Secure Data Retrieval on the Cloud:

Homomorphic Encryption meets Coresets

Adi Akavia*[1], Dan Feldman†[2] and Hayim Shaul‡[3]

[1] University of Haifa, akavia@cs.haifa.ac.il
[2] University of Haifa, dan.feldman@gmail.com
[3] University of Haifa hayim.shaul@gmail.com

Abstract. *Secure report* is the problem of a client that retrieves all records matching specified attributes from a database table at the server (e.g. cloud), as in SQL SELECT queries, but where the query and the database are encrypted. Here, only the client has the secret key, but still the server is expected to compute and return the encrypted result. Secure report is theoretically possible with Fully Homomorphic Encryption (FHE). However, the current state-of-the-art solutions are realized by a polynomial of degree that is at least linear in the number m of records, which is too slow in practice even for very small databases.

We present the first solution that is realized by a polynomial that attains degree independent of the number of records m, as well as the first implementation of an FHE solution to Secure report. This is by suggesting a novel paradigm that forges a link between cryptography and modern data summarization techniques known as coresets (core-sets), and sketches in particular. The key idea is to compute only a coreset of the desired report. Since the coreset is small, the client can quickly decode the desired report that the server computes after decrypting the coreset.

We implemented our main reporting system in an open source library. This is the first implemented system that can answer such database queries when processing only FHE encrypted data and queries. As our analysis promises, the experimental results show that we can run Secure report queries on billions records in minutes on an Amazon EC2 server, compared to less than a hundred-thousands in previous FHE based solutions.

Keywords: Fully Homomorphic Encryption · Secure Computation · Secure Outsourcing of Computation · Secure Search · Coresets · Sketches · Group Testing

1 Introduction

Outsourcing storage and computation to large third-party systems often called "the cloud" (server) has become the norm for organizations and individuals (client). Typically this involves a trust-relationship where the client is required to reveal her private records to the server who executes the computations for her. To avoid this undesirable exposure of her personal, proprietary, or other sensitive information, secure computation techniques [64, 38] offer an appealing alternative: it is feasible for the client and server to jointly compute the desired functionality, while revealing nothing beyond the designated output. A main

*This work was supported in part by the Center for Cyber Law & Policy at the University of Haifa in conjunction with the Israel National Cyber Directorate in the Prime Minister's Office.

†Supported by BSF grant no 2014627. This work was done in part while he was visiting the Simons Institute for the Theory of Computing.

‡Supported by grant no. 2014384 from the U.S.-Israeli Binational Science Foundation. Supported in part by the Bar Ilan Cyber Center.

challenge however in the design of secure computation is to attain low overhead over the un-secure version, in terms of both the communication and computation complexity. This translates to the goal of presenting the computation as a polynomial over a finite field with (a) low-degree and (b) low overall number of multiplications (see e.g. [7]).

Secure report. In this work we focus on secure outsourcing of *data storage and retrieval tasks* – a fundamental computational building block useful in numerous applications such as secure retrieval from a database, a corpus of tagged images, text documents, genomic data, and so forth. In the secure outsourcing context the client is the owner of data, queries, and retrieval outcome. The server is expected to provide the storage and desired data retrieval functionality while learning no (or minimal) information on the client's data, queries and retrieved outcome.

We address the *report* data retrieval functionality, where the data is an unsorted array of elements $array = (x_1, \ldots, x_m)$, the query ℓ corresponds to an agreed predicate $\text{isMatch}(x_i, \ell) \in \{0, 1\}$ (see below), and the retrieved outcome should be the list of (index, value) pairs for all array elements matching the query:

$$\mathcal{L} = \{\, (i, x_i) \mid \text{isMatch}(x_i, \ell) = 1 \,\}\,.$$

Namely, the *report functionality* involves two parties, a client and a server, where the client's input and output are $(array, \ell)$ and $\mathcal{L}$ respectively, and the server has no input or output other than the shared parameters (the security parameter, and upper bounds on the number and size of records, queries and output $\mathcal{L}$).

isMatch Implementation. The predicate $\text{isMatch}(x_i, \ell)$ can be instantiated with any matching criterion, in a black-box fashion, to address versatile retrieval tasks. For example, *exact-match retrieval* is addressed by taking isMatch to be an equality-test (i.e., $\text{isMatch}(x_i, \ell) = 1$ if-and-only-if $x_i = \ell$); *similarity-search* is addressed by taking isMatch to be the predicate accepting 1 if-and-only-if $\text{dist}(x_i, \ell) \leq \tau$ where τ is a threshold, and dist is a distance measure such as the Hamming, Euclidean, or Edit. Likewise, appropriate instantiations of isMatch on encrypted data yield wild-card matching, range queries, Boolean queries, and sub-array queries; See [65, 20, 52, 19, 50, 61, 51, 6]. In this work we address the complementary problem of retrieving the set of matching elements, as defined by the provided isMatch predicate.

For *Secure report*, the server should provide the report functionality obliviously of data, query and outcome. Furthermore, we make the following requirements: the protocol should use a *single-server* in *single-round* with *low communication* proportional only to the size of encrypted input and output, and perform *no pre-processing* of the cleartext data.

Motivating use-cases scenarios for our above requirements arise in settings where either: (a) We cannot assume non-collusion between multiple servers, or do not have availability of multiple servers; (b) Communication is intermittent, unreliable or costly, e.g., when communicating with mostly off-line clients as with sensors networks, posing a limiting factor on the number of communication rounds; (c) Communication is slow or expensive leading to the low bandwidth requirement; (d) it is infeasible to pre-process the cleartext data, e.g., for sorting, indexing or insertion into search-oriented data-structures such as search trees or hash tables. We note that retrieval from an unsorted data necessitates a linear-scan of the data even on cleartext data; yet it naturally arises in many use-cases. Examples for settings disallowing pre-processing the cleartext data follow:

- A-priori unknown matching criterion, as in ad-hoc SQL queries and our generic isMatch predicate;

- Range queries on high dimensional data, where index is infeasible due to size exponential in the number of attributes;
- Streaming data with each element discarded by the client immediately after being encrypted and uploaded to the server;
- Low capacity client too weak to store or sort the clear-text data, as in Internet-of-Things (IoT) devices;
- Fragmented data uploaded to the server from multiple distinct clients (e.g., agents, users, devices) with no single entity that holds entire clear-text data.

In all these examples, pre-processing the cleartext data is infeasible, whether because no single entity holds the entire cleartext data or has the capacity to process it, or because the required index is exponential large, or unknown in advance.

Threat Model. Our security requirement is that the adversary controlling the server learns no new information from participating in the protocol. For example, the adversary must not distinguish between two adversarially-chosen equal size queries ℓ (similarly, data arrays $array$) from participating in the protocol. We note that there is no need to consider adversaries controlling the client because the server has no input or output. We address computationally-bounded semi-honest adversaries, i.e., adversaries that follow the protocol but may try to learn additional information.

Secure report on FHE encrypted data. Protocols satisfying the above requirements are feasible using Fully Homomorphic Encryption (FHE) [57, 33, 34]. FHE is an encryption scheme that enables homomorphically-evaluating polynomials f over encrypted input; that is, given a ciphertext $[\![x]\!]$ encrypting the input x, it is possible to compute the ciphertext $[\![y]\!]$ encrypting the output $y = f(x)$ (with no access to a decryption key that would compromise security).

Secure report protocol using FHE would easily follow by providing a polynomial f realizing the above report functionality, i.e., f s.t.

$$f(array, \ell) = \mathcal{L}.$$

The protocol using f would be as follows. The client first generates public and secret keys for the FHE, keeps the secret key to herself, and sends to the server the public key and encrypted data elements $[\![array]\!] = ([\![x_1]\!], \ldots, [\![x_m]\!])$. The client can then repeatedly issue encrypted retrieval queries $[\![\ell]\!]$ for the server to homomorphically evaluate f on $[\![array]\!]$ and $[\![\ell]\!]$, sends the encrypted outcome

$$[\![\mathcal{L}]\!] = \{\, ([\![i]\!], [\![x_i]\!]) \mid \textsc{isMatch}(x_i, \ell) = 1\}$$

to the client which decrypts it and obtain the desired result $\mathcal{L}$.

Clearly this protocol has a single-server, single round, low bandwidth communication (proportional only to the encrypted input and output, while being independent of the complexity of the report functionality), and involves no preprocessing of the cleartext data. Moreover, security easily follows from the semantic security of the FHE scheme.

The problem is that known polynomials f for the report functionality have high degree $\Omega(m \cdot d)$, for m the number of data records and d the degree of the polynomial realizing the matching condition isMatch; See Appendix A. This is too slow with current FHE candidates and implementations.

The motivation for our work is to answer affirmatively the following question: **Is there an *efficient* protocol for Secure report on FHE encrypted data and query?**

1.1 Our Contribution

We propose a novel paradigm, named *coresets for homomorphic encryption* (CHE), that forges a link between data summarization techniques (known as "coresets" or "sketches") and secure computation on FHE encrypted data: Whereas it is not clear if low-degree polynomials exist for many classic tasks, such as search and report, our paradigm suggests instead for the server to compute a coreset for these problems for the goal of gaining a dramatic reduction in server's complexity, while essentially conserving the communication and client's complexity; See Section 2.

We demonstrate the strength of our CHE paradigm by applying it on the Secure report problem to dramatically reduce the server's complexity: reducing the degree of the polynomial evaluated by the server to be as low as the degree d of the polynomial realizing isMatch (improving over degree $\Omega(m \cdot d)$ in the direct polynomial). We stress that the degree in our solution is independent of the number of data records m. See Section 3.

Elaborating on the above, our Secure report solution improves over the direct polynomial in both the server's degree and overall multiplications by multiplicative factors $\Omega(m)$ and $\Omega(|\mathcal{L}| \cdot m)$ respectively. Conversely, our Secure report exhibit a slight degradation in the communication and client's complexity (by a factor $|\mathcal{L}|$ and $(|\mathcal{L}| + \log m)^{O(1)}$ respectively); this is a minor degradation when the number of matches $|\mathcal{L}|$ is moderate, say, logarithmic in m (whereas for large $|\mathcal{L}|$ transmitting all these matches would typically be undesirable, due to the high communication it entails even without the discussed overhead).

Experimental results. We implemented our Secure report solution in a C++ library building on HElib [43] FHE implementation, and ran extensive experiments. Our experimental results show that we can run our Secure report on *billions* of database records in a minute, on a single Amazon AWS server (compared to less than a hundred-thousand records with the direct polynomial, on the same hardware); See Fig. 1 and Section 5.

Our code is provided for the community in an open source library of Coresets for Homomorphic Encryption (CHElib) [3], to reproduce our experiments, to extend our results for real-world applications, and for practitioners at industry or academy that wish to use these results for their future papers or products.

Future research. We expect our coresets for homomorphic encryption paradigm to be further employed for degree reduction in the context of other secure computation tasks, possibly together with classical degree reduction techniques such as low-degree approximation [55, 59] and randomized polynomials [48]. Indeed, our paradigm was already employed in a follow-up work [5] to solve the secure search problem of returning the first-match (i.e., returning (i, x_i) for $i = \min\{\, i \in [m] \mid \text{isMatch}(x_i, \ell) = 1\}$), motivated by use-cases where the number of matches to report is too large.

1.2 Related Works

Secure search has been extensively studied employing a variety of cryptographic tools leading to solutions with versatile properties.

Most relevant scenario: Secure search on FHE encrypted data. The scenario most relevant to our settings is secure search on FHE encrypted data [34], when focusing on protocols with single-server, single-round, low-communication proportional only to the size of encrypted input and output, and with no pre-processing on cleartext data. We note that disallowing pre-processing necessitates a linear scan of the data, even if we were to compute on cleartext data and query.

Private Information Retrieval (PIR) [22] enables a client to retrieve a data item from a server holding a data array, while hiding from the server the client's query, the retrieved

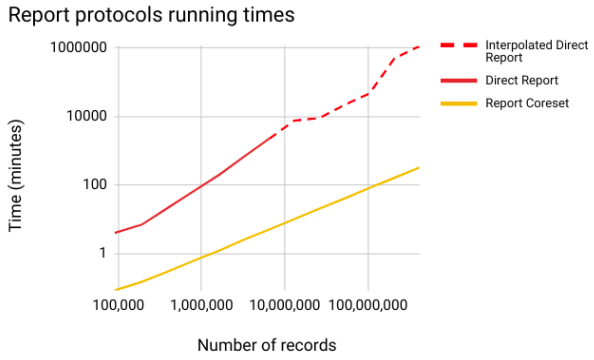

Figure 1: A **log-scaled** graph showing the server's running time (y-axis) on a single machine of Amazon's cloud, for different vector size (x-axis) of Secure Report (Protocol 3) over encrypted vector. In these experimets the vectors had 40 matching elements to be reported. The yellow line marks the running time of our report algorithm. The red line marks the running time of the direct report algorithm, where a solid line marks experiements that were made and a dashed line marks our interpolation based on our analysis.

data item, and the query and access patterns. The PIR query is typically the item's index $i \in [m]$. Nevertheless, constructions of PIR on FHE encrypted data [33, 11, 30] are easily extended for retrieving encrypted keywords provided the keyword *uniquely identifies* at most a single data item (uniqueness constraint). In contrast, in our work we do not require the query to be a unique identifier to a single data item.

Secure search on encrypted data, eliminating the aforementioned uniqueness constraint, was recently achieved [5], providing secure search on FHE encrypted hiding the content of data items and queries, as well as the query and access patterns. In [5] the retrieved item is the *first matching item* in the array; whereas retrieving subsequent items requires further interaction, another round for each fetch-next item to be retrieved. In contrast, in our work we retrieve all matches in a single round.

Other scenarios: relaxed settings. Numerous other related works on secure search are relevant, if relaxing the above settings to allow, for example, leaking information on the access or search patterns, searching on cleartext data, $k > 1$ servers, $R > 1$ rounds, communication bandwidth growing with database size m or with complexity of the computed functionality, allowing pre-processing the cleartext data, etc.

When allowing information leakage, as in leaking access and query patterns or revealing order, searchable and structured encryption (SE and STE) [60, 16] allow for highly efficient construction approaching plaintext search time. Furthermore, these constructions typically allow pre-processing the cleartext data for achieving sub-linear search time. Yet, the inherent information leakage has been often exploited to obtain the content of query and data items [14, 66, 49, 1, 36, 40, 41, 54].

When searching on unencrypted data (in contrast to encrypted data in our settings), indexing can be used to enforce uniqueness to allow using the aforementioned PIR on FHE encryption data protocols [15, 58]; this indexing however incurs considerable time and memory overhead. Furthermore, if allowing pre-processing of cleartext data on top of searching on unencrypted data, then we could employ PIR-by-Keywords [21] techniques to get sub-linear complexity.

When allowing $k > 2$ non-colluding servers, fast private queries on public (unencrypted) data systems were presented in [8, 62].

When allowing $R > 1$ rounds, $\log m$ communication overhead, and a client maintaining and updating state – Oblivious RAM (ORAM) [37] enable hiding data access patterns with sub-linear server time (breaking the linear scan barrier of our settings).

When allowing communication bandwidth to grow with the time to compute the search functionality, classical "pre-FHE" secure two-party computation (2PC*) techniques can be employed, such as garbled circuits and secret sharing [64, 39].

Likewise, when allowing communication bandwidth to grow with the number of records m, secure pattern matching (SPM) shows how to efficiently compute isMatch on FHE encrypted data item. Query [65, 20, 52, 19, 61, 51] can be employed by the server to compute the encrypted length m indicator vector indicating the desired matches, and transmitting this entire vector of length m to the client. This is clearly impractical for large databases where m is large.

Table 1: Comparison to related works for secure data retrieval on FHE encrypted data with single-server, single-round protocols, and no pre-processing of cleartext data. Rows correspond to approaches: Secure Pattern Matching [65, 20, 52, 19, 61, 51] (SPM), Private Information Retrieval [11, 30, 15, 58](PIR), Secure Search [5], the Direct polynomial (see Appendix A), and our Report-Coreset approach (CHElib). Columns correspond to desired properties: sub-linear communication and client's complexity, low degree for the polynomial homomorphically-evaluated by the server, handling query without uniqueness constraint, returning all matches (rather than a single match), and returning both index and record (rather than only the index). The parameter d is the degree of the specified matching predicate isMatch; and m is the number of records in the data array.

Approaches	Sub-linear Client	Server's degree	Non-unique Identifiers	Returns All Matches	Returns Records
SPM	×	d	✓	✓	×
PIR	✓	d	×	×	✓
Secure Search	✓	$d \cdot \log^3 m$	✓	×	✓
Direct method	✓	$d \cdot m$	✓	✓	✓
CHELib	✓	d	✓	✓	✓

2 New Paradigm: Coresets for Homomorphic Encryption

In this section we give an overview of how we propose to employ coresets, sketches and group-testing for reducing the complexity of computing on FHE encrypted data.

Coreset is a data summarization C of a set P of items (e.g. points, vectors or database records) with respect to a set Q of queries (e.g. models, shapes, classifiers, points, lines) and a loss function f, such that $f(P, q)$ is approximately the same as $f(C, q)$ for every query $q \in Q$. The goal is to have provable bounds for (i) the size of C (say, $1/\varepsilon$), (ii) the approximation error (say, $\varepsilon \in (0, 1)$) and (iii) construction time of C given (P, Q, f). We can then run (possibly inefficient) existing algorithms and heuristics on the small coreset C, to obtain provably approximated solution for the optimal query (with respect to f) of the original data.

Sketches are a special type of coresets, where given an $m \times d$ matrix P and a "fat", $s \times m$ matrix $A_{s,m}$, with $s \ll m$, the result $C = A_{s,m}P$ is an $s \times d$ ("sketch") vector. Many problems can be solved on the sketch C instead of the long vector P, by designing a corresponding (sketch) matrix $A_{s,m}$. For example, if the entries of $A_{s,m}$ are random

standard Gaussian variables, then we can approximate the k-means problem on C by the Johnson-Lindenstrauss Lemma; see [9]. In this paper we use sketches in the context of compressed sensing, or more precisely, Group Testing [31].

The coreset is a paradigm in the sense that its exact definition, structure and properties change from paper to paper. The term "coreset" was coined by Agarwal, Har-Peled, and Varadarajan [2] and originally used for optimization in computational geometry (e.g. [53, 23, 13]). Since then coresets were used in fields such as machine learning (e.g.[45, 12]), numerical algebra [63], graph theory (e.g. [29]), time series [46, 56, 35] streaming [26, 32, 24, 27] and pattern recognition [17, 25, 42, 28, 44].

Coresets for Homomorphic Encryption (CHE). In this paper we suggest a new paradigm that is inspired by the coreset paradigm, but for a very different application and context. Instead of running the complete secure algorithm on the server side and encrypted data, the server computes only (encrypted) coresets for the problem at hand and sends them back to the client. Since the coresets are small, communicating them to the client, decrypting them, and decoding the desired result from the coreset is relatively fast and require only little additional time on the client side.

In this paper we introduce the first two CHEs (cf. Fig. 2): one for reporting non-zero values in an array, and one for reducing the ring size from r to $O(\log r)$; see Section 4. These two coresets are then combined to give our secure solution to the report problem; see Theorem 1. The report coreset uses a sketch matrix to communicate the sparse binary vector of the reported indices to the client.

The key property that we show in this paper is that computing coresets on the server side securely (i.e., via polynomials) dramatically reduces its computation by reducing the degree of the polynomial from at least linear to logarithmic on the input size (CRT coreset) or even constant (report coreset).

Informally, in this paper a function S is an efficient coreset construction scheme for a problem if it has the following pair of properties.

(i) $S(P) \in \mathbb{R}^k$ is a short vector that can be computed efficiently. Usually $k = (\log m)^{O(1)}$ for an input vector P of length m.

(ii) $S(P)$ is a coreset or a "sketch" vector in the sense that the desired information, in a problem dependent sense, can be extracted efficiently and exactly from $S(P)$, without having access to (the large) original vector.

Another significant difference from traditional coreset and sketches papers is that the coresets in this paper are exact in the sense that the data reduction does not introduce any additional error ε.

Group Testing meets FHE for report coreset. Our report coreset in Protocol 3 uses a modern result from the field of Group Testing [31]. To our knowledge, this is the first application of Group Testing to FHE. The motivation is to return to the client the very large (m bits) but sparse binary (indicator) vector χ of the desired s indices that were computed in Line 2a of our report coreset (Protocol 3) by communicating a number of bits that is only poly-logarithmic in m. Specifically, entries $\chi(i) = \textsc{isMatch}(x_i, \ell)$ indicate whether data elements x_i match the query ℓ.

The server however sees only the ciphertexts $[\![\chi(i)]\!]$, encrypting the indicators $\chi(i)$, that the server obtains by homomorphically-evaluating the isMatch polynomial on encrypted elements $[\![x_i]\!]$ and query $[\![\ell]\!]$. Since the vector χ is encrypted, an efficient algorithm must be realized by a low-degree polynomial, so simple comparison of each entry to 1 is impossible by such a polynomial. Instead, Group Testing suggests to extract the set of 1's entries in χ from a small number of predefined sums on subsets entries in χ.

Our Report Coreset multiplies the (encrypted) indicator vector χ by a (sketch) matrix $A_{s,m}$ of size $k \times m$ where k is polynomial in s and $\log m$, i.e., we compute few *linear* combinations $A_{s,m} \cdot \chi$ of the entries of χ. In fact, since the sketch matrix $A_{s,m}$ is public, computing these linear combinations requires only an additively homomorphic encryption. Indyk, Ngo and Rudra [47] suggested such a binary matrix $A_{s,m}$ with $k = O(s^2 \log m)$ rows, so that our χ vector can be decoded from $A_{s,m} \cdot \chi$ in time that is also polynomial in s and $\log m$ on the client side.

Unfortunately, unlike χ, the vector $A_{s,m} \cdot \chi$ is not a binary vector; moreover, correct decoding of the sketch assumes this product is computed over the reals. In contrast, in the context of FHE, computation is typically within a finite ring, e.g., computation modulo p for a prime p, and not over the reals.[1] Taking a sufficiently large p (e.g., $p > m$ if $A_{m,s}$ is binary), would have resolved the correctness issue, albeit introducing an undesirable complexity increase, as larger p leads to a general slowdown in the homomorphic operations as well as size inflation of the keys and ciphertexts. To aggravating the problem further, if $\textsc{isMatch}(x_i, \ell)$ uses Fermat's Little Theorem (e.g., for testing equality), then the degree of the polynomial for computing χ in Line 2a of Protocol 3 grows at least linearly with p, resulting in a high-degree polynomial.

Our CRT Coreset offers a solution that guarantees to resolve both the correctness and complexity issues by executing the homomorphic evaluations in multiple rings (in parallel), where the CRT coreset is the tuple of results in all the rings. Our analysis follows from the Chinese Remainder Theorem (CRT) showing that computing modulo $O(\log m)$ small primes $p = \widetilde{O}(\log m)$, not only guarantees the efficiency and low-degree of the polynomials we evaluate, but also ensures that the correct outcome can be efficiently decoded from the CRT coreset.

In addition, the value s of the sparsity of χ is required to compute the sketch matrix $A_{s,m}$ but unknown to the server. These and other challenges are handled by the algorithms of our system.

3 Problem Statement and Main Result

In this section we formally define the Secure report problem and state our main theorem.

3.1 The Secure Report Problem

Our main example application for using coresets in the service of FHE is to solve the following report queries problem efficiently on a secure database.

For simplicity of the presentation, we first address the problem of reporting the list $\mathcal{I}$ consisting of all *indices* $i \in \{1, \ldots, m\}$ for the matching elements, i.e., all x_i so that $\textsc{isMatch}(x_i, \ell) = 1$. The extension for reporting the list $\mathcal{L}$ of index and value pairs (i, x_i), for $i \in \mathcal{I}$, is simple and requires no increase in the round complexity; see Section 4.2.2.

We focus here on key components of the definition, deferring to Section 3.3 some implementation details regarding the input representation and compatibility requirements for the $\textsc{isMatch}$ polynomial and the FHE.

Definition 1 (Report problem)**.** Let m be a positive integer, let $\mathcal{M}$ and $\mathcal{Q}$ the domains of data-elements and queries respectively, $\textsc{isMatch}\colon \mathcal{M} \times \mathcal{Q} \to \{0, 1\}$ a polynomial specifying the matching condition. In the *Report problem*, on input $array = (x_1, \ldots, x_m) \in \mathcal{M}^m$ and query $\ell \in \mathcal{Q}$, the output is the set $\mathcal{I}$ of all entries in $array$ matching ℓ:

$$\mathcal{I} = \{\, i \in \{1, \ldots, m\} \mid \textsc{isMatch}(x_i, \ell) = 1 \,\}.$$

[1] An exception is a recent FHE scheme computing over the reals, albeit inherently introducing rounding errors in each computation [18].

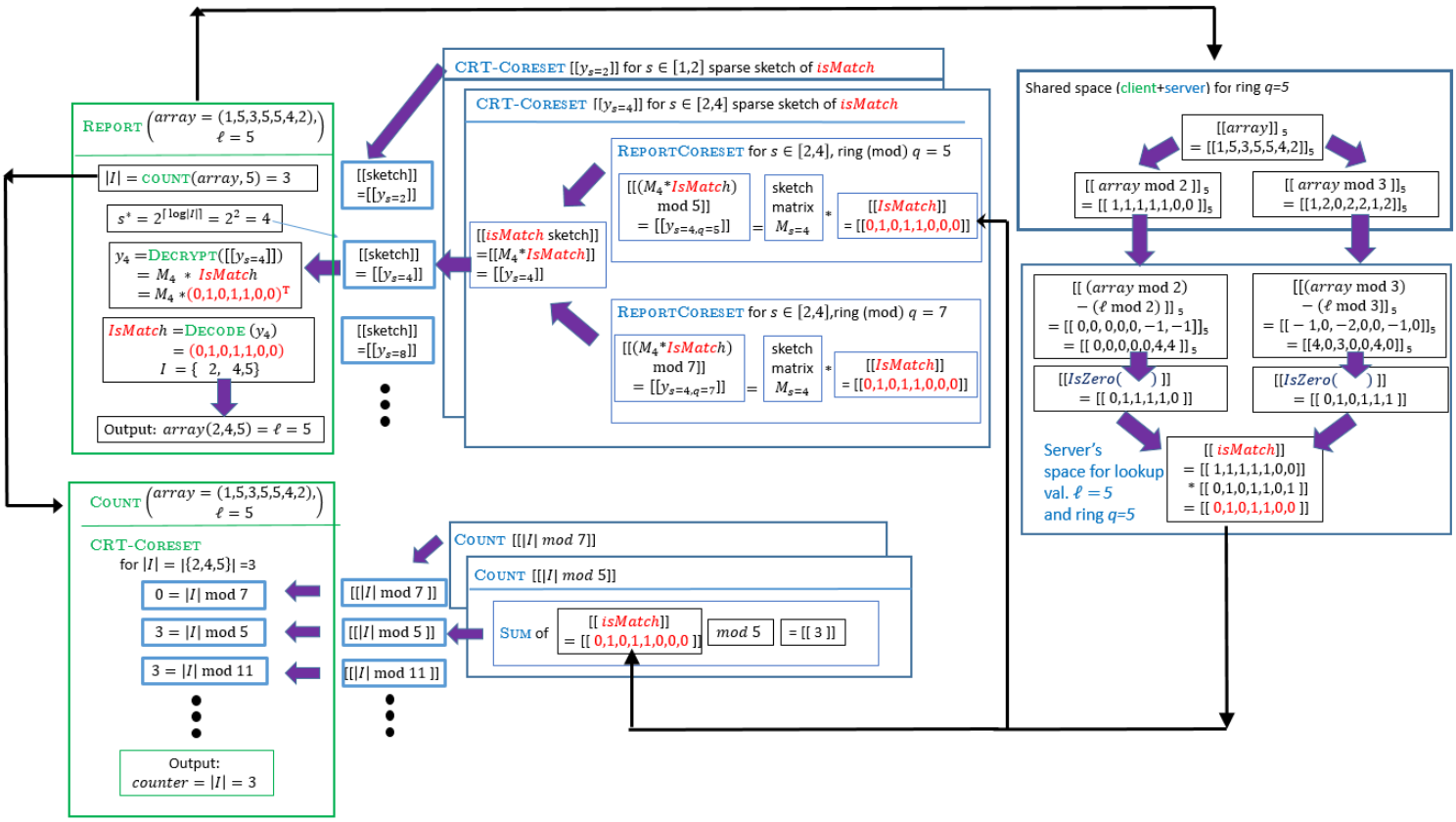

Figure 2: Overview of the suggested system for reporting the indices $I = \{2, 4, 5\}$ of the lookup value $\ell = 5$ in the array $(1, 5, 3, 5, 5, 4, 2)$. Green and blue titles correspond to algorithms that run on the client's and server's side, respectively. The rightmost boxes represent encrypted saved versions of the array. After the client calls REPORT on the right, a version is computed from the shared memory (top right) to each of the few ring values (only $q = 5$ is shown). ISMATCH (bottom right) contains binary results for each ring. The client computes the number $|I|$ of occurrences (bottom left) of ℓ in the array, by applying the CRT-CORESET on the COUNT results that are computed by the server. It then computes the closest power of 2 for $|I|$ (i.e., $s^* = 4$), decrypts the corresponding sketched vector ($y_s = 4$), and ignores the other sketches. The client then computes I efficiently from y_s. Each sketch vector is computed by the server using CRT-Coreset which computes, for every ring's size q, its REPORTCORESET (in the middle) that uses ISMATCH $= array - (\ell, \cdots, \ell)$ for q (bottom right).

In the context of our coreset for FHE paradigm, we relax the report problem to allow outputting a short sketch for $\mathcal{I}$, named, *report-coreset*, on which the client applies an efficient decoding algorithm to obtain $\mathcal{I}$. We assume an upper-bound $s \in [0, m]$ on the number of matches is given; See Section 4.2.3 for treatment of the case s is unknown. Looking ahead, for efficiency we'd like the report-coreset to be short and the decoding time to be efficient, specifically, we'd like them to be polynomial in s and $\log m$.

Definition 2 (Report coreset)**.** Let m, $array$, ℓ and $\mathcal{I}$ be as in Definition 1. Let $s \in [0, m]$ be an upper bound on the number of matches $|\mathcal{I}|$. A vector $y \in \mathbb{Z}^k$ is an *s-report coreset* for $(array, \ell)$ if, given only y, we can decode (compute) the set $\mathcal{I}$.

The secure-report problem is the report-problem where the input $(array, \ell)$ and report-coreset output y are replaced by their encryption.

Definition 3 (Secure-report coreset problem)**.** Let m, $array$, ℓ, $\mathcal{I}$, s and y be as in Definition 2. In the *secure-report problem*, on input ciphertexts $([\![array]\!], [\![\ell]\!])$, the output is a ciphertext $[\![y]\!]$ for an s-report coreset y for $(array, \ell)$.

Here $[\![array]\!] = ([\![x_1]\!], \ldots, [\![x_m]\!])$ is the entry-by-entry encryption of the data $array$, and $[\![\ell]\!]$ is the encryption of the query ℓ, where encryption is with an FHE scheme specified as parameters to the problem.

A secure-report protocol via coresets. The usage scenario is that the client first generates keys and uploads the encrypted data $[\![array]\!]$ to the server; the client can then issue repeated encrypted report queries $[\![\ell]\!]$, for which the server homomophically computes and sends to the client the s-report coreset $[\![y]\!]$; the client can then decrypt and decode to obtain the desired outcome $\mathcal{I}$.

In details, for m, $array$, ℓ, $\mathcal{I}$, s and y as in Definition 2, the protocol for Secure report via the secure-report coreset involves two parties, a client and a server with shared input parameters (see below); a client's input $(array, \ell)$ for $array$ a length m array of data-elements and ℓ a query (or lookup value); the client's output the list of matching elements $\mathcal{I}$; and no input or output for the server. The steps of the protocol are as follows First, in an initialization and data upload phase, the client generates keys for the FHE specified in the shared parameter, encrypts $array$ entry-by-entry, and uploads the corresponding ciphertexts $[\![array]\!]$ to the server. Next, in a search-phase, the client can repeatedly encrypt report queries ℓ and send the encrypted queries $[\![\ell]\!]$ to the server. The server then homomophically computes s-report coreset $[\![y]\!]$, and sends it to the client. The client then decrypts and decodes to obtain the desired outcome $\mathcal{I}$.

The shared input parameters for our protocol (Protocol 3) are λ, $\mathcal{E}$, p, $\mathcal{M}$, $\mathcal{Q}$, isMatch, m and s defined as follows: λ is the security parameter, p the smallest prime larger than m, $\mathcal{E} = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}, \mathrm{Eval})$ an FHE scheme that can compute homomorphic evaluations in $GF(p)$, $\mathcal{M}$ the data elements space, $\mathcal{Q}$ the queries-space, isMatch a polynomial realizing the matching criterion isMatch: $\mathcal{M} \times \mathcal{Q} \to \{0, 1\}$, m the number of data elements (for an upper-bound pad with zeros), and s an upper-bound on the number of matching data elements.

In our extension for the case that an upper bound s on the number of matches is unknown (Section 4.2.3), we omit s from the shared parameters.

In our optimization utilizing our CRT-coreset to reduce the ring size for the homomorphic-evaluations from linear to logarithmic in m (Section 4.2.2), we replace the large prime p by a small set $\mathcal{P} = \{p_1, \ldots, p_\kappa\}$ of small primes. Specifically, we set $\mathcal{P}$ to consist of the first κ primes larger than $\log_2 m$ for $\kappa = \lceil \log m / \log\log m \rceil$.

Remarks and extensions. For simplicity of the presentation we assume the entire data $array$ is uploaded in a single execution of the protocol, together with the keys generation. This can be easily modified. The keys can be generated at a separate time, possibly by a separate entity distributing the public key to the clients who will upload the data (data-sources) and to the server, and distributing the secret key to the clients who will issue search queries (search-clients). The upload can occur over time and from multiple data-sources, gradually uploading new encrypted data records for the server to append to its array of ciphertexts.

The question motivating this work, which we affirmatively answer is as follows:

> *Can we solve the secure-report problem via a server homomorphically evaluating of a polynomial of degree independent of the number of data elements m, and where the communication and client's is complexity sub-linear in m?*

3.2 Main Theoretical Result

The main result of this paper is the first solution for the Secure report-coreset problem on FHE encrypted data (see Definition 3) via polynomial of degree independent of the number m of elements in the data *array*. We employ this polynomial to give a secure-report protocol via coreset, in a single-server, single-round protocol with no pre-processing of cleartext data (no sorting, indexing, or search oriented data-structures are used). The protocol attains our complexity goals: The server homomorphically evaluates a polynomial of degree independent of the number m of elements in the data *array*, and the communication and client's complexity is sub-linear (in fact, poly-logarithmic) is m and quadratic in the upper-bound s on the number of matches.

Theorem 1 (basic settings, see extension below)**.** *There exists a protocol for secure-report via coresets satisfying the following, when executed on client's input array* $= (x_1, \ldots, x_m) \in \mathcal{M}^m$ *and* $\ell \in \mathcal{Q}$*, and shared parameters* $(\lambda, p, \mathcal{E}, \mathcal{M}, \mathcal{Q}, \text{isMatch}, m, s)$*, as defined in Section 3.1:*

- Structural requirements: *The protocol is a single-server, single-round protocol, with no preprocessing on cleartext data.*
- Correctness: *If* $|\mathcal{I}| \leq s$*, then the client's output is* $\mathcal{I} = \{\, i \mid \text{isMatch}(x_i, \ell) = 1\}$.
- Complexity: *The complexity of the protocol is as follows, when denoting by* d, μ *the degree and overall multiplication in the* isMatch *polynomial specified in the shared parameters:*
  - *The server homomorphically-evaluates over* $GF(p)$ *a polynomial of degree* d *and overall multiplications* $m \cdot \mu$ *(for* p *the smallest prime larger than* m*).*
  - *The communication consists of* $|\ell|$ *ciphertexts sent from the client and* $O(s^2 \log m)$ *ciphertexts sent from the server (for* $|\ell|$ *denoting the representation length of* ℓ*).*
  - *The client's running-time is polynomial in* $|\ell|$*,* λ*,* s *and* $\log m$.
- Security: *The protocol is secure against semi-honest adversary controlling the server.*

Our subsequent protocols offer some extension and optimizations and tradeoffs on the above theorem, as detailed next.

3.2.1 Extensions and Optimizations

The above theorem states our basic result that can be optimized and extended in several forms.

Outputting $\mathcal{L}$. The output can be extended to consist of (index, value) pairs for all matches, i.e., outputting

$$\mathcal{L} = \{\, (i, x_i) \mid \text{isMatch}(x_i, \ell) = 1\}\,.$$

This extension increased the client's run-time, communication complexity, and number of polynomials evaluated by the server by a factor w, for w the binary representation length of records x_i. There is no degradation however in the structural requirements or the complexity polynomials evaluated by the server.

Reducing the prime to logarithmic in m. The prime p can be reduced to $p = \widetilde{O}(\log m)$ using our CRT-coreset. In this case we essentially repeat our protocol in parallel over distinct fields $GF(p_1), \ldots, GF(p_\kappa)$ for $\kappa = \lceil \log_{\log m} m \rceil$ and $p_1, \ldots, p_\kappa$ the first κ primes larger than $\log_2 m$. This incurs a $\kappa = o(\log_2 m)$ increase on the client's run-time, communication complexity, and number of polynomials evaluated by the server. There is no degradation however in the structural requirements, or the complexity of the polynomials evaluated by the server.

Handling unknown number of matches. If a sufficiently small upper-bound s on the number of matches is unknown we offer two incomparable alternatives.

The first alternative is to add a communication round in which the client and server engage in a secure computation protocol to compute the number of matches $|\mathcal{I}|$, and then executing our Secure report protocol setting the upper-bound s to be at least $|\mathcal{I}|$ (possibly $s > |\mathcal{I}|$ to not reveal the exact count). Computing the number of matches can be done in a single-round protocol, where the server evaluates a polynomial of degree d and overall multiplications $\widetilde{O}(m \cdot \mu)$. Elaborating on the above, if allowing homomorphic evaluations over a large field $GF(p)$ for $p > m$, then counting the number of matches is straightforward (Section 4.2.3). To reduce the field size to logarithmic $p = \widetilde{O}(\log m)$, we propose employing our CRT-coreset (Section 4.2.2).

The second alternative is request that the server homomorphically evaluates the report coreset for $s \in \left\{2^0, 2^1, 2^2, \ldots, m\right\}$ (sending each resulting encrypted coresets as soon as its computation has completed). This does no increase the degree of the polynomials evaluated by the server's, but does have a high overall number of multiplications and servers communication complexity. Nonetheless, this approach might be relevant if the client can abort (preferably, covertly without the server knowing when the client aborts); this results in client's received communication and run-time that are still polynomial in λ, $|\ell|$, $|\mathcal{I}|$ and $\log m$, as desired.

3.3 Details: Compatibility Requirements

We next set some notation (below) and give the technical details on the compatibility requirements from isMatch (Section 3.3.1) and the requirements from the used FHE scheme (Section 3.3.2).

Notations. For a data record z or a lookup value, we denote by $|z|$ the binary representation length of z. For an integer m, we denote $[m] = \{1, \ldots, m\}$. We follow the convention that array indexes start from 1, with entry i denoted $array(i)$. Vectors are column vectors unless stated otherwise. We assume the array size m is a power of two (otherwise pad with zero). Logarithms are in base 2 unless explicitly stated otherwise.

3.3.1 Compatibility Requirements for isMatch

We next elaborate on the compatibility requirements for ensuring that (a) the input representation is compatible to the message space of the FHE scheme, and (b) the provided polynomial for isMatch is compatible with both the input representation and the ring for the FHE evaluations.

The polynomial isMatch specifies the matching criteria by mapping pairs $(x_i, \ell) \in \mathcal{M} \times \mathcal{Q}$ of data element and query to a Boolean value $\text{isMatch}(x_i, \ell) \in \{0, 1\}$ accepting value 1 if-and-only-if the element x_i is a match to the query ℓ. To plug-into our protocol the polynomial isMatch, we make few requirements for compatible with the input representation, compatibility with the underlying FHE, and for security.

Compatibility of isMatch to the input representation mean that data element and query pair (x_i, ℓ) must be in the domain of isMatch. A few examples follow. When the

matching criterion is an equality-test polynomial $\text{isMatch}: \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$ (Section 4.2.5), we require that data elements x_i and queries (or lookup value) ℓ are specified in binary representation of length w. When the matching criterion is an equality-test polynomial $\text{isMatch}: \{0,\ldots,p-1\}^w \times \{0,\ldots,p-1\}^w \to \{0,1\}$ over $GF(p)$ for a prime $p > 2$ (Section 4.2.5), we require that data elements x_i and query (or lookup value) ℓ are specified using w digits in base p. When the matching criterion is a range query, the elements and queries come from different domains, where for example elements may be integer values x_i and queries are a pair of integer endpoints (a, b) so that $\text{isMatch}(x_i, (a, b)) = 1$ if-and-only-if x_i is in the ranger (a, b). In general, for $\mathcal{M}$ and $\mathcal{Q}$ the spaces of data elements and queries respectively, the matching criterion is a function $\text{isMatch}: \mathcal{M} \times \mathcal{Q} \to \{0,1\}$.

Compatibility of isMatch with the underlying FHE means the following. First, the data elements space $\mathcal{M}$ and queries space $\mathcal{Q}$ must be contained in the message space for the FHE so that they can be encrypted. Second, the polynomial provided for isMatch must be over the same ring where the FHE computations are executed so that it can be homomorphically evaluated.

For security, we require that $\mathcal{M}$ consists of equal length data elements and $\mathcal{Q}$ consists of equal length queries, because the encryption does not hide length information; use padding as needed.

3.3.2 Compatibility Requirements for our Black-Box usage of FHE

Our protocols employ a fully (or, leveled) homomorphic encryption (FHE) in a black-box fashion: we require only a black-box usage of the standard algorithms for FHE (key generation, encryption, decryption, and evaluation). The only requirement we make on the scheme is that we can choose as a parameter the *plaintext modulus* to be a prime number p of our choice, so that the homomorphic operations are additions and multiplications modulo p. This is the case in many of the FHE candidates, for example, [10]. For security of our scheme we require that the FHE scheme is semantically secure.

To emphasize the plaintext modulus p we use the following notations for the standard algorithms specifying an FHE scheme $\mathcal{E} = (\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$:

- Gen is a randomized algorithm that takes a security parameter λ as input and a prime p, and outputs a public key $pk_p = (p, pk)$ and a secret key $sk_p = (p, sk)$ for plaintext modulus p, denoted:
$$(pk_p = (p, pk), sk_p = (p, sk)) \leftarrow \text{Gen}(1^\lambda; p).$$
- Enc is a randomized algorithm that takes pk_p and a plaintext message msg, and outputs a ciphertext $[\![msg]\!]_p$ for plaintext modulus p, denoted:
$$[\![msg]\!]_p \leftarrow \text{Enc}_{pk_p}(msg).$$
- Dec is an algorithm that takes sk_p and a ciphertext $[\![msg]\!]_p$ as input, and outputs a plaintext msg', denoted:
$$msg' \leftarrow \text{Dec}_{sk_p}([\![msg]\!]_p).$$
Correctness is the standard requirement that $msg' = msg$.
- Eval is a (possibly randomized) algorithm takes pk_p, a polynomial $f(x_1, \ldots, x_t)$, and a tuple of ciphertexts $([\![m_1]\!]_p, \ldots, [\![m_t]\!]_p)$, and outputs a ciphertext c, denoted:
$$c \leftarrow \text{Eval}_{pk_p}(f, [\![m_1]\!]_p, \ldots, [\![m_t]\!]_p).$$
Correctness is the requirement that decryption would return the message resulting from evaluating (modulo p) the polynomial $f()$ on inputs $m_1, \ldots, m_t$, $\text{Dec}_{sk_p}(\text{Eval}_{pk_p}(f, [\![m_1]\!]_p, \ldots, [\![m_t]\!]_p)) = f(m_1, \ldots, m_t) \mod p$. *Semantic security* implies that the resulting ciphertext c is computationally indistinguishable from a fresh ciphertext $[\![f(m_1, \ldots, m_t)]\!]_p$.

Shorthand notations we use in this writeup are detailed next. (a) We encrypt messages $m = (m_1, \ldots, m_t)$ digit-by-digit, and encrypt arrays entry-by-entry. We abuse notation by denoting the resulting tuple of ciphertextst by $[\![msg]\!]_p = ([\![m_1]\!]_p, \ldots, [\![m_t]\!]_p)$ for $[\![msg(i)]\!]_p \leftarrow \mathrm{Enc}_{pk_p}(m_i)$ and $[\![array]\!]_p = ([\![array(1)]\!]_p, \ldots, [\![array(m)]\!]_p)$. (b) When the modulus p is for a ciphertext $[\![m]\!]_p$ clear from context we omit it and write $[\![m]\!]$. (c) When the public key pk_p is clear from the context we use a more natural presentation of homomorphic evaluation:

4 The Secure Report via Coreset Protocol

In this section we give the details of our upload and report protocol, starting with the basic version of the protocols and continuing with the extensions, optimizations and trade-offs.

4.1 The Basic Protocol

In this section we give the details of our upload and report protocol (basic version), and analyze the protocol to prove Theorem 1; See a protocol summary in Fig. 3.

The Secure report protocol (basic version). The protocol begins with an initialization and data upload phase. In this phase the client generates keys for the FHE, encrypts the data elements $x_1, \ldots, x_m$, and sends the encrypted data and the public key to the server. The FHE is initialized here with plaintext modulus p so that $p > m$ for m the provided upper-bound on the number of data elements.

Next the protocols enters the report queries phase. In this phase the client can repeatedly issue encrypted report queries ℓ. For each such encrypted query $[\![\ell]\!]$, the server does the following. First the server homomorphically evaluates the specified matching polynomial isMatch to obtain the encrypted indicator vector $\chi = (\chi(1), \ldots, \chi(m))$, where

$$[\![\chi(i)]\!]_p := \textsc{isMatch}([\![x_i]\!]_p, [\![\ell]\!]_p)$$

for all $i \in [m]$. Next, the server homomorphically multiplies χ by an (s, m)-sketch matrix $A_{s,m}$, to obtain and send to the client the encryption of $y = A_{s,m} \cdot \chi$:

$$[\![y]\!]_p := A_{s,m} \cdot [\![\chi]\!]_p.$$

The client decrypts using her secret key to obtain y, and then decodes y using the decoding algorithms for the sketch to obtain χ.

Proof of Theorem 1. The protocol is a single-server, single-round protocol, that performs no pre-processing on the cleartext data.

Correctness is argued as follows. First observe that χ satisfies that $\chi(i) = 1$ if-and-only-if $\textsc{isMatch}(x_i, \ell) = 1$. This implies that χ is a binary length m vector with at most s positive entries (by the premise that the number of matches is upper-bounded by s). By the correctness of the sketch $A_{s,m}$, we know that when computing $y = A_{s,m}\chi$ over the real number, the sketch decoding algorithm returns the set of entries where χ is positive, that is, the correct output $\mathcal{I}$. In our case however computation is over $GF(p)$. Nonetheless, our choice of $p > m$ ensures that no overflow occurs, and the obtained result is identical to the result when computing over the reals. Namely, the output is the desired set $\mathcal{I} = \{\, i \in [m] \mid \textsc{isMatch}(x_i, \ell) = 1\}$.

Complexity analysis follows. The server homomorphically-evaluates over $GF(p)$ the composition of two polynomials: the isMatch polynomial and the linear polynomial that computes the product of a plaintext matrix and an encrypted vector. The former has degree d, whereas the latter add nothing to the degree because the linear combinations

specifying the entries $y(i)$ are computed using only homomorphic addition (by summing the encrypted entries $ind(j)$ for all j where the $A_{s,m}(i,j)=1$). Thus, the server computes a polynomial of degree d (the degree of isMatch). The overall number multiplications is μ multiplications to compute isMatch on each of the m data elements, resulting in a total of $m \cdot \mu$ multiplications. The communication consists $[\![\ell]\!]$ and $[\![y]\!]$. The consists of $|\ell|$, the latter consists of a ciphertext for each entry of y, that is, $O(s^2 \log m)$ ciphertexts as the number of rows in $A_{m,s}$. The client's running-time is the time to encrypt and decrypt the sent and received ciphertexts respectively, plus the time to decode y. The time to encrypt and decrypt is polynomial in the number of cihpertexts and the security parameter λ, the decoding time is polynomial in s and $\log m$. The overall running time is therefore polynomial in $|\ell|$, λ, s and $\log m$.

Security against a semi-honest adversary controlling the server follows from the semantic-security on the underlying FHE scheme, because the entire protocol can be simulated by an efficient simulator oblivious of the client's input and output. □

Shared Parameters: The security parameter λ, the FHE scheme $\mathcal{E} = (\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$, a prime $p > m$, the records space $\mathcal{M}$ and lookup values space $\mathcal{Q}$, a polynomial realizing the matching criterion isMatch: $\mathcal{M} \times \mathcal{Q} \to \{0,1\}$, an upper-bound on the number of data records m, and an upper-bound on the number of matches $s \in [0,m]$.

Inputs: The client's input is a data $array = (x_1, \ldots, x_m) \in \mathcal{M}^m$ and queries $\ell \in \mathcal{Q}$; The server has no input.

Outputs: The client's output is the set $\mathcal{I} = \{\, i \in [m] \mid \text{isMatch}(x_i, \ell) = 1\}$; The server has no output.

Initialization and upload phase The client does the following:

- Generate keys $(pk_p, sk_p) \leftarrow \text{Gen}(1^\lambda; p)$.
- Encrypt the data elements $[\![x_i]\!]_p \leftarrow \text{Enc}_{pk_p}(x_i)$ for all $i \in [m]$.
- Send to server pk_p and $([\![x_1]\!]_p, \ldots, [\![x_m]\!]_p)$.

Report queries phase:

1. To issue each report query ℓ, the client encrypts the query $[\![\ell]\!]_p \leftarrow \text{Enc}_{pk_p}(\ell)$ and sends $[\![\ell]\!]_p$ to the server.
2. The server then does the following:
   (a) Compute $[\![\chi(i)]\!]_p := \text{isMatch}([\![x_i]\!]_p, [\![\ell]\!]_p)$ for all $i \in [m]$.
   (b) Compute $[\![y]\!]_p := A_{s,m} \cdot [\![\chi]\!]_p$ for $A_{s,m}$ an (s,m)-sketch matrix (see Section 2) and $[\![\chi]\!]_p = ([\![\chi(1)]\!]_p, \ldots, [\![\chi(m)]\!]_p)$; send $[\![y]\!]_p$ to the client.
3. The client decrypts $y \leftarrow \text{Dec}_{sk_p}([\![y]\!]_p)$ and decodes y (see Section 2) to obtain the output $\mathcal{I}$.

Figure 3: Secure report protocol via report-coreset.

4.2 Extensions and Optimizations

We discuss extensions and optimizations of our protocol for reporting both value and index; handling the case that s is unknown; reducing the ring size for the homomorphic evaluation from linear to logarithmic in m; and handling dynamic data management.

4.2.1 Reporting values on top of indexes ($\mathcal{L}$)

To report the matching elements values $x_i \in \{0,1\}^w$ ($\mathcal{L}$) on top of their indexes ($\mathcal{I}$) we do the following. The server homomorphically evaluates $w+1$ sketches, instead of a single sketch: One sketch for χ (this is the same as in the basic protocol), plus w sketches – a sketch for each digit $j \in [w]$ in the binary representation of the matching data elements.

The sketch for the j-th digit is computed as follows. Denote the bits in the binary representation of x_i by $x_i(1), \ldots, x_i(w)$ (for $i \in [m]$). Consider the vector consisting of all the j-th bits of the data elements:

$$allData_j = (x_1(j), \ldots, x_m(j))$$

Observe that the server implicitly holds the encrypted version of $allData_j$ (encrypted entry-by-entry). To compute the sketch for the j-bit of the matching data elements the server first homomorphically maps $allData_j$ to the s-sparse vector $sparseData_j$ whose i-th entries ($i \in [m]$) are the product of $\chi(i)$ and $allData_j(i)$:

$$[\![sparseData_j(i)]\!]_p := [\![\chi(i)]\!]_p \cdot [\![allData_j(i)]\!]_p.$$

Next, the server homomorphically evaluates the sketch for $sparseData_j(i)$ by left-multiplying its encrypted version by the cleartext binary matrix $A_{s,m}$

$$[\![sketch_j]\!]_p := A_{s,m} \cdot [\![sparseData_j]\!]$$

The server then sends to the client all the encrypted sketches $[\![y]\!]_p$ and $[\![sketch_j]\!]_p$ for all $j \in [m]$. The client decrypts and decode $[\![y]\!]_p$ to obtain $\mathcal{I}$, and decrypts and decode each $[\![sketch_j]\!]_p$ to obtain the j-th entry of the matching elements Putting it all together, the protocol outputs

$$\mathcal{L} = \{\, (i, x_i) \mid i \in [m], \textsc{isMatch}(x_i, \ell) = 1 \}\,.$$

The complexity is affected as follows. There is a growth by a factor w in the server's overall multiplications, the communication complexity and the client's time. The degree of the homomorphically evaluated polynomial however does not suffer from any degradation.

4.2.2 Reducing the Modulus p to Quasi-Logarithmic in m

Our basic protocol requires that the homomorphic evaluations are done over $GF(p)$ for large prime modulus $p > m$. In this section we employ the CRT-coreset to reduce the modulus to nearly logarithmic in m, while repeating the small modulus protocol for $\kappa = \lceil \log m / \log\log m \rceil$ times.

The small modulus protocol is similar to our basic protocol, except for the following. First, the initialization, encryption and homomorphic evaluations are with respect to a small modulus $p = O(\log m)$. Second, the initialization upload and homomorphic evaluations are done in parallel for κ primes $p_1, \ldots, p_\kappa$ (specifically, the κ first primes larger than $\log m$). Third, the client receiving κ sketches:

$$[\![y]\!]_{p_1}, \ldots, [\![y]\!]_{p_\kappa}$$

does the following: decrypt all sketches to obtain the residues $y_j = A_{s,m}\chi \mod p_j$ for $j \in [\kappa]$, apply the CRT-decoding algorithm on these residues to obtain the value $y = A_{s,m}\chi$ over the reals, now apply the sketch-decoding algorithm (as in the basic protocol) to obtain $\mathcal{I}$.

In this modified protocol, the degree of the polynomial evaluated by the server is unchanged while the modulus is reduced to logarithmic $p = \widetilde{O}(\log m)$ (by the prime numbers theorem). The overall number of multiplications by the server and the communication and client's complexiyt grow by a factor of κ.

4.2.3 Handling Unknown Number of Matches

We discussed two approaches to address settings when an upper-bound s on the number of matches is unknown.

The first approach, yielding a 2-rounds protocol revealing and upper-bound s on $|\mathcal{I}|$. In this approach the client and server engage in a preliminary round of communication for the client to learns $|\mathcal{I}|$ (the server learns nothing). The client then sends to the server an upper-bound s on $|\mathcal{I}|$ as she pleases, and engages in the report protocol for known upper-bound s.

Computing $|\mathcal{I}|$ is as follows. For the simple case when that modulus p is larger than m, computing $|\mathcal{I}|$ is very simple: The server homomorphically evaluates

$$[\![cnt]\!]_p := \sum_{i\in[m]} [\![\chi(i)]\!]_p$$

and sends to the client who decrypts and learns $|\mathcal{I}|$. For the case where we require the use of small moduli $p = \widetilde{O}(\log m)$ the protocol employs the CRT-coreset: The server homorphically evaluates the above sum modulo p_j the $\kappa = O(\log m)$ small primes p_j:

$$[\![cnt]\!]_{p_j} := \sum_{i\in[m]} [\![\chi(i)]\!]_{p_j}$$

and sends the tuple of resulting ciphertexts to the client. The client decrypts $[\![cnt]\!]_{p_j}$ for all $j \in [\kappa]$ to obtain the residues modulo p_j of $|\mathcal{I}|$. From these residues the client computes $|\mathcal{I}|$ using the standard CRT-decode algorithm.

The second approach yields a 1-round protocol, albeit with high overall multiplications and outgoing communication on the server's side (communication that the client mostly can ignore). In this case, the server sends to the client $[\![|\mathcal{I}|]\!]_p$ as well as the messages send in the report protocol when executed with number of matches upper-bounds $s = 2^0, 2^1, \ldots, m$ (assuming here w.l.o.g that m is a power of two). The client decrypts $[\![|\mathcal{I}|]\!]_p$ to learn $|\mathcal{I}|$ and then decrypts only the sketch corresponding to the smallest s larger than $|\mathcal{I}|$. Although the Cloud has communication complexity of $\Omega(m)$, the user can abort the protocol after receiving communication that is polynomial in $|\mathcal{I}|$ and $\log m$. We remark that we assumed here that $p > m$; extension to small moduli p is via using the CRT-coreset analogously to the above.

4.2.4 Dynamic Data Management

We next elaborate on how to extending the Secure report functionality to dynamic data management: Insert, Update and Delete.

Insert. Insertion of additional data elements is simply be the client encrypting and sending the element for the server to append at the end of the encrypted array (and update the array size parameter m accordingly).

Update. To update the i-th data element x_i, the client issues an update request $(\text{UPDATE}, [\![i]\!], [\![diff]\!])$ with encrypted index i and encrypted difference value $diff = new - old$ for new the new value for x_i and old the current value. (If the client doesn't have old she can use PIR to retrieve it.) The server then homomorphically adds to each elements $[\![x_j]\!]$ of the stored array the value $\textsc{IsEqual}([\![i]\!], j) \cdot [\![diff]\!]$, where i and j are in binary representation and $\textsc{IsEqual}\colon \{0,1\}^{\log m} \times \{0,1\}^{\log m} \to \{0,1\}$ is the equality-test polynomial (see Section 4.2.5). This results in a new encrypted array $[\![x']\!]$ satisfying $x'_i = new$ and $\forall j \neq i$, x'_j is unchanged.

Delete. Deletion of elements can be implemented by updating them to a reserved "Deleted" symbol. Another option is switching the value of the element we wish to delete to that of the last element in the array and reducing the number of elements m by 1 (for cases when the dynamic size of the data is either maintained by the client, or is not a secret and can be maintained by the server).

4.2.5 isMatch Instantiation in our Experimental Results

For our experimental results we used an equality-test instantiation of isMatch for exact-match Secure report. The data and query and specified in binary representation.

Specifically we instantiate isMatch to using the following equality-test polynomial over $GF(p)$ for data and query in binary representation. The polynomial isMatch: $\{0,1\}^w \times \{0,1\}^w \to \{0,1\}$ is defined by:

$$\text{isMatch}((a_1,\ldots,a_w),(b_1,\ldots,b_w)) = \prod_{i=1}^{w}\left(1-(a_i-b_i)^2\right) \mod p.$$

of degree $2w$.

We remark that for $p = 2$ a degree w polynomial suffice, as there is no need to square the difference $a_i - b_i$. For large $p > 2$ the standard polynomial raises the difference by $p-1$ using Fermat's Little Theorem to guarantee that all non-zero values map to one. Nonetheless for data in binary representation, as considered here, taking the square suffice as we are guaranteed that the difference is in $\{-1,0,1\}$.

5 System & Experimental Results

In this section we describe experiments on the Secure report that we implemented based on our algorithms (Protocol 3). For example, it can report the locations of all 1's in a 10-sparse vector of size $3 \cdot 10^9 = 3,000,000,000$ entries in less than one minute by using a single machine on Amazon EC2 cloud. The system is fully open sourced, and all our experiments are reproducible. We hope to extend and improve the system in future papers together with both the theoretical and practical community.

5.1 The System

System Overview. The system maintains an encrypted database that is stored on Amazon's AWS cloud. The system gets from the user an encrypted lookup value ℓ to search for, and a column name $array$ in a database table of length m. The encryption is computed on the user's side using a secret key that is unknown to the cloud. The user can send the request through a web-browser, that can be run e.g. from a smart-phone or a laptop. The system then runs our Secure report coreset algorithm on the cloud, and returns a report coreset for $(array, \ell)$. The web browser then decrypts this coreset on the user's machine and uses it to compute the solution to the report query, which is the indices $i_1,\ldots,i_{cnt}$ in $array$ that matches ℓ. Database updates can be maintained between search calls, and support multiple users that share the same security key.

Hardware. Our system is generic but in this section we evaluate it on Amazon's AWS cloud. We use one of the standard suggested EC2 x1.32xlarge server, each with 64 2.4 GHz Intel Xeon E5-2676 v3 (Haswell) cores and 1,952 GigaByte of RAM.

Open Software and Security. The algorithms were implemented in $C++$. HELib library [43] was used for the FHE commands. The source of our system is open under the GNU v3 license and can be found in [3]. For our experiments below we use a security key of 80 bits of security.

5.2 Experimental Results

In this sub-section we describe our preliminary experiments with our system and explain the results. Due to lack of space we omit more results and description that can be found in the fuller version [4].

Data. We ran the system on a lookup value $\ell = 1$ in an array of m integers $array \in \{0, \ldots, m-1\}^m$. The vector was all zeroes except for s random indices, and different values for m and s were used. As expected, the actual values behind the encrypted records had no effect on the running times.

The Experiment. We ran Algorithm 3 for database table columns ranging from $m = 10$ to $m = 3,000,000,000 = 3 \cdot 10^9$ records, and $cnt \in \{10, 20, 40\}$.

Results. Our experimental results for a single machine on the cloud are shown as the circle points in Fig 1. The table of exact values appears in the full version [4].

The user's decoding time was negligible in all the experiments, so the cloud's time equals to the overall running time. For example, the graph shows that a single machine can report in about 2 minutes all 1's in a 20-sparse column of 3,000,000,000 binary entries.

Comparison to the direct approach. Our theoretical results proves that the running time of our new algorithm is only poly-logarithmic in the number m of entries compared to the direct approach which is polynomial in m. However, it assumes that both our and the direct algorithm may use m machines in parallel, i.e., a machine for each record. The goal of our experiment was to show a significant time reduction even using a single machine on the cloud where a running time that is linear in m is expected.

The graph in Fig. 1 is log scaled so a linear curve shows a polynomial relation, and its slope is the degree of the polynomial. Our experiments indeed show that our algorithm is linear in m as expected. The direct algorithm perform worse, both on small number of entries (the ratio is approximately 1000), but also asymptotically: The slope of the direct approach is about 1.5, indicating a running time of $O(m^{1.5})$ (as oppose to our $O(m)$). The main reason for the improvement in running time of our algorithm is that unlike the direct algorithm, our algorithm does not use multiplications to compute the sketch.

6 Conclusion

In this paper we give a secure protocol for the report problem on encrypted data elements and query. Our protocol retrieves all matching elements in a data array, where both data elements and query are encrypted using Fully Homomorphic Encryption (FHE).

Our protocol involves a single-server, single-round, and requires no pre-processing of the cleartext data. The complexity of our protocol, given an upper-bound s on the number of matches, is as follows: The server evaluates a polynomial of degree independent of the array size m; the communication and client's complexity is polynomial in s and $\log m$. The protocol is secure against semi-honest adversaries controlling the server. We presented extensions and optimizations of our protocol: handling the case that s is unknown; reducing the ring size for the homomorphic evaluation from linear to logarithmic in m;

Table 2: Server's running time of Report Coreset (Algorithm 3) as measured on a single machine on Amazon's cloud for different database *array* size (1^{st} column). In these experiments $cnt = 40$, i.e. at most 40 elements matched the report criteria. The 3^{rd} column shows the time in minute to compute report with the direct algorithm and The 4^{th} column shows the time in minute to compute report with our coreset algorithm. The 2^{nd} column shows the the speedup of our report coreset. Since we could not run the direct approach for too large databases we give estimations In parentheses are values based on our analysis.

Records	$\frac{time(direct)}{time(coreset)}$	Direct Repot (minutes)	Report Coreset (minutes)
89,600	48	4	0.083
192,000	47	7	0.15
396,800	73	22	0.3
806,400	109	67	0.61
1,625,600	164	199	1.21
3,264,000	268	689	2.56
6,540,800	450	2,281	10.16
13,094,400	(754)	(7,661)	20.55
26,201,600	(441)	(9,068)	40.58
52,416,000	(542)	(21,990)	82.38
104,844,800	(577)	(47,568)	163.56
209,702,400	(3,107)	(508,126)	328.08
419,417,600	(3,502)	(1,148,875)	653.07

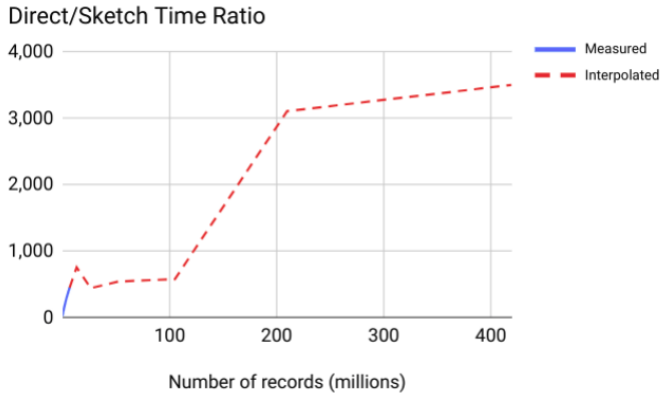

Figure 4: The speedup of using our report coreset (ratio between the direct algorithm and the report coreset) for arrays with $cnt = 40$, i.e. at most 40 records matched the query. The x-axis shows the array size (in millions) and the y-axis shows the ratio.

and handling dynamic data management. Furthermore, our protocol is compatible with versatile matching conditions isMatch, and can be utilized for example for exact match, similarity search, wild-card matching, range queries, Boolean queries and more.

We implemented our protocol in an open source library based on HElib, and ran experiments on Amazon's AWS EC2 cloud. Our experiments show that we can search in *billions of encrypted data elements for an encrypted query in a minute.*

To achieve our results we put forth our coreset paradigm for homomorphic encryption, and present a low-degree polynomial for Secure report. Both contributions could be useful for future works. The coreset for homomorphic encryption paradigm has already been employed in follow-up works. The low-degree polynomial could be useful in the broader

context of secure computation, using other techniques such as secret sharing beyond FHE.

The simplicity of our algorithm makes it a good candidate to run on small embedded systems. Motivating examples are data sources such as sensors of wearable devices uploading data to a server while keeping it hidden from the Server but still allowing to query the data. In this case the sensors take the role of the data source client in the running the upload phase of our protocol. With our protocol a network of weak sensor devices can store their encrypted data in a server for the search client to issue the report queries.

References

[1] M. A. Abdelraheem, T. Andersson, and C. Gehrmann. Inference and record-injection attacks on searchable encrypted relational databases. *IACR Cryptology ePrint Archive*, 2017:24, 2017.

[2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

[3] A. Akavia, D. Feldman, and H. Shaul. ReportLib: Open library for FHE report, 2018.

[4] A. Akavia, D. Feldman, and H. Shaul. Secure data retrieval on the cloud homomorphic encryption meets coresets. Cryptology ePrint Archive, Report 2018/1003, 2018. https://eprint.iacr.org/2018/1003.

[5] A. Akavia, D. Feldman, and H. Shaul. Secure search via sketching for homomorphic encryption. In *Proceedings of the 25th ACM Conference on Computer and Communications Security*. ACM, 2018.

[6] A. Akavia, C. Gentry, S. Halevi, and M. Leibovich. Setup-free secure search on encrypted data: Faster and post-processing free. Cryptology ePrint Archive, Report 2018/1235, 2018. https://eprint.iacr.org/2018/1235.

[7] O. Barkol and Y. Ishai. Secure computation of constant-depth circuits with applications to database search problems. In *Annual International Cryptology Conference*, pages 395–411. Springer, 2005.

[8] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu. Private database queries using somewhat homomorphic encryption. In *International Conference on Applied Cryptography and Network Security*, pages 102–118. Springer, 2013.

[9] C. Boutsidis, A. Zouzias, and P. Drineas. Random projections for k-means clustering. In *Advances in Neural Information Processing Systems*, pages 298–306, 2010.

[10] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[11] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106, 2011.

[12] V. Braverman, G. Frahling, H. Lang, C. Sohler, and L. F. Yang. Clustering high dimensional dynamic data streams. *arXiv preprint arXiv:1706.03887*, 2017.

[13] V. Braverman, A. Meyerson, R. Ostrovsky, A. Roytman, M. Shindler, and B. Tagiku. Streaming k-means on well-clusterable data. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 26–40. Society for Industrial and Applied Mathematics, 2011.

[14] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 668–679. ACM, 2015.

[15] G. S. qCetin, W. Dai, Y. Doröz, W. J. Martin, and B. Sunar. Blind web search: How far are we from a privacy preserving search engine? *IACR Cryptology ePrint Archive*, 2016:801, 2016.

[16] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 577–594. Springer, 2010.

[17] J. Chen and Q. Zhang. Bias-aware sketches. *Proceedings of the VLDB Endowment*, 10(9):961–972, 2017.

[18] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song. Homomorphic encryption for arithmetic of approximate numbers. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer, 2017.

[19] J. H. Cheon, M. Kim, and M. Kim. Optimized search-and-compute circuits and their application to query evaluation on encrypted data. *IEEE Trans. Information Forensics and Security*, 11(1):188–199, 2016.

[20] J. H. Cheon, M. Kim, and K. E. Lauter. Homomorphic computation of edit distance. In *Financial Cryptography Workshops*, pages 194–212, 2015.

[21] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords, 1997.

[22] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 41–50. IEEE, 1995.

[23] K. L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):63, 2010.

[24] E. Cohen and H. Kaplan. Tighter estimation using bottom k sketches. *Proceedings of the VLDB Endowment*, 1(1):213–224, 2008.

[25] R. Cole, D. Shasha, and X. Zhao. Fast window correlations over uncooperative time series. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 743–749. ACM, 2005.

[26] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *Proceedings of the 31st international conference on Very large data bases*, pages 13–24. VLDB Endowment, 2005.

[27] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 281–292. ACM, 2007.

[28] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.

[29] A. Czumaj, C. Lammersen, M. Monemizadeh, and C. Sohler. (1+ ε)-approximation for facility location in data streams. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1710–1728. Society for Industrial and Applied Mathematics, 2013.

[30] Y. Doröz, B. Sunar, and G. Hammouri. Bandwidth efficient PIR from NTRU. In *Financial Cryptography and Data Security - FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers*, pages 195–207, 2014.

[31] D. Du, F. K. Hwang, and F. Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 2000.

[32] M. Garofalakis, D. Keren, and V. Samoladas. Sketch-based geometric monitoring of distributed stream queries. *Proceedings of the VLDB Endowment*, 6(10):937–948, 2013.

[33] C. Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, Stanford, CA, USA, 2009. AAI3382729.

[34] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[35] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *VLDB*, volume 1, pages 79–88, 2001.

[36] M. Giraud, A. Anzala-Yamajako, O. Bernard, and P. Lafourcade. Practical passive leakage-abuse attacks against symmetric searchable encryption. In *14th International Conference on Security and Cryptography SECRYPT 2017*. SCITEPRESS-Science and Technology Publications, 2017.

[37] O. Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 182–194, New York, NY, USA, 1987. ACM.

[38] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[39] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[40] P. Grubbs, R. McPherson, M. Naveed, T. Ristenpart, and V. Shmatikov. Breaking web applications built on top of encrypted data. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1353–1364. ACM, 2016.

[41] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. Leakage-abuse attacks against order-revealing encryption. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 655–672. IEEE, 2017.

[42] S. Guha and A. McGregor. Graph synopses, sketches, and streams: A survey. *Proceedings of the VLDB Endowment*, 5(12):2030–2031, 2012.

[43] S. Halevi. Helib - an implementation of homomorphic encryption. https://github.com/shaih/HElib/, 2013.

[44] H. Huang and S. P. Kasiviswanathan. Streaming anomaly detection using randomized matrix sketching. *Proceedings of the VLDB Endowment*, 9(3):192–203, 2015.

[45] J. Huggins, T. Campbell, and T. Broderick. Coresets for scalable bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088, 2016.

[46] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *VLDB*, pages 363–372, 2000.

[47] P. Indyk, H. Q. Ngo, and A. Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1126–1142. SIAM, 2010.

[48] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *focs*, page 294. IEEE, 2000.

[49] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Ndss*, volume 20, page 12, 2012.

[50] M. Kim, H. T. Lee, S. Ling, S. Q. Ren, B. H. M. Tan, and H. Wang. Better security for queries on encrypted databases. *IACR Cryptology ePrint Archive*, 2016:470, 2016.

[51] M. Kim, H. T. Lee, S. Ling, B. H. M. Tan, and H. Wang. Private compound wildcard queries using fully homomorphic encryption. *IEEE Transactions on Dependable and Secure Computing*, 2017.

[52] K. E. Lauter, A. López-Alt, and M. Naehrig. Private computation on encrypted genomic data. *IACR Cryptology ePrint Archive*, 2015:133, 2015.

[53] J. M. Phillips. Coresets and sketches. *arXiv preprint arXiv:1601.00617*, 2016.

[54] D. Pouliot and C. V. Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1341–1352. ACM, 2016.

[55] A. A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[56] G. Reeves, J. Liu, S. Nath, and F. Zhao. Managing massive time series streams with multi-scale compressed trickles. *Proceedings of the VLDB Endowment*, 2(1):97–108, 2009.

[57] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[58] S. S. Roy, F. Vercauteren, J. Vliegen, and I. Verbauwhede. Hardware assisted fully homomorphic function evaluation and encrypted search. *IEEE Transactions on Computers*, 2017.

[59] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82. ACM, 1987.

[60] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000.

[61] H. Tang, X. Jiang, X. Wang, S. Wang, H. Sofia, D. Fox, K. Lauter, B. Malin, A. Telenti, L. Xiong, and L. Ohno-Machado. Protecting genomic data analytics in the cloud: state of the art and opportunities. *BMC Medical Genomics*, 9(1):63, Oct 2016.

[62] F. Wang, C. Yun, S. Goldwasser, V. Vaikuntanathan, and M. Zaharia. Splinter: Practical private queries on public data. In *NSDI*, pages 299–313. USENIX Association, 2017.

[63] D. P. Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[64] A. C.-C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.

[65] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba. Secure pattern matching using somewhat homomorphic encryption. In *Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop*, CCSW '13, pages 65–76, New York, NY, USA, 2013. ACM.

[66] Y. Zhang, J. Katz, and C. Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *USENIX Security Symposium*, pages 707–720, 2016.

A Direct Polynomial

We present the direct polynomial for returning all matches. We present here the polynomial as defined on cleartext data. To compute on encrypted data the polynomial is replaced by its homomorphic version, as standard.

Specifications for the direct polynomial. We first define the direct polynomial's parameters, input and output.

- The direct polynomial is parameterized by the number of elements m, a prime $p > m$ so that computations are in $GF(p)$, the number of matches to be returned $cnt \in \{1, \ldots, m\}$, the domain of data-elements $\mathcal{M}$ and of queries $\mathcal{Q}$, and a polynomial realizing the matching criterion $\textsc{isMatch}\colon \mathcal{M} \times \mathcal{Q} \to \{0, 1\}$.
- The input to the polynomial is the data $array = (x_1, \ldots, x_m) \in \mathcal{M}^m$ and query $\ell \in \mathcal{Q}$.
- The output is the indexes of the matching elements $output \in \{0, \ldots, m\}^{cnt}$, where for every $k \in [cnt]$, $output(k)$ is the index $i \in [m]$ of the k-th match for ℓ in $array$. That is, $output(k) = i$ if-and-only-if both the following holds:
  1. The i-th element is a match: $\textsc{isMatch}(x_i, \ell) = 1$, and
  2. There are $k-1$ preceding matches: $|\{\, j \in [i-1] \mid \textsc{isMatch}(x_j, \ell) = 1\}| = k-1$.

  In case $cnt > |\mathcal{I}|$, for $\mathcal{I}$ the set of all matches as defined in Section 3, the output on entries $k \in \{|\mathcal{I}| + 1, \ldots, cnt\}$ is 0.

Realization of the direct polynomial. We next specify how to realized the direct polynomial. Computation in the following is over $GF(p)$ for a prime $p > m$. For every $k \in [cnt]$, we define:

$$output(k) = \sum_{i=1}^{m} i \cdot \textsc{isMatch}(x_i, \ell) \cdot \left(1 - (\textsc{prevMatches}(i) - (k-1))^{p-1}\right)$$

where $\textsc{prevMatches}(i)$ is the number of matches in the sub-array $(x_1, \ldots, x_{i-1})$, as computed by the following polynomial:

$$\textsc{prevMatches}(i) = \sum_{j=1}^{i-1} \textsc{isMatch}(x_j, \ell).$$

Analysis of the direct polynomial. The analysis for the direct polynomial follows.

Correctness. To argue correctness observer first that $\textsc{prevMatches}(i)$ is the number of elements in $x_1, \ldots, x_{i-1}$ that match ℓ, namely:

$$\textsc{prevMatches}(i) = |\{\mathcal{I} \cap \{1, \ldots, i-1\}\}|.$$

Next, for every $k \in \{1, \ldots, |\mathcal{I}|\}$, by Fermat's Little Theorem (FLT), we have that

$$1 - (\textsc{prevMatches}(i) - (k-1))^{p-1} = \begin{cases} 1 & \text{if } \textsc{prevMatches}(i) = k-1 \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $\textsc{isMatch}(x_i, \ell) \cdot \left(1 - (\textsc{prevMatches}(i) - (k-1))^{p-1}\right)$ evaluates to 1 if-and-only-if both conditions (1) and (2) from the output definition above hold, and it is 0 otherwise. So,

$$i \cdot \textsc{isMatch}(array(i), \ell) \cdot \left(1 - (\textsc{prevMatches}(i) - (k-1))^{p-1}\right)$$

is equal to i if-and-only-if i is the index of the k-th matching element to ℓ in $array$, and it is 0 otherwise We conclude that the $output[k] = i$ for i the index of the k-th matching element to ℓ in $array$.

Complexity. We show that the direct polynomial has degree $\Omega(m \cdot d)$ for d the degree of $\textsc{isMatch}$. The direct polynomial for computing $output(k)$ is the product of several sub-polynomials: (i) The plaintext coefficient i, (ii) The $\textsc{isMatch}$ polynomial, and (iii) The product of $(p-1)$ polynomials, each of degree identical to the degree of $\textsc{prevMatches}$. The latter is equal to the degree of $\textsc{isMatch}$ (as it is the sum of $\textsc{isMatch}$ evaluations). The degree of a product of polynomials is the sum of their degrees. So the degree of the direct polynomial is $d + (p-1)d = pd$. We conclude, by assigning $p > m$, that the degree of the direct polynomial is $\Omega(m \cdot d)$.

Remarks. The requirement that $p > m$ is necessary for correctness. Otherwise, the output is incorrect due to two reasons. First, for $p \leq |\mathcal{I}|$, $\textsc{prevMatches}$ would have "false-zero" value whenever the number of previous-matches is a multiple of p, leading to incorrect output. Second, for $p \in (|\mathcal{I}|, m)$, the output values are the residues "$i \mod p$" of the desired indexes i, which is insufficient information for reconstructing i when $i \in [p+1, m]$.

We note that there are other possible implementations for the direct polynomial. In particular, the output of $\textsc{prevMatches}$ could be specified in binary representation. In the case, computations can be done over $GF(2)$, and the *comparison* of $\textsc{prevMatches}$ to $k-1$ can be done in low-degree $\log m$. However, the degree of *counting* the number of previous matches $\textsc{prevMatches}$ becomes $\Omega(m \cdot d)$.