The Case of Adversarial Inputs for Secure Similarity Approximation Protocols

Evgenios M. Kornaropoulos
Brown University
evgenios@cs.brown.edu

Petros Efstathopoulos
Symantec Research Labs
petros_efstathopoulos@symantec.com

***Abstract*—Computing similarity between high-dimensional data is a fundamental problem in data mining and information retrieval, with numerous applications—such as e-discovery and patient similarity. To address the relevant performance and scalability challenges, approximation methods are employed. A common characteristic among all privacy-preserving approximation protocols based on *sketching* is that the sketching is performed *locally* and is based on *common randomness*.**

Inspired by the power of attacks on machine learning models, we introduce the study of *adversarial inputs for secure similarity approximations*. To formally capture the framework of this family of attacks we present a new threat model where a party is assumed to use the common randomness to perturb her input 1) offline, and 2) before the execution of any secure protocol, so as to steer the approximation result to a maliciously chosen output. We define *perturbation attacks* under this adversarial model and propose attacks for the techniques of minhash and cosine sketching. We demonstrate the simplicity and effectiveness of the attacks by measuring their success on synthetic and real data from the areas of e-discovery and patient similarity.

To mitigate such perturbation attacks we propose a *server-aided* architecture, where an additional party, the server, assists in the secure similarity approximation by handling the common randomness as private data. We revise and introduce the necessary secure protocols so as to apply minhash and cosine sketching techniques in the server-aided architecture. Our implementation demonstrates that this new design can mitigate offline perturbation attacks without sacrificing the efficiency and scalability of the reconstruction protocol.

I. Introduction

Quantifying similarity between high-dimensional data points is a cornerstone problem in the area of data mining. The history of the problem goes back to 1901 with the influential work of Jaccard [53] and has a wide range of applications in today's software systems and services especially in the areas of healthcare [81], law [46], finance [45], recommendation engines [47], personalization systems [30], social networks [89], databases [90], earth science [71], link prediction [62], forensics [77]. The wide adoption of this concept in diverse fields highlights the importance of similarity computation—the spectrum of application is so broad that instead of listing them we refer the reader to books [60], [64], [80] describing some of the applications of similarity computation. Some of the applications above consider similarity computation between high-dimensional data in the presence of *strict privacy requirements*. As motivating examples, we consider two such areas: *electronic discovery* and *healthcare*.

Electronic discovery (or *e-discovery*) typically focuses on the discovery and identification of information among privacy-sensitive electronic files as part of a lawsuit or formal investigation. It has been reported that the area of legal forensic discovery is a $9.9 billion market [31]. The community of technologists and legal experts in the area has formed the Electronic Discovery Reference Model (EDRM) [1], which is a framework that describes standards for the recovery and discovery of digital information during the legal process—e.g., criminal evidence discovery. According to the EDRM paradigm, during the phase of "Preparation" the discovery model filters documents so as to shortlist the ones most interesting/relevant among a voluminous collection of data. Section 1.2 of EDRM's directive [2] explicitly lists *"Similarity Hashing"* as a recommended action to shortlist privacy-sensitive documents. Thus, similarity approximation is a vital component of this multi-billion dollar business.

The area of *patient similarity* has attracted attention from both industry [57] as well as the medical community [18], [44], [81]. The emerging area of personalized medicine, where patient similarity plays a central role, aims at treatments tailored to individual characteristics of each patient. To achieve this goal, one needs to organize similar patients into subgroups that have the same response to a given treatment. This approach has dramatically changed the area of pharmacogenetics [92]. From a computational perspective, entire research teams (e.g., [57]) are focusing on the problem of *patient similarity*, applying advanced algorithmic techniques so as to discover groups of patients with similar health record profiles, while aiming to provide high secrecy for the sensitive healthcare records. Health information exchange protocols are already in place [3], allowing patient similarity computation across hospitals of different US states. These are two of the many important examples highlighting not only the central role of similarity detection in important business areas, but also the need for performing such detection using secure and robust methods, due to the sensitivity of the analyzed data.

Secure Sketching. As computing exact similarity metrics on very large datasets is prohibitively expensive, state-of-the-art methods seek to *approximate* the similarity function that needs to be computed, by working with a succinct representation of the data that is called a *sketch*. Sketching is the mainstream approach for efficiently approximating a plethora of functions

and applications [13], [16], [17], [23], [25], [33], [39], [49], [50], [61], [66], [68], [51], [79], [83]. The seminal work by Feigenbaum *et al.* [35] set the foundation for secure multiparty computation of approximation functions. Furthermore, the community has made several important steps towards private computation on genomic data in a time-efficient and scalable manner [6], [11], [24], [29], [73]. Wang *et al.* [87] demonstrate the potential of secure approximations, by running a privacy-preserving similarity query for a human genome on 1 million records distributed across the U.S., in a couple of minutes. All of the above works only consider an honest-but-curious adversary. In this work we extend the threat model and demonstrate how easy it is to craft adversarial inputs for sketching algorithms within this new model.

On Crafting Adversarial Inputs. The sketching protocol as presented by Feigenbaum *et al.* [35] has two phases: 1) the sketching function is applied locally by each party, and 2) the reconstruction function is performed via secure multiparty computation. Our offline attack is mounted on the first phase by a data owner who exploits the fact that i) the *randomness* of the sketching algorithm is known to all the participants, and ii) the sketching algorithm is performed *locally*. Such an adversary can steer any similarity approximation between the perturbed data and any other data point to an *incorrect output*, regardless of the secure computation protocols of the second phase. Our first attack uses simple probabilistic arguments, and is mounted on the *minhash sketching*, which is deployed to measure the Jaccard similarity between two sets. Our second attack formulates a high-dimensional constrained optimization problem, and is mounted on the *cosine sketching*, which is deployed to measure the cosine similarity between two vectors.

Threat Model. In this threat model the *only action* the attacker is allowed to take is to change the *input data* to the sketching algorithm. This is because any other alteration that concerns, a) the steps of the locally computed sketching algorithm, b) the sketch computed, and c) the secure protocols, can be easily detected by applying verifiable computation mechanisms [38], [78]. We focus our attention to the threat related to the input data of the sketching algorithm, and leave as an open problem the task of deploying efficiently verifiable computation for the remaining steps (i.e., potential attacks on items a, b, and c above). The *input to the sketching algorithm* is the very first step of the pipeline and is provided directly by the user, therefore the protocols have no means of verifying if it is a legitimate input or an adversarially perturbed input. This new threat model formally capture the attack surface of malicious perturbations of the input data with the end-goal of *violating the correctness of the similarity approximation*.

Motives for Mis-approximating Similarity. The motivation for such attacks can be clearly demonstrated by considering the previously discussed examples of e-discovery and patient similarity (among others). In the case of e-discovery the plaintiff party is interested in correctly approximating similarity between privacy-sensitive documents so as to discover important evidence. On the contrary, the defending party might prefer to masquerade evidence by causing mis-approximation. Applying a perturbation attack on document similarity approximation algorithms will conceal important documents from the shortlisted set that will be thoroughly investigated on a criminal evidence discovery case, such an outcome violates the directive of EDRM [2]. In the case of patient similarity, a perturbation attack will cause a pair of patients that have similar medical profiles to be assigned to different subgroups. For instance, all future patients that are assigned to a subgroup with perturbed data will receive personalized treatment that is not effective or, even worse, lethal. Such an attacker not only causes a disrupted service on the patient similarity component of a personalized treatment engine, but also introduces liability for the participating parties.

Proposed Mitigation. To mitigate perturbation attacks we follow the standard server-aided paradigm [55] and formulate a *server-aided secure approximation architecture* that requires the participation of three parties, as opposed to two of the previous schemes. A new honest-but-curious entity—the server—stores the common randomness which is treated as private information. A user, who no longer knows the common randomness, is therefore forced to run a protocol with the server to build an encrypted sketch, as opposed to the local computation of the previous model's first phase. During the sketching protocol the user doesn't learn any information about the common randomness and the server doesn't learn any information about the user's data. The sketch-generation takes place *only once for each data point*, and the sketch can be reused for any future pairwise approximation. Most importantly, under this new server-aided framework the users *do not have direct access* to the common random input and thus they can not mount an offline perturbation attack. In this paper, we devise and implement new secure protocols in order to generate minhash and cosine sketches in our proposed architecture. Given a pair of sketches[1] our implementation achieves throughput of 30-600 approximations per second for data points with hundreds of dimensions.

Our Contributions:

- We identify and formalize the notion of *perturbation attacks* against secure multiparty approximation. We propose two attacks, the first is on the *minhash sketching* that is used to approximate the similarity between two sets, and the second is on the *cosine sketching* that is used to approximate the similarity between two vectors. We apply our attacks on both real and synthetic data.
- Following the paradigm of server-aided design, we propose a *server-aided* approach that mitigates offline perturbation attacks. In our setup, a server has exclusive access to the common randomness, and is assisting the clients in the sketch computation. Thus, a user does not learn any information about the common random input. Additionally, the server doesn't learn any information about the user's data[2].

[1]The parameterization, and consequently the efficiency, of the sketching instantiation depends on the approximation guarantees.

[2]Other than the result of the approximation of unknown inputs.

II. Related Work

Aside from the secure sketching protocols mentioned earlier, there is a rich body of protocol that devise a combination of semi-homomorphic cryptosystems and garbled circuits to operate on encrypted data [7], [14], [56], [70], [88]. The work by Mironov *et al.* [67] introduces the model of *sketching in adversarial environments* which is different in certain ways from what we consider in our work. Specifically, the work in [67] studies a model where a *single* party adversarially chooses the input *for all other parties* while they approximate joint functions on the adversarially chosen input. In their model, the adversarial inputs are provided to the parties in an *on-line* manner and thus the users update the sketch incrementally without being able to store the original information, much like in one-pass streaming algorithms. In our work, each party uses her own data which is stored locally. Our model is different from the data stream model, and follows more closely the published work on privacy-preserving sketches discussed above. The work by Naor *et al.* [72] introduces a new *adversarial model for Bloom filters*. The threat model of [72] is somewhat similar to our model, in the sense that both adversaries exploit the used randomness so as to violate the correctness of the computation. In terms of differences, our adversary has direct access to the randomness used, whereas for the case of [72] the adversary has only oracle access via the responses of the Bloom filter. Furthermore in our work sketching is just the first phase of the computation and the second phase consists of a secure computation protocol; on the contrary the work of [72] does not involve any form of secure computation.

There is a significant body of research focusing on the attack vectors that lay in the intersection of machine learning and privacy-preserving mechanisms [8], [21], [26], [36], [37], [65]. The line of research closer to our proposed attack is the work on Deep Learning in adversarial settings. Some works [5], [22], [76], [84] show how an adversary can *craft her input* so as to maximize the prediction error of a deep neural network (DNN). Interestingly, in this work we show that adversarial inputs are very effective not only with learning and classification mechanisms, e.g. DNN, but also with simple randomized algorithms, e.g. sketching.

III. Preliminaries and Background

k-Independent Hashing. Space and time-efficient hash functions provide rigorous guarantees about the distribution of their values, such a family is the family of k-*independent* hash functions. Let U be the domain of the inputs to the hash function and let $x \in U$ be a specific input. Let $p > |U|$ be a prime and $a_0, a_1, \ldots, a_{k-1} \in \mathbb{Z}_p$ be uniformly chosen values over the prime field $\mathbb{Z}_p$. A commonly used construction of a k-independent family is based on polynomials of degree $k-1$:

$$h(x) = (\alpha_{k-1}x^{k-1} + \ldots + \alpha_1 x + \alpha_0) \mod p.$$

A. Secure Sketching

Exact similarity computation between two data points takes at least linear time with respect to the size of the data, since we need to parse the data item for *every comparison* regardless of the similarity function. A way to overcome this overhead is to settle with an *approximation* of similarity.

Definition III.1. *(Def. 10.1 in [69]) A randomized algorithm gives an* (ϵ, δ)-approximation *for the value ν if the output ν' of the algorithm satisfies,* $\Pr(|\nu' - \nu| \leq \epsilon\nu) \geq 1 - \delta$.

We are interested in *sketching techniques* that are well-studied and widely applied in the area of data-mining and information retrieval [16], [17], [23], [25], [33], [50], [61], [68], [79], [83]. A benefit of sketching is that the succinct summary of the data, i.e., *the sketch*, is built once and can be reused in future pairwise approximations. Thus the super-linear overhead occurs only during the construction of the sketch which significantly speeds up the total time performance over a series of similarity approximations. The notion of a sketching protocol is defined as:

Definition III.2. *(Def. 8 in [35]) A* sketching protocol *for a 2-argument function $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ is:*

- *A* sketching function*, $\mathcal{S} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ mapping one input and a random string to a* sketch *consisting of a (typically) short string.*
- *A (deterministic)* reconstruction function *$\mathcal{G} : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$, mapping a pair of sketches to an approximate output.*

On inputs $\alpha, \beta \in \{0,1\}^n$, the protocol proceeds as follows. First, Alice and Bob locally compute a sketch $\sigma_A = \mathcal{S}(\alpha, r_{cmn})$ and $\sigma_B = \mathcal{S}(\beta, r_{cmn})$ respectively, where r_{cmn} is a common random input. Then, the parties exchange sketches, and both output locally $\hat{f} = \mathcal{G}(\sigma_A, \sigma_B)$. We denote by $\hat{f}(\alpha, \beta)$ the randomized function defined as the output of the protocol on inputs α, β. A sketching protocol as above is said to (ϵ, δ)-approximate f, if $\hat{f}$ (ϵ, δ)-approximates f.

We note that in this work we are interested in normalized similarity therefore the output of the sketching protocol takes values in $[0, 1]$. Following the terminology of Goldreich for multiparty computation (Section 7.2 [41]) we capture the above process with the following *functionality*:

$$\mathcal{F}_{\mathsf{Approx}}\big((\alpha, r_{cmn}), (\beta, r_{cmn})\big) \to (\hat{f}(\alpha, \beta), \hat{f}(\alpha, \beta)), \qquad (1)$$

where the first (resp. second) pair is the input of client C_A (resp. client C_B) and the output to both parties is the (ϵ, δ)-approximation $\hat{f}(\alpha, \beta)$. We note here that if the clients execute the sketching computation with *different randomness* then the output of the reconstruction is meaningless[3], thus the randomness must be the same. We emphasize that α, β are user-provided inputs and their legitimacy relies on the honesty and intention of the user.

A metric space is a set X accompanied with a *distance function* $d : X \times X \to \mathbb{R}$, or simply *distance*, that measures the distance between points $x, y \in X$. We are interested in

[3]This is equivalent to using different hash functions for the approximation of Jaccard similarity, or using different random vectors for the approximation of the cosine similarity.

the approximation of distance functions from which we can derive the similarity. Given the similarity we can compute the corresponding distance, and vice versa, thus the two terms are used interchangeably in the rest of the work.

B. Similarity Approximation

Approximating Jaccard Similarity. The *Jaccard similarity coefficient* (or Jaccard index) measures the similarity between two sets. Formally, given sets S_1, S_2 the Jaccard similarity coefficient and the Jaccard distance d_{Jac} are defined as:

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}, d_{Jac}(S_1, S_2) = 1 - J(S_1, S_2).$$

Minwise hashing [16], [17], or *minhashing*, is a technique for approximating the Jaccard index that has been successfully applied to numerous problems (e.g., [17], [61], [68], [79], [85]). Even though the analysis of the approximation is based on random permutations [16], in practice we use minhash functions that are defined as $h_i^{min}(S) = min_{x \in S}(h_i(x))$, where h_i is a k-independent hash function. Using κ distinct minhash functions one can build a *minhash sketch*, also called minhash signature, $\sigma(S)$ for input set S. Given two minhash sketches we approximate the Jaccard distance $\hat{d}_{Jacc}$ as follows:

$$\begin{aligned} \hat{d}_{Jacc}(S_1, S_2) &= \frac{1}{\kappa} d_H(\sigma(S_1), \sigma(S_2)), \\ \sigma(S) &= (h_1^{min}(S), \ldots, h_\kappa^{min}(S)), \end{aligned} \tag{2}$$

where d_H denotes the hamming distance between the two input arguments. The common random input r_{cmn} from Definition III.2 is used to initialize the minhash functions.

Mitzenmacher *et al.* [68] introduced an approximation technique using odd sketches. An *odd sketch* of set S, denoted as $odd(S)$, consists of 1) a bit array T of size u and 2) a hash function $h_{odd} : U \rightarrow [0, u-1]$. In order to approximate the Jaccard similarity via odd sketches one uses the values of the minhash sketch $\sigma(S) = (x_1,\ldots,x_\kappa)$ as the input set for the odd sketch. Whenever an item $x_i = h_i^{min}(S)$, where $i \in [1, \kappa]$, is hashed to the odd sketch T using function h_{odd}, the bit in position $h_{odd}(x_i)$ of T is flipped. We approximate the Jaccard index as follows [68]:

$$\hat{J}_{odd}(S_1, S_2) = 1 + \frac{u}{4\kappa} \ln\left(1 - \frac{2|odd(\sigma(S_1)) \Delta odd(\sigma(S_2))|}{u}\right), \tag{3}$$

where $|odd(\sigma(S_1)) \Delta odd(\sigma(S_2))|$ denotes the number of 1s in the sketch resulted after the exclusive-or operation over the odd sketches, κ denotes the number of independent minhash values, and u denotes the size of the odd sketch. Jaccard distance is approximated using eq. (3), as $\hat{d}_{Jacc}(S_1, S_2) = 1 - \hat{J}_{odd}(S_1, S_2)$. The common random input r_{cmn} is used to initialize h_{odd} and $h_1^{min}, \ldots, h_\kappa^{min}$. Thus all the parties of the sketching protocol (see Definition III.2) generate the same hash functions.

Approximating Cosine Similarity. The work of Charikar [23] introduced the notion of *cosine sketching* commonly used [33] to estimate the similarity between two vectors. Formally, let $\vec{v_1}, \vec{v_2} \in \mathbb{R}^n$ the *cosine similarity* as

$$C(\vec{v_1}, \vec{v_2}) = \frac{\vec{v_1} \cdot \vec{v_2}}{\|\vec{v_1}\|_2 \|\vec{v_2}\|_2}, d_{cos}(\vec{v_1}, \vec{v_2}) = (1 - C(\vec{v_1}, \vec{v_2}))/2, \tag{4}$$

where $\|\cdot\|_2$ is the Euclidean norm of the vector. The resulting similarity $C(\vec{v_1}, \vec{v_2})$ ranges from -1 to 1 which is interpreted as completely opposite and as exactly the same, respectively. The cosine sketching technique is based on sign random projections. Let $\vec{v} \in \mathbb{R}^n$ be a unit vector[4], then the cosine sketch is a κ-dimensional bit vector $\sigma(\vec{v}) = (\sigma_1, \ldots, \sigma_\kappa)$. The components σ_i for $i \in [1, \kappa]$ and the symmetric cosine sketch distance [63] are defined as:

$$\sigma_i = \begin{cases} 1, & \text{if } \vec{w_i}^T \cdot \vec{v} \geq 0 \\ 0, & \text{if } \vec{w_i}^T \cdot \vec{v} < 0, \end{cases}, \hat{d}_{cos}(\vec{v_1}, \vec{v_2}) = \frac{d_H(\sigma(\vec{v_1}), \sigma(\vec{v_2}))}{\kappa}, \tag{5}$$

where $\vec{w_i} \in \mathbb{R}^n$ is sampled uniformly at random from the set of n-dimensional unit vectors. The common random input r_{cmn} is used to initialize the vectors $\vec{w_i}$, for $i \in [1, \kappa]$.

C. Semi-Homomorphic Cryptosystems

We use the described notation to highlight that messages are encrypted under different cryptosystems.

Paillier Cryptosystem. The Paillier cryptosystem [75] is semantically secure. The term $[m]$ denotes the encryption of message m under the key pair $K_P = (PK_P, SK_P)$; from the additive homomorphism we have that $[m_1] \cdot [m_2] = [m_1 + m_2]$.

Goldwasser-Micali Cryptosystem. The Goldwasser - Micali (GM) cryptosystem [42] is semantically secure. The term $|m|$ denotes the encryption of the bit m under the key pair $K_{GM} = (PK_{GM}, SK_{GM})$; from the homomorphism we have that $|m_1| \cdot |m_2| = |m_1 \oplus m_2|$, where $\oplus$ is the XOR operation.

Damgård-Geisler-Krøigaard Cryptosystem. The Damgård-Geisler-Krøigaard (DGK) cryptosystem [27], [28] is semantically secure. The DGK cryptosystem is considered to be much more efficient [12], [34], [59] than Paillier due to its small plaintext space. The term $\langle m \rangle$ denotes the encryption of message $m \in \mathbb{Z}_u$ under the key pair $K_{DGK} = (PK_{DGK}, SK_{DGK})$. DGK is additively homomorphic; moreover, it embeds reductions modulo u to its homomorphic operations, therefore $\langle m_1 \rangle \cdot \langle m_2 \rangle = \langle (m_1 + m_2) \mod u \rangle$.

IV. Threat Model

In this work we consider a new threat model where the adversary can maliciously perturb *only* her input to the sketching algorithm which is executed offline and locally, a behavior that is challenging to detect. We form this new threat model so as to formally capture and study an *algorithmic blindspot* that permits the proposed family of attacks. At a high level, this new adversary does not interfere with the computation of the sketching, the reconstruction, and the communication, i.e., adversary follows the prescribed protocols *after* the perturbation of the input data. Thus, our threat model is *not*

[4]In case the input vector is not unit we convert it by normalizing.

the honest-but-curious. Our adversary is *not* trying to learn the input of the other party, the goal is to make his/her own data look different from what it really is with respect to the approximation. Consider the following class of protocols that compute the functionality $\mathcal{F}_{\text{Approx}}$ from the previous Section.

Class of Protocols for $\mathcal{F}_{\text{Approx}}$

- **Step 1:** Generate and distribute the common random input r_{cmn} to all the parties.
- **Step 2:** Each party inputs her data and r_{cmn} so as to locally compute the sketching function $\mathcal{S}$.
- **Step 3:** Parties run an MPC protocol that outputs the result of the reconstruction function $\mathcal{G}$.

Attack Surface of $\mathcal{F}_{\text{Approx}}$. We assume that the adversary participates in the above protocol. We distinguish two possible *offline* attacks on this class of protocols, the attacker can: 1) deviate from the correct execution of the locally computed sketching, and/or 2) execute the sketching correctly, but corrupt its output—and therefore the input to the reconstruction function $\mathcal{G}$. Both attacks can be detected using verifiable computation [38], [78], i.e., provide proof of correctness for the computation and the output of $\mathcal{S}$. Addressing such mitigations is outside the scope of our work and is left as future work. We focus on the remaining attack surface: since cryptographic techniques exist to detect the above attacks, the last resort for the adversary is to perturb the input to the sketching function.

Perturbing the Input to $\mathcal{S}$. To capture the remaining attack surface, in the new threat model we extend the above class of protocols by allowing the adversary to locally execute a function right before Step 2. Specifically, the adversary executes a randomized function Perturb that takes as an input the data point α and the common random input r_{cmn} outputted by Step 1. Function Perturb runs locally, without any interaction, and outputs a value α^+ that will serve as the new input to the sketching function $\mathcal{S}$. We emphasize that after the execution of Perturb the adversary *behaves in a semi-honest fashion*, i.e., she honestly follows the sketching function and honestly executes the MPC protocol. Thus, in our threat model the only malicious activity of the adversary is the local execution of Perturb.

V. Perturbation Attack

In this Section we define *perturbation attacks* on the class of protocols defined in Section IV. A successful attack on secure sketching protocols for a *distance function* yields a perturbed input such that although the pair (original input, perturbed input) is close with respect to the corresponding distance function, the approximation instantiation appears vastly distant. Thus, if one compares the sketch of *any* data point that is close to the original input, to the sketch of the perturbed input the distance is heavily mis-approximated.

To the best of our knowledge this work is the first that *concretely demonstrates* the pitfalls of using common random input r_{cmn} for secure sketching protocols. In this work we focus on distance functions, analogous definitions can be formed for other functions. Note that Definition III.2 deals with two inputs α and β from distinct users, whereas the following definition deals with the input of a single user and it perturbed version, i.e., α and α^+.

Definition V.1. *Let $\mathcal{F}_{Approx}$ be the functionality described in Equation (1) for a sketching approach of a distance function d. Let Perturb$(\cdot)$ be the function that adversary $\mathcal{A}$ can apply according to the threat model of Section IV. Let $\alpha \in X$ be a point of the metric space (X, d) with distance function d. Let r_{cmn} be the common random input to the sketching function $\mathcal{S}$. Then we say that Perturb$(\cdot)$ is a* successful (ν, ν')-perturbation attack *for sketching function $\mathcal{S}$ if for any α and r_{cmn}, Perturb(α, r_{cmn}) outputs a point α^+ such that:*

1) *The true distance between α, α^+ is ν, $d(\alpha, \alpha^+) = \nu$,*
2) *The approximate distance between α, α^+ is ν' according to $(\mathcal{S}, \mathcal{G})$ with input r_{cmn}, $\hat{d}_{(\mathcal{S},\mathcal{G})}(\alpha, \alpha^+) = \nu'$,*
3) *The inequality $|\nu' - \nu| > \epsilon\nu$ holds.*

where ϵ is the parameter of the (ϵ, δ) approximation guarantees of $\hat{d}_{(\mathcal{S},\mathcal{G})}$.

One might suggest that it is trivial to mount a successful perturbation attack by generating random data and call it α^+. This naive approach would successfully increase the approximate distance ν' (condition 2), but it would *heavily distort* the original input and as a result the true distance ν would increase as well, i.e., doesn't satisfy the inequality of condition 3. Intuitively, for the case where $\nu' > (1+\epsilon)\nu$, condition 3 guarantees that the perturbed data "appears" more distant from the original than it truly is even when we consider the valid approximation error ϵ. For the case where $\nu' < (1-\epsilon)\nu$, condition 3 guarantees that the perturbed data "appears" more similar from the original than it truly is. In this work we focus on the case $\nu' > (1+\epsilon)\nu$, thus the adversary wants to hide the high similarity by minimally perturbing the input. Due to the triangle inequality, if α appears distant to α^+ w.r.t. the approximation, then any data point β that is close to α will also appear distant to α^+ w.r.t. the approximation. We leave as an open problem the case where the adversary perturbs the data so as to make highly dissimilar items look similar.

On using Commitment Schemes. It appears that the perturbation attack can be avoided if we deploy commitment schemes [40] for the data *before* they receive r_{cmn}. Thus, any perturbation will be caught due to the binding property of the construction. This mitigation indeed works only if *all data* from *all the users* is available during the initialization of the system and no sketch is created thereafter. In all practical scenarios, however, the system is more dynamic—users generate additional data and join/leave at arbitrary times. If a user creates new data *after* the commitment phase then this new input can be perturbed since the randomness value is already known and the new data is not committed. One might argue that we can redistribute new randomness to all the clients periodically. This will defend against these attacks but it implies that every party must re-compute the sketches from scratch whenever new randomness is issued, which would go against the very reason we used sketching techniques in

	$\hat{d}_{Jac} \geq 0.9$									$\hat{d}_{Jac} = 1$								
	$s = 500$			$s = 1,000$			$s = 10,000$			$s = 500$			$s = 1,000$			$s = 10,000$		
κ	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time
10	0.01	1.00	0.01	0.008	1.00	0.03	0.0008	1.00	0.37	0.01	0.98	0.10	0.009	0.99	0.22	0.0009	0.99	2.52
50	0.08	1.00	0.07	0.043	1.00	0.15	0.004	1.00	1.47	0.09	0.95	1.32	0.047	0.96	3.04	0.005	0.98	38.9
100	0.15	1.00	0.14	0.082	1.00	0.30	0.008	1.00	3.00	0.16	0.89	4.07	0.090	0.92	9.23	0.009	0.96	120.1
200	0.27	1.00	0.34	0.159	1.00	0.59	0.018	1.00	5.25	0.28	0.82	12.60	0.166	0.86	27.6	0.019	0.94	380.1

TABLE I

EVALUATION OF THE PERTURBATION ATTACK ON MINHASH SKETCHES OVER SYNTHETIC DATA. THE TERM κ DENOTES THE SIZE OF THE SKETCH, s IS THE SIZE OF THE SET UNDER ATTACK, f_{SUCCESS} IS THE FREQUENCY OF SUCCESS OF THE PROBABILISTIC ALGORITHM 1. THE DATA POINTS SHOWN ARE THE AVERAGE OVER 5,000 INSTANTIATIONS. TIME IS MEASURED IN SECONDS.

the first place—to avoid processing the high-dimensional data points multiple times.

On the Level of Distortion. Many of the occasions where secure similarity approximation protocols are applied typically employ multiple layers of forensic investigation mechanisms or sanity checks. A legal document comprised of random words or a genomic expression with random data are easy to spot. Therefore, in order to *minimize the likelihood of getting detected* (e.g., by another mechanism in place or during an audit), the attacker is incentivized to minimize the amount of changes to the input data—making the changes less incriminating and harder to detect. Extensive transformations (e.g., substituting large amounts of data with random noise) are likely recorded in system logs, and can be incriminating as they demonstrate malicious intent. Another illustration of this intent comes from the case of adversarial inputs on facial recognition - wearing a mask that covers the entire face clearly shows intent of avoiding facial recognition whereas an attacker that is wearing a set of "adversarially" decorated 3D-printed glasses [82] can fool such a system into matching the attacker to any maliciously-chosen individual.

Objectives of Perturbation Attacks. Note that d_{Jac} and d_{cos} as defined in Section III-B take values from the range $[0, 1]$. Ideally, a successful (ν, ν')-perturbation attack 1) maximizes the approximate distance $\hat{d}$ so as α and α^+ appear as distant as possible, e.g., $\hat{d}_{Jac}(\alpha, \alpha^+) \approx 1$, while 2) minimizes the true distance between α and α^+, e.g., $d_{Jac}(\alpha, \alpha^+) \approx 0$. We present two such attacks that utilize different tools, namely a randomized algorithm and constrained optimization formulation, and provide different guarantees. We slightly abuse notation and indicate by $\hat{d}_{Jac}$ and $\hat{d}_{cos}$ the approximate distance that is returned by a sketching protocol $(\mathcal{S}, \mathcal{G})$.

A. Attacking Minhash Sketches

Minhash sketches are used for approximating the Jaccard distance between sets. We propose a perturbation attack on minhash sketches guaranteed to perform the minimum number of changes to the original input set, thus minimizing $d(\alpha, \alpha^+)$. The perturbation that we apply is in the form of *adding new elements* to the set.

Intuition. The adversary takes as input a set S and the common random input r_{cmn}. The goal is to augment S with the smallest number of new elements in order to create S^+, such that $\hat{d}_{Jac}(S, S^+) = 1$. Recall that the approximate Jaccard distance between two sets is maximized when their κ-dimensional sketches $\sigma()$ differ in all dimensions, i.e., quantity $\hat{d}_{Jac}$ in equation (2) is equal to 1. Thus, the adversary is looking for at most[5] κ new elements such that every dimension of sketch $\sigma(S^+)$ is different from $\sigma(S)$. We denote by t' the number of samples drawn from the metric space. The following algorithm describes the attack, the corresponding proof can be found in the full version [58] of this work.

Algorithm 1: Attack Perturb on Minhash Sketches

Input: Original set S, common randomness r_{cmn}, sketch size κ, attempts t' to augment the original set
Output: S^+ s.t. $\hat{d}_{Jac}(S, S^+) = 1$, $d_{Jac}(S, S^+) = \frac{\kappa}{s+\kappa}$

```
Use r_cmn to sample κ hash functions (h_1, ..., h_κ);
σ(S) ← (min_{x∈S}(h_1(x)), ..., min_{x∈S}(h_κ(x)));
S^+ ← S;
for i = 1 to t' do
    Sample an element z_i ∉ S uniformly at random;
    for j = 1 to κ do
        if h_j(z_i) < min_{x∈S}(h_j(x)) then
            S^+ ← S^+ ∪ {z_i};
        end
    end
end
```

Theorem V.1. *Let S be the set of s values from the range $[0, m]$ that is given as an input to Algorithm 1. Let κ be the number of dimensions of the minhash sketch according to (2). Then a quasilinear number t' of samples are enough for Algorithm 1 to mount a successful $(1, \frac{\kappa}{s+\kappa})$-perturbation attack for minhash sketching with probability at least*

$$\Pr\left(\{\textit{Succesful Attack}\} | (t' \geq 2c(s+1)\ln^3(s))\right) \geq 1 - \frac{6\kappa c^{1/2}}{s^c},$$

for any constant $c > 0$ assuming the codomain of the hash function is $\Omega(s \log^4(s))$.

Attacking Synthetic Data. We demonstrate the frequency of success and the efficiency of the perturbation attack on synthetic data. We tested setups that range across all different variables of the problem: 1) dimension of the sketch $\kappa \in \{10, 50, 100, 200\}$, 2) size of the set under attack $s \in \{500, 1000, 10000\}$, 3) desired mis-approximation $\hat{d}_{Jac}() = 1$ or $\hat{d}_{Jac}() \geq 0.9$. Works such as [63] deploy a sketch of 64 bits to capture similarity of a collection of 8 billion webpages.

[5]There is a case where the same new element of S^+ can contribute to more than one locations of the sketch $\sigma(S^+)$.

Therefore, we think that sketches with size in the 10-200 range are indicative of what might be used in practice. The attack is implemented in C++ where the elements of the original set are randomly generated numbers from a universe of size $2 \cdot 10^5$. We used 4-wise independent hash functions, and run 5,000 instantiations for each of the above setups. As observed in Table I, when the desired approximation is $\hat{d}_{Jac}() \geq 0.9$, the attack succeeds in *all* instantiations, and its execution time is less than 1 sec in most of the parameterizations. In this scenario it is enough for the adversary to discover smaller minhashes for 90% of the κ entries of the minhash sketch. Thus, if there are some small minhash values in the original sketch, the adversary can ignore those and "break" the rest of the sketch, whereas in the case of $\hat{d}_{Jac} = 1$ the adversary is forced to continue searching so as to "break" all κ minhashes. Overall, the frequency of success is extremely high, but there are a few cases for which the probabilistic guarantees of Theorem V.1 are not met. One explanation is that the analysis was performed assuming that hash functions are truly random, whereas in the experiment we use 4-wise independent hashing. Table I clearly demonstrates that the probabilistic perturbation attack on minhash sketches succeeds in the vast majority of the instantiations and the total time ranges from less than a second to a couple of minutes even when dealing with sets that contain thousands of elements.

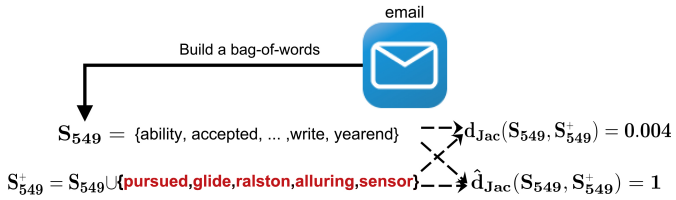

Fig. 1. Illustration of the perturbation attack on an e-discovery data. The adversary can add the 5 red-colored words in the original email with id 549 and the approximate distance of our instantiation will be 1 even though the exact distance is 0.004.

Attacking Real Data. To further verify the effectiveness of the attack we tested in real data using the bag of words dataset of Enron emails[6] which according to EDRM [1] has served for many years as *an industry-standard dataset* for e-discovery. We highlight that the findings of the attack on the synthetic data are expected to be similar to those on any real data, regardless of the context of the document, e.g. email, legal document. This is because the hash functions used are sampled uniformly at random and are independent of the input. In this real dataset every email is transformed into a multiset of words where the stop-words are removed. In this context Jaccard distance captures the similarity between any pair of emails. In our experiment we use the standard Rabin-Karp rolling hash function modulo $n = 105,943$. For simplicity we choose the size of the minhash sketch to be $\kappa = 5$ and the value of c to be 2 (see Theorem V.1). Without loss of generality, for the purposes of this evaluation we focus on email with id-549 (denoted as set S_{549}), with size $s = 1181$ words, 492 of which are unique.

The average time to mount 100 instantiations of the attack was 2.2 seconds. Specifically, 83 out of 100 instantiations mounted successfully a $(0.004, 1)$-perturbation attack and terminated *in less than 1 second*. The remaining 17 instantiations took between 3 to 22 seconds due to the fact that at least one of the minhash values of the original sketch was already too small (< 10). Figure 1 illustrates one of the successful attacks where by adding the 5 words {pursued, glide, ralston, alluring, sensor} in the current email, i.e. create S^{+}_{549}, the approximate distance becomes 1, while the real distance is 0.004. Thus, any future comparison between S^{+}_{549} and a similar email will result in mis-approximation.

B. Attacking Cosine Sketches

Cosine sketching is used for approximating the cosine distance between vectors. We propose a perturbation attack on cosine sketching guaranteed to output $\hat{d}_{cos}(\cdot) = 1$, while the exact distance between the perturbed and the original vectors depends on the solution of the formulated constrained non-convex optimization problem. Our perturbation is in the form of *adding a new vector* $\vec{x}$ to the original vector $\vec{v}$.

Algorithm 2: Attack Perturb on Cosine Sketch

Input: $\vec{v} \in \mathbb{R}^n, r_{cmn}, \kappa$
Output: $\nu, \vec{v^+} \in \mathbb{R}^n$ s.t. $\hat{d}_{cos}(\vec{v}, \vec{v^+}) = 1$, $d_{cos}(\vec{v}, \vec{v^+}) = \nu$

1 Use r_{cmn} to sample vectors $(\vec{w_1}, \ldots, \vec{w_\kappa})$ from the unit $(n-1)$-sphere

2 Solve the following optimization problem

$$\vec{x} = \underset{\vec{x} \in \mathbb{R}^n}{\text{argmax}} \quad \frac{\vec{v} \cdot (\vec{v} + \vec{x})}{||\vec{v}||_2 ||\vec{v} + \vec{x}||_2}$$

$$\text{subject to} \quad \text{sgn}(\vec{w_i}^T \vec{v}) \cdot (\vec{w_i}^T (\vec{v} + \vec{x})) \leq 0, \; i = 1, \ldots, \kappa.$$

$\nu = d_{cos}(\vec{v}, \vec{v} + \vec{x})$

3 **return** $\nu, \vec{v^+} = \vec{v} + \vec{x}$

Intuition. The adversary takes as input the original vector $\vec{v} \in \mathbb{R}^n$ and r_{cmn}. The goal is to add a new vector $\vec{x}$ to the original $\vec{v}$ in order to create $\vec{v^+}$ such that $\hat{d}_{cos}(\vec{v}, \vec{v^+}) = 1$. Recall that the approximate cosine distance between two vectors is maximized when their κ-dimensional sketches $\sigma()$ differ in all dimensions. Thus the addition of vector $\vec{x}$ to $\vec{v}$ must *change the sign* of the κ inner products with respect to Equation (5) and consequently flip the bits of the sketch $\sigma(\vec{v^+})$. Overall, the adversary wants to maximize the approximate cosine distance, handled by the constraints of the optimization problem, and minimize the exact cosine distance, handled by the objective function of the optimization.

In Algorithm 2 the function $\text{sgn}(x)$ has output -1 in case $x < 0$ and output $+1$ in case $x \geq 0$. The unit $(n-1)$-sphere is defined as the set of points $\{u \in \mathbb{R}^n : ||u|| = 1\}$. Notice that minimizing the exact cosine distance is equivalent to maximizing the cosine similarity as it is described in Equation (4), so our problem is formed as a maximization of the cosine similarity $C(\vec{v}, \vec{v} + \vec{x})$. Algorithm 2 requires to solve

[6] https://archive.ics.uci.edu/ml/datasets/Bag+of+Words

a non-convex, non-linear, high-dimensional constrained optimization problem. Furthermore the objective function presents discontinuity at point $\vec{x} = -\vec{v}$, see Figure 2. Since closed form solutions are generally challenging for this setup, we approximate the solution of the above problem using iterative algorithms from standard optimization toolboxes. Figure 2 visualizes the objective function for a toy example where $v \in \mathbb{R}^2$. Due to the lack of formal guarantees about the quality of the approximation, we present the effectiveness of the attack in a form of a remark.

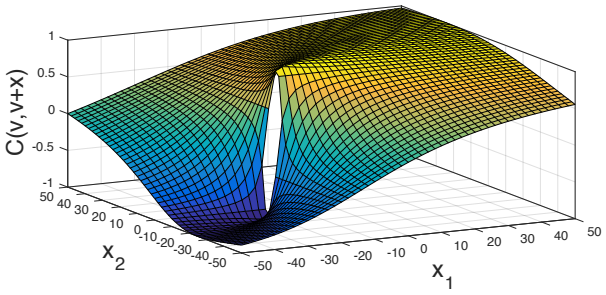

Fig. 2. An illustration of the objective function of the maximization problem of Algorithm 2 where $n = 2$ and $\vec{v} = (20, 10)$. The X-,Y-axis denote the x_1 and x_2 dimension of vector to be added, $\vec{x}$.

Remark V.1. *Let $\vec{v} \in \mathbb{R}^n$ be the vector that is given as an input to Algorithm 2. Let also $\vec{w}_i \in \mathbb{R}^n$ be a vector sampled from the unit $(n-1)$-sphere using r_{cmn} according to Algorithm 2, where $i = [1, \kappa]$. Then, Algorithm 2 is a successful $(\nu, 1)$-perturbation attack for cosine sketching, where ν is the achieved similarity of the perturbed input that is returned by the algorithm.*

Attacking Synthetic Data. We evaluated the performance of the attack on synthetic data using the interior point algorithm of MATLAB [4] where the input vector $\vec{v}$ is an n-dimensional vector where the value of each element is chosen uniformly at random from $[0, 10^5]$. We tested setups that range across the different variables of the problem: 1) the number of dimensions of the vector under attack $n \in \{500, 1000, 5000\}$, and 2) the size of the sketch under attack $\kappa \in \{10, 50, 100, 200\}$. To generate vectors $\vec{w}_i \in \mathbb{R}^n$ we sampled vectors from the $(n-1)$-sphere of unit radius centered at the origin. We run the above setups with 10 different common randomness inputs r_{cmn} and present the mean. As one may observe in Table II, the approximate distance is always $\hat{d}_{cos} = 1$ which implies that all the returned solutions were part of the feasible region of the optimization problem. The value of the exact cosine distance d_{cos} between the original and the perturbed data depends on the returned solution of the optimization problem. Note that different solution methods can potentially result in even lower d_{cos} values. Depending on the optimization toolbox and the number of dimensions the time performance may vary, in our case all the executions terminated within a couple of minutes.

Attacking Real Data. We demonstrate the perturbation attack of Algorithm 2 on a real dataset[7] of human gene-expression levels that can be found in the work of Notteramn *et al.* [74]. The authors perform a clustering analysis on the vectors of gene-expression levels so as to capture similarity patterns between healthy patients, patients with adenoma and patients with adenocarcinoma. It is rather common to perform similarity-based analysis on genomic data with the goal of understanding and diagnosing diseases at the molecular level. We highlight that the findings of the attack on the synthetic are expected to be similar to those on any real data. This is because the generative model of the input vector $\vec{v}$ does not affect the sign of the inner product with a random vector $\vec{w}$. We approximate the solution of the optimization problem using the interior point algorithm from MATLAB [4]. We use a cosine sketch of $\kappa = 100$ dimensions and we repeat the experiment for 10 different initializations of the vectors $(\vec{w_1}, \ldots, \vec{w_\kappa})$. The input vector is denoted as $\vec{v}$ and it has $n = 7,086$ dimensions each of which is a gene-expression measured with a DNA microarray. We report that all of the instantiations successfully satisfied the optimization constraints and thus resulted in $\hat{d}_{cos}(\vec{v}, \vec{v^+}) = 1$. The average ν value was 0.0033 with a maximum value of 0.0039. Therefore, on average we mounted a successful $(0.0033, 1)$-perturbation attack. One of the recorded instantiations is illustrated in Figure 3 where it shows that if the adversary perturbs $\vec{v}$ to form $\vec{v^+}$ then according to the cosine sketching initialization we have $\hat{d}_{cos}(\vec{v}, \vec{v^+}) = 1$, even though their exact cosine distance is $d_{cos}(\vec{v}, \vec{v^+}) = 0.0036$.

[7] http://genomics-pubs.princeton.edu/oncology/

TABLE II
EVALUATION OF THE PERTURBATION ATTACK ON COSINE SKETCHES OVER SYNTHETIC DATA. THE DATA POINTS SHOW THE AVERAGE VALUE OVER 10 INSTANTIATIONS.

	$n = 500$		$n = 1,000$		$n = 5,000$	
κ	d_{cos}	$\hat{d}_{cos}$	d_{cos}	$\hat{d}_{cos}$	d_{cos}	$\hat{d}_{cos}$
10	0.005	1	0.002	1	0.0005	1
50	0.02	1	0.01	1	0.002	1
100	0.05	1	0.02	1	0.005	1
200	0.11	1	0.06	1	0.01	1

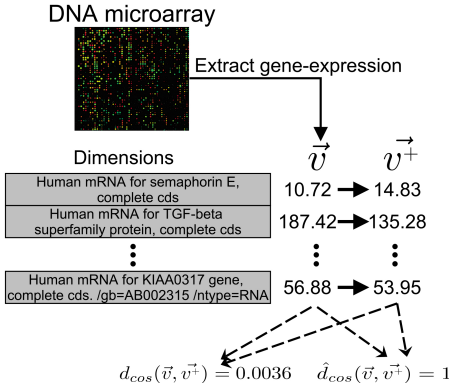

Fig. 3. Illustration of the perturbation attack on a gene-expression of an adenoma patient. The proposed attack on vector $\vec{v}$ outputs the perturbed $\vec{v^+}$ with approximate cosine distance 1 even though the exact distance is 0.0036.

TABLE III

AN OVERVIEW OF THE PROTOCOLS. FOR BREVITY WE ASSUME THAT THE PUBLIC KEYS OF THE SERVER $PK_P^{(S)}, PK_{GM}^{(S)}$ AND THE CLIENT $PK_P^{(C)}, PK_{GM}^{(C)}, PK_{DGK}^{(C)}$ ARE PUBLICLY AVAILABLE AND THUS NOT PASSED AS AN INPUT TO THE PROTOCOLS.

Protocol	Client (C) Input	Server (S) Input	Client (C) Output	Server (S) Output	Summary
PrvComparison*	a	b	-	$[t]$	[t=1] if $a < b$, [0] otherwise
EncComparison*	$SK_P^{(C)}, SK_{GM}^{(C)}, l$	$[a], [b], l$	t	-	t=1 if $a < b$, 0 otherwise
EncComparison2*	$SK_P^{(C)}, SK_{GM}^{(C)}, l$	$[a], [b], l$	-	$\|t$	$\|t = 1\|$ if $a < b$, $\|0\|$ otherwise
ChangePartyEnc*	$SK_{GM}^{(C)}$	$SK_{GM}^{(S)}, \|b\|$	$\|b\|$	-	Encrypts $\|b\|$ under $SK_{GM}^{(S)}$
kIndHashing	$SK_P^{(C)}, x, k, p$	$\{a_i\}_{i=0}^{k-1}, p$	-	$[h]$	$[(\sum_{i=0}^{k-1} a_i x^i) \mod p]$
EncHashing*	$SK_P^{(C)}, k, p$	$[x], \{a_i\}_{i=0}^{k-1}, p$	-	$[h]$	$[(\sum_{i=0}^{k-1} a_i x^i) \mod p]$
FindMin*	$SK_{GM}^{(C)}, SK_P^{(C)}, l$	$\{[y_i]\}_{i=1}^{n}, l$	-	$[min]$	$[\min_i y_i]$
UpdateOddSketch	$SK_{GM}^{(C)}, SK_P^{(C)}, SK_{DGK}^{(C)}, u, k$	$[x], \{a_i\}_{i=0}^{k-1}, u, (\|skt_0\|, \ldots, \|skt_{u-1}\|)$	-	$(\|skt'_0\|, \ldots, \|skt'_{u-1}\|)$	Update odd sketch with x
SketchingCosine	$\vec{v}, SK_P^{(C)}, SK_{GM}^{(C)}$	$\{\vec{w_i}\}_{i=1}^{\kappa}, SK_{GM}^{(S)}$	$(\|\sigma_1\|, \ldots, \|\sigma_\kappa\|)$	-	Encr. cosine signature
SketchingOdd	$S, k, u, SK_{GM}^{(C)}, SK_P^{(C)}, SK_{DGK}^{(C)}$	$\{h_i^{min}\}_{i=1}^{\kappa}, h_{odd}, p, SK_P^{(S)}, SK_{GM}^{(S)}$	$(\|\sigma_1\|, \ldots, \|\sigma_\kappa\|)$	-	Encr. odd-minhash signature

VI. SERVER-AIDED APPROXIMATION

In this Section we reframe the architecture of secure sketching protocols so that we can 1) still use the well-studied sketching techniques based on the common random input r_{cmn}, and 2) *eliminate* the possibility of an offline perturbation attack. In our proposed *server-aided* design we introduce a new semi-honest entity, i.e., the server S, that has exclusive access to the common random input r_{cmn} and assists in the sketching protocols. Compared to previous approaches, the main difference of our design is that a client *does not have direct access* to the common random input. The sketching function that was previously a local computation (as described in Section IV), is replaced by a two-party protocol denoted as Sketching between the server and the client. We capture the new *functionality* as follows:

Functionality $\mathcal{F}_{\text{S-approx}}$

- *Input:* Party C_A provides v_A, party C_B provides v_B, party S provides r_{cmn}.
- *Output:* All three parties receive $\hat{d}(v_A, v_B)$.

Notice that in case client C_A (similarly for client C_B) observes the values of σ_A, then it is possible for the C_A to infer r_{cmn}, which is an attack that defeats the purpose of the server-aided model. For example, in the case where r_{cmn} is used to sample k-independent hash functions then the set of values of σ_A consists of the evaluations of the above hash functions. An adversary that observes the output of the polynomial-based hash function can easily infer the coefficients of the hash function by solving a system of equations [48]. In our design, protocol Sketching outputs the *encrypted* sketch σ_A to C_A so as to avoid the above type of attacks.

The Real Model. Let Π be a three-party protocol computing the functionality $\mathcal{F}_{\text{S-approx}}$. For ease of exposition we consider the execution of Π in the presence of an adversary $\mathcal{A}$ as being coordinated by a nonuniform environment $\mathcal{Z} = \{\mathcal{Z}_\lambda\}$, much like [20], [52]. In the beginning $\mathcal{Z}$ gives input $(1^\lambda, v_A)$ to C_A, input $(1^\lambda, v_B)$ to C_B, input $(1^\lambda, r_{cmn})$ to S, and gives z and X to $\mathcal{A}$, where z denotes an auxiliary input and $X \in \{C_A, C_B, S\}$ is the corrupted party. At this point the parties interact with each honest party behaving as instructed by Π. At the end of the protocol, adversary $\mathcal{A}$ gives to $\mathcal{Z}$ an output which is an arbitrary function of $\mathcal{A}$'s view. Additionally, $\mathcal{Z}$ gets the output of the honest parties. Finally, environment $\mathcal{Z}$ outputs a bit. We denote as $\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda)$ the random variable that represents the value of this bit.

The Ideal Model. In this model there is a trusted party that computes $\mathcal{F}_{\text{S-approx}}$ on behalf of the parties. Similar to the real model, environment $\mathcal{Z}$ gives inputs $(1^\lambda, v_A)$ and $(1^\lambda, v_B)$ to parties C_A and C_B, respectively. It gives input $(1^\lambda, r_{cmn})$ to S, and also gives z and X to $\mathcal{A}'$ where $X \in \{C_A, C_B, S\}$ indicates the corrupted party. All the parties send their input to the trusted party. The trusted party computes $\mathcal{F}_{\text{S-approx}}$ and sends $\hat{d}(v_A, v_B)$ to all the parties. In the next step $\mathcal{A}'$ outputs to $\mathcal{Z}$ an arbitrary function of the view of $\mathcal{A}'$. The honest parties also give their output to $\mathcal{Z}$. As a final step $\mathcal{Z}$ outputs a bit. We denote as $\mathsf{IDEAL}_{\Pi,\mathcal{A}',\mathcal{Z}}(\lambda)$ the random variable that represents the value of this bit.

Definition VI.1. *Let* Π *be a three-party protocol for computing* $\mathcal{F}_{\text{S-approx}}$ *functionality. We say that* Π securely computes $\mathcal{F}_{\text{S-approx}}$ *in the presence of semi-honest adversaries corrupting one party if for any* PPT *semi-honest adversary* $\mathcal{A}$ *there exists a* PPT *semi-honest adversary* $\mathcal{A}'$ *such that, for every polynomial size circuit family* $\mathcal{Z} = \{\mathcal{Z}_\lambda\}$ *corrupting at most one party, the following is negligible:*

$$|\Pr[\mathit{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda) = 1] - \Pr[\mathit{IDEAL}_{\Pi,\mathcal{A}',\mathcal{Z}}(\lambda) = 1]|.$$

Notice that if the adversary were to corrupt both a client and the server then she would have access to the common random input, and thus become capable of mounting a perturbation attack. We note here that the server-aided approach has been successfully deployed [11], [54], [55] in various other problems. The proposed perturbation attacks of the previous Section are based on the fact that all clients have offline and direct access to the common random input r_{cmn}. Under our server-aided design an adversary can only attempt an *online* attack, hoping to infer the r_{cmn} from the value of $\hat{d}(\cdot)$, by performing a series of Sketching and Reconstruct executions. Using rate-limiting techniques (e.g., [55]) one can mitigate such an online attack. This scenario, however, is beyond the scope of this paper.

Composition of Building Blocks. We define separate building blocks that can be combined and the proof of security

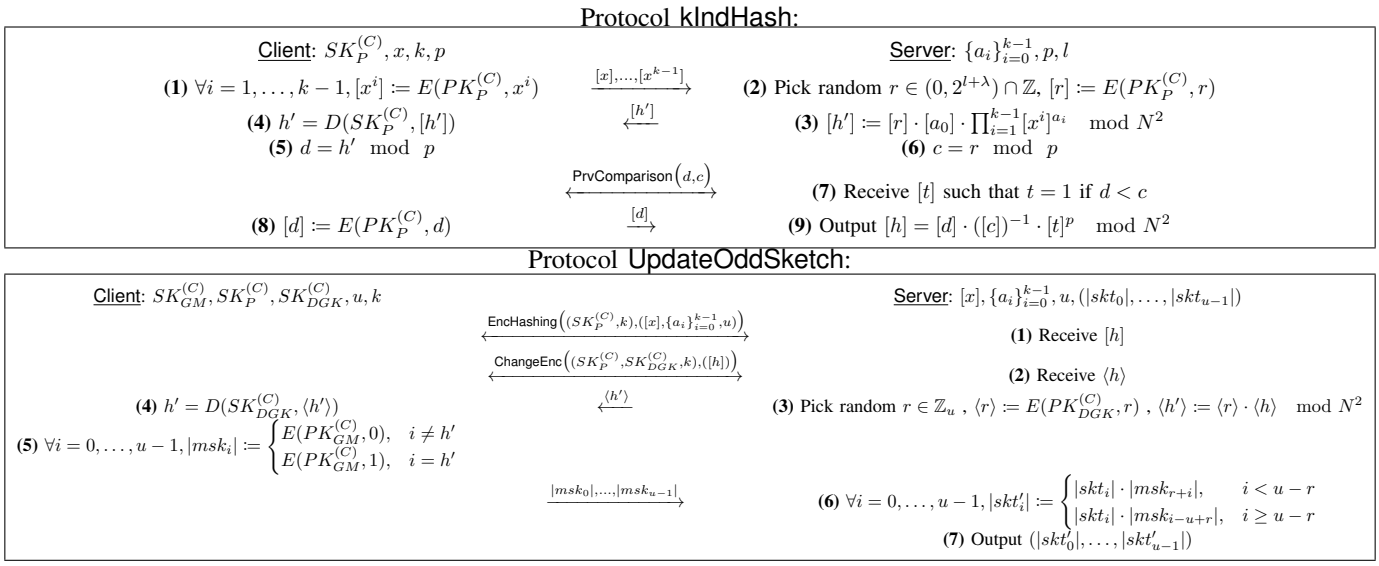

Fig. 4. Two-party protocols between a client and the server that are used as building blocks for sketching.

for the overall construction can be derived using modular composition [19]. The model is called *hybrid model with ideal access to functions* $f_1, \ldots, f_m$ or simply $(f_1, \ldots, f_m)$-hybrid model . In the real life experiment we assume the existence of an incorruptible trusted party T for evaluating $f_1, \ldots, f_m$; all parties hand their input to T and they receive the corresponding output. As a next step, the ideal evaluation of f at each step is replaced with the invocation of a protocol—we refer the reader to [19] for a detailed exposition. In case the function returns an encrypted output, a party passes a public key as an input and we assume that the necessary encryption algorithm is hardwired to the corresponding function. Table III summarizes all the two-party protocols, which in our case are executed between the server and the client. Using the above building blocks we construct a secure two-party analogue for minhashing (via odd sketches) and cosine skething. Due to lack of space, protocols that are marked with $*$ in Table III (simple modification of already proposed protocols [7], [14], [86] or new protocols) can be found in the full version [58] of this work. We note that we follow the protocol and encryption notation established by the work of Bost *et al.* [14].

A. Building Blocks

k-Independent Hashing over Encrypted Data. The functionality of $\mathcal{F}_{\text{kIndHash}}$ is as follows. The input of the server is the bit length l and the set of parameters of a k-independent hash function—i.e., the coefficients $\{a_i\}_{i=0}^{k-1}$, the prime p of a $(k-1)$ degree polynomial on $\mathbb{Z}_p$. The client has the input x which is used to evaluate the polynomial on $\mathbb{Z}_p$. The degree of the polynomial as well as the modulo p are considered to be known to both parties. At the end of the protocol the server receives the evaluation of the polynomial $a_0 + a_1 x + \ldots + a_{k-1} x^{k-1} \mod p$ that is encrypted with the client's public key. We do not use a private polynomial evaluation technique due to the fact that we require the output to be *encrypted*. The server should not learn any information about the client's input x and the client should not learn any information about the coefficients $\{a_i\}_{i=0}^{k-1}$ of the polynomial. A more thorough exposition of the protocol is provided in the full version [58] of this work.

Lemma VI.1. *Protocol kIndHash correctly and securely computes $\mathcal{F}_{kIndHash}$ in the ($\mathcal{F}_{PrvComp}$)-hybrid model.*

Update Encrypted Odd Sketch. The functionality of $\mathcal{F}_{\text{UpdateOddSketch}}$ is as follows. The input of the server consists of i) the bits of an odd sketch $(skt_0, \ldots, skt_{u-1})$ encrypted with the client's public key, ii) the parameters of the $(k-1)$-degree polynomial that is used as the hash function h_{odd}, and iii) the input x of the polynomial encrypted with client's public key. The input of the client is the set of secret keys. At the end of the protocol the server receives an updated odd sketch where the bit in location $h_{odd}(x)$ of the sketch is flipped, while the client receives no output. The server and the client should not learn which bit of the odd sketch is flipped or the input x of the polynomial. One new idea of our design is the use of DGK with message space $\mathbb{Z}_u$, where u is also the length of the sketch, so as to securely translate the hash value into a bit-mask, and eventually apply the mask to the original sketch. A thorough overview of the protocol is provided in the full version [58] of this work.

Lemma VI.2. *Protocol UpdateOddSketch correctly and securely computes $\mathcal{F}_{UpdateOddSketch}$ in the ($\mathcal{F}_{EncHashing}$, $\mathcal{F}_{ChangeEnc}$)-hybrid model.*

B. Protocols for the Server-Aided Model

Approximating Jaccard Distance via Odd Sketches. We employ the protocols of the previous subsection as building blocks to *securely* approximate Jaccard distance using the approach by Mitzenmacher *et al.* [68]. As denoted in Figure 5, the input of the server consists of the set of κ minhash functions $\{h_i^{min}\}_{i=1}^{\kappa}$, the hash function for the creation of

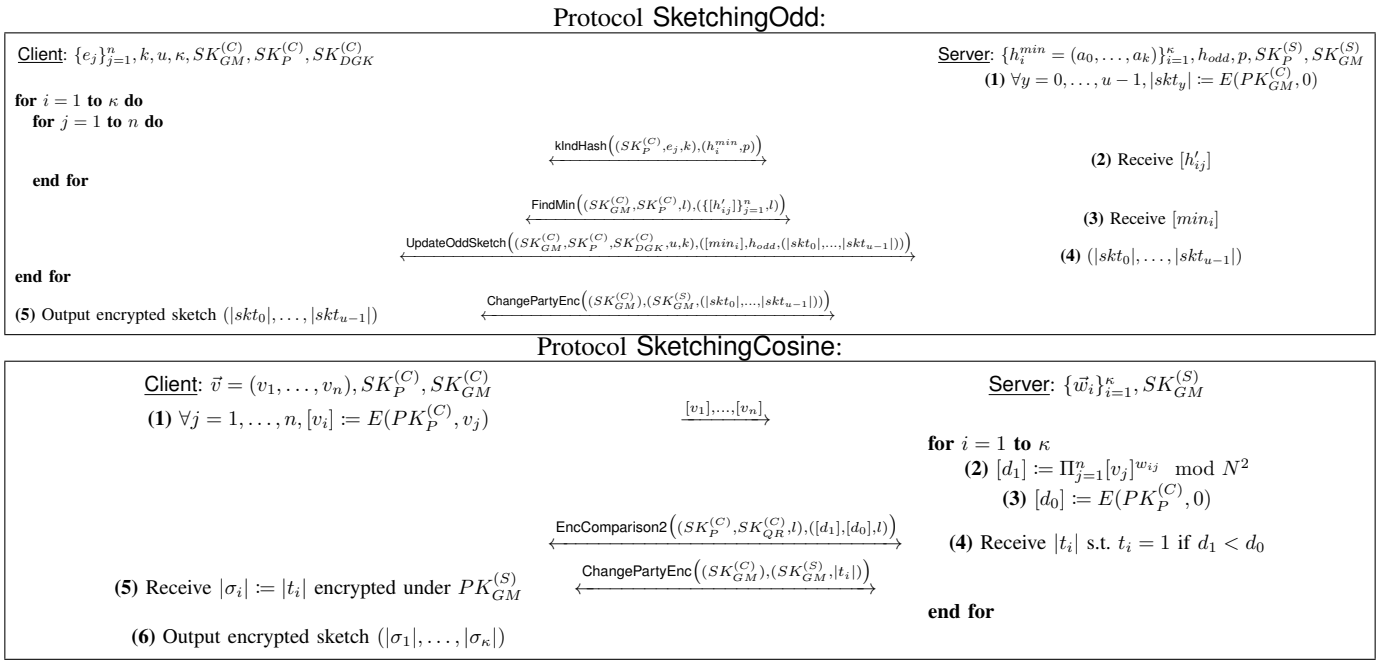

Fig. 5. The sketching protocols between the server and the client for the server-aided model.

the odd sketch h_{odd}, as well as the corresponding secret keys. Recall that $\{h_i^{min}\}_{i=1}^{\kappa}$ and h_{odd} are generated using the common randomness r_{cmn} that can only be accessed by the server. The input of the client consists of her data, denoted as the elements $\{e_j\}_{j=1}^n$, as well as the publicly known moduli p, u, and the secret keys. At the end of the protocol the client receives the odd sketch encrypted with the server's public key.

Lemma VI.3. *Protocol SketchingOdd correctly and securely computes $\mathcal{F}_{SketchingOdd}$ in the ($\mathcal{F}_{kIndHashing}$, $\mathcal{F}_{FindMin}$, $\mathcal{F}_{UpdateOddSketch}$, $\mathcal{F}_{ChangePartyEnc}$)-hybrid model.*

Approximating Cosine Distance via Cosine Sketching. We approximate cosine distance as follows. The input of the server consists of the vectors $\vec{w_i}$, that are sampled uniformly at random from the $(n-1)$-sphere. The input of the client consists of her data which is represented by the vector $\vec{v}$. Note that vectors $\vec{w_i}$ are generated using the common randomness r_{cmn} that can only be accessed by the server. At the end of the protocol the client receives the cosine sketch encrypted with the server's public key.

Lemma VI.4. *Protocol SketchingCosine correctly and securely computes $\mathcal{F}_{SketchingCosine}$ in the ($\mathcal{F}_{EncComparison2}$, $\mathcal{F}_{ChangePartyEnc}$)-hybrid model.*

Reconstruct Protocol. The power of the sketching techniques that we chose for approximating Jaccard distance and cosine distance lies in the fact that their reconstruction function is *simple* and *efficient*. Both techniques follow the same reconstruction process which performs an exclusive-or operation between the two sketches, and then counts the number of 1 values (see Equations (3) and (5)). Taking advantage of the homomorphic properties of the GM cryptosystem we build an efficient Reconstruct protocols. See Figure 6.

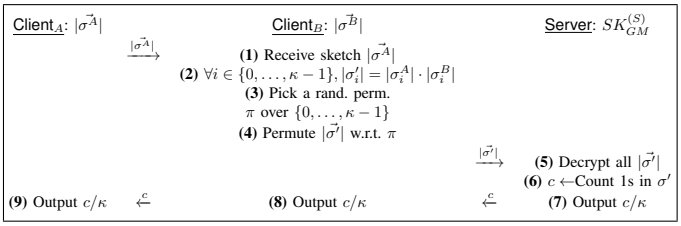

Fig. 6. The reconstruction of SketchingCosine between the server and the clients. The reconstruction for SketchingOdd is the same for steps **(1)**-**(6)**; steps **(7)**, **(8)** follow the reconstruction of Equation (3).

Lemma VI.5. *Protocol Reconstruct is correct and secure in the semi-honest model.*

On the Choice of Building Blocks. Since our protocols follow a modular design, one can substitute the proposed building blocks with protocols that follow other MPC techniques so as to further optimize the performance of our constructions. The work presented in this paper is meant to present to principles of this modular design and is not representative of a highly-optimized implementation. According to the work of Bost *et al.* [14], comparison protocols that utilize specialized homomorphic cryptosystems [34], [86] are more efficient when the input is encrypted. Thus, our implementation invokes variations of the above protocols, namely EncComparison and EncComparison2. For the comparison protocol on unencrypted inputs, Bost *et al.* [14] denote that a garbled circuit approach [9] results in a more efficient implementation. In our implementation we followed the work of Veugen [86], and therefore one can further speedup our

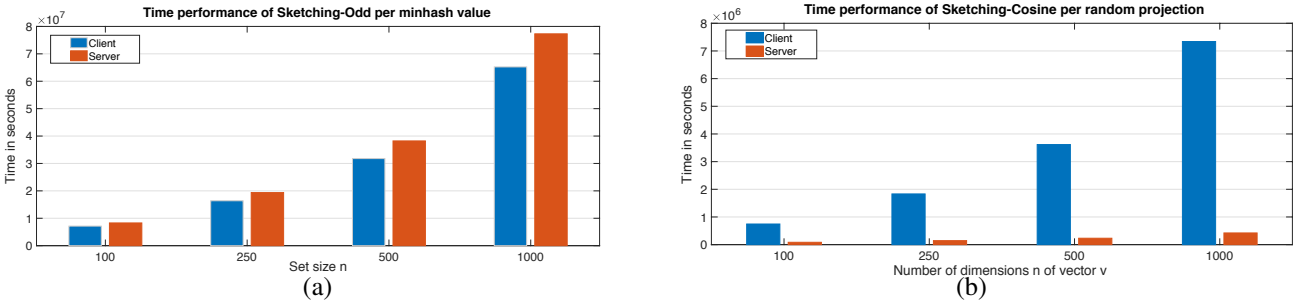

Fig. 7. Subfigure (a): Time performance for varied set size of the SketchingOdd protocol. Time averaged for a single minhash over five runs. Subfigure (b): Time performance for varied number of vector dimensions of the SketchingCosine protocol. Time averaged for a single random projection over five runs.

implementation by invoking a garbled circuit design instead. We note that well-known protocols that are purely based on garbled circuits for functionality such as FindMin can not be deployed because the input of the FindMin is a set of *encrypted* inputs (see Table III). A similar argument holds for the output of kIndHashing which is encrypted. Thus, to the best of our knowledge, the most promising speedup opportunity would be opting for garbled circuit designs for the simplest building blocks, such as comparison.

VII. Scalability Evaluation

Implementation Setup. We implemented the proposed protocols in C++ using existing libraries as well as newly implemented building blocks. For serializing the communication between the server and client we use Protocol Buffers [43]. All the arithmetic operations are performed with the gmp multiple precision library [32]. We use the Advanced Crypto Software Collection [10] implementation of the Paillier cryptosystem, and an open-source implementation of the GM cryptosystem. We implemented the DGK cryptosystem in C++ following the design principles of [10] and the directions of the original work [27], [28].

For the minhashing via odd sketching protocols we choose the security parameter $\lambda = 100$. Given the scale of our experiments the k-independent hashing setup is the following: we choose $k = 4$ and a prime p that is at least an order of magnitude larger than the size of the set—i.e., $p \geq 10n$. As explained in the description of protocol UpdateOddSketch, prime u of the DGK cryptosystem is set to have the same value as the length of the odd sketch. As it is also noted in [14] the parameterization of Paillier has to be such that the homomorphic operations do not overflow the message space. To accomplish this instantiation we analyze the two phases of the protocol. The first phase is the kIndHashing computation; let l' be the maximum bit-length of the inputs x. In step **(1)** of protocol kIndHashing involves $(k-1)$ exponentiations among which the plaintext x^{k-1} can have the maximum length of $l'_{\max} = (k-1)l'$ bits. Step **(3)** of protocol kIndHashing involves $(k-1)$ multiplications and $(k+1)$ additions of numbers that are at most $l'_{\max}$ bits long. Therefore it is sufficient for N to be such that $\log N \geq (k^2-k-2)(l'/2)+2+\lambda$. After the execution of kIndHashing the numbers involved in protocols PrvComparison and EncComparison are $\log p$ bits long, since they are hash values. Thus protocols PrvComparison and EncComparison operate on integers that are at most $l = \log p$ bits long. Consequently, it is sufficient for N to be such that $\log N > \log p + \lambda + 1$. We satisfy the above inequalities by choosing $\log N \geq 1024$.

Regarding the protocols for cosine sketching, we also choose a security parameter $\lambda = 100$. Recall that vectors $\vec{w}_i = (w_{i1}, \ldots, w_{in})$ are sampled uniformly at random from the $(n-1)$-sphere, so each value w_{ij} is a real number. We can transform the above real numbers to integers by multiplying with a constant K and rounding, allowing us to interpret w_{ij} as part of Paillier's message space. The purpose of the random projection is to compute the sign of the inner product thus one can choose a relatively small K. In our implementation we choose $K = 1000$. Similarly to the previous instantiation, the parameterization of Paillier should not overflow by the homomorphic operations of the encrypted inner product that is performed in step **(2)** of protocol Sketching-Cosine. Let l be the maximum length in bits of the entries in $\vec{v}$. Then step **(2)** of protocol Sketching-Cosine involves the multiplication of a $\log K$ bit long integer with an l bit long integer. Thus, it is sufficient for N to be such that $\log N \geq \log K + l + n$. Finally, in our implementation, both GM and DGK have moduli that are at least 1024 bits long. The implementation of the protocols and the serialization of the server is around 1400 lines, while the client is around 1100 lines.

Scalability. We evaluate the scalability of the server-aided design based on the described implementation setup. In Figure 7 we present the recorded computation time for the sketching protocols on a commercial laptop with 2.6 GHz Intel Core i5 CPU and 8GB DDR3 RAM.

The client and server have similar time performance for the SketchingOdd protocol. This is mostly because both parties are subject to a slowdown by a similar number of encrypt/decrypt operations. The time performance presented in Figure 7-(a) is for a a single minhash value (i.e., $\kappa = 1$), and an odd sketch of 151 bits (i.e., $u = 151$). Note that the computational overhead scales linearly with κ: for $\kappa > 1$ we have the same computational overhead as the one depicted in Figure 7-(a), only κ times larger. Notice, however, that the computation for each of the κ dimensions of the sketch is *in-*

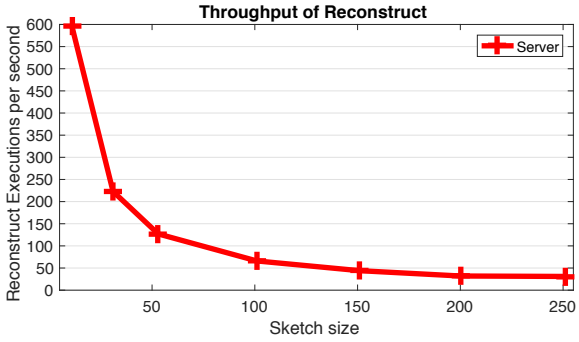

Fig. 8. Throughput of the server Reconstruct protocol for varied sketch sizes. Average values over 5 runs.

dependent of each other, thus the overall task is parallelizable. On the other, hand the computational overhead of the client in protocol SketchingCosine is significantly higher than the one of the server. This is mainly caused by the encryption of each dimension of $\vec{v}$, which translates to a large number of exponentiations taking place in step **(1)** of the protocol. Furthermore, the performance of the server (time) is measured when we have a single random projection, i.e., $\kappa = 1$, thus steps **(2)**-**(4)** of SketchingCosine are repeated only once. Similar to the case of SketchingOdd, for $\kappa > 1$ the overall task is highly parallelizable into κ tasks. The communication overhead of the sketching protocols for various values of n is depicted in Table IV.

TABLE IV
COMMUNICATION OVERHEAD OF SKETCHING. AVERAGE OVER 5 RUNS.

Protocol	n			
	100	250	500	1000
Sketch-Odd	1112 KB	2930 KB	6218 KB	13201 KB
Sketch-ShimHash	69 KB	165 KB	324 KB	644 KB

In our design we prioritize the *speedup the reconstruction protocol*, since it is the protocol that is executed multiple times throughout the lifetime of the system—once for every pairwise approximation. On the contrary, the sketching protocol is invoked only once for every high-dimensional data point, so as to create the sketch. Thus, using odd sketches (rather than regular minhashing) introduced, indeed, some overhead in the overall sketching protocol but resulted in a fast and more scalable reconstruction protocol. Generally, the reconstruction protocol from the server's perspective is the same, regardless of whether we are approximating Jaccard or cosine similarity, since the only task performed by the server is to decrypt κ ciphertexts encrypted under GM. The end result is a rather scalable performance illustrated in Figure 8.

VIII. DISCUSSION

Moving forward, it would be interesting to study even stealthier attacks where the perturbation is tailored to the specific context of the application. Under *context-aware* attacks the adversary perturbs the data in a way that is relevant to the semantics of the data. For instance, if the first dimension of the vector under attack represents "age" then it is preferable not to change the value to negative. In our work we focused on perturbation mechanisms that *add* information. In a different setup it might be preferable to *remove* existing information or *transform* small pieces of data to something equivalent, e.g., in the case of legal document, phrases that have the same meaning. On the defensive end it would be interesting to develop practical *robust* sketching methods in adversarial environments, e.g. see concurrent work in this direction by Boyle *et al.* [15]. Another direction would be the design of secure approximation protocols where the approximation does not rely on a fixed randomness but rather on fresh randomness via a sampling-based approach, similarly to Zadeh *et al.* [91].

IX. CONCLUSION

In this paper we introduced and studied the effectiveness of adversarial inputs for secure similarity approximation protocols. We proposed concrete perturbation attacks for the well-studied minhash and cosine sketching techniques, and measured the performance and scalability of the attacks on both real and synthetic data, while tuning various parameters. Subsequently, we formally defined a server-aided model that mitigates the aforementioned attacks. We also proposed new sketching protocols for this architecture, building upon state-of-the-art sketching techniques. Our design and implementation aimed at speeding up the reconstruction protocols, as they constitute the part of the overall computation that is executed most frequently—thus having the most severe impact on overall performance. We evaluated the implementation of the proposed protocols and demonstrated that this architecture achieves the desired scalability for the reconstruction process, with reasonable performance.

ACKNOWLEDGMENTS

The authors would like to thank Seny Kamara for his comments on the proofs of Section VI. This work was done while the first author was visiting Symantec Research Labs. The first author was partially supported by the U.S. National Science Foundation and by the Kanellakis Fellowship at Brown University.

REFERENCES

[1] "EDRM: Creating Practical Resources to Improve E-Discovery and Information Governance," www.edrm.net/, Accessed: 2017-11-27.
[2] "EDRM: Processing Guide," www.edrm.net/frameworks-and-standards/edrm-model/processing/, Accessed: 2017-11-27.
[3] "Principles and Strategy for Accelerating Health Information Exchange (HIE)," Department of Health & Human Services, USA, 2013.
[4] "MATLAB and Optimization Toolbox Release 2016a, The Mathworks, Inc., Natick, Massachusetts, United States." 2016.
[5] A. Athalye, N. Carlini, and D. A. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *in Proc. of the 35th ICML 2018*, 2018, pp. 274–283.
[6] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering gattaca: Efficient and secure testing of fully-sequenced human genomes," in *ACM CCS* , 2011, pp. 691–702.
[7] F. Baldimtsi and O. Ohrimenko, "Sorting and searching behind the curtain," in *FC*, 2015, pp. 127–146.
[8] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010.

[9] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *IEEE S&P*, 2013, pp. 478–492.

[10] J. Bethencourt. (2010) Advanced Crypto Software Collection. [Online]. Available: http://acsc.cs.utexas.edu/libpaillier/

[11] M. Blanton and F. Bayatbabolghani, "Efficient server-aided secure two-party function evaluation with applications to genomic computation," *PoPETs*, vol. 2016, no. 4, pp. 144–164, 2016.

[12] M. Blanton and P. Gasti, "Secure and efficient protocols for iris and fingerprint identification," in *ESORICS*, 2011, pp. 190–209.

[13] C. Blundo, E. De Cristofaro, and P. Gasti, "EsPRESSO: Efficient privacy-preserving evaluation of sample set similarity," *J. Comput. Secur.*, vol. 22, no. 3, pp. 355–381, 2014.

[14] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine Learning Classification over Encrypted Data," in *NDSS*, 2015.

[15] E. Boyle, R. LaVigne, and V. Vaikuntanathan, "Adversarially Robust Property-Preserving Hash Functions," in *in Proc. of 10th ITCS 2019*, vol. 124, pp. 16:1–16:20.

[16] A. Z. Broder, "On the resemblance and containment of documents," in *In Compression and Complexity of Sequences (SEQUENCES)*, 1997, pp. 21–29.

[17] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *J. Comput. Syst. Sci.*, vol. 60, no. 3, pp. 630–659, 2000.

[18] S.-A. Brown, "Patient similarity: Emerging concepts in systems and precision medicine," vol. 7, 11 2016.

[19] R. Canetti, "Security and composition of multiparty cryptographic protocols," *J. Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.

[20] ——, "Security and composition of cryptographic protocols: A tutorial (part i)," *SIGACT News*, vol. 37, no. 3, pp. 67–92, Sep. 2006.

[21] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *Proc. of 25th USENIX Sec. 16*, Aug. 2016, pp. 513–530.

[22] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, 2017, pp. 39–57.

[23] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *STOC*, 2002, pp. 380–388.

[24] Y. Chen, B. Peng, X. Wang, and H. Tang, "Large-scale privacy-preserving mapping of human genomic sequences on hybrid clouds," in *NDSS*, 2012.

[25] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, Apr. 2005.

[26] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *Proc. of the 10th ACM SIGKDD Inter. Conf. on Knowledge Disc. and Data Mining*, ser. KDD '04, 2004, pp. 99–108.

[27] I. Damgård, M. Geisler, and M. Krøigaard, "Efficient and secure comparison for on-line auctions," in *ACISP*, 2007, pp. 416–430.

[28] ——, "A correction to 'efficient and secure comparison for on-line auctions'," *IJACT*, vol. 1, no. 4, pp. 323–324, 2009.

[29] G. Danezis and E. De Cristofaro, "Fast and private genomic testing for disease susceptibility," in *Proc. of the 13th WPES*, 2014, pp. 31–34.

[30] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 271–280.

[31] Deloitte, "GCC eDiscovery survey, How ready are you?" https://tinyurl.com/jp2qwey, 2015.

[32] G. development team. (2016) GMP: The GNU multiple precision arithmetic library. [Online]. Available: https://gmplib.org/

[33] W. Dong, M. Charikar, and K. Li, "Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces," in *SIGIR*, 2008, pp. 123–130.

[34] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proc. of 9th PETS*, 2009, pp. 235–253.

[35] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright, "Secure multiparty computation of approximations," *ACM Trans. on Algorithms*, vol. 2, no. 3, pp. 435–472, 2006.

[36] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *ACM CCS*, 2015, pp. 1322–1333.

[37] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *USENIX Security*, 2014, pp. 17–32.

[38] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *CRYPTO*, 2010, pp. 465–482.

[39] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, ser. VLDB '99, 1999, pp. 518–529.

[40] O. Goldreich, *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[41] ——, *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[42] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270 – 299, 1984.

[43] Google. (2017) Protocol buffers. [Online]. Available: http://code.google.com/apis/protocolbuffers/

[44] A. Gottlieb, G. Stein, E. Ruppin, R. Altman, and R. Sharan, "A method for inferring medical diagnoses from patient similarities," vol. 11, p. 194, 09 2013.

[45] C. W. J. Granger and P. Newbold, *Forecasting economic time series*. Academic Press, 2014.

[46] M. R. Grossman and G. V. Cormack, "Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review," *Rich. JL & Tech.*, vol. 17, p. 1, 2010.

[47] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "Wtf: The who to follow service at twitter," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 505–514.

[48] J. Hastad and A. Shamir, "The cryptographic security of truncated linearly related variables," in *STOC*, 1985, pp. 356–362.

[49] T. H. Haveliwala, A. Gionis, D. Klein, and P. Indyk, "Evaluating strategies for similarity search on the web," in *Proceedings of the 11th International Conference on World Wide Web*, ser. WWW '02, 2002, pp. 432–442.

[50] M. Henzinger, "Finding near-duplicate web pages: A large-scale evaluation of algorithms," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '06. ACM, 2006, pp. 284–291.

[51] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, ser. STOC '98, 1998, pp. 604–613.

[52] Y. Ishai, J. Katz, E. Kushilevitz, Y. Lindell, and E. Petrank, "On achieving the "best of both worlds" in secure multiparty computation," *SIAM J. Comput.*, vol. 40, no. 1, pp. 122–141, 2011.

[53] P. Jaccard, "Etude de la distribution florale dans une portion des alpes et du jura," vol. 37, pp. 547–579, 01 1901.

[54] S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian, "Scaling private set intersection to billion-element sets," in *FC*, 2014, pp. 195–215.

[55] S. Keelveedhi, M. Bellare, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage," in *USENIX*, 2013, pp. 179–194.

[56] A. Khedr, P. G. Gulak, and V. Vaikuntanathan, "SHIELD: scalable homomorphic implementation of encrypted data-classifiers," *IEEE Transactions on Computers*, vol. PP:99, 2015.

[57] M. S. Kohn, J. Sun, S. Knoop, A. Shabo, B. Carmeli, D. Sow, T. Syed-Mahmood, and W. Rapp, "IBM's Health Analytics and Clinical Decision Support," in *Yearbook of Medical Informatics 9.1*, 2014, pp. 154–162.

[58] E. M. Kornaropoulos and P. Efstathopoulos, "Breaking and fixing secure similarity approximations: Dealing with adversarially perturbed inputs," Cryptology ePrint Archive, Report 2017/850, 2017.

[59] R. L. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 82–105, 2013.

[60] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of massive datasets*. Cambridge university press, 2014.

[61] P. Li and A. C. König, "b-bit minwise hashing," in *WWW*, 2010, pp. 671–680.

[62] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *journal of the Association for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[63] G. S. Manku, A. Jain, and A. Das Sarma, "Detecting near-duplicates for web crawling," in *WWW*, 2007, pp. 141–150.

[64] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.
[65] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," *IEEE Security & Privacy*, vol. 14, no. 3, pp. 68–72, 2016.
[66] L. Melis, G. Danezis, and E. D. Cristofaro, "Efficient private statistics with succinct sketches," in *NDSS*, 2016.
[67] I. Mironov, M. Naor, and G. Segev, "Sketching in adversarial environments," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1845–1870, 2011.
[68] M. Mitzenmacher, R. Pagh, and N. Pham, "Efficient estimation for high similarities using odd sketches," in *WWW*, 2014, pp. 109–118.
[69] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. New York, NY, USA: Cambridge University Press, 2005.
[70] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy, SP*, 2017, pp. 19–38.
[71] M. Mudelsee, *Climate time series analysis*. Springer, 2013.
[72] M. Naor and E. Yogev, "Bloom filters in adversarial environments," in *Advances in Cryptology - CRYPTO'15*, 2015, pp. 565–584.
[73] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang, "Privacy in the genomic era," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 6:1–6:44, Aug. 2015.
[74] D. Notterman, U. Alon, A. Sierk, and A. Levine, "Transcriptional gene expression profiles of colorectal adenoma, adenocarcinoma, and normal tissue examined by oligonucleotide arrays," *Cancer Research*, vol. 61, pp. 3124–3130, 2001.
[75] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999, pp. 223–238.
[76] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE EuroS&P*, 2016.
[77] J.-H. Park, M. Kim, B.-N. Noh, and J. B. Joshi, "A similarity based technique for detecting malicious executable files for computer forensics," in *Information Reuse and Integration, 2006 IEEE International Conference on*. IEEE, 2006, pp. 188–193.
[78] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *IEEE S&P*, 2013, pp. 238–252.
[79] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. New York, NY, USA: Cambridge University Press, 2011.
[80] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*. Springer, 2011, pp. 1–35.
[81] A. Sharafoddini, A. J. Dubin, and J. Lee, "Patient similarity in prediction models based on health data: A scoping review," *JMIR Med Inform*, vol. 5, no. 1, Mar 2017.
[82] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proc. of the 23rd 2016 ACM CCS*, 2016, pp. 1528–1540.
[83] A. Shrivastava and P. Li, "In defense of minhash over simhash," in *Proc. of AISTATS*, 2014, pp. 886–894.
[84] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013.
[85] T. Urvoy, E. Chauveau, P. Filoche, and T. Lavergne, "Tracking web spam with html style similarities," *ACM Trans. Web*, vol. 2, no. 1, pp. 3:1–3:28, 2008.
[86] T. Veugen, "Encrypted integer division and secure comparison," *Int. Journal of Applied Cryptology*, vol. 3, no. 2, pp. 166–180, 2014.
[87] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *ACM CCS*, 2015, pp. 492–503.
[88] D. J. Wu, T. Feng, M. Naehrig, and K. E. Lauter, "Privately evaluating decision trees and random forests," *PETS*, 2016.
[89] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 981–990.
[90] X. Yan, P. S. Yu, and J. Han, "Substructure similarity search in graph databases," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 766–777.
[91] R. B. Zadeh and A. Goel, "Dimension independent similarity computation," *Journal of Machine Learning Research*, vol. 14, pp. 1605–1626, 2013.
[92] P. Zhang, F. Wang, J. Hu, and R. Sorrentino, "Towards personalized medicine: Leveraging patient similarity and drug similarity analytics," vol. 2014, pp. 132–6, 04 2014.

X. Supplementary Material

A. Overview of Protocols

Overview of FindMin. Initially the server assigns the first encrypted value as the current minimum $[min]$. Then we compare the current minimum with the next encrypted value using the protocol EncComparison, which outputs the result of the comparison without revealing the encrypted values to the key holder (i.e., the client). Notice, however, that if the server iterates through the ciphertexts in the originally given order then the client can learn the index of the minimum value. To overcome this the server picks a random permutation π that is applied before any pairwise comparison (step **(1)**). Thus the client learns the index of the minimum value with respect to the secret random permutation that the server applied. After the execution of the comparison protocol the client returns a re-encryption $[c_i]$ of the smallest among the input values $[min], [y_{\pi(i)}]$, so as not to reveal to the server which of the two ciphertexts is smaller. Re-encryption (denoted as Refresh) can be achieved by either decrypting and re-encrypting the ciphertext, or by using the homomorphic properties of the cryptosystem to refresh the randomness. Since the client can decrypt $[min]$ and $[y_{\pi(i)}]$, the server blinds the ciphertexts using r_i and s_i so as to create the blinded ciphertexts $[b_i]$ and $[c_i]$. In the final step we deal with two cases. If the result of the comparison is $min < y_{\pi(i)}$ (i.e., $t_i = 1$) the server subtracts the blinding r_i from the value that the client returned. Otherwise the server subtracts s_i. Protocol FindMin performs $n-1$ encrypted comparisons of l bit integers, $8(n-1)$ homomorphic operations and $n-1$ roundtrips.

B. A Note on the Security Proofs

The security proofs take the classic simulation based approach for semi-honest adversaries on the hybrid model with ideal access to functions [19] and show that a party's view in a protocol execution is simulatable given its input, its output (if any), and access to a series of ideal functionalities. On the one hand we have the hybrid world were protocols have access to functions that are invoked by specific step of the protocol and on the other hand we have the ideal world where the simulator lives. Thus, the participating parties learn nothing from the protocol's execution beyond what can be derived from their input. For the proofs we refer the reader to the full version of our work [58].

Protocol EncHashing

Client: $SK_P^{(C)}, k, p$		Server: $[x], \{a_i\}_{i=0}^{k-1}, p$
(2) $\forall i = 2, \ldots, k-1, h_i = D(SK_P^{(C)}, [h_i])$	$\xleftarrow{[h_2],\ldots,[h_{k-1}]}$	**(1)** $\forall i = 2, \ldots, k-1$, Pick $r_i \in (0, 2^l) \cap \mathbb{Z}$, $[h_i] \coloneqq [x]^{r_i} \mod N^2$
(3) $\forall i = 2, \ldots, k-1, [h'_i] \coloneqq E(PK_P^{(C)}, h_i^i)$	$\xrightarrow{[h'_2],\ldots,[h'_{k-1}]}$	**(4)** $\forall i = 2, \ldots, k-1, [x_i] \coloneqq [h'_i]^{r_i^{-i}} \mod N^2$
(6) $h' = D(SK_P^{(C)}, [h'])$	$\xleftarrow{[h']}$	**(5)** Pick $r \in \mathbb{Z}_u$, $[h'] \coloneqq [r] \cdot [a_0] \cdot [x]^{a_1} \prod_{i=2}^{k-1} [x_i^i]^{a_i} \mod N^2$
(7) $d = h' \mod p$		**(8)** $c = r \mod p$
	$\xleftrightarrow{\text{PrvComparison}(d,c)}$	Receive $[t]$ such that $t = 1$ if $d < c$
(9) $[d] \coloneqq E(PK_P^{(C)}, d)$	$\xrightarrow{[d]}$	**(10)** Output $[h] = [d] \cdot ([c])^{-1} \cdot [t]^p$

Protocol EncComparison

Client: $SK_P^{(S)}, SK_{GM}^{(S)}, l$		Server: $[a], [b], l$
		(1) $[x] \coloneqq [2^l] \cdot [b] \cdot [a]^{-1} \mod N^2$
		(2) Pick a random $r \in (0, 2^l) \cap \mathbb{Z}$
(4) $z = D(SK_P^{(C)}, [z])$	$\xleftarrow{[z]}$	**(3)** $[z] \coloneqq [x] \cdot [r] \mod N^2$
(5) $d \coloneqq z \mod 2^l$		
		(6) $c \coloneqq r \mod 2^l$
	$\xleftrightarrow{\text{PrvComparison}(d,c)}$	Receive $\lvert t' \rvert$ such that $t' = 1$ if $c < d$
(7) $\lvert z_l \rvert \leftarrow E(PK_{GM}^{(C)}, z_l)$	$\xrightarrow{\lvert z_l \rvert}$	
		(8) $\lvert r_l \rvert \coloneqq E(PK_{GM}^{(C)}, r_l)$
(10) Output $t = D(SK_{GM}^{(C)}, \lvert t \rvert)$	$\xleftarrow{\lvert t \rvert}$	**(9)** $\lvert t \rvert \coloneqq \lvert z_l \rvert \cdot \lvert r_l \rvert \cdot \lvert t' \rvert \cdot \lvert 1 \rvert$

Protocol FindMin:

Client: $SK_{GM}^{(C)}, SK_P^{(C)}, l$		Server: $\{[y_i]\}_{i=1}^n, l$
		(1) Pick a rand. permutation π over $\{1, \ldots, n\}$
		(2) $[min] \coloneqq [y_{\pi(1)}]$
for $i = 2$ **to** n **do**		
(3) Receive bit t_i s.t. $t_i = 1$ if $min < y_{\pi(i)}$	$\xleftrightarrow{\text{EncComparison}((SK_P, SK_{GM}, l), ([min], [y_{\pi(i)}], l))}$	
		(4) Pick random $r_i, s_i \in (0, 2^{l+\lambda}) \cap \mathbb{Z}$
	$\xleftarrow{[b_i],[c_i]}$	**(5)** $[b_i] \coloneqq [min] \cdot [r_i] \mod N^2$, $[c_i] \coloneqq [y_{\pi(i)}] \cdot [s_i] \mod N^2$
if t_i is 1 **then**		
(6a) $[c_i] \coloneqq \text{Refresh}([b_i])$		
else		
(6b) $[c_i] \coloneqq \text{Refresh}([c_i])$		
end if	$\xrightarrow{[c_i],[t_i]}$	**(7)** $[min] \coloneqq [c_i] \cdot ([t_i] \cdot [-1])^{s_i} \cdot [t_i]^{-r_i} \mod N^2$
end for		
		(8) Output $[min]$

Fig. 9. Protocol EncComparison is a slight modification of the comparison protocol found in [14], [86]. Protocol EncComparison2 is the same up to step **(9)** where it terminates by outputting $|t|$ to the Server.