On Private Similarity Search Protocols

Sven Laur and Helger Lipmaa
Laboratory for Theoretical CS, Department of CS&E
Helsinki University of Technology, P.O.Box 5400, FIN-02015 HUT, Espoo, Finland
Email: {slaur,helger}@tcs.hut.fi

***Abstract*— In a private similarity search (PSS) protocol, a client receives from the database the entry, closest to her query, without either the client or the database getting to know more information than necessary. PSS protocols have potentially wide application in areas like bioinformatics, where precise queries might be impossible. We show that the previously proposed PSS protocols by Du and Atallah have serious weaknesses; in particular, some of their protocols can be broken by a semi-honest third party who observes a relatively small amount of traffic. In several cases, we show that even maximally securified versions of these protocols—when used as proposed by Du and Atallah—are not private in the sense, needed in the practice. We propose a few protocols that are better from the privacy viewpoint, but none of the proposed protocols is really efficient.**

***Index Terms*—Cryptanalysis, cryptographic protocols, privacy-preserving data-mining, private similarity search.**

I. INTRODUCTION

In a private similarity search (PSS) protocol [1], a client receives from the database (the index of the) the entry, closest to her query, without either the client or the database getting to know more information. Similarity search is used in many cases where, e.g., finding the exact match is impossible or infeasible, when the data is corrupted by noise or the user is really interested in similar objects. A canonical application area of the PSS is bioinformatics, together with related fields like biometrics. A motivating task could be, given a DNA sample of a criminal, to find the closest match in the genome database without compromising the safety of honest citizens. To implement such applications, one must address the privacy concerns. Otherwise, both clients and database maintainers would be discouraged to participate in such services.

One can expect private similarity search to be a hard problem, in particular since at some point during the PSS protocol, the participants have to find minimum over non-public distances. Private minimum finding is a well-known cryptographic hard problem and only generic inefficient protocols for it are known [2].

Thus, one cannot expect to design really practical protocols for the PSS that are secure in the sense of secure two-party computations [3]. The main goal of published research on the PSS [1] has been to propose *efficient* PSS protocols that are secure according to somewhat less stringent notions. For example, in [1] the authors propose several PSS protocols that are relatively efficient but make use of a conditionally trusted third party Ursula, who is assumed to follow the protocol and not to collaborate with any other party. Moreover, the protocols from [1] are only claimed to be secure against *ciphertext-only attack.*

Our contributions. We show that several protocols in [1] are insecure even following the weak security definitions of their authors. First, [1] proposes protocols for the PSS with respect to the Euclidean distance. This protocol essentially employs a private minimal scalar product (PMSP) protocol, using the help of Ursula to find the minimum. The PMSP protocol masks the real distances by using *additive masking functions*.

We show that the PMSP protocol of Du and Atallah [1] is completely insecure against ciphertext-only attacks. Namely, we show that a conditionally trusted third party Ursula can recover the queries by observing a small amount of traffic and using a straightforward matrix equations. Our attack against this protocol succeeds with a very high probability as soon as the database has a reasonable size. As a consequence, the full PSS protocol becomes insecure in practice.

After that, we show that all PMSP protocols, that are computed by using additive masking functions, must reveal the differences between the scalar products. We then propose two new SMSP protocols that do not reveal anything else, except these differences, in the security-model proposed by Du and Atallah (security against ciphertext-only attacks with the help of a semi-honest third party Ursula). This protocol is as efficient as the Du-Atallah PMSP protocol and quantifiably more secure.

However, ciphertext-only security is not sufficient in many practical applications, since additional public information about the database may easily disclose private information. We argue that practical PSS protocol must at least withstand statistical attacks (where the attacker knows something non-trivial about the database) and known-plaintext attacks. We show that any PMSP protocol, where Ursula learns distance differences, is insecure against very simple statistical attacks. While a refinement to the additive masking (the use of order-preserving affine transformation, defined later in this paper) can resist simple statistical attacks, it is not sufficient for more elaborate attacks. Finally, in Sect. IV, we propose the divide and conquer technique that provides more security, but it is also computationally more demanding.

We stress that also the protocols, proposed in this paper, are applicable only in limited environments, e.g., when only the ciphertext-only attacks are allowed. Still, the provably secure alternative of using Yao's alternative garbled circuit evaluation [4] is computationally too costly for large databases, and therefore our new protocols could be used in the practice when a trade-off between security and efficiency is desired. It is a major open problem to design a PSS protocol that is both secure and efficient.

Road-map. Section II introduces the reader to notation and preliminaries. Section III describes our attacks against MINDASP protocol and improved PSS protocols. Section IV gives a brief overview of other possible attack scenarios and solutions.

II. NOTATION AND PRELIMINARIES

Notation. A PSS database can consist of either discrete, continuous or hybrid vectors. For example, genome data can be represented over discrete alphabet, whereas biometric data is inherently continuous. For discrete data, we use elements of some quotient ring $\mathbb{Z}_t$, and we think of continuous parameters as fixed point real numbers that are mapped into $\mathbb{Z}_t$ by using a suitable affine transform.

For n-dimensional vectors, two standard distance functions are the Euclidean distance $d_2(\boldsymbol{x}, \boldsymbol{y}) := \left[\sum_i (x_i - y_i)^2\right]^{1/2}$ and the Manhattan distance $d_1(\boldsymbol{x}, \boldsymbol{y}) := \sum_i |x_i - y_i|$, where all calculations are done in $\mathbb{R}$. (If made in $\mathbb{Z}_t$, no modular reductions should appear in the computations.)

Cryptographic background. A public-key cryptosystem Π is a triple (G, E, D) of probabilistic polynomial-time algorithms for key-generation, encryption and decryption. A cryptosystem Π is homomorphic if $E_K(m_1; r) \cdot E_K(m_2; s) = E_K(m_1 + m_2; r \cdot s)$, where $+$ is a group operation and $\cdot$ is a groupoid operation, and semantically secure if no probabilistic polynomial-time adversary can distinguish between random encryptions of two elements, chosen by herself. See, e.g., [5], for an efficient semantically secure homomorphic public-key cryptosystem.

Linear-algebraic background. The next results are necessary to quantify our attacks in the next section. For the sake of completeness, we present them together with proofs.

First, let us denote the ring of $m \times n$ matrices over the ring $\mathbb{Z}_t$ by $\mathsf{Mat}_{m\times n}(\mathbb{Z}_t)$. Recall that a matrix equation $M\boldsymbol{x} = \boldsymbol{y}$ over the finite field $\mathsf{GF}(q) = \mathbb{Z}_q$ can have at most $q^{n-\mathsf{rank}(M)}$ solutions. Generally, when $\mathbb{Z}_t$ is a ring, the solution of $M\boldsymbol{x} = \boldsymbol{y}$ is unique iff the matrix M is left-invertible. Let $\mathcal{L}$ be the set of left-invertible matrices. Define $\mathsf{P}_{m\times n}(t) := \Pr[M \leftarrow \mathsf{Mat}_{m\times n}(\mathbb{Z}_t) : M \in \mathcal{L}]$.

Lemma 1: Let $m \geq n$, let $q, q_1, \ldots, q_\ell$ be some primes. Then the following claims hold: (a) $\mathsf{P}_{m\times n}(q) = \prod_{k=0}^{n-1}\left(1 - q^{-m+k}\right)$;
(b) $\mathsf{P}_{m\times n}(q^r) = \mathsf{P}_{m\times n}(q)$;
(c) $\mathsf{P}_{m\times n}(q_1^{r_1}\cdots q_\ell^{r_\ell}) = \prod_{i=1}^{\ell} \mathsf{P}_{m\times n}(q_i)$.

Proof: (a) Since $\mathbb{Z}_q$ is a finite field, a matrix M over $\mathbb{Z}_q$ is left-invertible iff all columns of M are linearly independent. Now, if the first k columns of M are linearly independent, they span a vector space of dimension k and size q^k. The probability that the next column avoids this space is $1 - q^k/q^m$ and thus probability that all n columns are linearly independent is $\prod_{k=0}^{n-1}(1 - q^{-m+k})$.
(b) Follows directly since M has a left-inverse modulo q^r iff $M \mod q$ has a left-inverse modulo q. (c) By the Chinese remainder theorem, matrices $M_i \equiv M \mod q_i^{r^i}$ for every i have left-inverses iff M does. Random sampling in $\mathbb{Z}_t$ is equivalent to random sampling modulo $q_i^{r_i}$ for every i, and thus the probability in this case is just a product of the probabilities given by case (a). ■

Private similarity search. A private similarity search (PSS) protocol has two parties, the querier Alice and the database owner Bob. Alice's private input is a vector (query) $\boldsymbol{x}$; Bob's private input is the database with vector elements $\boldsymbol{y_1}, \ldots, \boldsymbol{y_m}$. Assume that the similarity between two vectors is determined by a public distance (score) function $d(\cdot,\cdot)$. During a PSS protocol, Alice learns the *match index* b, s.t. $\boldsymbol{y_b} = \arg\min_{\boldsymbol{y_i}} d(\boldsymbol{x}, \boldsymbol{y_i})$. and the corresponding *match score* $d(\boldsymbol{x}, \boldsymbol{y_b}) = \min_i d(\boldsymbol{x}, \boldsymbol{y_i})$. If b is not unique, Bob may return a randomly chosen index that minimises b. Alice must gain no new information, except her private output (the match index and the match score). Bob must gain no new information. Some PSS protocols use a conditionally trusted third party Ursula, who must gain no new information during the protocol.

Du-Atallah protocol for finding minimal Euclidean distance. The task of finding the closest match can be simplified when $d(\cdot,\cdot)$ is the Euclidean distance. Then one can ignore the square root and compute $d^2(\boldsymbol{x}, \boldsymbol{y_i}) := \sum_{j=1}^{n}(x_j - y_{ij})^2$, where $\boldsymbol{y_i} = (y_{i1}, \ldots, y_{in})$. Moreover, since $d^2(\boldsymbol{x}, \boldsymbol{y_i}) = \boldsymbol{x}^2 - 2\boldsymbol{x}\cdot\boldsymbol{y_i} + \boldsymbol{y_i}^2$ and $\boldsymbol{x}^2$ is a constant known to Alice, it is sufficient for Alice to learn the minimal value of $-2\boldsymbol{x}\cdot\boldsymbol{y_i} + \boldsymbol{y_i}^2$ over all i-s. The latter task can be reduced to the private minimal scalar product problem by defining new vectors $\boldsymbol{x'} := (-2x_1, \ldots, -2x_n, 1)$ and $\boldsymbol{y'_i} := (y_{i1}, \ldots, y_{in}, \sum_{j=1}^{n} y_{ij}^2)$ for all i-s; then $\boldsymbol{x'}\cdot\boldsymbol{y'_i} = -2\boldsymbol{x}\cdot\boldsymbol{y_i} + \boldsymbol{y_i}^2$.

Since only generic inefficient protocols are known for the minimum finding [2], the hardest part of any reasonable PSS protocol is to find the minimum over the distances. To overcome this issue Du and Atallah proposed in [1] to use a trusted third party Ursula. Their proposed protocol, MINDASP, is depicted by Protocol 1.

INPUT: A query $\boldsymbol{x}$ and a database $\boldsymbol{y_1}, \ldots, \boldsymbol{y_m}$.
OUTPUT: The index b and the score $\boldsymbol{x}\cdot\boldsymbol{y_b}$.

1) Alice and Bob jointly generate two random numbers r^A, r^B.
2) For every row i:
   a) Alice and Bob jointly generate two random vectors $\boldsymbol{R_i^A}, \boldsymbol{R_i^B}$.
   b) Alice sends $\boldsymbol{w_i^A} \leftarrow \boldsymbol{x} + \boldsymbol{R_i^A}$ and $s_i^A \leftarrow \boldsymbol{x}\cdot\boldsymbol{R_i^B} + r^A$ to Ursula.
   c) Bob sends $\boldsymbol{w_i^B} \leftarrow \boldsymbol{y_i} + \boldsymbol{R_i^B}$ and $s_i^B \leftarrow \boldsymbol{R_i^A}\cdot\boldsymbol{w_i^B} + r^B$ to Ursula.
   d) Ursula computes and stores $v_i \leftarrow \boldsymbol{w_i^A}\cdot\boldsymbol{w_i^B} - s_i^A - s_i^B$.
3) Ursula finds an index b for which $v_b = \min_i v_i$. She sends b and v_b to Alice. Alice outputs $v_b + r^A + r^B$.

Protocol 1: MINDASP protocol.

During this protocol, for every i, Ursula learns the value $v_i = \boldsymbol{x}\cdot\boldsymbol{y_i} - r^A - r^B$ for r_A and r_B unknown to her. Therefore, provided that Ursula is honest, Alice learns the correct answer. Though Ursula gains new non-trivial information about the query, the authors of [1] argue that it is not sufficient to reveal either the match scores, the queries or the database. However, [1] does not give any further analysis. It also does not specify how to choose random values. In the case of a discrete search space, we should use $\mathbb{Z}_t$ with t being larger than any intermediate scalar product. More precisely, let $\Delta := \max\{\boldsymbol{x}\cdot\boldsymbol{y_i} - \boldsymbol{x}\cdot\boldsymbol{y_j}\}$; then we must have $t > 2\Delta$ since otherwise, Ursula cannot determine the smallest v_i. Thus, in the discrete case, Alice and Bob first agree on a safe $\mathbb{Z}_t$, perform all calculations in $\mathbb{Z}_t$ and choose all necessary random numbers uniformly from $\mathbb{Z}_t$. If the vectors are continuous, Alice and Bob have to embed real numbers into large enough $\mathbb{Z}_t$. More precisely, they should first fix the precision parameter p and use transformation $x \mapsto \lfloor p\cdot x\rceil$. Therefore, we must have $t > p^2\Delta$, since otherwise Ursula cannot determine the minimum.

Du-Atallah security model. In the security model of Du and Atallah, only ciphertext-only attacks with the next restrictions, are allowed: Ursula colludes with nobody and has no prior information about queries and a database. A protocol is considered secure in this model if Ursula cannot restore any queries, database vectors or corresponding match scores. It is still required that Bob must learn no new information and Alice must only learn the match index and the match score. It is easy to see that during the MINDASP protocol, Ursula can trivially learn distance differences. This does not directly imply that ciphertext-only attacks are dangerous to this protocol in the Du-Atallah security model, as the differences themselves do not reveal any vectors or match scores.

III. Analysis of SMSP Protocols

Efficient ciphertext-only attack against MINDASP. Recall that m is the number of database elements and n is the dimension of the vectors.

Lemma 2: Assume that the MINDASP protocol is executed over the ring $\mathbb{Z}_t$. If $m > n$, a semi-honest Ursula can reconstruct $\boldsymbol{x}$ with probability $\mathsf{P}_{(m-1)\times n}(t)$.

Proof: For any i, $s_i^B = (\boldsymbol{w_i^A} - \boldsymbol{x})\cdot\boldsymbol{w_i^B} + r^B$ and thus $\boldsymbol{w_i^B}\cdot\boldsymbol{x} = \boldsymbol{w_i^A}\cdot\boldsymbol{w_i^B} + r^B - s_i^B$. Hence, Ursula obtains $m-1$ equations $(\boldsymbol{w_i^B} - \boldsymbol{w_1^B})\cdot\boldsymbol{x} = \boldsymbol{w_i^A}\cdot\boldsymbol{w_i^B} - \boldsymbol{w_1^A}\cdot\boldsymbol{w_1^B} - (s_i^B - s_1^B)$. The claim now follows from Lemma 1. ■

As an example, if $t = 10$, $n = 6$ and $m = 7$, the success probability is roughly 0.22, while $m = 11$ raises it to 0.94. If $m > 2n$, the success rate will be almost 1.

Improved protocols. Next, we propose two alternative PMSP protocols that achieve the security goal, proposed by Du and Atallah: namely, Alice learns exactly the closest match and the match index, Bob gains no new information and Ursula learns only the distance differences $d_{ij} := d(\boldsymbol{x},\boldsymbol{y_i}) - d(\boldsymbol{x},\boldsymbol{y_j})$.

The MINTSP protocol, depicted by Protocol 2, is a simple tweak to MINDASP that achieves the required security level. (Note that scalar products in the MINTSP protocol represent distances up to an additive constant.)

Lemma 3: Assume the participants are semi-honest. During a single run of Protocol 2, Alice learns only the match score and match index, Bob learns nothing and Ursula learns only the scalar product differences $d_{ij} := \boldsymbol{x}\cdot\boldsymbol{y_i} - \boldsymbol{x}\cdot\boldsymbol{y_j}$.

INPUT: A query $\boldsymbol{x}$ and a database $\boldsymbol{y_1},\ldots,\boldsymbol{y_m}$.
OUTPUT: The index b and the score $\boldsymbol{x}\cdot\boldsymbol{y_b}$.

1) Alice and Bob jointly a random query key r.
2) For every row i:
   a) Alice and Bob randomly choose vectors $\boldsymbol{R_i^A}, \boldsymbol{R_i^B}$ and a scalar r_i.
   b) Alice sends $\boldsymbol{w_i^A} \leftarrow \boldsymbol{x} + \boldsymbol{R_i^A}$ and $s_i^A \leftarrow \boldsymbol{x}\cdot\boldsymbol{R_i^B} + r_i$ to Ursula.
   c) Bob sends $\boldsymbol{w_i^B} \leftarrow \boldsymbol{y_i} + \boldsymbol{R_i^B}$ and $s_i^B \leftarrow \boldsymbol{R_i^A}\cdot\boldsymbol{w_i^B} - r - r_i$ to Ursula.
   d) Ursula computes and stores $v_i \leftarrow \boldsymbol{w_i^A}\cdot\boldsymbol{w_i^B} - s_i^A - s_i^B$.
3) Ursula finds the minimising index b such that $v_b = \min_i v_i$. Then sends b and v_b to Alice, who finds the score $v_b - r$.

Protocol 2: The MINTSP protocol

Proof: Correctness is clear, since $\boldsymbol{w_i^A}\cdot\boldsymbol{w_i^B} - s_i^A - s_i^B = (\boldsymbol{x} + \boldsymbol{R_i^A})\cdot(\boldsymbol{y_i} + \boldsymbol{R_i^B}) - (\boldsymbol{x}\cdot\boldsymbol{R_i^B} + r_i) - (\boldsymbol{R_i^A}\cdot(\boldsymbol{y_i} + \boldsymbol{R_i^B}) - r - r_i) = \boldsymbol{x}\cdot\boldsymbol{y_i} + r$. It is straightforward to simulate the views of Alice and Bob, and therefore nothing will be leaked to them. To prove that nothing else, except the values d_{ij}, is leaked to Ursula, we show how to perfectly simulate views of Ursula by a simulator who knows all the differences d_{ij}. Consider the view of Ursula. In a valid protocol run, Ursula sees tuples $(\boldsymbol{w_i^A}, \boldsymbol{w_i^B}, s_i^A, s_i^B)$, such that $\boldsymbol{w_i^A}\cdot\boldsymbol{w_i^B} - s_i^A - s_i^B = \boldsymbol{x}\cdot\boldsymbol{y_i} + r$. Since $\boldsymbol{R_i^A}, \boldsymbol{R_i^B}, r_i$ are chosen uniformly, the triple $(\boldsymbol{w_i^A}, \boldsymbol{w_i^B}, s_i^A)$ has also a uniform distribution. Consequently, the simulator can choose $\bar{v}_1, \bar{\boldsymbol{w}}_i^{\boldsymbol{A}}, \bar{\boldsymbol{w}}_i^{\boldsymbol{B}}, \bar{s}_i^A$ at random and compute $\bar{s}_i^B = \bar{\boldsymbol{w}}_i^{\boldsymbol{A}}\cdot\bar{\boldsymbol{w}}_i^{\boldsymbol{B}} - \bar{s}_i^A - \bar{v}_1 - d_{i1}$. ■

To reduce the number of required random bits and the communication, Alice and Bob can use a pseudo-random generator for generating $r, \boldsymbol{R_i^A}, \boldsymbol{R_i^B}, r_i$. Then they have to agree only on random seed s and only send the tuple $(\boldsymbol{w_i^A}, \boldsymbol{w_i^B}, s_i^A, s_i^B)$ to Ursula. This reduces the communication between Alice and Bob to a few hundred bits.

Provided that vectors belong or can be safely embedded into $\mathbb{Z}_t^n$, where $\mathbb{Z}_t$ is the plaintext space of the used semantically secure homomorphic public-key cryptosystem $\Pi = (G, E, D)$, one can alternatively use the next communication-efficient MINHSP protocol, depicted by Protocol 3. Note that the same key K can be used in multiple protocols.

ALICE'S INPUT: A query $\boldsymbol{x} = (x_1,\ldots,x_n)$
BOB'S INPUT: A database $\boldsymbol{y_1},\ldots,\boldsymbol{y_m}$, where $\boldsymbol{y_i} = (y_{i1},\ldots,y_{in})$.
OUTPUT: The index b and the score $\boldsymbol{x}\cdot\boldsymbol{y_b}$.

1) Ursula generates a new private-public key pair for Π and sends the public key K to Alice and Bob.
2) Alice and Bob choose a random r from the plaintext space of Π.
3) For each $j \in [n]$, Alice sends $c_j \leftarrow E_K(x_j; t_j)$ to Bob, where t_j is a fresh random number.
4) For each row i, Bob sends $s_i \leftarrow \prod c_j^{y_{ij}} \cdot E_K(r; t)$, for a fresh random number t, to Ursula.
5) Ursula decrypts the result and sends the match index b and $v_b \leftarrow D_K(s_b)$ to Alice. Alice computes the score $v_b - r$.

Protocol 3: The MINHSP protocol

Alice's and Bob's privacy in the MINHSP protocol relies on the semantical security of Π. The security proof for Protocol 3 is very standard and therefore omitted. The MINHSP protocol requires n encryptions by Alice, and mn exponentiations by Bob. The number of exponentiations can be amortised. For example, if vectors $\boldsymbol{y_i}$ consist of binary data, Bob will not have to perform any exponentiations. Hence, the most demanding computational burden of m decryptions is placed on Ursula. The communication of the MINHSP protocol is only $m+n$ ciphertexts, whereas the MINTSP protocol requires sending at least $2m(n+1)$ scalars. Since the MINTSP protocol is computationally more efficient, we will have a trade-off between communicational and computational complexity.

IV. SECURITY AGAINST MORE ELABORATED ATTACKS

To avoid the costly minimum finding operation, the previous protocols make use of a trusted third party Ursula. However, Ursula learns some non-trivial information—namely, the distance differences—about the queries and the database. Next, we will analyse how Ursula can abuse this information and what could be the possible counter-measures.

Known-plaintext attacks. As Ursula obtains the list of distance differences $d_{ij} = \boldsymbol{x}\cdot\boldsymbol{y_i} - \boldsymbol{x}\cdot\boldsymbol{y_j}$, the knowledge of $\boldsymbol{x}\cdot\boldsymbol{y_i}$ for any single i reveals all scalar products. If Ursula knows $r > n$ database elements $\boldsymbol{y_{i_0}}, \ldots, \boldsymbol{y_{i_r}}$, she will obtain r equations $d_{i_k i_0} = \boldsymbol{x} \cdot (\boldsymbol{y_{i_k}} - \boldsymbol{y_{i_0}})$. Consequently, she can restore all query vectors $\boldsymbol{x}$, provided that $\boldsymbol{y_{i_1}} - \boldsymbol{y_{i_0}}, \ldots, \boldsymbol{y_{i_r}} - \boldsymbol{y_{i_0}}$ are linearly independent. The latter holds for a random database with probability, given by Lemma 1. By a similar argument, the knowledge of $r > n$ linearly independent query vectors to the same database reveals all differences $\boldsymbol{y_j} - \boldsymbol{y_1}$.

Some simple attacks can be avoided by randomly permuting the database elements before each query. This forces Ursula to determine the values $v_{j_0}, \ldots, v_{j_r}$ that are paired with $\boldsymbol{y_{i_0}}, \ldots, \boldsymbol{y_{i_r}}$. Since the number of valid pairings is $m(m-1)\cdots(m-r+1)$, where m is the number of database elements, such attacks become infeasible for most databases, at least if assuming that Ursula does not have any extra knowledge about the database.

Statistical attacks. However, the random permuting of the database rows does not provide absolute protection since Ursula still learns the multi-set $\{d(\boldsymbol{x}, \boldsymbol{y_i}) + r\}$. If n is large and the database vectors contain enough entropy, then one can approximate the empirical distance distribution with a data-independent distribution (this is caused by the *curse of high dimensions*). This statement is of course purely qualitative; quantitative estimates require the knowledge of the distribution of query and database vectors. For example, if the database and query vectors are uniformly and at random chosen from $\{0,1\}^n$, then the distribution of d_2^2 can be approximated with the Gaussian distribution $\mathcal{N}(\frac{n}{2}, \frac{n}{4})$. Similar results can be obtained if the vector components are assumed to be (weakly) independent; then the desired results follow from the theorems of weak convergence.

Assume now that the distance distribution $\mathcal{D}$ is known to Ursula and for a single query, all distances are independently sampled from $\mathcal{D}$. Consequently, Ursula can compute several point estimators like the mean value, the median or the variance and then use the obtained knowledge to reconstruct the match score. For example, in the case of the MINTSP and the MINHSP protocols, one can compute the expected value of v_i, $\mathsf{Exp}(v_i) := \frac{1}{m}\sum_{i=1}^{m} v_i = \mathsf{Exp}(d_i) + r$. Therefore, $d(\boldsymbol{x}, \boldsymbol{y_b}) = v_b - \mathsf{Exp}(v_i) + \mathsf{Exp}(d_i)$. Since by assumption, all d_i-s are sampled randomly from $\mathcal{D}$, then the standard central limit theorem assures that $\mathsf{Exp}(d_i)$ has the Gaussian distribution $\mathcal{N}(\mu, \sigma^2/m)$, where μ and σ are the mean and the variance of the distribution $\mathcal{D}$. Let $d_* = v_b - \mathsf{Exp}(v_i) + \mu$, then one can use standard results from statistics to show that the match score $d(\boldsymbol{x}, \boldsymbol{y_b})$ is in the interval $d_* \pm \sigma^2/\sqrt{m}$ with the probability 68%. For example, if $n = 100$, $\mu = 50$ and $m > \sigma^2 = 625$, Ursula can with probability 68% infer the match score with precision ± 1. Ursula can estimate the variance $\sigma^2 \approx \mathsf{Exp}((v_i - \mathsf{Exp}(v_i))^2) = 1/m \sum_i (v_i - \mathsf{Exp}(v_i))^2$ directly from the multi-set $\{d(\boldsymbol{x}, \boldsymbol{y_i}) + r\}$ and thus also compare the different match scores without knowing the distribution.

Detection of identical queries. In all presented protocols, Ursula can detect identical queries, since identical (or even just similar) queries have identical (resp., similar) distance multi-sets $\{d(\boldsymbol{x}, \boldsymbol{y_i})\}$. Thus, with a high probability, Ursula can decide whether two queries were equal or not. The rate of false positives—two different queries that have have identical or similar distance multi-sets—depends on database vectors, but is certainly small. E.g., this rate can be computed if all distances are sampled independently from $\mathcal{D}$. If the queries are similar, $d(\boldsymbol{x_1}, \boldsymbol{x_2}) < \tau$, then $|d(\boldsymbol{x_1}, \boldsymbol{y_i}) - d(\boldsymbol{x_2}, \boldsymbol{y_i})| < \tau$. If v_i^1 and v_i^2 are ordered lists of the values v_i obtained in the PSS protocols then for similar queries, $|v_i^1 - v_i^2 - v_1^1 + v_1^2| < \tau$.

Order preserving transformations. The previously described protocols make use of the simplest order preserving transformation $d_i \mapsto d_i + r$. This map corresponds to the use of one-time pad, and thus the knowledge of a single distance compromises all distances. If say a randomised affine transformation $d_i \mapsto sd_i + r + \epsilon$, with a random distortion ϵ, is used, we get a bit more security but the database can still be completely determined by a known distance pair.

In the MINTSP and MINHSP protocols, it is straightforward to replace the additive masking function with an affine transformation. First note that not all affine transformations preserve order: (a) $\Delta := \max\{\boldsymbol{x} \cdot \boldsymbol{y_i} - \boldsymbol{x} \cdot \boldsymbol{y_j}\} < t/(2s)$ must hold to avoid modular reduction; (b) $0 < \epsilon < s$ must hold to avoid local re-ordering. Thus, to implement affine masking, Alice must choose s randomly from valid interval $s \in [s_{min}, s_{max}]$, where s_{min} and s_{max} are chosen so that the resulting transformation would preserve order, and then use $s\boldsymbol{x}$ instead of $\boldsymbol{x}$, while Bob must use $r + \epsilon$ instead of r. However, as not all values of s are valid, Ursula knows that $d_i - d_j \in [(v_i - v_j)/s_{max}, (v_i - v_j)/s_{min}]$. Hence, the ratio s_{max}/s_{min} classifies the maximum uncertainty, though the knowledge of Δ sometimes allows to exclude part of the interval. Therefore, the knowledge of the mean value μ of $\mathcal{D}$ does not reveal the match score if $s_{max}/s_{min} > \mu/(\mu - d_{min})$, where $d_{min} = \min_i d_i$.

Divide-and-conquer technique. The underlying idea of divide-and-conquer approach (see Protocol 4) is to find minimum in several stages with a tournament scheme.

INPUT: A query $\boldsymbol{x}$ and a database $\boldsymbol{y_1},\dots,\boldsymbol{y_m}$.
OUTPUT: The index b and the score $\boldsymbol{x}\cdot\boldsymbol{y_b}$.
INITIAL STAGE

1) Bob randomly permutes the database and divides it into k random almost equally-sized blocks $\boldsymbol{y^{(1)}},\dots,\boldsymbol{y^{(k)}}$.
2) For every $j\in\{1,\dots,k\}$:
   a) Run a minimal scalar product protocol on inputs $\boldsymbol{x}$ and $\boldsymbol{y^{(j)}}$, so that Ursula obtains the match index b_j and $v_{b_j}\leftarrow\min_i \boldsymbol{x}\cdot\boldsymbol{y_i^{(j)}}$.
   b) Ursula generates two random values $b_j^B \mod k$ and $d_j^B \mod t$, and sends them to Bob. She sends $b_j^A\leftarrow b_j-b_j^B \mod k$ and $d_j^A\leftarrow v_{b_j}-d_j^B \mod t$ to Alice.

SECOND STAGE

1) Alice and Bob jointly generate a random permutation π.
2) Alice and Bob choose a random key r.
3) For every index j, Ursula learns $v_{\pi(j)}=d_{\pi(j)}^A+d_{\pi(j)}^B+r$.
4) Ursula sends minimising index $\tau:=\arg\min_j v_{\pi(j)}$ to Alice.
5) Alice uses private information retrieval to get d_τ^B and b_τ^B and compute d_b and b.

Protocol 4: Divide and conquer algorithm

As the minima are taken over smaller k-element blocks, the scheme is more resistant against statistical attacks and the empirical point estimates are less precise. It also makes harder to detect identical (similar) queries, since Ursula sees only a small random fraction shares $v_{i_1},\dots,v_{i_k}$ in every stage.

It is easy to generalise this protocol to an arbitrary number of stages. Again, instead of the additive masking, Alice and Bob can use more complex order-preserving transformations, for example affine transformation. Consider for example the extreme case, where there are only two values in each block. Then statistical attacks are not applicable and the distance difference is bounded to interval $[(v_1-v_2)/s_{max},(v_1-v_2)/s_{min}]$. Moreover, Ursula cannot link different stages—for each comparison there are two equiprobable ways to assign winner. Consequently, it is infeasible to look through all possible tournament trees. Therefore, the scheme is secure against statistical attacks.

This protocol is computationally more demanding since computationally-private information retrieval protocols are not cheap computationally or communicationally. (The currently most communication-efficient protocol [6] has communication complexity $\Theta(\log^2 n\cdot k+\log n\cdot\ell)$, where n is the size of the database, k is the security parameter and ℓ is the bit-length of transferred strings.) Still, it is several orders of magnitude more efficient than Yao's garbled circuit evaluation [2]. The latter requires roughly one oblivious transfer per input bit and a large garbled circuit description.

Attacks against some other PSS protocols from [1]. Random permutations are not always applicable and thus known-plaintext attacks can be dangerous. For example, Du and Atallah proposed two protocols (see sections 4.4.1 and 4.5.1 from [1]) for PSS for Euclidean distance, when the database itself is outsourced to potentially hostile party. In both cases, Alice outsources her database $\boldsymbol{y_1},\dots,\boldsymbol{y_m}$ in a garbled form to Bob. Omitting unnecessary details, Bob receives $\boldsymbol{z_i}=Q\boldsymbol{y_i'}$ where Q is a random invertible matrix known only to Alice. Each query vector is in form $\boldsymbol{q}=Q^{-1}\boldsymbol{x'}$ so that Bob can compute $v_i=\boldsymbol{q}\cdot\boldsymbol{z_i}=\boldsymbol{x'}\cdot\boldsymbol{y_i'}$.

In the SSO protocol $\boldsymbol{x'}$ and $\boldsymbol{y_i'}$ is chosen so that $v_i=(\boldsymbol{x}-\boldsymbol{y_i})^2+r$ for all i-s. Therefore, the leakage of $\boldsymbol{y_{i_0}},\dots,\boldsymbol{y_{i_r}}$ has a devastating effect. Bob can solve the linear equations $v_{i_k}-v_{i_0}+\boldsymbol{y_{i_0}}^2-\boldsymbol{y_{i_k}}^2=-2\boldsymbol{x}\cdot(\boldsymbol{y_{i_k}}-\boldsymbol{y_{i_0}})$ and determine query vectors. Then he can use revealed query vectors $\boldsymbol{x_1},\dots,\boldsymbol{x_s}$ to reconstruct the database.

In the SSCO protocol, Bob obtains $v_i=s(\boldsymbol{x}-\boldsymbol{y_j})^2+r$. Thus Bob can treat equality $v_{i_k}-v_{i_0}=-2s\boldsymbol{x}\cdot(\boldsymbol{y_{i_k}}-\boldsymbol{y_{i_0}})+s(\boldsymbol{y_{i_k}}^2-\boldsymbol{y_{i_0}}^2)$ as linear equation with unknowns $\boldsymbol{x_*}=s\boldsymbol{x}$ and s. Similarly, the revealed query vectors $\boldsymbol{x_1},\dots,\boldsymbol{x_k}$ enable to reconstruct the database. To conclude, both protocols are insecure in practice—only a small leakage of database vectors compromises the entire garbled database.

Acknowledgements: This work was partially supported by the Estonian Information Technology Foundation, the Finnish Defence Forces Research Institute of Technology and by the Finnish Academy of Sciences.

REFERENCES

[1] W. Du and M. J. Atallah, *Protocols for Secure Remote Database Access with Approximate Matching*, ser. Advances in Information Security. Kluwer Academic Publishers, Boston, 2001, vol. 2, p. 192, http://www.wkap.nl/prod/b/0-7923-7399-5.

[2] M. Naor, B. Pinkas, and R. Sumner, "Privacy Preserving Auctions and Mechanism Design," in *The 1st ACM Conference on Electronic Commerce*, Denver, Colorado, Nov. 1999.

[3] O. Goldreich, *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[4] A. C.-C. Yao, "Protocols for Secure Computations (Extended Abstract)," in *23rd Annual Symposium on Foundations of Computer Science*. Chicago, Illinois, USA: IEEE Computer Society Press, 3–5 Nov. 1982, pp. 160–164.

[5] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology — EUROCRYPT '99*, ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592. Prague, Czech Republic: Springer-Verlag, 2–6 May 1999, pp. 223–238.

[6] H. Lipmaa, "An Oblivious Transfer Protocol with Log-Squared Total Communication," International Association for Cryptologic Research, Tech. Rep. 2004/063, Feb. 25 2004. [Online]. Available: http://eprint.iacr.org/2004/063/