Sketching in Adversarial Environments[*]

Ilya Mironov
Microsoft Research
Silicon Valley Campus
mironov@microsoft.com

Moni Naor[†] [‡]
Weizmann Institute of Science
Rehovot, Israel
moni.naor@weizmann.ac.il

Gil Segev[‡]
Weizmann Institute of Science
Rehovot, Israel
gil.segev@weizmann.ac.il

ABSTRACT

We formalize a realistic model for computations over massive data sets. The model, referred to as the *adversarial sketch model*, unifies the well-studied sketch and data stream models together with a cryptographic flavor that considers the execution of protocols in "hostile environments", and provides a framework for studying the complexity of many tasks involving massive data sets.

The adversarial sketch model consists of several participating parties: honest parties, whose goal is to compute a pre-determined function of their inputs, and an adversarial party. Computation in this model proceeds in two phases. In the first phase, the adversarial party chooses the inputs of the honest parties. These inputs are sets of elements taken from a large universe, and provided to the honest parties in an on-line manner in the form of a sequence of insert and delete operations. Once an operation from the sequence has been processed it is discarded and cannot be retrieved unless explicitly stored. During this phase the honest parties are not allowed to communicate. Moreover, they do not share any secret information and any public information they share is known to the adversary in advance. In the second phase, the honest parties engage in a protocol in order to compute a pre-determined function of their inputs.

In this paper we settle the complexity (up to logarithmic factors) of two fundamental problems in this model: testing whether two massive data sets are equal, and approximating the size of their symmetric difference. We construct *explicit and efficient* protocols with sublinear sketches of essentially optimal size, poly-logarithmic update time during the first phase, and poly-logarithmic communication and computation during the second phase. Our main technical contribution is an explicit and deterministic encoding scheme that enjoys two seemingly conflicting properties: incrementality and high distance, which may be of independent interest.

[*]Due to space limitations a more complete version is available at http://www.wisdom.weizmann.ac.il/∼naor.

[†]Incumbent of the Judith Kleeman Professorial Chair. Research supported in part by a grant from the Israel Science Foundation.

[‡]Most of the work was done at Microsoft Research.

Categories and Subject Descriptors

F.1.1 [**Theory of Computation**]: Models of Computation; F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity.

General Terms

Algorithms, Theory.

Keywords

Massive data sets, sketch model, data stream model.

1. INTRODUCTION

The past two decades have introduced striking technological breakthroughs in information collection and storage capabilities. These breakthroughs allowed the emergence of enormous collections of data, referred to as massive data sets, such as the World Wide Web, Internet traffic logs, financial transactions, the human genome, census data and many more. This state of affairs introduces new and exciting challenges in analyzing massive data sets and extracting useful information.

From a computational point of view, most of the traditional computational models consider settings in which the input data is easily and efficiently accessible. This is, however, usually not the case when dealing with massive data sets. Such data sets may either be stored on highly constrained devices or may only be accessed in an on-line manner without the ability to actually store any significant fraction of the data. In recent years several computational approaches which are suitable for computing over massive data sets have been developed, such as sketch and lossy compression schemes [13, 23], data stream computations [3, 21, 26], and property testing [25, 36].

Motivated by the challenges posed by computational tasks involving massive data sets, and by the existing approaches for performing such tasks, we formalize a realistic model of computation which we refer to as the *adversarial sketch model*. This model can be seen as unifying the standard sketch model and the data stream model together with a cryptographic flavor that considers the execution of protocols in "hostile environments". The model under consideration provides a framework for studying the complexity of many fundamental and realistic problems that arise in the context of massive data sets. In what follows we briefly describe the standard sketch model and the data stream model, as well as our approach for modeling computations in hostile environments in this context.

The standard sketch model. In the standard sketch model the input is distributed among several parties. Each party runs a compression procedure to obtain a compact "sketch" of its input, and these sketches are then delivered to a referee. The referee has to compute the value of a predetermined function applied to the inputs of the parties by using only the sketches and not the actual inputs. The parties are not allowed to communicate with each other, but are allowed to share a random reference string which is chosen independently of their inputs. This string can be used, for example, to choose a random hash function that will be applied by each party to obtain a compressed sketch of its input. The standard sketch model fits many scenarios in which a massive data set is partitioned and stored in a distributed manner in several locations. In each location a compressed sketch of the stored data is computed, and then sent to a central processing unit that uses only the sketches and not the actual data. The main performance criterion for protocols in this model is the size of the sketches. We note that this model is the public-coin variant of the simultaneous communication model introduced by Yao [40].

The data stream model. In the data stream model the input is received as a one-way stream. Once an element from the stream has been processed it is discarded and cannot be retrieved unless it is explicitly stored in memory, which is typically small relative to the size of the data stream. The data stream model captures scenarios in which computations involve either massive data sets that are stored on sequential magnetic devices (for which one-way access is the most efficient access method), or on-line data which is continuously generated and not necessarily stored. The main performance criteria for algorithms in this model is the amount of storage they consume and the amount of time required for processing each element in the stream. For a more complete description of this model, its variants and the main results we refer the reader to a survey by Babcock et al. [5].

The adversarial factor. In the standard sketch model described above, it is assumed that the parties share a random string, which is chosen independently of the inputs held by the parties[1]. In many real-life scenarios, however, it is not at all clear that such an assumption is valid. First, since the parties are assumed not to communicate with each other, this enforces the introduction of trust in a third party to set up the random string. In many situations such trust may not be available, and if the shared string is set up in an adversarial manner there are usually no guarantees on the behavior of the protocol. That is, there may be "bad" choices of the shared string that cause the protocol to fail with very high probability. Second, even when a truly random string is available, this string may be known to an adversary as well (and in advance), and serve as a crucial tool in attacking the system. For example, an adversary may be able to set the inputs of the parties after having seen the random string. Thus, when considering computations in a setting where the inputs of the parties may be adversarially chosen, it is usually not justified to assume independence between the shared random string and the inputs of the parties[2]. For these reasons we are interested in exploring the feasibility and efficiency of computations over massive data sets in hostile environments. In such environments the honest parties do not share any secret information, and any public information they share is known to the adversary in advance who may then set the inputs of the parties. Protocols designed in such a model have significant security and robustness benefits.

[1]We note that in the data stream model, when dealing only with insertions, several deterministic algorithms are known, most notably those based on the notion of *core-sets* (see, for example, [2, 6]).

Sketching in adversarial environments. We consider a model with three participating parties: two honest parties, Alice and Bob, and an adversarial party[3]. Computation in this model proceeds in two phases. In the first phase, referred to the as the *sketch phase*, the adversarial party chooses the inputs of Alice and Bob. These inputs are sets of elements taken from a large universe $\mathcal{U}$, and provided to the honest parties in an on-line manner in the form of a sequence of insert and delete operations. Once an operation from the sequence has been processed it is discarded and cannot be retrieved unless explicitly stored. This phase defines the input sets $S_A \subseteq \mathcal{U}$ and $S_B \subseteq \mathcal{U}$ of Alice and Bob, respectively. During this phase the honest parties are completely isolated in the sense that (1) they are not allowed to communicate with each other, and (2) the sequence of operations communicated to each party is hidden from the other party. In addition, we assume that the honest parties do not share any secret information, and that any public information they share is known to the adversary in advance. In the second phase, referred to as the *interaction phase*, Alice and Bob engage in a protocol in order to compute (or approximate) a pre-determined function of their input sets.

When designing protocols in the adversarial sketch model we are mainly interested in the following performance criteria: (1) the amount of storage (i.e., the size of the sketches), (2) the update time during the sketch phase (i.e., the time required for processing each of the insert and delete operations), and (3) the communication and computation complexity during the interaction phase.

The most natural question that arises in this setting is to characterize the class of functions that can be computed or approximated in this model with sublinear sketches and poly-logarithmic update time, communication and computation. In the standard sketch model, a large class of functions was shown to be computed or approximated with highly compressed sketches whose size is only poly-logarithmic in the size of the input. Therefore, one can ask the rather general question of whether the adversarial sketch model "preserves sublinearity and efficiency". That is, informally:

Is any function, computable in the standard sketch

[2]Typical examples include: (1) Plagiarism detection – two parties wish to compute some similarity measure between documents. In this case the inputs (i.e., the documents) are chosen by the assumed plagiarizer. (2) Traffic logs comparison: two internet routers wish to compare their recent traffic logs. The inputs of the routers can be influenced by any party that can send packets to the routers.

[3]For concreteness we focus in this informal discussion on the simplest case where only two honest parties are participating in the computation. We note that the model naturally generalizes to any number of honest parties.

model with highly compressed sketches, also computable in the adversarial sketch model with sublinear sketches and poly-logarithmic update time, communication and computation?

In fact, one can consider a relaxed variant of this question that does not take into account the adversarial factors. Namely, the question of characterizing the class of functions that can be computed in the standard sketch model in an incremental manner. A negative answer to this question in turn implies a negative answer to the above question.

1.1 Our Contributions

In this paper we study the two fundamental problems of testing whether two massive data sets are equal, and approximating the size of their symmetric difference. For these problems we provide an affirmative answer to the above question. We construct *explicit and efficient* protocols with sketches of essentially optimal size, and poly-logarithmic update time, communication, and computation. We settle the complexity, up to logarithmic factors, of these two problems in the adversarial sketch model.

Our main technical contribution, that serves as a building block of our protocols, is an explicit and deterministic encoding scheme that enjoys two seemingly conflicting properties: incrementality and high distance. That is, the encoding guarantees that (1) for any set S and element x the encodings of the sets $S \cup \{x\}$ and $S \setminus \{x\}$ can be easily computed from the encoding of S by modifying only a small number of entries, and (2) the encodings of any two distinct sets significantly differ with respect to a carefully chosen weighted distance. In addition, the scheme enables efficient (linear time) decoding. We believe that an encoding scheme with these properties can find additional applications, and may be of independent interest. In what follows we formally state our results[4].

Equality testing. An equality testing protocol in the adversarial sketch model is parameterized by the size N of the universe of elements from which the sets are taken, and by an upper bound K on the size of the sets to be tested[5]. Our construction provides an explicit protocol, and in addition a non-constructive proof for the existence of a protocol that enjoys slightly better guarantees[6]. We prove the following theorem:

THEOREM 1.1. *In the adversarial sketch model, for every N, K and $0 < \delta < 1$ there exists a protocol for testing the equality of two sets of size at most K taken from a universe of size N with the following properties:*

1. *Perfect completeness: For any two sequences of insert and delete operations communicated to the parties that lead to the same set of elements, the parties always output* `Equal`.
2. *Soundness: For any two sequences of insert and delete operations communicated to the parties that do not lead to the same set of elements, the parties output* `Not Equal` *with probability*[7] *at least $1 - \delta$.*
3. *The size of the sketches, the update time during the sketch phase, and the communication complexity during the interaction phase are described in Table 1.*

A rather straightforward reduction of computations in the private-coin simultaneous communication model to computations in the adversarial sketch model implies that the size the sketches in our protocols is essentially optimal (the following theorem is stated for protocols with constant error).

THEOREM 1.2. *Equality testing in the adversarial sketch model requires sketches of size $\Omega\left(\sqrt{K \log(N/K)}\right)$.*

Approximating the size of the symmetric difference. We construct a protocol that enables two parties to approximate the size of the symmetric difference between their two input sets determined during the sketch phase. We prove the following theorem:

THEOREM 1.3. *In the adversarial sketch model, for every N, K, $0 < \delta < 1$ and constant $0 < \rho \le 1$, there exists a protocol for approximating the size of the symmetric difference between two sets of size at most K taken from a universe of size N with the following properties:*

1. *For any two sequences of insert and delete operations communicated to the parties that lead to sets with symmetric difference of size $\Delta_{\texttt{OPT}}$, the parties output $\Delta_{\texttt{APX}}$ such that*
$$\Pr\left[\Delta_{\texttt{OPT}} \le \Delta_{\texttt{APX}} \le (1+\rho)\Delta_{\texttt{OPT}}\right] > 1 - \delta \ .$$
2. *Sketches of size $O\left(\sqrt{K \log N} \cdot (\log\log K + \log(1/\delta))\right)$.*
3. *Update time $O(\log K \cdot \log N)$.*
4. *Communication complexity*
$O\left(\left(\log^2 K + \log K \cdot \log\log N\right)(\log\log K + \log(1/\delta))\right)$.

As with the equality testing protocol, our construction provides an explicit protocol as well. The explicit protocol guarantees that $\Pr\left[\Delta_{\texttt{OPT}} \le \Delta_{\texttt{APX}} \le \mathrm{polylog}(N)\Delta_{\texttt{OPT}}\right] > 1 - \delta$, and the size of sketches, update time and communication complexity match those stated in Theorem 1.3 up to poly-logarithmic factors.

Dealing with multisets and real-valued vectors. For simplicity, we stated the above results assuming that the inputs of the parties are sets. Both of our protocols, however, can in fact be used when the inputs of the parties are multisets containing at most K distinct elements each, and

[4]Our protocols have the property that, during the interaction phase, the amount of computation is linear in the amount of communication. Therefore, for simplicity, we omit the computation cost and only state the communication complexity.

[5]We note that the upper bound K on the size of the sets only imposes a restriction on the size of the sets at the end of the sketch phase. During the sketch phase the parties should be able to deal with sets of arbitrary size, and nevertheless the size of the sketches refers to their maximal size during the sketch phase. A possible adversarial strategy, for example, is to insert all the N possible elements and then to delete $N - K$ of them.

[6]The poly-logarithmic gap between the implicit and the explicit parameters in this paper is due to a poly-logarithmic gap between the optimal and the known explicit constructions of dispersers (see, for example, [38]). Any improved explicit construction of dispersers will, in turn, improve our explicit protocols.

[7]The probability is taken only over the internal coin tosses of the honest parties.

	Non-explicit protocol	**Explicit protocol**
Size of sketches	$O\left(\sqrt{K \cdot \log N \cdot \log(1/\delta)}\right)$	$\sqrt{K \cdot \mathrm{polylog}(N) \cdot \log(1/\delta)}$
Update time	$O(\log K \cdot \log N)$	$\mathrm{polylog}(N)$
Communication	$O\left(\left(\log^2 K + \log K \cdot \log\log N\right) \cdot \log(1/\delta)\right)$	$\mathrm{polylog}(N) \cdot \log(1/\delta)$

Table 1: The non-explicit and explicit parameters of the equality testing protocol.

even real-valued vectors of length N with at most K non-zero entries. In these cases, the equality protocol determines whether the inputs are equal as multisets or as real-valued vectors, and the symmetric difference protocol approximates the number of distinct elements in the symmetric difference between the multisets or the number of non-zero entries in the difference between the vectors.

For concreteness, in this paper we present our protocols for multisets, and note that they can be easily adapted for real-valued vectors given an appropriate representation. When applied to multisets, an additional parameter of the problems under consideration is an upper bound, M, on the number of appearances of any element in each multiset. This generalization, however, hardly affects the performance of our protocols. The only change is that the size of sketches, the update time, the communication and the computation increase by a multiplicative factor of $\log M$.

1.2 Related Work

Due to the recent blow-up in the emergence of massive data sets, extensive work has been devoted to designing sketch-based algorithms for many tasks, such as estimating various similarity and distance measures, compressed data structures, histogram maintenance, and many more. It is far beyond the scope of this paper to present an exhaustive overview of this ever-growing line of work. The reader may find Bar-Yossef's Ph.D. thesis [7] (and the many references therein) and the Handbook of Massive Data Sets [1] as sources of preliminary information and reference. We focus only on the main results that are relevant to our setting.

Gibbons and Matias [23] denoted as *synopsis data structures* any data structures that are substantively smaller than their base data sets. Their goal was to design data structures that support queries for massive data sets while minimizing the number of disk accesses. They constructed such data structures for estimating frequency moments, estimating the number of distinct elements, identifying frequent elements and maintaining histograms and quantiles. Broder et al. [13, 14] developed efficient sketching techniques to determine the syntactic similarity of documents. They defined a similarity measure known as *resemblance*, which was shown to be applicative in eliminating near-duplicates of web pages. Sketch computations were found useful in particular for approximate nearest-neighbor algorithms in high-dimensional spaces by Indyk and Motwani [28] and by Kushilevitz, Ostrovsky and Rabani [31]. Both of these approaches were based on compressed sketches for estimating various distance measures, such as the Hamming distance and L^p norms. Sketches for approximating such distance measures were also developed by Feigenbaum et al. [20, 21] and by Indyk and Woodruff [29]. Additional sketch-based approximations include an edit distance approximation due to Bar-Yossef, Jayram, Krauthgamer and Kumar [8], and Cosine similarity and earth mover distance due to Charikar [16]. We note that the above mentioned results rely on public randomness which enables highly compressed sketches (usually of poly-logarithmic size), which are much smaller than the possible sketches in the adversarial model (given the lower bound stated in Theorem 1.2).

Compressed sensing. Sketch computations are of key importance in *compressed sensing* [15, 18], a rapidly developing field of research in signal processing. Algorithms for compressed sensing receive as input a signal and output a short sketch of the signal, usually via a small set of non-adaptive linear measurements, that can be used to approximate the signal (see, for example, [15, 17, 24, 27]). Compressed sensing is most effective when the signal can be well approximated using much fewer vectors from some fixed basis than the signal's nominal dimension.

Early results on compresses sensing showed that the set of measurements can be chosen from some distribution, and the reconstruction algorithm was guaranteed to be correct for any specific signal with high probability over the choice of the set of measurements. However, for any set of measurements it may be possible to choose a specific signal on which the reconstruction fails. Recently, sets of measurements with significantly stronger uniform recovery properties have been discovered. Specifically, it is possible to choose one set of measurements which is "good" (in the above sense) for all signals. Although this coincides with the approach underlying the adversarial sketch model, two desiderata of protocols in the adversarial sketch model are incompatible with existing compressed sensing mechanisms: poly-logarithmic update time and sublinear space requirement.

1.3 Overview of Our Techniques

In this section we provide an informal overview of the main techniques underlying our protocols. We focus here on the problem of testing whether two massive data sets are equal, as it already illustrates the main ideas. We first make an attempt to point out the difficulties of designing an efficient equality testing protocol in the adversarial sketch model by demonstrating that known solutions to several relaxations do not seem to extend to the model under consideration.

Relaxation 1: The standard sketch model. In the standard sketch model there are highly efficient solutions which take advantage of the fact that the parties share a random string which is chosen independently of their inputs. A very simple protocol proceeds as follow: The shared random string is a description of a randomly chosen function h from a family of pair-wise independent functions that map sets of size K to $\{0, 1\}$. The sketch of each party consists of a single bit obtained by applying h to its input set[8]. Clearly, if the sets are equal then the sketches are equal as well, and if the sets are not equal then the sketches differ with probability $1/2$ over the choice of h. Such an approach was demonstrated by Blum et al. [11] (in the context of memory checking) to efficiently support incremental updates by using

[8]More specifically, the parties agree ahead of time on a canonical representation for sets, and apply h to the representations of their sets.

ϵ-biased hash functions [33] instead of pair-wise independent hash functions.

Relaxation 2: The private-coin simultaneous communication model. The above solution heavily relies on the shared random string available to the parties. When such a string is not available, but arbitrary access to the inputs during the sketch phase is allowed, the model is equivalent to the private-coin simultaneous communication model [30, 40] (this is exactly the standard sketch model without shared randomness). In this model the complexity of the equality function is well-studied and tight bounds are known. Babai and Kimmel [4] (generalizing Newman and Szegedy [34]) proved that in any equality testing protocol in this model it holds that $s \times t = \Omega(K)$ where s and t are the amount of communication sent to the referee by the parties when comparing two K-bit strings.

In the simultaneous communication model there is an explicit protocol that matches this lower bound. For simplicity we present the protocol for comparing two K-bit strings. Alice and Bob agree ahead of time on an error-correcting code $C : \{0,1\}^K \rightarrow \{0,1\}^{O(K)}$ for which any two distinct codewords differ on at least a $(1-\epsilon)$-fraction of their entries (for some constant $0 < \epsilon < 1$). Alice and Bob encode their inputs, and view the codewords as $O(\sqrt{K}) \times O(\sqrt{K})$ matrices. Alice sends a random row to the referee and Bob send a random column. The referee compares the bits at the intersection of these row and column, and outputs `Equal` if and only if they match. Clearly, if the inputs of the parties are equal then the bits in the intersection always match, and if the inputs are not equal then bits differ with probability at least $1-\epsilon$.

Such a solution does not seem to extend to the adversarial sketch model in which the parties are given only restricted access to their inputs. The inputs are provided in an on-line manner and once an operation from the sequence has been processed it is discarded and cannot be retrieved. Therefore it does not seem possible for the parties to encode their inputs in an incremental manner (i.e., with only a small cost for *each* update operation) and still obtain codewords that significantly differ. Overcoming the inherent difficulty of constructing an *incremental encoding scheme with high distance* is the main idea underlying our protocols.

Relaxation 3: Comparing "close" sets. The final relaxation we consider is not a relaxation of the model, but a relaxation of the problem. Suppose that we are guaranteed that the symmetric difference between the inputs sets S_A and S_B of the honest parties is rather small. In this case there is a simple protocol with highly compressed sketches, very fast updates and low communication. Denote by ℓ the maximal size of the symmetric difference between the two sets. The honest parties agree ahead of time on a mapping T from the universe $\mathcal{U}$ of elements to vectors over some domain with the following simple property: For every ℓ distinct elements $x_1, \ldots, x_\ell \in \mathcal{U}$ the vectors $T(x_1), \ldots, T(x_\ell)$ are linearly-independent. As a concrete mapping, for example, we assume that $\mathcal{U} = [N]$ and use $T : [N] \rightarrow \mathrm{GF}(Q)^\ell$ defined by $T(x) = (1, x, \ldots, x^{\ell-1})$ for some prime Q in the range $[N, 2N]$. The sketch of a set S is given by $\sum_{x \in S} T(x)$, and is clearly easily updated given an on-line sequence of insert and delete operations that determine the set S. In the interaction phase the parties compare their sketches and output `Equal` if and only if the sketches are equal. Then, for equal sets the sketches are always equal. If the sets are not equal but the size of their symmetric difference is at most ℓ then the sketches always differ. The property of this solution is that the size of the sketch and the communication complexity are proportional to the size of the assumed symmetric difference between the sets and not to the size of the sets. We use this solution as a tool in our construction[9].

Conflicting properties: incrementality vs. high distance. Our main technical contribution is a deterministic encoding that enjoys two seemingly conflicting properties:

- **Incrementality:** Given an encoding of a set $S \subseteq \mathcal{U}$ and an additional element $x \in \mathcal{U}$, the encodings of the sets $S \cup \{x\}$ and $S \setminus \{x\}$ can be easily computed from the encoding of S without having to recompute the entire encoding of the new set. By "easily computed" we mean that only a small number of modifications is required (for example, poly-logarithmic in the size of the encoding of S).

- **High distance:** For any two distinct sets $S_A, S_B \subseteq \mathcal{U}$ of size at most K, the encodings of the sets significantly differ.

Clearly, if we consider the Hamming distance between codewords then such an encoding scheme does not exist: take any set $S \subseteq \mathcal{U}$ of size less than K, and an element $x \notin S$. Then, on one hand the incrementality property implies that the encodings of S and of $S \cup \{x\}$ differ only on very few entries, while on the other hand the high distance property implies that they differ on a significant fraction of the entries.

In our construction we manage to circumvent this conflict by considering a more generalized *weighted* distance, in which different entries are associated with different weights. More specifically, we map each set $S \subseteq \mathcal{U}$ to a *logarithmic number* of codewords $C(S) = \{C_1^S, \ldots, C_k^S\}$, and consider the following distance measure:

$$\mathrm{dist}\left(C(S_A), C(S_B)\right) = 1 - \prod_{i=1}^{k} \left(1 - \mathrm{d_H}\left(C_i^{S_A}, C_i^{S_B}\right)\right) \ ,$$

where $\mathrm{d_H}$ denotes the normalized Hamming distance. This approach enables us to construct an incremental encoding scheme in which the above distance between any two distinct codewords is at least $1-\epsilon$, for constant ϵ. The challenge in constructing such an encoding is to minimize the number k of codewords while enabling incremental updates. The number k of codewords corresponds in our protocols to the communication complexity during the interaction phase. When $k = 1$, the above distance is the normalized Hamming distance and in this case, as discussed above, the encoding cannot be incremental. When k is rather large, the encoding may be incremental, but this will lead to high communication complexity in our protocols. In our encoding, we show that $k \approx \log K$ suffices in order to enjoy "the best of the two worlds": updates require only a poly-logarithmic number of modifications, and the normalized distance between any two distinct codewords at least is $1-\epsilon$.

[9]Similar ideas are well-known in coding theory and also found applications in various other settings (for example, public-key traitor tracing [12], amortized communication complexity [19], signal processing [39] and many more). The set of linearly independent vectors is usually derived from the parity-check matrix of a linear error-correcting code.

We make a concentrated effort to provide an *explicit* encoding without relying on public randomness. The construction is based on a certain form of unbalanced bipartite graphs with random-like properties, that can be easily shown to be satisfied by random graphs. Obtaining explicit constructions, however, is much more subtle. In many cases, one can replace random bipartite graphs with explicit constructions of extractors, as was very recently done by Indyk [27] in the context of explicit constructions for compressed sensing [15, 18]. Explicit constructions of extractors, however, are still rather far from optimal. In this paper we emphasize the importance of identifying the *minimal properties* needed for the constructions, and as a result of identifying these properties we manage to base our constructions on "weaker" objects – dispersers, instead of extractors. Explicit constructions of dispersers are more practical, and are optimal up to poly-logarithmic factors.

More specifically, the encoding scheme is based on unbalanced bipartite graphs, which we refer to as *bounded-neighbor dispersers*. Very informally, we use these graphs as a deterministic way of mapping elements into "buckets" such that any set of elements of certain size does not lead to too many "overflowing buckets". In each bucket we apply a local encoding that "resolves collisions" among the elements mapped to the bucket[10]. A formal description of our encoding scheme is provided in Section 2.

The equality testing protocol. Given such an incremental encoding scheme, our protocol proceeds as follows: During the sketch phase each party incrementally updates an encoding that corresponds to the set defined by the sequence of insert and delete operations it receives from the adversary. That is, at the end of the sketch phase the sketches held by the parties are the encodings of their inputs sets. In the interaction phase, the parties compare a few entries of their sketches, and output `Equal` if and only if they all match. The size of the sketches in this protocol, however, is not sublinear in the size of the sets. In order to overcome this difficulty, we follow the approach described above in the simultaneous communication model, and have each party store and update only a small random sample of the entries of its codeword. Such a small sample (of size square-root of the size of a codeword) will still allow the parties to compare a few random entries. This results in a very efficient protocol with sketches of size $\widetilde{O}(\sqrt{K})$, poly-logarithmic update time during the sketch phase and poly-logarithmic communication and computation during the interaction phase. Moreover, the above mentioned lower bound of Babai and Kimmel [4] implies that the size of the sketches in our protocol is essentially optimal. A formal description of the protocol is provided in Section 3.

1.4 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we describe the encoding scheme which serves as a building block of our protocols. In Section 3 we present an equality testing protocol in the adversarial sketch model, and in Section 4 we demonstrate that a similar approach can be refined and extended to approximate the size of the symmetric difference between two sets. In Section 5 we present constructions of bounded-neighbor dispersers that are used to instantiate our protocols. Section 6 provides some concluding remarks and open problems.

2. THE INCREMENTAL ENCODING

In this section we present the encoding scheme that serves as the basis of our protocols. The scheme encodes multisets, and is parameterized as follows: N – the size of the universe $\mathcal{U}$ of elements (for simplicity we assume that $\mathcal{U} = [N]$), K – the maximal number of distinct elements in each multiset, and M – the maximal number of appearances of each element in any multiset. In the longer version of this paper we show that the scheme can be augmented to enable fast decoding as well[11].

The encoding of a multiset $S \subseteq [N]$ consists of a sequence of codewords $C(S) = \{C_0^S, \dots, C_{k+1}^S\}$, where $k = \lceil \log_{1+\rho} K \rceil$ for some constant $\rho > 0$, with the following properties[12]:

- **Incrementality:** Given an encoding $C(S)$ of a multiset $S \subseteq \mathcal{U}$ and an additional element $x \in \mathcal{U}$, the encodings of $S \cup \{x\}$ and $S \setminus \{x\}$ can be computed from the encoding of S by modifying only a poly-logarithmic number of entries in each codeword C_i^S.
- **High distance:** For any two distinct multisets $A, B \subseteq \mathcal{U}$, each containing at most K distinct elements and no element appears more than M times, there exists an integer $0 \le i \le k+1$ for which the codewords C_i^A and C_i^B differ on a $(1-\epsilon)$-fraction of their entries (note that this in particular implies distance at least $1-\epsilon$ according to the metric discussed in Section 1.3).

The scheme incorporates two encodings: a *global encoding* that maps each element $x \in [N]$ to several entries of each of the $k+2$ codewords, and a *local encoding* that applies to the sets of elements mapped to each entry. More specifically, given a multiset $S \subseteq [N]$, we denote by $S_{i,y} \subseteq S$ the *set* of elements in S that are mapped to entry y of the i-th codeword, denoted $C_i^S[y]$ (note that we consider $S_{i,y}$ as a set and not as a multiset). We construct a mapping with the following property: for any two distinct multisets $A, B \subseteq [N]$ that contain at most K distinct elements each, there exists an index i for which $1 \le |A_{i,y} \Delta B_{i,y}| \le \ell$ for a significant fraction of the entries y and for some integer ℓ (where Δ denotes symmetric difference). The local encoding will then guarantee that $C_i^A[y] \ne C_i^B[y]$ by using the solution for "close" sets described in Section 1.3 as relaxation 3.

For each codeword C_i, we view the global encoding as a bipartite graph $G = (L, R, E)$, where the set of vertices on the left, L, is identified with the universe of elements $[N]$, and the vertices on the right, R, are identified with the entries of C_i. An element $x \in [N]$ is mapped to entry $C_i[y]$ if and only if the edge (x, y) exists. We are interested in bipartite graphs with the following property: For every set $S \subseteq [N]$ of size roughly K', at least $(1-\epsilon)$-fraction of the vertices $y \in R$

[10]Ideas along these lines were also used by Moran et al. [32] for designing history-independent data structures and conflict resolution algorithms.

[11]Although our protocols do not take advantage of this property, an encoding scheme with the three properties of incrementality, high distance and fast decoding may be of independent interest and find additional applications beyond the setting of this paper.

[12]For the equality testing protocol it is sufficient to only consider $\rho = 1$. For approximating the size of the symmetric difference it will be useful to consider the more general case.

have at least one and at most ℓ neighbors from S. Now consider any two different multisets $A, B \subseteq [N]$ that contain at most K distinct elements each, and suppose that the number of distinct elements in the multiset $S = A \Delta B$ is roughly K'. Then, such a property of the bipartite graph corresponding to the i-th codeword implies that $1 \leq |A_{i,y} \Delta B_{i,y}| \leq \ell$ for at least $(1-\epsilon)$-fraction of the entries $C_i[y]$. We refer to such graphs as *bounded-neighbor dispersers*. Formally, we define:

Definition 2.1. *Let $G = (L, R, E)$ be a bipartite graph. For a set $S \subseteq L$ and an integer ℓ we denote by $\Gamma(S, \ell)$ the set of all vertices in R that have at least one and at most ℓ neighbors in S.*

Definition 2.2. *A bipartite graph $G = (L, R, E)$ is a $(K, \epsilon, \rho, \ell)$-bounded-neighbor disperser if for every $S \subseteq L$ such that $K \leq |S| < (1+\rho)K$, it holds that $|\Gamma(S, \ell)| \geq (1-\epsilon)|R|$.*

For such graphs we denote $|L| = N$, and in addition we assume that all the vertices on the left have the same degree D, which is called the left-degree of the graph.

The local encoding uses the solution described in Section 1.3 that applies whenever we are guaranteed to compare multisets for which the number of distinct elements in their symmetric difference is rather small. Denote by ℓ the bound on the size of the symmetric difference provided by the global encoding. That is, for any two different multisets $A, B \subseteq [N]$ that contain at most K distinct elements each the global encoding guarantees that there exists an index i for which $1 \leq |A_{i,y} \Delta B_{i,y}| \leq \ell$ for a significant fraction of the entries y. The local encoding consists of a mapping T from the universe $\mathcal{U}$ of elements to vectors over some domain with the following property: For every ℓ distinct elements $x_1, \ldots, x_\ell \in \mathcal{U}$ the vectors $T(x_1), \ldots, T(x_\ell)$ are linearly-independent. As a concrete mapping, for example, we use $T : [N] \to \mathrm{GF}(Q)^\ell$ defined by $T(x) = (1, x, \ldots, x^{\ell-1})$ for some prime Q such that $\max\{N, M\} < Q \leq 2\max\{N, M\}$. The local encoding in each entry $C_i^S[y]$ is given by $\sum_{x \in S_{i,y}} \sharp(x, S) \cdot T(x)$, where $\sharp(x, S)$ denotes the number of appearances of x in S. This is clearly easily updated given an on-line sequence of insert and delete operations that determine the multiset S.

A formal description. Let $G_0, \ldots, G_{k+1}$ denote a sequence of bipartite graphs $G_i = (L = [N], R_i, E_i)$ with left-degree D_i. The graphs are constructed such that each G_i is a $(K_i = (1+\rho)^i, \epsilon, \rho, \ell)$-bounded-neighbor disperser for some constant $0 < \epsilon < 1$ and an integer $\ell \geq 1$. The encoding consists of a sequence of codewords $C_0, \ldots, C_{k+1}$ (initialized with all zero entries). Each codeword C_i is identified with the right side R_i of the bipartite graph G_i, and contains $|R_i|$ entries denoted by $C_i[1], \ldots, C_i[|R_i|]$. An additional tool in our construction is a mapping T from the universe $[N]$ to vectors over some domain that was described above.

Figure 1 describes the incremental update operations of the encoding. That is, given an encoding $\{C_0, \ldots, C_{k+1}\}$ of a multiset $S \subseteq [N]$ and an element $x \in [N]$, Figure 1 describes the required modifications to the codewords in order to obtain the encodings of $S \cup \{x\}$ and of $S \setminus \{x\}$. We note that the update operations naturally extend to deal with real-valued multiplicities of elements. The following two lemmata state properties of the encoding that will be used in our protocols: unique encoding and high distance.

```
Insert(x, {C_0, ..., C_{k+1}}):
1: for $i = 0$ to $k+1$ do
2:     for all neighbors $y$ of $x$ in the graph $G_i$ do
3:         $C_i[y] \leftarrow C_i[y] + T(x)$

Delete(x, {C_0, ..., C_{k+1}}):
1: for $i = 0$ to $k+1$ do
2:     for all neighbors $y$ of $x$ in the graph $G_i$ do
3:         $C_i[y] \leftarrow C_i[y] - T(x)$
```

Figure 1: The insert and delete operations.

Lemma 2.3. *Any two sequences of insert and delete operations that lead to the same multiset result in the same encoding.*

Lemma 2.4. *Fix two distinct multisets $A, B \subseteq [N]$, each containing at most K distinct elements and no element appears more than M times. Denote by $C^A = \{C_0^A, \ldots, C_{k+1}^A\}$ and by $C^B = \{C_0^B, \ldots, C_{k+1}^B\}$ the encodings of A and B, respectively, and denote by d the number of distinct elements in $A \Delta B$. Then, for $i = \lfloor \log_{1+\rho} d \rfloor$, the codewords C_i^A and C_i^B differ on at least $(1-\epsilon)$-fraction of their entries.*

In Table 2 we describe the size of the codewords and the update time. The generic parameters are obtained directly from the parameters of the bounded-neighbor dispersers $G_0, \ldots, G_{k+1}$. For simplicity, we assume in Table 2 that $M \leq N$, but note that this is not essential for our construction. The non-explicit and explicit parameters are obtained by instantiating these graphs with the explicit and non-explicit constructions from Theorem 5.1 and Corollary 5.4, respectively.

Finally, we note that in the implicit parameters (where the construction of bounded-neighbor dispersers guarantees $\ell = 1$) in the special case that $M = 1$ (i.e., we consider sets and not multisets), it is not necessary to use the mapping T described above over $\mathrm{GF}(Q)$. Instead, it is possible to only store the parity of the number of elements mapped to each entry of the codeword. This enables us to reduce the codeword size to $O(K \cdot \log(N/K))$ and the update time to $O(\log K \cdot \log N)$.

3. THE EQUALITY TESTING PROTOCOL

The incremental encoding scheme described in Section 2 serves as our main tool in designing an equality testing protocol in the adversarial sketch model.

First attempt. In the sketch phase, Alice and Bob incrementally encode the sequences of insert and delete operations they receive (using the encoding scheme from Section 2 with parameter $\rho = 1$) and obtain the encodings $C^A = \{C_0^A, \ldots, C_{k+1}^A\}$ and $C^B = \{C_0^B, \ldots, C_{k+1}^B\}$ of their multisets S_A and S_B, respectively. In the interaction phase, the honest parties compare a random entry from each of the codewords. More specifically, Alice picks $k+2$ uniformly distributed entries $y_0, \ldots, y_{k+1}$ and sends the values $(y_0, C_0^A[y_0]), \ldots, (y_{k+1}, C_{k+1}^A[y_{k+1}])$ to Bob. Bob compares these entries to the corresponding entries of his codewords, and if $C_i^B[y_i] = C_i^A[y_i]$ for every $0 \leq i \leq k+1$ then they output `Equal`. Otherwise, they output `Not Equal`.

Lemma 2.3 guarantees that the protocol is perfectly complete. That is, any two sequences of insert and delete operations that lead to the same multiset of elements induce the

	Generic parameters	**Non-Explicit**	**Explicit**
Codeword size	$\ell \log N \cdot \sum_{i=0}^{k+1} \lvert R_i \rvert$	$O(K \cdot \log^2 N)$	$K \cdot \mathrm{polylog}(N)$
Update time	$\ell \log N \cdot \sum_{i=0}^{k+1} D_i$	$O(\log K \cdot \log^2 N)$	$\mathrm{polylog}(N)$

Table 2: The codeword size and update time of the encoding scheme.

same representation, and therefore the parties will always output `Equal`. Lemma 2.4 guarantees the soundness of the protocol. That is, if Alice and Bob are given two sequences of operations that do not lead to the same multisets, then there exists an integer $0 \le i \le k+1$ for which the codewords C_i^A and C_i^B differ on at least $(1-\epsilon)$-fraction of their entries (recall that this is the distance property guaranteed by the encoding scheme). In this case the parties output `Not Equal` with probability at least $1-\epsilon$.

The protocol. A drawback of the above protocol is that the size of the sketches (i.e., the size of the encodings C^A and C^B) is not sublinear in the size of the input sets S_A and S_B. To overcome this undesirable property, we modify the protocol as follows: The parties view each codeword C_i (corresponding to C_i^A and to C_i^B) as a square matrix. Prior to the sketch phase, Alice chooses uniformly distributed rows $a_0, \dots, a_{k+1}$ and Bob chooses uniformly distributed columns $b_0, \dots, b_{k+1}$. During the sketch phase Alice only stores and updates the entries of each codeword C_i^A along row a_i, and Bob only stores and updates the entries of each codeword C_i^B along column b_i. Notice that each party now stores only a square-root of the entries of each codeword, and this leads to sketches which are of size $\widetilde{O}(\sqrt{K})$. In the interaction phase, for every $0 \le i \le k+1$ Alice and Bob compare the entry at the intersection of row a_i and column b_i in the codewords C_i^A and C_i^B, and output `Equal` if and only if they all match.

Amplifying the success probability. The protocol described above guarantees that if Alice and Bob are given two sequences of operations that do not lead to the same sets, then they both output `Not Equal` with probability at least $1-\epsilon$ (where ϵ in our constructions of bounded neighbor-dispersers is some constant). More generally, given an accuracy parameter $0 < \delta < 1$, we would like to amplify the probability that they output `Not Equal` to $1-\delta$. Naively, this can be achieved by having Alice and Bob compare $s = O\left(\frac{1}{1-\epsilon}\log\frac{1}{\delta}\right)$ uniformly and independently chosen entries of each codeword. This will require each party to store and update s rows or columns of each codeword, resulting in sketches of size $\widetilde{O}(\sqrt{K}\cdot\log(1/\delta))$. We now show that it is possible, however, to reduce this amount to only $\widetilde{O}\left(\sqrt{K\cdot\log(1/\delta)}\right)$. We partition each codeword C_i (corresponding to C_i^A and to C_i^B) to s disjoint parts of roughly equal size (the partition is part of the description of the protocol, and is known to the adversary in advance). Alice and Bob will compare a random entry from each such part. This will require each of them to store and update only $O(\sqrt{|C_i|/s})$ entries from each part, and therefore a total of $O(\sqrt{|C_i|s})$ entries from each codeword. This turns out to have essentially the same effect as comparing s independently chosen entries from each codeword, and results in sketches of size $\widetilde{O}\left(\sqrt{K\cdot\log(1/\delta)}\right)$. A formal proof of the properties of the protocol is provided in the longer version of this paper.

4. SYMMETRIC DIFFERENCE APPROXIMATION

We present a protocol for approximating the size of the symmetric difference of two massive data sets in the adversarial sketch model. We note that the protocol can be applied also when the inputs of the parties are multisets, and in this case it approximates the number of distinct elements in the symmetric difference.

The approximation ratio of our protocol depends on the properties of the graphs $G_0, \dots, G_{k+1}$ used to construct the incremental encoding scheme in Section 2, and stating the result requires introducing the following notation. Recall that each $G_i = (L = [N], R_i, E_i)$ is a $(K_i, \epsilon, \rho, \ell)$-bounded-neighbor disperser with left-degree D_i. We let

$$r = \max_{0 \le i \le k+1} \frac{K_i D_i}{(1-\epsilon)|R_i|} \ .$$

Theorem 5.1 shows that such graphs exist with $r \approx 1$, and Corollary 5.4 provides an explicit construction with $r = \mathrm{polylog}(N)$.

The protocol. In the sketch phase, Alice and Bob incrementally encode the sequences of insert and delete operations they receive (using the encoding scheme from Section 2 with parameter ρ), and obtain the encodings $C^A = \{C_0^A, \dots, C_{k+1}^A\}$ and $C^B = \{C_0^B, \dots, C_{k+1}^B\}$ of their multisets S_A and S_B, respectively. In the interaction phase, the parties compare $s = \Theta\left(\frac{1}{(1-\epsilon)\rho^2}\log\frac{\log_{1+\rho}K}{\delta}\right)$ independently chosen and uniformly distributed entries from each of the $k+2$ codewords. For every $0 \le i \le k+1$ denote by d_i the number of differing entries out of the s entries that the parties compare from the i-th codeword. The parties output $\Delta_{\mathtt{APX}} = (1+\rho)^{i+1}$ for the maximal i for which $d_i \ge \frac{1}{2}\left(1+\frac{1}{1+\rho}\right)(1-\epsilon)s$. If there is no such i, then the parties output $\Delta_{\mathtt{APX}} = 0$.

In order to reduce the size of the sketches, we observe once again that the parties are not required to store and update their entire codewords. The parties view each codeword C_i (corresponding to C_i^A and to C_i^B) as a square matrix, and store and update only the entries that correspond to s random rows or columns from each codeword. The following lemma states the approximation guarantee, and a formal proof is provided in the longer version of this paper.

Lemma 4.1. *For any two sequences of insert and delete operations communicated to the parties that lead to multisets with symmetric difference that contains $\Delta_{\mathtt{OPT}}$ distinct elements, the parties output $\Delta_{\mathtt{APX}}$ such that*

$$\Pr\left[\Delta_{\mathtt{OPT}} \le \Delta_{\mathtt{APX}} \le (1+\rho)^3 r \Delta_{\mathtt{OPT}}\right] > 1-\delta \ .$$

5. BOUNDED-NEIGHBOR DISPERSERS

Given N, K and ρ we are interested in constructing a $(K, \epsilon, \rho, \ell)$-bounded-neighbor disperser $G = (L = [N], R, E)$, such that ϵ, ℓ and $|R|$ are minimized. A rather standard argument shows the existence of a bounded-neighbor disperser with essentially optimal parameters: a constant ϵ, $\ell = 1$ and $|R| = O(K \log(N/K))$. We prove the following theorem:

THEOREM 5.1. *For every N, $K \leq N$ and constant $0 < \rho \leq 1$, there exists a $(K, \epsilon = 1 - \frac{\rho}{(1+\rho)^4}, \rho, \ell = 1)$-bounded-neighbor disperser $G = (L, R, E)$, with $|L| = N$, left-degree $D = O(\log(N/K))$, and $|R| = \left\lceil \frac{(1+\rho)^2}{\rho} KD \right\rceil$.*

As for explicit constructions, we show that any disperser [37] is also a bounded-neighbor disperser for some parameters[13]. Once again, we emphasize the importance of basing our protocols on dispersers and not on extractors: Whereas the existing explicit constructions of extractors are rather far from optimal, the existing explicit constructions of disperser are optimal up to poly-logarithmic factors. This yields a significant performance differences.

Dispersers are combinatorial objects with many random-like properties. Dispersers can be viewed as functions that take two inputs: a string that is not uniformly distributed, but has some randomness; and a shorter string that is completely random, and output a string whose distribution is guaranteed to have a large support. Dispersers have found many applications in computer science, such as simulation with weak sources, deterministic amplification, and many more (see [35] for a comprehensive survey).

DEFINITION 5.2. *A bipartite graph $G = (L, R, E)$ is a (K, ϵ)-disperser if for every $S \subseteq L$ of size at least K, it holds that $|\Gamma(S)| \geq (1 - \epsilon)|R|$, where $\Gamma(S)$ denotes the set of neighbors of the vertices in S.*

LEMMA 5.3. *Any (K, ϵ)-disperser $G = (L, R, E)$ with left-degree D is a $(K, \epsilon', \rho = 1, \ell)$-bounded-neighbor disperser, for $\epsilon' = \frac{1+\epsilon}{2}$ and $\ell = \left\lceil \frac{4DK}{(1-\epsilon)|R|} \right\rceil$.*

Lemma 5.3 can be instantiated, for example, with the disperser construction of Ta-Shma, Umans and Zuckerman [38]. In this case we obtain the following corollary:

COROLLARY 5.4. *For every n and k there exists an efficiently computable $(K = 2^k, \epsilon = 3/4, \rho = 1, \ell)$-bounded-neighbor disperser $G = (L, R, E)$, with $|L| = N = 2^n$, $|R| = \Theta(K/\log^3(N))$, left-degree $D = \text{polylog}(N)$ and $\ell = \text{polylog}(N)$.*

6. CONCLUDING REMARKS

Relying on computational assumptions. We considered the adversarial sketch model in an information-theoretic setting (i.e., we did not impose any restrictions on the computational capabilities of the adversary). In any realistic setting, however, it is reasonable to assume that the adversary is polynomially bounded . It will be interesting to explore whether computational assumptions can significantly improve the efficiency of protocols in the adversarial sketch model. For example, the existence of incremental collision resistant hash functions [9, 10] implies an equality testing protocol with highly compressed sketches which dramatically circumvents the lower bound stated in Theorem 1.2 in the computational setting. A major drawback of existing constructions of such hash functions is that they either rely on a random oracle, or are inefficient (more specifically, the construction of Bellare, Goldreich and Goldwasser [9] can be proved secure without a random oracle, but in this case the size of the description of each hash function is too large to be used in practice – linear in the number of input blocks)[14].

In addition, for the problem of approximating the size of the symmetric difference we are not aware of any protocol in the computational setting that similarly improves our protocol. It would be very interesting to take advantage of computational assumptions and construct such a protocol with highly compressed sketches.

Preserving sublinearity and efficiency. As discussed in Section 1, the most natural question that arises in the context of the adversarial sketch model is to characterize the class of functions that can be computed or approximated in this model with sublinear sketches and poly-logarithmic update time, communication and computation. In particular, we have asked whether the adversarial sketch model "preserves sublinearity and efficiency" of problems from the standard sketch model.

In this paper we provided an affirmative answer for the problems of testing whether two massive data sets are equal, and approximating the size of their symmetric difference. It would be interesting to consider other distances and similarity measures that can be efficiently approximated in the standard sketch model (See Section 1.2). For example, an intriguing measure, due to its application in eliminating near-duplicates of web pages is the resemblance measure [14], defined as

$$r(S, T) = \frac{|S \cap T|}{|S \cup T|} \ .$$

There are highly compressed sketches for estimating the resemblance between two sets using a collection of min-wise independent permutations [13]. It is not clear, however, that without shared randomness this technique can result in sketches that can be updated in an efficient incremental manner. It would be interesting to construct an efficient protocol for approximating resemblance in the adversarial sketch model.

Acknowledgments

We thank Ziv Bar-Yossef, Danny Segev, and the anonymous referees for many useful comments.

7. REFERENCES

[1] J. Abello, P. M. Pardalos, and M. G. C. Resende, editors. *Handbook of Massive Data Sets.* Kluwer Academic Publishers, 2002.

[2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. of the ACM*, 51(4):606–635, 2004.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comp. Syst. Sci.*, 58(1):137–147, 1999.

[13] A similar observation was used by Moran et al. [32] who defined the notion of *bounded-neighbor expanders*, and showed that it can be satisfied by any disperser with certain parameters. Our graphs have slightly weaker properties, and this enables more efficient constructions. That is, the parameters of dispersers are better preserved.

[14] An additional construction that does not rely on random oracles can be based on the techniques of Gennaro, Halevi and Rabin [22] that require hash functions that output prime numbers, but this does not result in an efficient construction.

[4] L. Babai and P. G. Kimmel. Randomized simultaneous messages: Solution of a problem of Yao in communication complexity. In *12th CCC*, pages 239–246, 1997.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *21st PODS*, pages 1–16, 2002.

[6] M. Badoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *34th STOC*, pages 250–257, 2002.

[7] Z. Bar-Yossef. *The Complexity of Massive Data Set Computations*. PhD thesis, UC Berkeley, 2002.

[8] Z. Bar-Yossef, T. S. Jayram, R. Krauthgamer, and R. Kumar. Approximating edit distance efficiently. In *45th FOCS*, pages 550–559, 2004.

[9] M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: The case of hashing and signing. In *CRYPTO '94*, pages 216–233, 1994.

[10] M. Bellare and D. Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In *EUROCRYPT '97*, pages 163–192, 1997.

[11] M. Blum, W. S. Evans, P. Gemmell, S. Kannan, and M. Naor. Checking the correctness of memories. *Algorithmica*, 12(2/3):225–244, 1994.

[12] D. Boneh and M. K. Franklin. An efficient public key traitor tracing scheme. In *CRYPTO '99*, pages 338–353, 1999.

[13] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *J. Comp. Syst. Sci.*, 60(3):630–659, 2000.

[14] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.

[15] E. J. Candès and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. on Infor. Theory*, 52(12):5406–5425, 2006.

[16] M. Charikar. Similarity estimation techniques from rounding algorithms. In *34th STOC*, pages 380–388, 2002.

[17] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In *SIROCCO*, pages 280–294, 2006.

[18] D. L. Donoho. Compressed sensing. *IEEE Trans. on Infor. Theory*, 52(4):1289–1306, 2006.

[19] T. Feder, E. Kushilevitz, M. Naor, and N. Nisan. Amortized communication complexity. *SIAM J. Comput.*, 24(4):736–750, 1995.

[20] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright. Secure multiparty computation of approximations. *ACM Trans. on Alg.*, 2(3):435–472, 2006.

[21] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L^1-difference algorithm for massive data streams. *SIAM J. Comput.*, 32(1):131–151, 2002.

[22] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *EUROCRYPT '99*, pages 123–139, 1999.

[23] P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. In *10th SODA*, pages 909–910, 1999.

[24] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. One sketch for all: fast algorithms for compressed sensing. In *39th STOC*, pages 237–246, 2007.

[25] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. of the ACM*, 45(4):653–750, 1998.

[26] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. In *External memory algorithms*, pages 107–118. American Mathematical Society, 1999.

[27] P. Indyk. Explicit constructions for compressed sensing of sparse signals. In *19th SODA*, pages 30–33, 2008.

[28] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *30th STOC*, pages 604–613, 1998.

[29] P. Indyk and D. P. Woodruff. Polylogarithmic private approximations and efficient matching. In *3rd TCC*, pages 245–264, 2006.

[30] I. Kremer, N. Nisan, and D. Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

[31] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000.

[32] T. Moran, M. Naor, and G. Segev. Deterministic history-independent strategies for storing information on write-once memories. In *34th ICALP*, pages 303–315, 2007.

[33] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[34] I. Newman and M. Szegedy. Public vs. private coin flips in one round communication games. In *28th STOC*, pages 561–570, 1996.

[35] N. Nisan and A. Ta-Shma. Extracting randomness: A survey and new constructions. *J. Comp. Syst. Sci.*, 58(1):148–173, 1999.

[36] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

[37] M. Sipser. Expanders, randomness, or time versus space. *J. Comp. Syst. Sci.*, 36(3):379–383, 1988.

[38] A. Ta-Shma, C. Umans, and D. Zuckerman. Lossless condensers, unbalanced expanders, and extractors. *Combinatorica*, 27(2):213–240, 2007.

[39] H. S. Witsenhausen and A. D. Wyner. Interframe coder for video signals. U.S. patent 4,191,970, 1980.

[40] A. C. Yao. Some complexity questions related to distributive computing. In *11th STOC*, pages 209–213, 1979.