + (1 - a) \times D_{POS\_digram}(X, Y)$$

We vary a from 0 to 1 (in 0.1 increments) to determine impact on rr and er.

With our training dataset, values of a between 0.7 and 0.8 lead to $er <\ 10^{-5}$. There are two reasons for limiting er this way: (1) $er \approx 10^{-5}$ is relatively high and could lead to

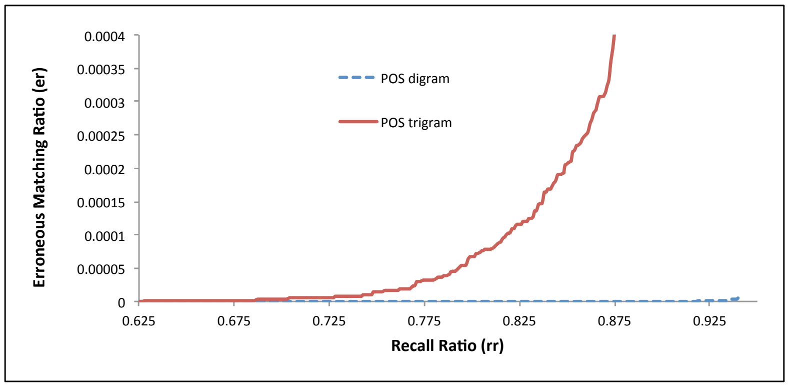

Fig. 3: Error and recall ratio of POS digrams and trigrams

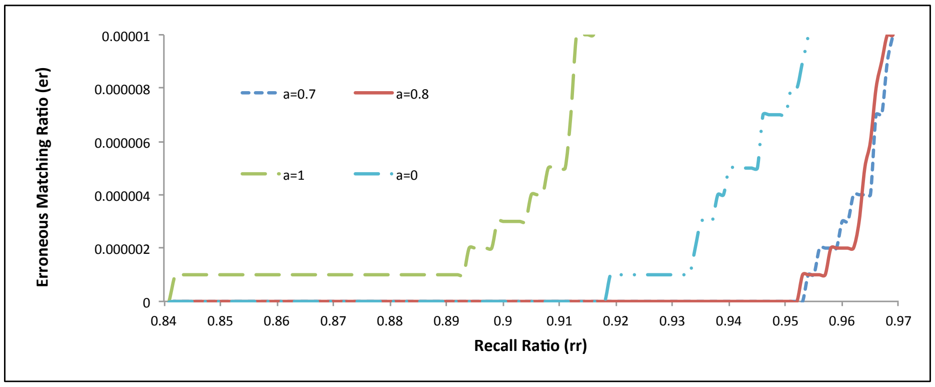

Fig. 4: error and recall ratio of combining character and POS digrams

poor approximation of Ψ when v, w are very large[4], and (2) for our dataset, there is no a value that gives better performance over the full range of rr.

Figure 4 summarizes the experiments. Combining character and POS digram features yields increased matching accuracy. Since $a = 0.7$ and $a = 0.8$ provide roughly the same performance, we pick $a = 0.7$. We choose ε that yeilds $rr = 95.3\%$ and $er = 0\%$ in Tr. We test selected ε on Te and the results are virtually identical ($rr = 95.5\%$ and $er = 0\%$). Note that, when selecting the threshold, we choose ε such that it maximizes rr, while keeping $er = 0\%$ to reduce inaccuracy of approximating Ψ incurred by larger er values.

When combining character and POS digrams, the resulting feature set size contains $2,701$ features: the former contribute 676 (26^2) and the latter – 2025 (45^2) features. Even both digram types are a subset of Write-Print features, they perform significantly better than the entire Write-Print feature set; see Figure 1.

[4] Note that the number of errors grows proportionally to $v \cdot w$.

3.4 Approximation Error

Though er and rr represent good metrics for determining accuracy of matching algorithms, they do not offer easy-to-interpret information for the *number of matching users* algorithm. We therefore define matching user approximation error (mr) as:

$$mr = \frac{abs(|\widehat{\Psi}| - |\Psi|)}{|\Psi|}$$

Since our choice for ε leads to $er = 0\%$, mr mainly depends on rr. Given our accuracy results, $|\widehat{\Psi}| = rr \cdot |\Psi|$. Thus, $mr = 1 - rr$, i.e., $mr < 5\%$. This shows that our review matching technique closely approximates Ψ with $\widehat{\Psi}$.

4 Cryptographic Preliminaries

Security model. We use the standard model for secure two-party computation in the presence of semi-honest (also known as honest-but-curious) participants. In this model, participants follow prescribed protocol behavior, while trying to learn or infer additional information beyond that obtained during normal protocol execution. A protocol is considered secure in the semi-honest model if the view of protocol execution for each party is computationally indistinguishable from the view simulated using that party's input and output only. This means that protocol execution does not reveal any additional information to participants. A more formal definition is as follows:

Definition 1. *Suppose participants P_1 and P_2 run a protocol π that computes function $f(\mathsf{in}_1, \mathsf{in}_2) = (\mathsf{out}_1, \mathsf{out}_2)$, where in_i and out_i denote P_i's input and output, respectively. Let $\mathrm{VIEW}_\pi(P_i)$ denote P_i's view during the execution of π. It is formed by P_i's input, internal random coin tosses r_i, and messages $m_1, \ldots, m_t$ passed between parties during execution:*

$$\mathrm{VIEW}_\pi(P_i) = (\mathsf{in}_i, r_i, m_1, \ldots, m_t).$$

We say that π is secure in the semi-honest model, if for each P_i, there exists a probabilistic polynomial time simulator S_i such that

$$\{S_i(\mathsf{in}_i, f_i(\mathsf{in}_1, \mathsf{in}_2))\} \equiv \{\mathrm{VIEW}_\pi(P_i), \mathsf{out}_i\},$$

where "$\equiv$" denotes computational indistinguishability.

Homomorphic encryption. Our protocols require existence of a semantically secure additively homomorphic encryption scheme. In such a scheme, $\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) = \mathsf{Enc}(m_1 + m_2)$, and, therefore, $\mathsf{Enc}(m)^a = \mathsf{Enc}(a \cdot m)$. While any such scheme (e.g., Paillier [44]) would suffice, the construction by Damgård et al. [23,22] (DGK) is of particular interest here.

DGK was designed to work with small plaintext spaces and has shorter ciphertext size than other similar schemes. A DGK public key consists of: (i) a small (possibly prime) integer u that defines plaintext space, (ii) a k-bit RSA modulus $N = pq$ where p and q are $k/2$-bit primes, such that, if v_p and v_q are t-bit primes, and $uv_p|(p-1)$ and $uv_q|(q-1)$, and (iii) elements $g, h \in \mathbb{Z}_N^*$ such that g has order uv_pv_q and h has order v_pv_q. Given a message $m \in \mathbb{Z}_u$, encryption is performed as: $\mathsf{Enc}(m) = g^m h^r \bmod N$, where $r \leftarrow \{0,1\}^{2.5t}$.

Homomorphic-based comparison. Our protocols rely on privacy-preserving comparison to determine whether the distance between two feature vectors is below a threshold. Such a distance (d) is computed in the encrypted domain by the server, and compared (also in its encrypted form) with threshold ε.

We base our comparison protocol on that of Erkin et al. [26]. It relies on the observation that $d < \varepsilon$ is true iff the l-th bit of $a = 2^l + d - \varepsilon$ is 1 (for $2^l >> d$ and $l >> \varepsilon$). Given $\mathsf{Enc}(d)$, encryption of a is computed by S as $\mathsf{Enc}(a) = \mathsf{Enc}(2^l) \cdot \mathsf{Enc}(d) \cdot \mathsf{Enc}(\varepsilon)^{-1}$. Encryption of the l-th bit of a is then: $\mathsf{Enc}(a_l) = \mathsf{Enc}(2^{-l} \cdot (a - (a \bmod 2^l)))$. Value a is available to S only in encrypted form, and computing $a \bmod 2^l$ in the encrypted domain requires interaction between C and S:

> S "masks" $\mathsf{Enc}(a)$ by selecting a random r and computing $\mathsf{Enc}(\hat{a}) = \mathsf{Enc}(a) \cdot \mathsf{Enc}(r)$. Then, S sends $\mathsf{Enc}(\hat{a})$ to C, who decrypts it and returns the encryption of $c = \hat{a} \bmod 2^l$ to S. Next, S "unmasks" $\mathsf{Enc}(c)$ by computing $\mathsf{Enc}(c) \cdot \mathsf{Enc}(r)^{-1} = \mathsf{Enc}(a \bmod 2^l)$.

5 Privacy-Preserving Computation of Matching List

We now present a protocol for Privacy-Preserving Computation of Matching List (PPCML). It involves two participants: C and S. At the end, C learns the set of users in its input that match those in S's input, while S learns nothing. For simplicity's sake, we represent C's input as a single feature vector, corresponding to one user, while S's input is a set of w feature vectors, from w users. This protocol can be trivially extended to the case where both parties input a *set* of feature vectors.

Weighted Average. As discussed in Section 3.3, we use a weighted average distance function $D_{combined}$ with $a = 0.7$. $D_{combined}$ can be also computed as a square Euclidean distance function between a feature vector for user in C and a user in S. This is done by updating the weights of the feature vector by multiplying all digram feature weights by $\sqrt{a}$, and all POS digram feature weights by $\sqrt{1-a}$.

Scaling. Since our protocol can only process integer vectors, we first need to scale values in feature vectors from the domain $[0,1] \subset \mathbb{R}$ to $[0, 10^h] \subset \mathbb{N}$ by multiplying all features by 10^h for some h. Intuitively, larger h allow for better precision. However, the number of bits required to represent values in $[0, 10^h]$ – and therefore the cost of our protocol – increases with h. Our experiments showed that $h = 4$ provides a reasonable tradeoff between cost and accuracy. With this scaling, we obtain a scaled ε value that gives exactly the same rr and er as the non-scaled version in both Tr and Te. Moreover, we determined that using $h > 4$ does not improve precision and recall significantly.

Protocol Input:
C: feature vector $X = (x_1, \ldots, x_\ell)$ and key-pair (pk, sk).
S: $\mathcal{Y} = \{Y_1, \ldots, Y_w\}$ where $Y_m = (y_{m,1}, \ldots, y_{m,\ell})$, for $0 < m \leq w$ is a feature vector.

Protocol Output:
C: 1, if Euclidean distance between X and any vector in $\mathcal{Y}$ is below ε,[5] and 0 otherwise.
S: nothing.

[5] The protocol implements $D(X,Y) \overset{?}{<} \varepsilon$ instead of $D(X,Y) \overset{?}{\leq} \varepsilon$ as defined in Section 3.1. In our setting, $(D(X,Y) \overset{?}{\leq} \varepsilon) = (D(X,Y) \overset{?}{<} \varepsilon')$ for $\varepsilon' = \varepsilon + 1$.

Protocol Steps:

1. For $i = 1, \dots, \ell$, C computes $\{\langle \mathsf{Enc}(x_i), \mathsf{Enc}(x_i^2)\rangle\}$ and sends results to S.
2. For $m = 1, \dots, w$ and $j = 1, \dots, \ell$, S computes $\{\mathsf{Enc}(y_{m,j}^2)\}$.
3. For $m = 1, \dots, w$, S computes encrypted square Euclidean distance between X and Y_m as:

$$\mathsf{Enc}(d_m) = \mathsf{Enc}\left(\sum_{i=1}^{\ell}(x_i - y_{m,i})^2\right) = \prod_{i=1}^{\ell}\left(\mathsf{Enc}(x_i^2)\mathsf{Enc}(y_{m,i}^2)\mathsf{Enc}(x_i)^{(-2y_{m,i})}\right)$$

4. For each $m = 1, \dots, w$, S and C invoke an instance of the privacy-preserving comparison protocol [26] to determine whether $d_m < \varepsilon^2$, i.e., S learns $\mathsf{Enc}(\delta_m)$, where $\delta_m = 1$ iff $d_m < \varepsilon^2$.
5. S computes $\mathsf{Enc}(\alpha) = \prod_{m=1}^{w}\mathsf{Enc}(\delta_m)$. Note that α represents the number of vectors in $\mathcal{Y}$ for which square Euclidean distance from X is less than ε^2.
6. S returns $u = \mathsf{Enc}(\alpha)^r = \mathsf{Enc}(\alpha \cdot r)$, where r is a random element chosen uniformly from the message space (except 0).
7. C computes $z = \mathsf{Dec}(u) = \alpha \cdot r$. If $z \neq 0$, C outputs 1; otherwise it outputs 0.

Although there are techniques for computing square roots using secure multi-party computation, e.g., [40], their performance is quite below par for our application. Fortunately, comparison of Euclidean distance with ε can be performed without computing any square roots, by comparing ε with the square of Euclidean distance (see Step 4).

In practice, C's input would contain multiple feature vectors. C can simply run the protocol multiple times – once per input vector. Security of the protocol would be unaffected, except that S would learn the upper bound on the number of vectors in C's input.

In the rest of paper, we use the term PPCML to refer to the combination of (possibly) multiple instance of the protocol above, one per feature vector of C. Security analysis of the protocol sketched out above is provided in Appendix A.

6 Privacy-Preserving Computation of Matching List Size

We now extend PPCML by restricting $C's$ knowledge to the number of users that occur in both$C$ and S, i.e., we obtain Privacy-Preserving Computation of Matching List *Size* (S-PPCML). In this protocol, each party's input is a set of feature vectors. C learns the matching list (set intersection) size while S only learns the upper bound on the number of C's users.

Protocol Input:

C: set of feature vectors $\mathcal{X} = \{X_1, \dots, X_v\}$, with $X_n = (x_{n,1}, \dots, x_{n,\ell})$ and key pair (pk, sk).

S: set $\mathcal{Y} = \{Y_1, \dots, Y_w\}$ where $Y_m = (y_{m,1}, \dots, y_{m,\ell})$ is a feature vector.

Protocol Output:

C: number of feature vectors $X_n \in \mathcal{X}$ with Euclidean distance less than ε for at least one vector from $\mathcal{Y}$; i.e., $|\widehat{\Psi}|$.

S: nothing.

Protocol Steps:

1. For each $n = 1,\ldots,v$ and $i = 1,\ldots,\ell$, C computes $\{\langle \mathsf{Enc}(x_{n,i}), \mathsf{Enc}(x_{n,i}^2)\rangle\}$ and sends them to S.
2. For each $m = 1,\ldots,w$ and $j = 1,\ldots,\ell$, S computes $\{\mathsf{Enc}(y_{m,j}^2)\}$.
3. For each $n = 1,\ldots,v$ and $m = 1,\ldots,w$, S computes encrypted square Euclidean distance between X_n and Y_m as

$$\mathsf{Enc}(d_{n,m}) = \mathsf{Enc}\left(\sum_{i=1}^{\ell}(x_{n,i} - y_{m,i})^2\right) = \prod_{i=1}^{\ell}\left(\mathsf{Enc}(x_{n,i}^2)\mathsf{Enc}(y_{m,i}^2)\mathsf{Enc}(x_{n,i})^{(-2y_{m,i})}\right)$$

4. For each $n = 1,\ldots,v$ and $m = 1,\ldots,w$, S and C interact in a privacy-preserving manner to compare $\mathsf{Enc}(d_{n,m})$ against ε^2; S learns $\mathsf{Enc}(\delta_{n,m})$, where $\delta_{n,m} = 1$ iff $d_{n,m} < \varepsilon^2$.
5. For each $n = 1,\ldots,v$, S computes $\mathsf{Enc}(\alpha_n) = \prod_{m=1}^{w}\mathsf{Enc}(\delta_{nm})$. Note that α_n represents the number of vectors in $\mathcal{Y}$ that fall within ε of X_n.
6. For each $n = 1,\ldots,v$, S and C interact in a privacy-preserving manner to compare $\mathsf{Enc}(\alpha_n)$ to 0. Let β be the outcome of this comparison – i.e., $(\beta_n = 1)$ iff $(\alpha_n > 0)$; S learns $\mathsf{Enc}(\beta_n)$.
7. S computes $\mathsf{Enc}(\gamma) = \prod_{n=1}^{v}\mathsf{Enc}(\beta_n)$ and sends it to C.
8. C decrypts and outputs γ, which corresponds to the number of users it shares with S.

6.1 Protocol Optimizations: AS-PPCML

We now discuss some optimizations.

Dataset-Dependent Optimizations The goal of Step 6 in the S-PPCML protocol is to "combine" multiple matches between a single feature vector from C and multiple vectors from S into one. According to our experiments, the value of ε selected in Section 3.3 allows us to keep error rate at 0 (with our dataset) and matching rate at 95% *without* performing Step 6. Therefore, removing this step has virtually no impact on the result of the computation. We refer to this modified version of the protocol as *Approximate* S-PPCML (AS-PPCML).

Garbled Circuits As shown in [15,47], comparison protocols can be implemented more efficiently using garbled circuits, rather than homomorphic encryption. Therefore, we can easily optimize the S-PPCML protocol by replacing homomorphic-based comparison with one using a garbled circuit.

For each X_n and Y_m from $C's$ and $S's$ inputs, respectively, S computes encrypted Euclidean distance between the two as in our S-PPCML protocol. Then S "obfuscates" the result by multiplying it with a random value $r_{n,m}$. The obfuscated value is returned to C, which inputs it into the comparison circuit. S inputs ε and $r_{n,m}$. The circuit adds $-r_{n,m}$ to C's input in order to "unmask" it, and compares the result with ε. C only learns the outcome of the comparison, while S learns nothing.

We implemented this comparison circuit based on the design of efficient circuits for addition modulo 2^N and comparison described in [37].

Other Optimizations We perform as much computation as possible in the unencrypted domain. In particular, both S and C compute, in the clear, summation of the squares of all elements in their feature vectors.

6.2 Optimized Protocol

The protocol below includes all the aforementioned optimizations.

– **Protocol Input:** C's input is a set of feature vectors $\mathcal{X} = \{X_1, \ldots, X_v\}$, with $X_n = (x_{n,1}, \ldots, x_{n,\ell})$ and key pair (pk, sk). S's input is $\mathcal{Y} = \{Y_1, \ldots, Y_w\}$ where $Y_m = (y_{m,1}, \ldots, y_{m,\ell})$ is a feature vector.
– **Protocol Output:** C's output is the number of feature vectors $X_n \in \mathcal{X}$ that have square Euclidean distance smaller than ε^2 with at least one vector from $\mathcal{Y}$; i.e., $|\widehat{\Psi}|$.

Protocol steps:
1. For each $n = 1,\ldots,v$ and $i = 1,\ldots,\ell$, C encrypts $\{\langle \mathsf{Enc}(x_{n,i}), \mathsf{Enc}(c_n) = \mathsf{Enc}(\sum_{i=1}^{\ell} x_{n,i}^2)\rangle\}$ and sends results to S.
2. For each $m = 1,\ldots,w$, S computes $\{\mathsf{Enc}(s_m) = \mathsf{Enc}(\sum_{j=1}^{\ell} y_{m,j}^2)\}$.
3. For each $n = 1,\ldots,v$ and $m = 1,\ldots,w$, S computes the encrypted square Euclidean distance between X_n and Y_m as
$$\mathsf{Enc}(d_{n,m}) = \mathsf{Enc}\left(\sum_{i=1}^{\ell}(x_{n,i} - y_{m,i})^2\right) = \mathsf{Enc}(c_n) \cdot \mathsf{Enc}(s_m) \cdot \prod_{i=1}^{\ell}\left(\mathsf{Enc}(x_{n,i})^{(-2y_{m,i})}\right)$$
4. For each $n = 1,\ldots,v$ and $m = 1,\ldots,w$, S randomizes the value computed in the previous step as: $\mathsf{Enc}(\hat{d}_{n,m}) = \mathsf{Enc}(d_{n,m}) \cdot \mathsf{Enc}(r_{n,m})$, where $r_{n,m}$ is uniformly selected from the message space. Then, S shuffles these values and sends them to C.
5. C decrypts all $\{\mathsf{Enc}(\hat{d}_{n,m})\}$; C and S evaluate a garbled circuit over input $\{\mathsf{Enc}(\hat{d}_{n,m})\}$ for C and $\{-r_{n,m}\}, \varepsilon^2$ for S. The circuit implements functionality $(\hat{d}_{n,m}+(-r_{n,m})) < \varepsilon^2$, where addition is performed modulo 2^N for some N.
6. C outputs $\gamma = \sum_{n=1}^{v}\sum_{m=1}^{v}\delta_{n,m}$.

7 Implementation and Performance

In this section we provide implementation details for our protocols, and report on performance measurements. All protocols are implemented in C. Our code is compiled using GCC 4.2 and relies on the GMP library to implement number-theoretic cryptographic operations and on OpenSSL for symmetric cryptography. Tests are run under Ubuntu 8.04 LTS.

Measurements are performed on a machine with two quad-core 2.5 GHz Intel Xeon CPUs and 16 GB memory. In order to provide results comparable with the state of the art, we restrict our code to run on a single CPU core. However, since there is no data dependency in the steps that represent the bulk of the computation, our protocols scale virtually linearly with the number of available cores.

We instantiated DGK with a 1024-bit modulus. We also set the security parameter $t = 160$ and $u = 2^{20}$, since the largest plaintext value in our dataset does not require over 19 bits. Our garbled circuit implementation uses the OT protocol in [33] for transferring keys corresponding to input wires. It reduces OT_L^M to $\mathrm{OT}_\kappa^\kappa$. We set the security parameter $\kappa = 80$, $M = 20$ (since we selected $u = 2^{20}$) and $L = 128$ (the symmetric key size)[6]. We

[6] L is dictated by the key size of AES – used to encrypt input wires in the garbled circuit – rather than by security reasons. In fact, using an 80-bit key would provide the desired level of security. However, performance-wise there would be virtually no difference.

assume that the data-independent part of OT is performed by C and S prior to running AS-PPCML. All performance results in this section correspond to the average of 50 runs. Step-3 of the PPCML protocol is optimized by pushing most of the computation to the unencrypted domain: S computes $\sum_i y_{n,i}^2$ and then encrypts the result.

On-Line Computation Complexity.

Table 2: Breakdown of the server- and client-side on-line computation of our PPCML protocol for $v = w = 300$

Server		Client	
Step-3: Euclidean Distance	518.9 s	Step-4: Comparison	1096.83 s
Step-4: Comparison	125.15 s	Step-7: Decryption	39.11 ms
Step-5: Multiplication	179.4 ms		
Step-6: Exponentiation	7.386 ms		
Total	≈ 10.7 min	Total	≈ 18.3 min

Table 2 illustrates our measurements, where both C and S hold 300 feature vectors (i.e., $v = w = 300$). For C, the total cost is dominated by the homomorphic comparison, while the most expensive step for S is the computation of the Euclidean distance.

Table 3: Breakdown of the server- and client-side on-line computation of our basic S-PPCML protocol for $v = w = 300$

Server		**Client**	
Step-3: Euclidean Distance	518.9 s	Step-4: Comparison	1096.83 s
Step-4: Comparison	125.15 s	Step-6: Comparison	3.66 s
Step-5: Multiplication	179.4 ms	Step-8: Decryption	0.13 ms
Step-6: Comparison	417.2 ms		
Step-7: Multiplication	0.598 ms		
Total	≈ 10.7 min	Total	≈ 18.3 min

Table 4: Breakdown of the server-side and client-side on-line computation of our AS-PPCML protocol for $v = w = 300$

Server		**Client**	
Step-3: Euclidean Distance	518.9 s	Step-5-a: Decryptions	11.7 s
Step-4: Randomization	180 ms	Step-5-b: Comparison	11.1 s
Step-7: Comparison	10.8.s		
Total	≈ 8.8 min	Total	22.8 s

Tables 3 shows the computation cost of our basic S-PPCML protocol, while Table 4 shows the breakdown of the computations of the AS-PPCML Protocol. The use of a garbled circuit for comparing Euclidean distance with the threshold has a great impact on the performance of the AS-PPCML protocol. In particular, total time is reduced by a 1.2x factor for the server and by a 48x for the client.

On-Line Communication Complexity. The on-line communication cost is proportional to $v \cdot w$. Let $|N|$ indicate the number of bits corresponding to a DGK ciphertext. The following exchanges of information contribute to the total bandwidth (on-line) required by the PPCML protocol:

- The encrypted vectors sent by C to S account for $((2701 + 1) \cdot v) \cdot |N|$ bits.
- The homomorphic-based comparison – $(2 \cdot M + 3) \cdot w \cdot v \cdot |N|$ bits.
- The results sent by S to C – $v \cdot |N|$ bits.

Thus, the on-line data exchanged between C and S amounts to $(2701 \cdot v + (2 \cdot M + 3) \cdot w \cdot v + v) \cdot |N|$ bits. In our setting, this amounts to 572 MB.

Similarly, the on-line communication cost of the S-PPCML protocol is $(2701 \cdot v + (2 \cdot M + 3) \cdot w \cdot v + (2 \cdot M + 3) \cdot v + 1) \cdot |N|$ bits, i.e., 573 MB in our setting.

Finally, the AS-PPCML protocol relies on a garbled circuit for comparison, which incur on-line communication cost of $2 \cdot M \cdot (L + \kappa) \cdot w \cdot v$ bits. Therefore the total cost of the AS-PPCML protocol is $((2701 \cdot v + w \cdot v) \cdot |N| + 2 \cdot M \cdot (L + \kappa) \cdot w \cdot v)$ bits, corresponding to 200 MB in our setting.

8 Conclusion

In this paper we have introduced a set of protocols that implement PPCML and S-PPCML/AS-PPCML functionalities. The first allows two parties representing two user communities – e.g., two review websites – to *privately* determine which users belong to both communities. The second protocol allows the parties to privately compute how many users they have in common. Our protocols compare user-generated content rather than user identifiers, such as user-IDs or IP addresses.

We implement our protocols and measure their performance on commodity hardware. Our results indicate that the overhead introduced by the privacy-preserving computation is relatively small. In particular, two parties which hold 300 users each can determine the number of common users in a matter of minutes.

As for the future work, we plan to optimize our protocols for multi-core CPUs. Parallel implementation of our protocols can provide significant speedup, allowing clusters with hundreds of CPUs to run protocols over sets of millions of users.

References

1. Amazon. `http://www.amazon.com`.
2. Facebook. `http://www.facebook.com`.
3. Facebook Reports Third Quarter 2012 Results. `http://investor.fb.com/releasedetail.cfm?ReleaseID=715607`.
4. Google+. `https://plus.google.com`.
5. Linkedin. `http://www.linkedin.com`.
6. TripAdvisor. `http://www.tripadvisor.com`.
7. Twitter. `http://www.twitter.com`.

8. Yelp. `http://www.yelp.com`.
9. Yelp – About Us. `http://www.yelp.com/about`.
10. Yelp – Terms of Service. `http://www.yelp.com/static?country=US&p=tos`.
11. A. Abbasi and H. Chen. Writeprints: A Stylometric Approach to Identity-Level Identification and Similarity Detection in Cyberspace. In *ACM Transactions on Information Systems*, 2008.
12. M. Almishari and G. Tsudik. Exploring linkability in user reviews. In *European Symposium on Research in Computer Security*, 2012.
13. R. Baeza-Yates. Modern Information Retrieval. In *Addison-Wesley Longman Publishing Co., Inc.*, 1999.
14. C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
15. M. Blanton and P. Gasti. Secure and efficient protocols for iris and fingerprint identification. In *ESORICS*, pages 190–209, 2011.
16. C. Blundo, E. De Cristofaro, and P. Gasti. EsPRESSo: Efficient Privacy-Preserving Evaluation of Sample Set Similarity. In *ESORICS Workshop on Data Privacy Management (DPM)*, 2012.
17. A. Border. On the resemblance and containment of documents. Compression and Complexity of Sequences, 1997.
18. M. Brennan and R. Greenstadt. Practical Attacks Against Authorship Recognition Techniques. In *IAAI*, 2009.
19. S. Choi, J. Katz, R. Kumaresan, and H. Zhou. On the security of the free-xor technique. In Ronald Cramer, editor, *Theory of Cryptography*, volume 7194 of *Lecture Notes in Computer Science*, pages 39–53. Springer Berlin / Heidelberg, 2012.
20. R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT*, pages 280–300, 2001.
21. E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and Private Computation of Cardinality of Set Intersection and Union. In *CANS*, 2012.
22. I. Damgård, M. Geisler, and M. Krøigård. A correction to efficient and secure comparison for on-line auctions. Cryptology ePrint Archive, Report 2008/321, 2008.
23. I. Damgård, M. Geisler, and M. Krøigård. Homomorphic encryption and secure comparison. *Journal of Applied Cryptology*, 1(1):22–31, 2008.
24. I. Damgård, M. Geisler, and M. Krøigård. Asynchronous multiparty computation: Theory and implementation. In *Public Key Cryptography (PKC)*, pages 160–179, 2009.
25. E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography*, 2010.
26. Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Privacy Enchancing Technologies Symposium (PETS)*, pages 235–253, 2009.
27. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt*, 2004.
28. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.
29. C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, 2008.
30. C. Hazay and K. Nissim. Efficient Set Operations in the Presence of Malicious Adversaries. In *PKC*, 2010.
31. W. Henecka, S. Kogl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: Tool for Automating Secure Two-partY computations. In *ACM Conference on Computer and Communications Security (CCS)*, pages 451–462, 2010.
32. F. Iqbal, H. Binsalleeh, B. Fung, and M. Debbabi. A unified data mining solution for authorship analysis in anonymous textual communications. In *Information Sciences (INS): Special Issue on Data Mining for Information Security*, 2011.

33. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious tranfers efficiently. In *Advances in Cryptology – CRYPTO*, pages 145–161, 2003.
34. S. Jarecki and X. Liu. Fast secure computation of set intersection. In *SCN*, 2010.
35. N. Jindal and B. Liu. Opinion Spam and Analysis. In *ACM International Conference on Web Search and Data Mining*, 2008.
36. Lea Kissner and Dawn Song. Privacy-preserving set operations. In *Crypto*, 2005.
37. V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology and Network Security (CANS)*, pages 1–20, 2009.
38. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 486–498, 2008.
39. D. Lewis. Naive(bayes) at forty:the independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*, 1998.
40. M. Liedel. Secure distributed computation of the square root and applications. In *ISPEC*, pages 277–288, 2012.
41. D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – a secure two-party computation system. In *USENIX Security Symposium*, pages 287–302, 2004.
42. A. McDonald, S. Afroz, A Caliskan, A Stolerman, and R. Greenstadt. Use Fewer Instances of the Letter "i": Toward Writing Style Anonymization. In *PETS*, 2012.
43. A. Narayanan, H. Paskov, N. Gong, J. Bethencourt, E. Stefanov, E. Shin, and D. Song. On the Feasibility of Internet-Scale Author Identification. In *IEEE Symposium on Security and Privacy*, 2012.
44. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238, 1999.
45. B. Pinkas, T. Schneider, N. Smart, and S. Williams. Secure two-party computation is practical. In *Advances in Cryptology – ASIACRYPT*, volume 5912 of *LNCS*, pages 250–267, 2009.
46. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *ACM Symposium on Theory of Computing (STOC)*, pages 73–85, 1989.
47. A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *International Conference on Information Security and Cryptology (ICISC)*, pages 229–244, 2009.
48. E. Stamatatos. A Survey of Modern Authorship Attribution Methods. In *Journal of the American Society for Information Science and Technology*, 2009.
49. P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Addison-Wesley, 2005.
50. K. Toutanova, D. Klein, C. Manning, and Y. Singer. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *HLT-NAACL*, 2003.
51. A. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.

A Security Analysis

Security of the protocol presented in Section 5 is based on that of security assumptions about our building blocks. In particular, we assume that DGK encryption is semantically secure. This was shown in [23,22] under the RSA setting.

We now outline how to simulate the view of C and S using each party's inputs and outputs only. We show that such simulation is indistinguishable from a real execution of the protocol. This allows us to claim that the protocol is secure in the honest-but-curious (HbC) model.

C's input consists of a feature vector and a private key, while its output is a single bit b. Given these values, the simulator constructs messages to C as follows: during the comparison protocol (Step 4) the simulator sends encryptions of random values to C. Since DGK is semantically secure, C cannot detect it. Then, if $b = 0$ the simulator returns to C $u = \mathsf{Enc}(0)$ and $u = \mathsf{Enc}(r)$ (for a random r) otherwise. Since the outcome of decryption is distributed identically to that what C expects, simulation cannot be detected.

S's input is a database consisting of w feature vectors; S has no output. The simulator encrypts two random values per each element of the feature vector and sends them to S. Since DGK is semantically secure, S cannot detect that the message from the simulator represents encryption of random values. During privacy-preserving comparison, the simulator sends encryption of random values to S (Step 4). S, however, cannot decide with any non-negligible probability that these values are indeed random.

An analogous argument extends to the protocols in Section 6. However, security of these protocols relies on two additional assumptions: (1) oblivious transfer used is for garbled circuit evaluation is secure; and (2) garbled circuit evaluation is secure.

Assumption (1) holds if the hash function used to instantiate the oblivious transfer protocol in [33] is either correlation-robust, or modeled as a random oracle. Also, [33] requires the use of a secure pseudorandom generator. With respect to (2), security of garbled circuits with "free-XOR" was proven under the assumption that the hash function is correlation-robust under the definition of [19], or is instantiated as a random oracle.