Distance Computation between Two Private Preference Functions

Alberto Blanco, Josep Domingo-Ferrer, Oriol Farràs, and David Sánchez

Universitat Rovira i Virgili
Dept. of Computer Engineering and Mathematics
UNESCO Chair in Data Privacy
Av. Països Catalans 26
E-43007 Tarragona, Catalonia
{alberto.blanco,josep.domingo,oriol.farras,david.sanchez}@urv.cat

Abstract. We consider the following problem: two parties have each a private function, for example one that outputs the party's preferences on a set of alternatives; they wish to compute the distance between their functions without any of the parties revealing its function to the other. The above problem is extremely important in the context of social, political or business networks, whenever one wishes to find friends or partners with similar interests without having to disclose one's interests to everyone. We provide protocols that solve the above problem for several types of functions. Experimental work demonstrates that privacy preservation does not significantly distort the computed distances.
Keywords: Private distance computation, privacy, social networks, utility functions, preferences, user profiles, private matching.

1 Introduction

Shyness often has a rational component. Getting to know a stranger normally requires us to disclose some of our privacy to that stranger. Indeed, in a fair relationship there is normally a tit-for-tat approach, in which each of the parties must disclose something in order to learn something. It would be more privacy-preserving and less risky if two parties could determine whether they have similar interests without *prior* disclosure of such interests to each other. Of course, the more similar their interests turn out to be, the greater the *posterior* mutual disclosure: in the extreme case, if their interests turn out to be at distance 0, total mutual disclosure occurs.

In game-theoretic terms, the above problem can be expressed as two players being interested in determining how close their utility

functions are without disclosing these utility functions to each other. In particular, this allows forming coalitions with homogeneous interests without prior utility disclosure.

Solving this problem is very relevant in many practical situations:

- In social networks, users could find friends or other users to follow who share their interests, without being forced to disclose their own private interests (*e.g.* religion, sexual orientation, health condition, etc.). For example, in the social network PatientsLikeMe [12] users currently need to disclose their diseases in order to find users with similar ailments, a privacy loss which could be mitigated by our approach.
- Grooming attacks in social networks could be hindered to a good extent with our approach. Note that the groomer would need to guess the victim's interests in order to become her/his friend.
- Targeted consumer profiling could be also made possible. A company could create dummy users of a social network with the profile of the consumers it is looking for. In this way, the company could identify communities of potential consumers with the desired profile, without encroaching on the privacy of people who do not fit the profile being sought.
- In commercial transactions, parties could determine whether the values they assign to a collection of goods are similar, without prior disclosure of their chosen value(s). For example, in some cases a company may be interested in associating with companies with *dissimilar* interests, in order to form a complementary partnership, rather than associating with companies with too similar interests, which may be regarded as competitors.
- In job recruitment, companies and candidates would be able to confidentially determine to what extent the corporate vision is shared by each candidate. Thanks to the privacy-preserving mechanism, a lot of different factors could be included in the evaluation, without the company being forced to reveal its strategic goals to unsuccessful candidates or the latter needing to disclose their views.

1.1 Contribution and plan of this paper

We present in this paper privacy-preserving protocols that allow computing the distances between various types of functions. After that, we report on experimental work that shows that privacy preservation does not cause a significant distortion in the computed distance.

Section 2 defines several cases of privacy-preserving distance computation between functions, depending on the nature of the function and the type of distance being considered. Section 3 describes a protocol for privacy-preserving distance computation based on set intersection. Section 4 reports on experimental work. Section 5 describes related work. Conclusions and future research lines are summarized in Section 6.

2 Taxonomy of distance computation between functions

The protocol to compute the distance between two functions in a privacy-preserving way depends on the nature of the functions and the way the distance is to be measured. We next discuss several cases.

2.1 Case A: counting common qualitative preferences

In this case, the interests or preferences of each player are characterized as binary features on various independent topics. For example, in a social networks like Facebook, users are asked to provide their opinions on different topics as "like" or "do not like". In PatientsLikeMe [12], users are requested to detail their medical histories as binary selections from sets of alternatives (diseases, symptoms, etc.).

We can consider the preferences or profile of a player as a set containing his/her opinions or personal details. Let this set be X for the first player $\mathcal{C}$ and Y for second player $\mathcal{S}$. Then the distance between the interests of $\mathcal{C}$ and $\mathcal{S}$ can be evaluated as the multiplicative inverse of the size of the intersection of X and Y, that is $1/|X \cap Y|$, when the intersection is not empty. If it is empty, we say that the distance is ∞.

Clearly, the more the coincidences between X and Y, the more similar the preferences of both players, and the smaller the distance between them.

2.2 Case B: correlating qualitative preferences

As in the previous case, the players' preferences are expressed as qualitative features. However, if these features are not independent (*e.g.* related diseases) or they are not binary (*e.g.* expressed as free textual answers to questionnaires), the distance between the players' profiles cannot be computed as the size of the intersection between their sets of features. For example, if $\mathcal{C}$ suffers from anorexia and $\mathcal{S}$ from bulimia, there is some coincidence between them because they both present eating disorders. This coincidence must be captured by the resulting distance.

Assume we have a correlation function $s : E \times E \mapsto \mathbb{Z}_+$ that measures the similarity between the values in the sets of features of $\mathcal{C}$ and $\mathcal{S}$, where E is the domain where the sets of features of both players take values. For nominal features (*e.g.* disease names), semantic similarity measures can be used for this purpose [13]; for numerical features that take values over bounded and discrete domains (*e.g.* ages, zip codes), standard arithmetic functions can be used. Assume further that both players know this function s from the very beginning.

Here the distance between the set X of $\mathcal{C}$'s features and the set Y of $\mathcal{S}$'s features can be computed as

$$1/(\sum_{x \in X} \sum_{y \in Y} s(x, y))$$

when the denominator is nonzero. If it is zero, we say that the distance is ∞.

2.3 Case C: quantitative preference functions

In this case, we want to compute the difference between two quantitative functions over the same domain, which define the preferences or profiles of the two players. We assume that they are integer functions. That is, $\mathcal{C}$ has a private preference function $f : E \to \mathbb{Z}$ and $\mathcal{S}$ has a private preference function $g : E \to \mathbb{Z}$.

A way to measure the dissimilarity between f and g is to compute $d(f,g) = \sum_{i=1}^{t} |f(x_i) - g(x_i)|$, where $D = \{x_1, \ldots, x_t\}$ is a representative discrete subset of elements from E.

This scenario fits well the usual way of learning, modeling and managing social network user profiles in the literature [1, 16, 21]. It consists in associating a vector of weights to each user, where each weight expresses the preference of the user on a certain topic (*e.g.* sports, science, health, etc.). Users are thus compared according to the distance between their vectors of weights.

3 Computing distances based on set intersection

It will be shown further below that the above three cases A, B and C can be reduced to computing the cardinality of set intersections. Hence, we first review the literature for solutions to secure two-party computation of set intersection cardinality.

Secure multiparty computation (MPC) allows a set of parties to compute functions of their inputs in a secure way without requiring a trusted third party. During the execution of the protocol, the parties do not learn anything about each other's input except what is implied by the output itself. There are two main adversarial models: honest-but-curious adversaries and malicious adversaries. In the former model, the parties follow the protocol instructions but they try to obtain information about the inputs of other parties from the messages they receive. In the latter model, the adversary may deviate from the protocol in an arbitrary way.

We will restrict here to a two-party setting in which the input of each party is a set, and the desired output is the cardinality of the intersection of both sets. The intersection of two sets can be obtained by using generic constructions based on Yao's garbled circuit [20]. This technique allows computing any arithmetic function, but for most of the functions it is inefficient. Many of the recent works on two-party computation are focused on improving the efficiency of these protocols for particular families of functions.

Freedman, Nissim, and Pinkas [4] presented a more efficient method to compute the set intersection, a *private matching scheme*, that is

secure in the honest-but-curious model. A private matching scheme is a protocol between a client $\mathcal{C}$ and a server $\mathcal{S}$ in which $\mathcal{C}$'s input is a set X of size $i_{\mathcal{C}}$, $\mathcal{S}$'s input is a set Y of size $i_{\mathcal{S}}$, and at the end of the protocol $\mathcal{C}$ learns $X \cap Y$. The scheme uses polynomial-based techniques and homomorphic encryption schemes.

Several variations of the private matching scheme were also presented in [4]: an extension of the malicious adversary model, an extension of the multi-party case, and schemes to compute the cardinality of the set intersection and other functions. Constructing efficient schemes for set operations is an important topic in MPC and has been studied in many other contributions. Several works such as [2, 3, 5, 10, 15] present new protocols to compute the set intersection cardinality.

We now proceed to specifying protocols to deal with cases A, B and C above. In all cases, the distance between the private preferences of the two parties is computed using a protocol that yields the cardinality of a set intersection.

3.1 Case A

$\mathcal{C}$ inputs X and $\mathcal{S}$ inputs Y, and they want to compute $|X \cap Y|$ without revealing their own set.

In the protocol specified below, $\mathcal{C}$ inputs $X = \{a_1, \ldots, a_s\} \subseteq E$, $\mathcal{S}$ inputs $Y = \{b_1, \ldots, b_t\} \subseteq E$, where s and t are known, and finally $\mathcal{C}$ learns $|X \cap Y|$. For $\mathcal{S}$ to learn also $|X \cap Y|$, the protocol below should be run a second time (sequentially or concurrently) with the roles of $\mathcal{C}$ and $\mathcal{S}$ being exchanged.

We will use the protocol described in [4] for the cardinality of the set intersection, that is secure in the honest-but-curious model. The homomorphic encryption scheme we use is the Paillier cryptosystem [11]. The protocol exploits the property that, given three elements m_1, m_2, m_3, it is possible to efficiently compute $Enc(m_1 + m_2)$ and $Enc(m_1 \cdot m_3)$ from $Enc(m_1)$, $Enc(m_2)$, and m_3. We assume that $\mathcal{C}$ and $\mathcal{S}$ agree on a way to represent the elements of E as elements of the Enc function. They also agree on a special string m. The protocol is outlined next.

Step 1 $\mathcal{C}$ chooses the secret-key parameters, and publishes its public keys and parameters.

Step 2 $\mathcal{C}$ computes the polynomial $p(x) = \prod_{i=1}^{s}(x - a_i)$.

Step 3 $\mathcal{C}$ sends $Enc(p_0), \ldots, Enc(p_s)$ to $\mathcal{S}$, where p_i is the coefficient of degree i of p.

Step 4 $\mathcal{S}$ picks a random element $r_j \in \mathbb{Z}_n$ for every $1 \leq j \leq t$. $\mathcal{S}$ computes $Enc(r_j \cdot p(b_j) + m)$ for $1 \leq j \leq t$. Then $\mathcal{S}$ sends these ciphertexts to $\mathcal{C}$.

Step 5 $\mathcal{C}$ decrypts the received ciphertexts. The result of each decryption is m or a random element.

If the size of the domain of Enc is much larger than $|X|$, the scheme computes $|X \cap Y|$ with high probability: indeed, the number of times that m is obtained in the last step indicates the number of common elements in X and Y. Observe that $\mathcal{C}$ learns $|X \cap Y|$, but $\mathcal{C}$ does not learn any additional information about Y or $X \cap Y$ (in particular, $\mathcal{C}$ does not learn the elements of these sets). Moreover $\mathcal{S}$ cannot distinguish between cases in which $\mathcal{C}$ has different inputs.

3.2 Case B

Here, $\mathcal{C}$ inputs X and $\mathcal{S}$ inputs Y, two sets of features, and they want to know how close X and Y are without revealing their set. In the protocol below, only $\mathcal{C}$ learns how close X and Y are; for $\mathcal{S}$ to learn it as well, the protocol should be run a second time (sequentially or concurrently) with the roles of $\mathcal{C}$ and $\mathcal{S}$ being exchanged.

We assume that the domain of X and Y is the same, and we call it E. The closeness or similarity between elements is computed by means of a function s. In particular, we consider functions $s : E \times E \to \mathbb{Z}_+$. Observe that Case A is a particular instance of this Case B in which $s(x, x) = 1$ and $s(x, y) = 0$ for $x \neq y$.

Let Y be the input of $\mathcal{S}$. For every $z \in E$, $\mathcal{S}$ computes $\ell_z = \sum_{y \in Y} s(z, y)$. Observe that ℓ_z measures the overall similarity of z and Y. Let $Y' = \{z \in E : \ell_z > 0\}$. It is common to consider functions satisfying $s(z, z) > 0$ for every $z \in E$, and so in general $Y \subseteq Y'$.

A protocol for such a computation can be obtained from the previous protocol, by replacing Step 4 with the following one:

Step 4' For every $z \in Y'$, $\mathcal{S}$ picks ℓ_z random elements $r_1, \ldots, r_{\ell_z} \in \mathbb{Z}_n$ and computes $Enc(r_j \cdot p(z) + m)$ for $1 \leq j \leq \ell_z$. Then $\mathcal{S}$ sends all these ciphertexts to $\mathcal{C}$.

Thus, for every $z \in Y'$, $\mathcal{S}$ sends ℓ_z ciphertexts. In Step 5 $\mathcal{C}$ will recover m from these ciphertexts only if $z \in X$. Hence, at the end of the protocol the total number of decrypted messages that are equal to m is

$$\sum_{x \in X} \ell_x = \sum_{x \in X} \sum_{y \in Y} s(x, y),$$

that is, the sum of similarities between the elements of X and Y. This clearly measures how similar X and Y are. At the end of the protocol $\mathcal{C}$ knows $|Y'|$ and $\mathcal{S}$ knows $|X|$. Besides that, $\mathcal{C}$ and $\mathcal{S}$ cannot gain any additional knowledge on the elements of each other's set of preferences.

3.3 Case C

Here $\mathcal{C}$ inputs a private function f and $\mathcal{S}$ inputs a private function g, and they want to know how close these functions are without revealing them to each other.

The value $d(f, g)$ will be computed in a vectorial setting. We assume that $f, g : E \to \mathbb{Z}_+$. Note that if f or g take negative values, then $\mathcal{C}$ and $\mathcal{S}$ can define $f' : E \to \mathbb{Z}_+ : x \mapsto f(x) + c$ and $g' : E \to \mathbb{Z}_+ : x \mapsto g(x) + c$ for some large enough constant $c \in \mathbb{Z}_+$. Observe that $d(f, g) = d(f', g')$.

Given $D = \{x_1, \cdots, x_t\} \subseteq E$ publicly known, $\mathcal{C}$ defines the vector $\mathbf{u} = (u_1, \ldots, u_t) \in \mathbb{Z}_+^t$, where $u_i = f(x_i)$ for $i = 1, \ldots, t$, and $\mathcal{S}$ defines $\mathbf{v} = (v_1, \ldots, v_t) \in \mathbb{Z}_+^t$, where $v_i = g(x_i)$ for $i = 1, \ldots, t$. The problem described in Section 2.3 can be reduced to computing $\|\mathbf{u} - \mathbf{v}\| = \sum_{i=1}^{t} |u_i - v_i|$.

Given $\mathbf{u}$ and $\mathbf{v}$, we define the sets $X = \{(i, j) : u_i > 0 \text{ and } 1 \leq j \leq u_i\}$ and $Y = \{(i, j) : v_i > 0 \text{ and } 1 \leq j \leq v_i\}$. Following the protocol for computing the cardinality of the set intersection presented above, $\mathcal{C}$ and $\mathcal{S}$ can compute $|X \cap Y|$ in a private way (the protocol needs to be run twice with the roles of $\mathcal{C}$ and $\mathcal{S}$ being exchanged in the second run). Observe that

$$\begin{aligned} |X \cap Y| &= |\{(i, j) : u_i > 0 \text{ and } v_i > 0 \text{ and } 1 \leq j \leq \min\{u_i, v_i\}\}| \\ &= \sum_{1 \leq i \leq t} \min\{u_i, v_i\}. \end{aligned}$$

According to [4], in addition to learning $|X \cap Y|$, during the protocol $\mathcal{S}$ learns $|X|$ and $\mathcal{C}$ learns $|Y|$. Hence both $\mathcal{C}$ and $\mathcal{S}$ can compute

$$\begin{aligned} |X| + |Y| - 2|X \cap Y| = \\ = \sum_{i=1}^{m} \max\{u_i, v_i\} + \min\{u_i, v_i\} - 2\sum_{i=1}^{m} \min\{u_i, v_i\} = \\ = \sum_{i=1}^{m} \max\{u_i, v_i\} - \min\{u_i, v_i\} = \sum_{i=1}^{m} |u_i - v_i| = \|\mathbf{u} - \mathbf{v}\| \end{aligned}$$

in a private way.

4 Experimental analysis

This section illustrates the applicability of the proposed protocols to compare profiles of social network users in a privacy-preserving way.

Empirical work was based on 16 Twitter users selected among the most relevant ones from `WeFollow` [18] and `WhoToFollow` [19]. These web sites rank and classify Twitter users in a set of categories. As done in [16, 17], we took the two 2012 most influential users for each of the following eight categories: Arts, Health, Shopping, Science, Computers, Sports, Society and Business.

Both client and server use the following computing environment: Asus S56C computer with an Intel core i7 3517U 8GB RAM DDR3 1600Mhz, running Ubuntu 13.10 and Java7 (opendjk-1.7). The size of the keys used is 1024 bits. The implementation of the Paillier cryptosystem is the one in [14], which we patched to evaluate polynomials using Horner's method.

We profiled each Twitter user by following the procedure described in [16]. In a nutshell, we extract the noun phrases from the user's most recent set of 100 tweets, and we classify them in the above eight categories. Then, the contribution of each noun phrase to the corresponding category is measured as its informativeness, computed from its distribution in the Web. The aggregated contributions of all the noun phrases of a category measure the interest of the user in the category topic. Profiles are thus represented by a set of vectors containing eight normalized weights, each one quantifying the interest

of the user in each of the eight categories. For example, the profile of the Twitter user CERN, which corresponds to the European Center for Nuclear Research, is: {*Arts=15.1%, Health=0.27%, Shopping=1.79%, Science=47.93%, Computers=7.5%, Sports=5.45%, Society=10.65%, Business=11.31%*}, which shows a clear preference for Science-related topics. Thus, user profiles can be seen as preference functions that can be completely represented by a discrete set of eight quantitative features. This fits the case C discussed above, whose privacy preserving protocol is presented in Section 3.3.

To evaluate the suitability of the privacy-preserving protocol in terms of accuracy, we first computed the pairwise distance d between all 16 user profiles as described in Section 2.3: $d(f,g) = \sum_{i=1}^{m} |f(x_i) - g(x_i)|$, where $x_i \in \{$*Arts, Health, Shopping, Science, Computers, Sports, Society, Business*$\}$, and f and g represent the profiles of two different users by assigning the user's weight to each input category x_i. Then, we computed the same pairwise distances using the privacy-preserving protocol described in Section 3.3.

Since our protocol assumes that the functions f and g to be compared output integer values, in a first experiment we rounded weights to the nearest integer. To measure the accuracy of the results, we computed the average error between the distances obtained by straightforward (non-private) computation and the distances obtained by our privacy-preserving distance computation. Such an average error was 1.69% with a standard deviation of 2.25%. This shows that our protocol does not cause a significant distortion in spite of the rounding needed to accommodate values.

On the other hand, the average run time for a privacy-preserving distance computation was 36.7 seconds, whereas it was negligible for a non-private computation. From direct analysis of the protocol (Section 3.3), it can be seen that the run time for a privacy-preserving distance computation depends on the number of weights to be compared (eight) and on the ranges of such weights. Since we rounded percentages to be integers between 0 and 100, the range of weights was 100.

In situations in which reducing the response time is especially important, one may sacrifice some accuracy to speed by using an integer representation with a smaller range. For example, by dividing all weights by 10 and rounding to the nearest integer, we reduce the

weight range to 10, which in turn reduces the required number of encryption/decryption steps in the set intersection protocol by an order of magnitude. We did this and we obtained an average run time of 2.7 seconds per privacy-preserving distance computation. However, the average error with respect to non-private distances was 18.49% with a standard deviation of 17.8%, which illustrates how the (lack of) accuracy in the input discretization impacts on the (lack of) accuracy of the output.

Last but not least, we examined scalability. Figure 1 shows the growth of the computing times for $\mathcal{C}$ and $\mathcal{S}$ as the size of X and Y grow. The linear behavior is clear, with $\mathcal{S}$ having a higher computational load than $\mathcal{C}$. More on computational complexity is discussed in Section 5 below.

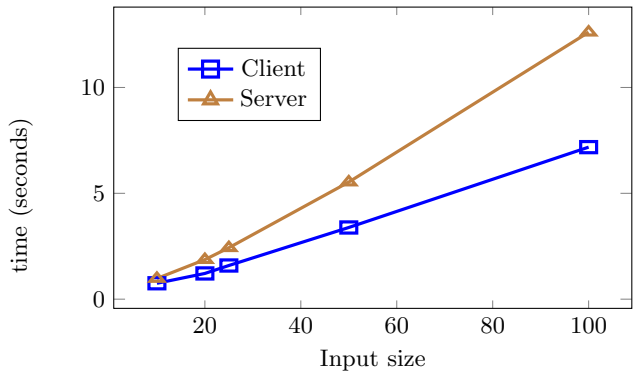

Fig. 1. Execution time of the client $\mathcal{C}$ and the server $\mathcal{S}$ for different input sizes

5 Related work

We consider the problem of computing the distance between two private utility or preference functions. In the proposed cases, we deal with discrete or discretized domains, which allows us to lean on the literature on private record matching. In this type of matching, the problem is slightly different: it consists of matching the records of the same entity (individual, company, etc.) distributed across different data sets, while keeping those data sets private to their owners. There are three main approaches in the private record matching literature: sanitization-based, cryptographic and hybrid.

In *sanitization-based* methods, matching is performed on perturbed versions of the private data sets, in order to protect them against disclosure; a survey of perturbation/sanitization methods and record linkage methods can be found in [6]. This class of methods is usually computationally efficient but it incurs an obvious accuracy toll: matching on perturbed data sets is less accurate than matching on the original data sets. Indeed, false positive and false negative matches can appear.

Cryptographic methods are based on secure multiparty computation and they achieve privacy without accuracy loss. As mentioned above, we follow this approach, as we use MPC to compute the cardinality of set intersection, specifically the protocol in [4]. Our approach could be easily adapted to use other set intersection cardinality protocols in the literature, such as the ones mentioned at the beginning of Section 3 ([2, 3, 5, 10, 15]).

The communication complexity of our protocol is $O(i_{\mathcal{C}} + i_{\mathcal{S}})$, where $i_{\mathcal{C}}$ and $i_{\mathcal{S}}$ are the sizes of $\mathcal{C}$'s input and $\mathcal{S}$'s input, respectively. $\mathcal{C}$'s computation complexity is $O(i_{\mathcal{C}}+i_{\mathcal{S}})$, while $\mathcal{S}$'s computation complexity is in fact $O(i_{\mathcal{C}}i_{\mathcal{S}})$, but it can be reduced to $O(i_{\mathcal{C}} \log\log i_{\mathcal{S}})$ [4]. The computational complexity of the scheme in [3] is linear in $i_{\mathcal{C}}+i_{\mathcal{S}}$. Other protocols like the ones presented in [10] do not differentiate the users $\mathcal{C}$ and $\mathcal{S}$: both receive the set intersection cardinality at the end of the protocol.

There are also solutions for the private computation of the set intersection cardinality of $n > 2$ sets from n parties, and constructions that are secure in the malicious adversary model [2–5, 10, 15]. In a private matching scheme, the size of the inputs is known by both parties. Some techniques presented in [2] allow hiding the size of the inputs, but the communication and computation complexity of the resulting protocol is higher.

Hybrid methods try to achieve a trade-off between the sanitization-based and cryptographic methods, to retain as much accuracy as possible while keeping computational complexity reasonable. The idea is to use a blocking phase that operates over sanitized data in order to rule out pairs of records that do not satisfy the matching condition. Then cryptographic MPC matching is applied only to blocks of both data sets which contain pairs of candidate matches. Methods following this approach are [7] (where k-anonymity is used as saniti-

zation), [8] (where differential privacy is used as sanitization) and [9] (which refines [8]). The approach proposed in our paper could also be adapted to the hybrid approach, as the final MPC stage of this approach can also be implemented via set intersection.

6 Conclusions and future research

Computing the distance between the private preference functions held by two different parties is very relevant in a number of applications. We have motivated several application scenarios in which the private functions express the preferences or profiles of the parties or, in game-theoretic terms, the utility of the players. These scenarios include finding friends or partners with similar interests in social networks, hindering grooming attacks, etc.

We have defined the problem for several types of private functions and, for each function type, we have given a protocol that solves the problem based on two-party secure computation of the cardinality of a set intersection. Empirical work shows that preserving the privacy of the preferences does not meaningfully affect the accuracy of the distances obtained.

Future work will involve designing a protocol to compute the distance between two private functions based on their *whole* domain E, rather than on a discrete subset D. This general approach will require dealing with continuous domains.

Acknowledgments

This work was partly supported by the Government of Catalonia under grant 2009 SGR 1135, by the Spanish Government through projects TIN2011-27076-C03-01 "CO-PRIVACY", TIN2012-32757 "ICWT", IPT-2012-0603-430000 "BallotNext" and CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES", and by the European Commission under FP7 projects "DwB" and "Inter-Trust". J. Domingo-Ferrer is partially supported as an ICREA Acadèmia researcher by the Government of Catalonia. The authors are with the UNESCO Chair in Data Privacy, but they are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization.

References

1. F. Abel, Q. Gao, G.J. Houben and K. Tao, "Semantic enrichment of Twitter posts for user profile construction on the social web", in *ESWC'11*, pp. 375–389, 2011.
2. M. Blanton and E. Aguiar, "Private and oblivious set and multiset operations", in *ASIACCS 2012*, Springer, pp. 40-41, 2012.
3. E. De Cristofaro, P. Gasti and G. Tsudik, "Fast and private computation of cardinality of set intersection and union", in *CANS 2012*, Springer, pp. 218-231, 2012.
4. J. Freedman, K. Nissim and B. Pinkas, "Efficient private matching and set intersection", in *EUROCRYPT 2004*, Springer, pp. 1-19, 2004.
5. S. Hohenberger and S. Weis, "Honest-verifier private disjointness testing without random oracles", in *PET 2006*, Springer, pp. 277–294, 2006.
6. A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. Schulte Nordholt, K. Spicer and P.-P. de Wolf, *Statistical Disclosure Control*, Wiley, 2012.
7. A. Inan, M. Kantarcioglu, E. Bertino and M. Scannapieco, "A hybrid approach to private record linkage", in *Proc. IEEE 24 Intl. Conf. Data Eng.-ICDE 2008*, pp. 496-505, 2008.
8. A. Inan, M. Kantarcioglu, G. Ghinita and E. Bertino, "Private record matching using differential privacy", in *Proc. of the 13th Intl. Conference on Extending Database Technology-EDBT'10*, pp. 123-134, 2010.
9. A. Inan, M. Kantarcioglu, G. Ghinita and E. Bertino, "A hybrid approach to private record matching", *IEEE Transactions on Dependable and Secure Computing*, 9(5):684-698, 2012.
10. L. Kissner and D. X. Song, "Privacy-preserving set operations", in *CRYPTO 2005*, Springer, pp. 241–257, 2005.
11. P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes", in *EUROCRYPT 1999*, Springer, pp. 223-238, 1999.
12. PatientsLikeMe, `http://www.patientslikeme.com`
13. D. Sánchez, M. Batet, D. Isern and A. Valls, "Ontology-based semantic similarity: A new feature-based approach", *Expert Systems with Applications*, 39(9):7718-7728, 2012.
14. The Homomorphic Encryption Project (thep). `https://code.google.com/p/thep/`. Accessed March 17, 2014.
15. J. Vaidya and C. Clifton, "Secure set intersection cardinality with application to association rule mining", *Journal of Computer Security*, 13(4):593-622, 2005.
16. A. Viejo, D. Sánchez and J. Castellà-Roca, "Preventing automatic user profiling in Web 2.0 applications", *Knowledge-based Systems*, 36:191-205, 2012.
17. A. Viejo, D. Sánchez and J. Castellà-Roca, "Using profiling techniques to protect the user's privacy in Twitter", in *MDAI 2012*, Springer, pp. 161-172, 2012.
18. WeFollow, `http://wefollow.com`
19. WhoToFollow, `http://whotofollow.net`
20. A.C.-C. Yao, "How to generate and exchange secrets", in *FOCS 1986*, pp. 162-167, 1986.
21. K. Zoltan and S. Johann, "Semantic analysis of microposts for efficient people to people interactions", in *RoEduNet11*, pp. 14, 2011.