Electronic Colloquium on Computational Complexity, Report No. 22 (2007)

Algebraic Lower Bounds for Computing on Encrypted Data

Rafail Ostrovsky* William E. Skeith III†

Abstract

In cryptography, there has been tremendous success in building primitives out of homomorphic semantically-secure encryption schemes, using homomorphic properties in a black-box way. A few notable examples of such primitives include items like private information retrieval schemes and collision-resistant hash functions (e.g. [14, 6, 13]). In this paper, we illustrate a general methodology for determining what types of protocols can be implemented in this way and which cannot. This is accomplished by analyzing the computational power of various algebraic structures which are preserved by existing cryptosystems. More precisely, we demonstrate lower bounds for algebraically generating generalized characteristic vectors over certain algebraic structures, and subsequently we show how to directly apply this abstract algebraic results to put lower bounds on algebraic constructions of a number of cryptographic protocols, including PIR-writing and private keyword search protocols. We hope that this work will provide a simple "litmus test" of feasibility for use by other cryptographic researchers attempting to develop new protocols that require computation on encrypted data. Additionally, a precise mathematical language for reasoning about such problems is developed in this work, which may be of independent interest.

1 Introduction

One of the central problems in cryptography is that of finding a public key encryption scheme that would allow "computation on encrypted data". In its full generality the problem could be simply stated as follows: to find a public key encryption scheme such that given encryptions of arbitrary plaintexts $\mathcal{E}(x_1), \ldots, \mathcal{E}(x_n)$ it is possible *without the decryption key* to compute $\mathcal{E}(f(x_1, \ldots, x_n))$ for any polynomial-time computable function f. Naturally, if one can find a public-key cryptosystem that is "fully homomorphic", i.e. allows operations on ciphertext that preserve the structure of a ring, and hence allows computation of the ubiquitous *"NAND"* operation on the underlying plaintext, it would give a general solution to the above problem. Indeed, the reason this is such a central problem is that it would allow for incredible ability to arbitrarily manipulate encrypted data without sacrificing privacy. This problem was posed nearly 30 years ago by Rivest, Adelman and Dertouzos [21]. We

*Department of Computer Science and Department of Mathematics, UCLA, E-mail: rafail@cs.ucla.edu

†Department of Mathematics, University of California, Los Angeles. E-mail: wskeith@math.ucla.edu, wskeith@ucla.edu.

ISSN 1433-8092

do not know if such an encryption scheme exists in its full generality, though various partial answers are known: One partial answer is single-homomorphic encryption: given $\mathcal{E}(x)$ and $\mathcal{E}(y)$, where x and y come from some abelian group, there exist cryptosystems that can compute $\mathcal{E}(x * y)$, where $*$ is the group operation. Examples include ElGamal [9], where the group operation is multiplication, Goldwasser and Micali [10] where the operation is addition modulo 2, and Pallier [20] where the group operation is addition modulo a large composite. Recent progress by Boneh, Goh and Nissim [3] showed that more is possible: they designed a cryptosystem that allows an arbitrary number of additions and a single multiplication (of the underlying plaintext) by manipulating ciphertexts only. Another approach at building fully-homomorphic encryptions was considered by Sander, Young, and Yung [23], but only applied to Boolean operations and doubled the ciphertext size at every step. As a result, one could only perform a few Boolean operations before the ciphertext size became impractical. A partial negative result was given by Boneh and Lipton [4].

Many useful protocols and primitives have been derived from such homomorphic schemes in a "black box" way, essentially just manipulating the homomorphic properties to construct various systems. Prominent examples include single-database private information retrieval (PIR) and collision-resistant hashing (see [14, 6, 13]). In this work, we show a variety of natural tasks that *cannot* be accomplished in this way. More accurately we'll illustrate a single basic task that cannot be algebraically accomplished (with small communication) with various types of algebra (e.g., that of any abelian group). This result will give us a nice criterion or "litmus test" for determining the feasibility of constructing communication-efficient protocols in general, and a very strong result for constructing protocols based on the black box use of homomorphic encryption. Along the way, we'll also develop a mathematical language and technique for reasoning about such questions, which may be of independent interest.

1.1 Our Results

A central element of this paper, from which we will derive a number of results, is an algebraic lower bound for a certain task. The task in question is that of specifying "characteristic vectors" over a group. For a group G, we call a vector $(v_1, ..., v_n) \in G^n$ "characteristic" for a set $S \subset [n]$ if $v_i \neq \mathbf{0}_G$ if and only if $i \in S$. We'll show that it is impossible to "algebraically" specify characteristic vectors of singleton subsets of $[n]$ over *any abelian group* with communication complexity less than $\mathcal{O}(n)$. A formal statement of this idea appears as Theorem 2.3.

This statement holds for *all* abelian groups. For intuition, one may consider the case of linear algebra, in which the group G is of prime order, and has a field structure which could be put upon it. It is a relatively simple exercise to prove this very special case of the theorem, just arguing about the degree of vector spaces. However, this technique does not get very far. As the reader will see from Example 2.2, these ideas don't apply to general abelian groups G, even when they are cyclic. In fact, there is not even a well-defined notion of degree in this setting. A "degree-based" argument could be carried out via free-module analysis, but it will greatly complicate and obfuscate matters, and furthermore it will yield a weaker version of the theorem. Our more abstract approach and more general result will be of utility later on, when we generalize to other structures.

Additionally, we prove a smooth trade-off in communication complexity as the size of S

(the number of non-identity elements in the characteristic vectors) increases, and as mentioned, we also generalize to other algebraic structures, which contain virtually all such that are preserved by known homomorphic encryption schemes. In particular, we prove results for *any abelian group* as well as results for arbitrary rings, in a setting restricted to polynomials of total degree t. (Note that the cryptosystem of Boneh, Goh, and Nissim [3] gives an example of such an algebraic structure for the special case of $t = 2$, where polynomials of total degree 2 are the most general items that one has the ability to compute on encrypted data.) Finally, we'll show a number of natural cryptographic protocols that would imply the functionality of generating characteristic vectors, and hence derive algebraic lower bounds for the communication involved in these protocols as well.

As one will see after an examination of our algebraic results, they are in fact quite general. Since the results for abelian groups apply to all affine maps, this rules out many possibilities which do not necessarily come from group formulas. (For example, arbitrary endomorphisms may now be included in the class of "formulas" even though there is no way to compute all endomorphisms via an abelian group formula.) In particular, even if one changes their representation of data to be not just one group element, but many, and furthermore manipulates each of these elements independently, our results will still apply (this is a simple consequence of Corollary 2.11).

We also note that a general language for formally discussing these ideas is presented here, using category-theoretic ideas. This helps unify our discussion, and make formal definitions possible at the right level of abstraction (since as mentioned, a number of different algebraic structures are addressed here). As a final note, using this language we demonstrate that with any simple *non-abelian* group structure one *can* compute all finite functions via group formulas (thus, the existence of any cryptosystem homomorphic over a simple non-abelian group implies a fully-homomorphic encryption). This work can be found in the later sections, and somewhat generalizes that of [1] and [17], however it is essentially a different, and constructive version of [25] and may be of independent interest.

1.2 Related Work

The lower bounds that we consider are most closely related to computational lower bounds on number theoretic problems when algorithms are restricted only to underlying group operations. For example, Boneh and Lipton [4] examine the computational difficulty breaking any algebraically homomorphic (over a field) cryptosystem. In contrast, our lower bounds are on communication complexity and apply to a wide variety of algebraic structures. Other related works are that of Shoup [24] and Maurer and Wolf [16], which consider computational difficulty of the discrete logarithm problem, and other number-theoretic problems in cyclic groups, provided that the algorithms do not exploit any specific properties of the representation of group elements.

Our lower bounds are geared towards communication complexity and program size, rather than computational complexity, but similar to these works, we focus only on algorithms that utilize nothing other than the underlying algebraic structures. However, we consider a far greater variety of structures in our work (including arbitrary abelian groups and bounded degree polynomials over rings).

1.3 Overview, Motivation and Intuition

Often times, novel cryptographic protocols are developed using homomorphic encryption as building block (and often it is the only necessary ingredient). Many basic protocols can be constructed in this way, for example, private information retrieval, oblivious transfer, and collision-resistant hashing, to name a few. Indeed, such methods have accomplished much in the past, and continue to prove themselves as fruitful techniques. However, the types of algebraic structures available in homomorphic encryption are quite limited. Not much beyond the structure of an abelian group can be preserved under an encryption scheme. Quite clearly, abelian groups have limited computing power. If one simply examines the number of distinct m-variable "formulas" in a finite abelian group G of order k in comparison to the number of G-valued functions (as set maps) that depend on m variables, one can't help but notice a great discrepancy in cardinality (the fraction is in fact negligible as the number of variables increases). So indeed, there is much that cannot be computed using only abelian group formulas. But what are these functions? Furthermore, in what sense can they not be computed?

As mentioned before, there have been many protocols of great utility derived from homomorphic encryption over abelian groups (e.g. [14, 6, 13]). However, as the authors believe, for every such useful protocol in the literature, there are many dead ends, lying at the bottom of stacks of paper upon researchers' desks. But until now, there has not been much formal proof that these dead ends are actually just that. This work provides some basic proofs of lower bounds for a few very straightforward protocols, based on these algebraic assumptions. But more importantly, it develops techniques and methods for reasoning about such tasks, which hopefully will be of use to many other researchers, in the context of many other protocols.

1.4 Summary and Techniques

To summarize, we present here a formal study of what can be computed solely using the operations of various algebraic structures (although certain results hold under weaker assumptions). The structures primarily studied are a superset of the structures that (to date) can be preserved with a homomorphic encryption scheme. Hence, in terms of generic algebraic methods, the lower bounds shown here serve as practical lower bounds for such techniques. We prove results for the entirety of abelian groups, and also for rings, in a setting that uses polynomials of some bounded total degree as formulas. (See the cryptosystem of [3] for motivation of this idea.)

Given the large cardinality discrepancy mentioned above with respect to abelian group algebraic formulas, it is not surprising that various protocols cannot be algebraically implemented in such a way. However, it may be quite surprising that such simple and natural protocols cannot be implemented, and also that the techniques used to prove such statements are in fact rather elementary. Basically, what is shown is that all "formulas" over abelian groups correspond to "affine" group maps, which as we will show have a fair amount of structure. These "affine" group maps are then analyzed over general modules over a finite ring R, and a basic result regarding characteristic vectors over a group follows from the analysis. This result is quite strong from an algebraic point of view, since it was proved for arbitrary affine maps, not all of which come from formulas over a group. However, every

(abelian) group formula does come from an affine map. With this abstract formulation and proof, it will then be relatively easy to extend the result to other more general structures, such as polynomials of bounded total degree over a ring.

To even begin a mathematical discussion on this subject, a formulation of the idea of "algebraic formula" is needed, and accordingly, we have provided a thorough formalization in this work using the language of category theory. This study is actually somewhat extensive, with numerous examples provided, and as such has been placed near the final sections of the paper. This material may be interesting in its own right, but it is not the focus of this work. It is merely a tool for understanding, formalizing, and for properly stating the definitions we use here. Formalizations of the idea of computing with algebra are given there as well.

2 Preliminaries and Basic Results

2.1 Notations

For a brief index of mathematical notations we use, see Section 5. Most notations are standard, and most algebraic notations are consistent with [12].

2.2 Generating Encryptions of Characteristic Vectors: Motivation

This example provides a simple description of a protocol that can't be non-trivially implemented with abelian group algebra. Later, we'll show a variety of problems (usually related to PIR or PIR-writing) which would imply a protocol like this. Hence, these too cannot be implemented with abelian group algebra.

We could, at this point, formalize a cryptographic protocol about generating characteristic-type vectors over a group, but it may be convenient to postpone such a definition and instead get right to the main point, which is algebra. So, we will explain in simple terms the algebraic task we are trying to accomplish. Consider the following problem:

Let $n, m \in \mathbb{Z}^+$, and let G be an abelian group. Define the following elements $v_i \in G^n$:

$$v_i = (\mathbf{0}_G, ..., \mathbf{0}_G, x_i, \mathbf{0}_G, ..., \mathbf{0}_G)$$

where $x_i \neq \mathbf{0}_G$ appears in the i-th position.[1] Let $\{\mathfrak{m}_i\}_{i=1}^n \subset G^m$ and let f be an arbitrary affine group map in m variables from $G^m \longrightarrow G^n$, i.e., $f = f_m + c$ where $f_m : G^m \longrightarrow G^n$ is linear and $c \in G^n$. Note that these affine maps can express all possible abelian group formulas on a set of variables (see Definition 4.2). The question is

Question 2.1 (Informal) *If $f(\mathfrak{m}_i) = v_i$ for all $i \in [n]$, what can be said about $|G^m|$? In particular, how small can it be?*

We will soon answer this question in a variety of contexts, but first we'll give an example to help motivate the non-triviality of the question and our lower bound. The phrasing used regarding the size estimation was deliberate: we don't isolate or bound m alone, because we cannot bound m in a non-trivial way. It is in fact possible to accomplish the above result

[1] We give x an index i simply to show that it need not be uniform across all vectors.

with $m = 1$, even for a cyclic group. However, as we'll show in our lower bound, this comes at the cost of increasing the size of G.

Example 2.2 *Let $n \in \mathbb{Z}^+$, and let $N = \prod_{i=1}^{n} p_i$, where p_i is the i-th prime number. Define $G = \mathbb{Z}_N$. Define integers $\{z_i\}_{i=1}^{n}$ as follows:*

$$z_i = \prod_{j \neq i} p_j$$

Then, since all the primes were distinct, it is easy to verify that

$$(z_i z_j \neq 0 \bmod N) \iff (i = j)$$

So, we could define a linear function $f = (f_1, ..., f_n)$ from $G \longrightarrow G^n$ by $f_i(x) = z_i \cdot x$, and we would have $f(z_i) = v_i$, for some elements $v_i \in G^n$ which fit the above description of a complete set of characteristic vectors.

However, in the preceding example, notice that n different primes had to divide the order of G. Hence, $|G| > 2^n$ is of exponential size in n. We will show that even using affine maps, this is always the case: to generate n orthogonal-type characteristic vectors with m group elements always requires a group G such that G^m has exponential size in n, although the statement we prove has a more abstract setting.

2.3 A Basic Algebraic Result

Here, we will make precise the relationship regarding n and the size of an abelian group that can algebraically generate a complete set of n characteristic vectors over an abelian group G.

Theorem 2.3 *Let $n \in \mathbb{Z}^+$ and let G, A be abelian groups. Let $V = \{v_i\}_{i=1}^{n} \subset G^n$ be any collection of elements so that the j-th position of v_i is $\mathbf{0}_G$ if and only if $i \neq j$. Then if $F = f + c$ is an affine map from $A \longrightarrow G^n$ such that $V \subset F(A)$ then we have $\log(|A|) \in \Omega(n)$. More specifically, if $A \subset G^m$, we have that*

$$\log(|G|) \geq \frac{n}{m+1}$$

We'll break the majority of the proof into a lemma and a few simple observations. To begin, we'll prove the following lemma which will help us analyze affine maps and translated characteristic vectors.

Lemma 2.4 *Let R be a finite ring with identity, and let M be a (unitary) R-module. Let $\Omega = \{\omega_i\}_{i=1}^{k} \subset M$ be a finite collection of elements. Let $\Omega' = \{(\omega_i + c)\}_{i=1}^{k}$ for some fixed element $c \in M$. Then $\langle \Omega' \rangle$, the module generated by Ω', increases in size by at most a factor of $|R|$ over the size of $\langle \Omega \rangle$. I.e.,*

$$\frac{|\langle \Omega' \rangle|}{|\langle \Omega \rangle|} \leq |R|$$

Proof: Recall that for any submodules A, B of a module there is always a surjection $A \oplus B \longrightarrow A + B$ since $A \oplus B$ is a coproduct and $A + B$ is generated by $A \cup B$. Hence $|\langle \Omega \rangle + Rc| \leq |\langle \Omega \rangle \oplus Rc| \leq |\langle \Omega \rangle||R|$. Since clearly $\langle \Omega' \rangle \subset \langle \Omega \rangle + Rc$ as M is unitary over R, this in fact completes the proof. ■

In light of Lemma 2.4, we need only to analyze "un-translated" characteristic-type vectors. If they generate a large module, then so will the translated vectors. It is quite clear any such module generated by elements like those in V will be exponential in size, however to be complete, we provide a formal proof.

Observation 2.5 *Let G be a finite abelian group. Let $n \in \mathbb{Z}^+$. Define elements $v_i \in G^n$ by $v_{ij} = \delta_{ij} \cdot \alpha_i$ for some $\alpha_i \neq 0 \in G$, and $\delta_{ij} \in \mathbb{Z}$ with $\delta_{ii} = 1$ for all i and $\delta_{ij} = 0$ for $i \neq j$. Let $H = \langle \{v_i\}_{i=1}^n \rangle$, the subgroup of G^n generated by the v_i. Then $|H| \geq 2^n$.*

Proof: Note that $H \simeq \bigoplus_{i=1}^n \langle v_i \rangle$ since clearly $\langle v_i \rangle \cap \langle v_j \rangle = \{0\}$ for all $i \neq j$, and since by definition H is generated by the v_i. Also, for all $i \in [n]$ we have that $|\langle v_i \rangle| \geq 2$ since $\alpha_i \neq 0$ which completes the proof. ■

We'll also make use of a few very elementary observations from group theory.

Observation 2.6 *Let G be an abelian group and let $a, b \in G$ with $x = \text{ord}(a), y = \text{ord}(b)$. Then $\text{ord}(ab) \mid \text{lcm}(x, y)$.*

Observation 2.7 *Let G, H be groups, and let $f : G \longrightarrow H$ be a homomorphism. Then for all $g \in G$, we have that $\text{ord}(f(g)) \mid \text{ord}(g)$.*

Observation 2.8 *Let G be a group, and let $(a, b) \in G \times G$. Then $\text{ord}((a, b)) = \text{lcm}(\text{ord}(a), \text{ord}(b))$.*

Observation 2.9 *Let G be an abelian group, and suppose that there exists $N \in \mathbb{Z}^+$ such that $N \cdot g = \mathbf{0}_G$ for all $g \in G$, where $\cdot$ denotes $\mathbb{Z}$-module action. Then, G is a $\mathbb{Z}_N$-module, where the action is inherited from that of $\mathbb{Z}$.*

We are now ready to complete the proof of Theorem 2.3.

Proof: **(Theorem 2.3)** Recall that $F = f + c$ was an affine map. By assumption, we have that $V \subset F(A)$, and so, by Observation 2.5 we have that $|\langle F(A) \rangle| \geq |\langle V \rangle| \geq 2^n$.

Next, consider the elements $c \in G^n$ and the $v_i \in G^n$. Note that $(v_i - c) \in f(A)$ by assumption. Now let's examine the order of these elements. Define $V' = \{v_i - c\}_{i \in [n]}$. By Observation 2.7, we know that all of the $v_i - c$ have order that divides $|A|$, since they are images of elements of A under a homomorphism. But then, by Observation 2.8, we can see that if $c = (c_1, ..., c_n)$, then all of the c_i must have order that divides $|A|$ as well, or else the order of at least one of the $(v_i - c)$ would have order not dividing $|A|$. Hence c has order dividing $|A|$. Now, by Observation 2.6, we have that the v_i also have order dividing $|A|$. Therefore by Observation 2.9, $\langle V \rangle$ is in fact a $\mathbb{Z}_{|A|}$-module, as of course is $\langle V' \rangle$ since it is in fact the image of some submodule of A (possibly all of A) under a homomorphism. Then, by Lemma 2.4 (with $\langle V \cup V' \rangle$ playing the role of M, if you'd like) we have that

$$\frac{2^n}{|\langle V' \rangle|} \leq |\mathbb{Z}_{|A|}| = |A|$$

and hence

$$2^n \leq |\langle V' \rangle||A| \leq |A|^2$$

so that $|A| \geq 2^{n/2}$, and $\log(|A|) \in \Omega(n)$ as desired.

More specifically, if $A \subset G^m$, then all objects involved are $\mathbb{Z}_{|G|}$-modules, and hence

$$\frac{2^n}{|G|} \leq |f(A)| \leq |G^m|$$

So that $|G|^{m+1} \geq 2^n$ and hence $\log(|G|) \geq \frac{n}{m+1}$. ■

2.4 Functions that Change Multiple Values

We can also generalize this algebraic result to include other types of vectors, where $F(\mathfrak{m}_i)$ has the i-th component non-identity, but possibly some other number of positions are non-identity elements as well. If the function F has the ability to change arbitrary subsets of c elements for a constant c, then our original results clearly apply, as you could re-organize G^n as a product $G^c \times \cdots \times G^c$ with n/c components. (Without loss of generality, we assume $c|n$.) However, the bounds still apply for less powerful classes of functions. We will show that *any* function that produces vectors with $c(n)$ or fewer non-identity positions at a time has communication complexity $\Omega(n/c(n))$, provided only that it is complete- i.e., for every position, it has the ability to produce a vector that is non-identity in that position. Here, $c(n)$ is any positive function of n, and note also that the number of non-identity positions per $\mathfrak{m}_i$ need not be uniform- we only ask that it is bounded by $c(n)$. We'll prove this by showing that we can always re-organize G^n into a product of larger components (of size $c(n)$) so that the original function F produces orthogonal characteristic-type vectors in the original sense, only over $(G^{c(n)})^{n/c(n)}$. Then, the proof follows immediately from the original result. Consider the following lemma.

Lemma 2.10 *Let $c \in \mathbb{Z}^+$. Let $\{S_k\}_{k\in\Gamma}$ be a collection of sets such that $S_k \subseteq [n]$, $|S_k| \leq c$ for all $k \in [n]$ and such that the $\{S_k\}$ form a cover of $[n]$, i.e., $\bigcup_{k\in\Gamma} S_k = [n]$. Then there exists $X \subseteq [n]$ and a sub-collection of sets $\{S_{k_j}\}_{k_j\in\Lambda\subseteq\Gamma}$ such that $S_{k_j} \cap S_{k_{j'}} \cap X = \varnothing$ whenever $j \neq j'$ yet $S_{k_j} \cap X \neq \varnothing$ for at least $\lceil n/c \rceil$ of the sets S_{k_j}.*

Proof: Suppose $\{S_k\}_{k\in\Gamma}$ is such a cover of $[n]$. In this finite case, it is clear that every cover has a minimal sub-cover, i.e., a collection $\{S_{k_j}\}_{j=1}^m$ that still covers $[n]$ such that for any other sub-cover $\{S_{k'_j}\}_{j=1}^{m'}$ we have that $m \leq m'$. [2] Let $\{S_{k_j}\}_{j=1}^m$ be a minimal sub-cover. For this sub-cover, define for every $i \in [n]$

$$N_i = |\{S_{k_j} \mid i \in S_{k_j}\}|$$

Note that $N_i > 0$ for all i since the $\{S_{k_j}\}$ form a cover of $[n]$. Define $X \subset [n]$ as follows:

$$X = [n] \setminus \{i \mid N_i > 1\}$$

Now clearly, X has the property that $S_{k_j} \cap S_{k_{j'}} \cap X = \varnothing$ whenever $j \neq j'$, but it is also true that $S_{k_j} \cap X \neq \varnothing$ for every $j \in [m]$. To see this, suppose that for some $j \in [m]$ we have

[2] Clearly $m < \infty$ since $\mathcal{P}([n])$ is finite, so there is no loss of generality.

$S_{k_j} \cap X = \varnothing$, i.e., that $S_{k_j} \subset [n] \setminus X$. This statement says that every element of S_{k_j} is also in at least one other set in the sub-cover. Hence,

$$S_{k_j} \subset \bigcup_{j' \neq j} S_{k_{j'}}$$

and thus $\{S_{k_{j'}}\}_{j' \neq j}$ is also a subcover of all of $[n]$ that is smaller than our original, contradicting the minimality that we assumed. So, we have that $S_{k_j} \cap X \neq \varnothing$ for every $j \in [m]$. To complete our proof, we simply note that since $|S_k| \leq c$ for every k, any sub-cover must have at least $\lceil n/c \rceil$ sets, just by counting. ■

Corollary 2.11 *Let $n \in \mathbb{Z}^+$ and let G, A be abelian groups. Let $w(x)$ be a positive valued function and let $V = \{v_i\}_{i=1}^n \subset G^n$ be any collection of elements so that the i-th position of v_i is not equal to $\mathbf{0}_G$, and at most $w(n)$ total positions of v_i are non-identity for all $i \in [n]$. Then if $F = f + c$ is an affine map from $A \longrightarrow G^n$ such that $V \subset F(A)$ then we have* $\log(|A|) \in \Omega(n/w(n))$.

Proof sketch: This is an easy consequence of the lemma. In the language of the lemma, set the S_k to be the set of indexes in $[n]$ corresponding to all the positions with non-identity elements in v_k. By the lemma, we can find a subset of the indexes X and a subset of the sets S_k of size at least $n/w(n)$ so that the S_{k_j} are disjoint on X. We can then re-organize the product G^n according to the v_{k_j} and where they differ from the identity. Then, transform $f + c$ into a map on this restricted product of G's, just using the component functions (i.e., the compositions with the projections from $G^n \longrightarrow G$). So, we have some product like:

$$G^{s(k_1)} \times \cdots \times G^{s(k_{n/w(n)})}$$

Then pad $G^{s(k_j)}$ with extra products of G and redefine the maps accordingly (set those components of the new constant c to $\mathbf{0}_G$, and the new components of f to the trivial map. You now have exactly the original situation from Theorem 2.3, only with a new value of n, which happens to be $n/c(n)$. So, we can conclude $\log(|A|) \in \Omega(n/w(n))$. ■

2.5 Polynomials of Bounded Total Degree

Recently, new cryptosystems have been developed with additional homomorphic properties (see [3]), which provide the ability to compute on ciphertext, polynomials of total degree at most 2. Here, we will generalize our original algebraic result to apply to algebraic functions of the form of any polynomial of total degree t, over a ring R. Although the following result will apply to the ring of polynomials over any ring R (it need not have an identity or be commutative), this result has the most meaning in the case of commutative rings with identity, since in this case the ring of multivariate polynomials coincides precisely with our notion of "algebraic formula", which is formalized in a category theoretic sense in Section 4.1. (For a non-commutative ring, there's a more general structure that serves as the set of all formulas.)

Corollary 2.12 *Let $n \in \mathbb{Z}^+$ and let R be any ring. Let $V = \{v_i\}_{i=1}^n \subset R^n$ be any collection of elements so that the j-th position of v_i is not equal to $\mathbf{0}_R$ precisely when $j = i$, for all*

$i, j \in [n]$. Then if $F : R^m \longrightarrow R^n$ is such that $F = (F_1, ..., F_n)$ with each $F_i \in R[X_1, ..., X_m]$ of total degree less than or equal to t (a constant) and has $V \subset F(R^m)$ then we have

$$\left(\sqrt[t]{\log(|R|)}\right) m \in \Omega(\sqrt[t]{n})$$

In particular, if $|R|$ is independent of n, then $m \in \Omega(\sqrt[t]{n})$.

Proof: Let $f \in R[X_1, ..., X_m]$ be of total degree t. We do not assume that R is commutative, so there could be a total of $\sum_{k=0}^{t} m^k = \mathcal{O}(m^t)$ terms. Note that we can simulate f with a polynomial $\overline{f} \in R[Y_1, ..., Y_{\mathcal{O}(m^t)}]$ of total degree 1. I.e., if $N = \sum_{k=0}^{t} m^k$ and

$$f = \sum_{k=1}^{N} \left[r_k \prod_j X_{\alpha_k(j)} \right] + r_0$$

where $\alpha_k : [t] \longrightarrow [m]$, then we'll set

$$\overline{f} = \sum_{k=1}^{N} r_k Y_k + r_0$$

Then of course

$$f(X_1, ..., X_m) = \overline{f}(\prod_j X_{\alpha_1(j)}, ..., \prod_j X_{\alpha_N(j)})$$

The point being that any function which f computes, we can compute with $\overline{f}$. But now observe that $\overline{f} - r_0 \in \mathrm{Hom}_{\mathbb{Z}}((R,+)^{m^t}, (R,+))$. Componentwise applying this to $F = (F_1, ..., F_n)$ as given above, we have a function $\overline{F} : (R,+)^{m^t} \longrightarrow (R,+)^n$ which is an affine map, and which can simulate the functionality of F. Hence, it will of course have the property that $V \subset \overline{F}((R,+)^{m^t})$. Now we can directly apply Theorem 2.3 to this situation since $(R,+)$ is of course an abelian group. Therefore,

$$\log(|R|^{m^t}) \in \Omega(n)$$

and hence $\log(|R|)m^t \in \Omega(n)$ so that

$$\left(\sqrt[t]{\log(|R|)}\right) m \in \Omega(\sqrt[t]{n})$$

as desired. ■

3 Applications of Algebraic Results

We will discuss here a number of protocols which are both easy to state, and would provide desirable functionalities, yet under algebraic assumptions, they cannot be very well implemented with existing technology.

3.1 Private Database Modification (PIR Writing)

As seen in [5], the ability to privately modify an encrypted database in a communication efficient way could provide a valuable tool for private computation. One very natural approach to such a problem, is to proceed in a manner analogous to many PIR protocols, and use homomorphic encryption as a building block.

The protocol would then communicate encrypted values which encode the modification to take place, and then the database owner would execute some algebraic operations on the encrypted database and the modification description to update the database contents. Then, since all of the communication consisted only of encrypted values, CPA-type security comes easily. Unfortunately, we have very limited structures available to homomorphic encryption schemes. Almost always, what is preserved is the operation of an abelian group. At best, the ability to evaluate polynomials of total degree 2 is provided (see [3]). It will follow from our preliminary algebraic results, that these type of algebraic protocols cannot be very well implemented with existing encryption schemes. We'll often speak of "algebraic" maps, which will usually mean functions that are obtainable from some type of formula involving only the operations of the algebraic structure. A precise, formal, and detailed exposition of this idea is given in Section 4, especially Section 4.1.

We've placed some of the formal protocol-type definitions in the appendix to improve the readability, since there isn't much surprising about them, and most readers of this paper could likely re-invent them in a few minutes. We'll instead give an informal description of the protocol here. For precise statements, see Definitions 5.1, 5.2 and 5.3, as well as Definition 5.4 which are discussed at length in the appendix.

Let **U** be a user that wishes to update the database, and denote by **DB** the database owner. We'll summarize a protocol for algebraic database modification between **U** and **DB** via the following steps, in which we assume that G is an abelian group. Below, we'll just describe the algebra involved. In an actual protocol, everything will be computed on ciphertext in some homomorphic encryption scheme over G.

1. **U** selects $\mathfrak{m}_i = (g_1, ..., g_m)$ to modify position i and sends $\mathfrak{m}_i$ to **DB**.
2. **DB** computes an algebraic function $F(X, \mathfrak{m}_i, H)$ of the database $X \in G^n$, the modification description $\mathfrak{m}_i$, and other inputs of his own, $H \in G^\epsilon$.
3. **DB** replaces X by $X' = F(X, \mathfrak{m}_i, H)$

Clearly the algebra involved in this protocol implies the ability to algebraically generate complete sets of characteristic vectors:

Claim 3.1 *An algebraic protocol for database modification over an abelian group implies an algebraic function (affine map) with a complete set of characteristic vectors in the image.*

Proof sketch: Define a database $X = \{\mathbf{0}_G\}_{i=1}^n$, which is the identity in all positions. Apply **DB**'s function to obtain $X' = F(X, \mathfrak{m}_i, H)$ where $\mathfrak{m}_i$ describes a modification for position i. Then clearly $X' = v_i$, a characteristic vector in G^n, non-identity at position i. ■

Therefore, by Theorem 2.3, if we build such a protocol based on a homomorphic cryptosystem over *any* abelian group, it will necessarily have linear communication complexity.

Note the strong sense in which this is true: abelian group formulas always correspond to affine maps, but certainly not every affine map comes from such a formula.[3] Furthermore, Theorem 2.3 did not even assume that the groups were the same. So, even if the database elements are encrypted in some other cryptosystem and over some other group than the descriptions, and even if we were provided the ability to compute all algebraic maps from one to the other on ciphertext, we still couldn't produce a non-trivial protocol over abelian groups.

We'll summarize these ideas as

Corollary 3.2 *There are no non-trivial Algebraic Oblivious Database Modifiers over an abelian group. I.e., any oblivious database modifier based on the operations of an abelian group has communication complexity* $\Omega(n)$.

3.2 Algebraic and Homomorphic Protocols for PIR

As a second corollary, we consider "algebraic", or "homomorphic" protocols for private information retrieval. One may have observed, as the authors have, that the query results for PIR protocols usually fall into one of two categories: either (a) they have no (or very limited) algebraic value[4] or homomorphic properties, or (b) the server side communication is non-constant, i.e., the results of a query return many items, not just an encryption of one value in the database[5]. A protocol for private information retrieval that returns encryptions of single values which retain algebraic and homomorphic properties could be a very useful tool in private computation[6], especially in non-interactive settings. In what follows, we present evidence that the absence of such protocols is not just a random coincidence.

We'll try to establish a basic definition that captures the properties that we desire, and encapsulates most existing work possessing these properties. Suppose that the values in a database have some algebraic structure. For now, say that of an abelian group. We will denote the return value of a PIR query for the i-th position of a database by $\mathrm{PIR}(i)$, which consists of one or more encrypted database elements. Let $S_i = \{s_j\}_{j=1}^{k}$ denote the set of values from the database that are returned by a PIR query for position i.

Suppose for a moment that the domain from which PIR query returns reside has the algebraic structure of a group, say $(G', \star)$. To name just a few, we have the PIR protocols of [14], [6], [7] as examples of such systems. Suppose also that the database elements themselves also come from a domain having the algebraic structure of a group, say $(G, \cdot)$.[7] Then we make the following definition:

[3]For a simple example, consider $G = \mathbb{Z}_p \times \mathbb{Z}_p$ and $\varphi \in \mathrm{Hom}\mathbb{Z}(G, G)$ by $(a, b) \mapsto (b, a)$. So, we've shown that even if we allow **DB** to somehow compute arbitrary affine maps on the ciphertext values, it still does not suffice to accomplish this task.

[4]See the work of [7] for an example of such a PIR protocol having limited algebraic value.

[5]See [14] for such an example, but virtually every PIR based on homomorphic encryption (over an abelian group) has this property.

[6]For example, in the keyword search of [18], the dictionary size could be reduced.

[7]There is no assumption that the group representing the query returns are the same as the database elements, or even that they are encryptions of database elements, exactly. It could be the case that as a part of the encryption, the group that the database elements come from is first homomorphically transformed, and then transformed back as a part of decryption. The general way that we've stated our algebraic results allows us to reason about such a general definition.

Definition 3.3 *Using the notation established above, we say that a PIR protocol is* **homomorphic** *if for a given database $X \in G^n$, we have that $\mathcal{D}(\mathrm{PIR}(i) \star \mathrm{PIR}(j)) = S_i \cdot S_j$ where $\mathcal{D}$ is the function from the PIR protocol that decrypts the query results.*

Note also that for such a PIR protocol to be of much utility as a subroutine in some private computation, it is almost essential to have $|S_i| = 1$, or at least bounded by a small constant. If not, then the party which is to perform a computation on the return values of a homomorphic PIR query will likely not have any information about where the relevant element is in the query results. Hence, if such a party wishes to perform a computation on t variables obtained via homomorphic PIR queries, it would in general require repeatedly performing the computation on all $|S_i|^t$ possible sequences to ensure that the right variables were involved at least once. Furthermore, it may not be possible for any party to distinguish which of the resulting outputs in fact corresponds to the desired computation, even after decryption.

Finally note that from the definition of homomorphic PIR, we see that the results of queries are in fact encryptions of elements in some homomorphic cryptosystem. To create such a PIR protocol, a very natural approach is to manipulate the algebraic structure of some such homomorphic cryptosystem. This motivates the following definition.

Definition 3.4 *We say that a PIR protocol is* **algebraic** *if the following hold:*

1. *A query consists of an ordered sequence of ciphertexts in some cryptosystem where the plaintext set A has some algebraic structure.*

2. *To process a query, the database owner computes on ciphertext some algebraic function of the query's array, this function being determined by the contents of the database to obtain an array of ciphertext which will be the results of the query.*

For precise definitions regarding "algebraic function", please see Section 4, specifically Definition 4.2. For abelian groups, we'll again have affine maps as our model of algebraic functions.

Corollary 3.5 *Consider an abelian group algebraic PIR protocol with sender-side communication complexity $g(n)$ and server-side communication complexity $h(n)$. Then $g(n)h(n) = \Omega(n)$. More specifically, if $k(n)$ is any positive integer-valued function and if the server's response consists of $k(n)$ encrypted values, then the sender-side communication complexity is $\Omega(n/k(n))$.*

Proof: We will show in a straight-forward way that the algebra involved for any such PIR protocol will imply the ability for algebraically generating a complete set of characteristic vectors, and hence, cannot be done with with small communication using the algebra of an abelian group.

Let $\delta_{ij} \in \{0, 1\}$ defined by $\delta_{ij} = 1 \iff i = j$. Define databases $\{X(i)\}_{i=1}^n$ by the formula $X(i)_j = \delta_{ij}\alpha$ where $\alpha \neq 0_G \in G$. Each one of these databases has an associated function $F_i : G^{m+\epsilon} \longrightarrow G^k$, a homomorphism of groups. Again, the ϵ variables provided by the database owner are independent of the m variables that comprise the query. (If not, the

function can of course be re-written to make it so.)[8] Now define maps $f_i \in \mathrm{Hom}_{\mathbb{Z}}(G^{m+\epsilon}, G)$ by $f_i(x) = \sum_{j=1}^{k} F_i(x)_j$ (i.e., we just compose F_i with the map from the sum to G that is guaranteed to exist by the universal mapping property of coproducts). Next, define $F = (f_1, ..., f_n)$ so that $F \in \mathrm{Hom}_{\mathbb{Z}}(G^{m+\epsilon}, G^n)$. Note that in terms of the m variables corresponding to the queries, $F' = f + c$ is an affine map from $G^m \longrightarrow G^n$, again, since the ϵ variables from the database are independent. Now by construction, this affine map has a complete set of characteristic vectors in the image. However, each vector has k non-identity entries. So, by Lemma 2.10 and Corollary 2.11, we see that this protocol for oblivious database modification has communication complexity $\Omega(n/k)$. However, this is precisely the sender-side complexity of the PIR protocol as well. So, since the protocol clearly has server-side complexity $\Omega(k)$ (with a tight bound if the group size is a constant) then we see that the product of the complexities is of course $\Omega(n)$. ■

Using Corollary 2.12, we can generalize this result to cryptosystems that may have additional homomorphic properties (see [3]), showing $\Omega(\sqrt[t]{n})$ bounds if total degree t polynomials over a ring R can be computed on ciphertext.

For example, if given a cryptosystem that allows polynomials of fixed total degree t to be computed on ciphertext over some ring R, we can easily construct an algebraic PIR protocol with sender-side communication $\Theta(\sqrt[t]{n})$ and server-side complexity $\Theta(1)$ (see [3], or section 5 for details of a simple example). However, this is in fact meets a lower bound: In general, if such a protocol has sender-side complexity $g(n)$ and server-side complexity $h(n)$, then we can show that $g(n)h(n) = \Omega(\sqrt[t]{n})$, which is a simple consequence of Corollary 2.12.

3.3 Private Keyword Searching [18]

As another relatively simple corollary, we resolve (under our algebraic assumptions) an open problem posed by Ostrovsky and Skeith [18] regarding extending the query semantics for private searching on streaming data. We show that without new homomorphic encryption schemes with additional properties, their methods cannot be extended to perform conjunctive queries.

Corollary 3.6 *The problem of private keyword search on streaming data as proposed in Ostrovsky and Skeith [18], has no non-trivial algebraic solution for a conjunctive query of two or more terms if the underlying cryptosystem is only group homomorphic over an abelian group.*

Remark: We will assume the same basic framework as developed in [18] for a solution and show that there is no such solution that performs conjunctive queries. Specifically, we assume that a dictionary with an associated array of ciphertexts is used to conditionally encrypt documents as in [18].

Proof: First note that the protocol inherently gives rise to an algebraic method for generating complete sets of characteristic vectors: Suppose that the dictionary D has size m. Each word has its role in the query encoded via an encrypted group element, say in

[8] Also, note again that this contains all possible algebraic formulas of the inputs, as well as potentially a large number of maps that are not necessarily obtainable from such formulas.

some group G. Look at the encoded dictionary (un-encrypted) as the set G^m. Suppose we have a protocol as described in [18] for some query that involves k variables. Running this protocol on m^k documents which run over all unique k-tuples from the dictionary gives us a set of characteristic vectors inside of $G^{(m^k)}$. So, we can think of this as an algebraic map from $G^m \longrightarrow G^{(m^k)}$, which (unless the query is somewhat trivial) contains a complete set of characteristic vectors in the image. But, now the question is how many positions are non-identity in each vector? This of course depends on the query. Suppose that the query is a disjunction of terms. Each vector in $G^{(m^k)}$ will have at least $m^{k-1}k$ positions that are non-identity, since $k-1$ entries could be arbitrary as long as one contains a keyword. So, the ratio of total positions to non-identity positions is less than m and our algebraic lower bounds give no contradiction (which of course should be the case since [18] gives such a construction). But now consider a conjunctive query, just of two terms. In the same way as described above, this gives rise to an algebraic function for characteristic vectors from $G^m \longrightarrow G^{(m^2)}$, however this time we have $\mathcal{O}(1)$ positions of each vector are non-identity. So, applying Corollary 2.11, we see that no such protocol can exist based on an abelian group. More generally, from Corollary 2.12, we see that if given the ability to compute total degree t polynomials, we can construct a protocol that executes a conjunction of *at most* t terms. ■

We believe that this example illustrates particularly well a situation in which the bounds proved in this work are especially useful. The entire method of [18] critically depends on the ability to generate these types of characteristic vectors so that the final representation is an encryption in a homomorphic scheme. This is the case since the functionality of characteristic vectors is used as a subroutine for the larger procedure, and so to continue the computation (i.e., writing to the buffer, etc.) it is necessary that the output have algebraic value. So, since we have proven that this subroutine is impossible to implement in the critically important manner desired, it seems that improving the work of [18] would require either a completely new approach, or new designs of homomorphic encryption schemes, such as fully-homomorphic encryption.

It is this type of information that we hope will save researchers time and effort in the future. Applying these bounds may not give an absolute impossibility, but it can quickly eliminate a very large space of what might otherwise seem to be feasible approaches to the problem.

4 Formalizations Regarding Computation by Algebra

4.1 Unification of Algebraic Formulas

We speak often of the idea of an "algebraic formula" over some algebraic structure, and it will be very convenient to make definitions at such a level of abstraction. Clearly, we need an abstract definition of "algebraic formula" before we can continue. We hope to establish an appropriate definition in this section that applies to a wide variety of categories of algebraic structures.

To begin, we'll examine rings of polynomials in an abstract setting, and see if we can't extract the ideas about it that characterized it as the set of "algebraic formulas". Let R be a commutative ring with identity. This is one of the most natural structures to our intuition,

and we will use it to extend our intuition to other possibly less natural structures. What plays the role of an arbitrary algebraic formula in n variables over R? We can add and multiply, but there isn't much else we can compute from just the operations of R. This leads us towards the ring $R[x_1, ..., x_n]$ serving as the set of all generic algebraic formulas over R. Next note that in this situation, the ring $R[x_1, ..., x_n]$ satisfies an interesting universal mapping property: we always have an *evaluation map* that takes an assignment of elements to variables and gives us a homomorphism from $R[x_1, ..., x_n] \longrightarrow R$ that "evaluates" each polynomial using that assignment of variables. This evaluation map is exactly what we are after when talking about a general algebraic formula. More formally, and a bit more abstractly, consider the following claim (see [12] for details) which we state here without proof:

Claim 4.1 *Let R, S be commutative rings with identity. Let $\varphi : R \longrightarrow S$ be a homomorphism and let $\alpha : X = \{x_i\}_{i=1}^n \longrightarrow S$ be a set map. Then there exists a unique homomorphism $\overline{\varphi} : R[x_1, ..., x_n] \longrightarrow S$ such that*

1. $\overline{\varphi}|_X = \alpha$, *and*
2. $\overline{\varphi}|_R = \varphi$

Here, we identified the set of variables, X, and the ring R as subsets of the ring of polynomials $R[X]$. More generally, we'll have canonical injections $\iota_X : X \hookrightarrow R[X]$ and $\iota_R : R \hookrightarrow R[X]$. Then the conditions on $\overline{\varphi}$ become $\alpha = \overline{\varphi} \circ \iota_X$ and $\varphi = \overline{\varphi} \circ \iota_R$, rather than the restrictions being equal.

As it turns out this mapping property is exactly what we need to characterize algebraic formulas in general. We'll see in a moment that any such object is in fact unique up to isomorphism. We begin with the following definition.

Definition 4.2 *Let $\mathfrak{C}$ be a concrete category. Let A be an object in $\mathfrak{C}$, and let X be a set. We define the object of* A-algebraic formulas over X *to be an object $F_A[X]$ in $\mathfrak{C}$ together with maps $\iota_X : X \longrightarrow F_A[X]$ and $\iota_A : A \longrightarrow F_A[X]$ such that for any object B in $\mathfrak{C}$, morphism $\varphi : A \longrightarrow B$ and set map $\alpha : X \longrightarrow B$, there exists a unique morphism $\overline{\varphi} : F_A[X] \longrightarrow B$ such that*

1. $\alpha = \overline{\varphi} \circ \iota_X$ *and*
2. $\varphi = \overline{\varphi} \circ \iota_A$

The following uniqueness theorem justifies our giving a specific name to such an object as above.

Theorem 4.3 *Let A be an object in a concrete category $\mathfrak{C}$. Suppose that (F, X, ι_X, ι_A) and $(F', X', \iota'_{X'}, \iota'_A)$ are both objects of A-algebraic formulas over X, X' respectively, and suppose that there exists a bijection $\beta : X \hookrightarrow\!\!\!\!\rightarrow X'$. Then F, F' are equivalent objects in $\mathfrak{C}$.*

Proof: We need to show that there are morphisms in $\mathrm{Hom}(F, F')$ and $\mathrm{Hom}(F', F)$ such that the composition is the identity morphism. Consider the map $\alpha : X \longrightarrow F'$ defined by $\alpha = \iota_{X'} \circ \beta$ and define $\varphi = \iota'_A$. Since F is a set of A-algebraic formulas over X, this gives us a unique morphism $\overline{\varphi} : F \longrightarrow F'$ such that

- $\iota_{X'} \circ \beta = \overline{\varphi} \circ \iota_X$ and
- $\iota'_A = \overline{\varphi} \circ \iota_A$

And symmetrically, we have a map $\overline{\varphi}' : F' \longrightarrow F$ such that

- $\iota_X \circ \beta^{-1} = \overline{\varphi}' \circ \iota_{X'}$ and
- $\iota_A = \overline{\varphi}' \circ \iota'_A$

But now, consider the map $\overline{\varphi}' \circ \overline{\varphi} \in \mathrm{Hom}(F, F)$. Note that $\overline{\varphi}' \circ \overline{\varphi} \circ \iota_X = \overline{\varphi}' \circ \iota_{X'} \circ \beta$ and $(\overline{\varphi}' \circ \iota_{X'}) \circ \beta = \iota_X \circ \beta^{-1} \circ \beta = \iota_X = \iota_X \circ \beta^{-1} \circ \beta = \iota_X$. Very similarly, $\overline{\varphi}' \circ \overline{\varphi} \circ \iota_A = \overline{\varphi}' \circ \iota'_A = \iota_A$. So, composing $\overline{\varphi}' \circ \overline{\varphi}$ with ι_X, ι_A gives us back ι_X, ι_A, respectively. However, applying the universal property of F to itself, using ι_X in the place of α and ι_A in the place of φ, we see that there is a unique map with the properties we've just demonstrated $\overline{\varphi}' \circ \overline{\varphi}$ to have. But, note of course that the identity morphism, $\mathbf{1}_F$, is also such a map, and therefore $\overline{\varphi}' \circ \overline{\varphi} = \mathbf{1}_F$. Similarly, we can show that $\overline{\varphi} \circ \overline{\varphi}' = \mathbf{1}_{F'}$, and hence $\overline{\varphi}$ is an equivalence of objects in $\mathfrak{C}$. I.e., up to isomorphism, the set of A-algebraic formulas (on bijective sets of variables) is unique. ■

4.1.1 Examples

Here we'll give a short list of examples to illustrate the definition, and to (hopefully) see that it very well matches our intuition about what algebraic formulas should be. Many of the incarnations of these objects will involve free objects, a somewhat related idea, and as such, we'll begin with a brief account of free objects before stating the specific examples.

Recall that a *free object* in a concrete category $\mathfrak{C}$ (one in which the objects can be thought of as sets) is an object F with a set X and a map $\iota : X \longrightarrow F$ such that for any other object A of $\mathfrak{C}$ and for any set map $\phi : X \to A$, there is a unique morphism of $\mathfrak{C}$, $\overline{\phi} : F \to A$ such that $\overline{\phi} \circ \iota = \phi$. In this case we say that F is free on the set X.

It is sometimes convenient to identify the set X with its image $\iota(X)$ in F, and then the mapping property becomes $\overline{\phi}|_X = \phi$.

Perhaps the most commonly known example is that of a vector space V over a field $\mathbb{F}$. In this language, we have that V is a free object in the category of all $\mathbb{F}$-modules (it is free on any basis). Another simple example is that $\mathbb{Z}^n$ is a free object in the category of abelian groups. It is easy to verify that it is free on the set of vectors $\{(1, 0, ..., 0), ..., (0, ..., 0, 1)\}$, for example.

Now, let's move on to some concrete examples.

Example 4.4 *(Commutative Rings.) If R is a commutative ring with identity, we have that the set of all R algebraic formulas on variables $\{x_i\}_{i=1}^n$ is simply $R[x_1, ..., x_n]$.*

It is quite straightforward to construct the needed maps. As mentioned, the "evaluation" map is what does the trick. All that is needed is a proper formalization, which can be found in [12]. ✓

This seems to match our intuition very well regarding algebraic formulas. However, what about a field $\mathbb{F}$? This is of course a commutative ring with identity, but it seems like we could

accomplish more with division, and perhaps rational functions would correspond better to algebraic formulas. But notice a critical difference in this case: if the denominator has roots in $\mathbb{F}$, then we cannot freely assign variables to values and evaluate such a function on that assignment. This could have serious implications in cryptographic computation as well- in this scenario, it may be the case that attempting such an evaluation which leads to "division by zero" will produce some distinguishable behavior in a method that shouldn't disclose any information about the underlying values.

Note that in non-commutative rings, the set of formulas is more complex than just $R[x_1, ..., x_m]$ since coefficients and variables cannot always be written so concisely. We'll see this in more detail later on.

Example 4.5 *(Groups.) For a group G, the set of all G-algebraic formulas on a set of variables X is the free product of the free group on X with G, i.e., $F(X) * G$.*

A general formula seems to correspond to a word in a free group, only mixed in with elements of G, which lines up exactly with $F(X) * G$. Using the universal mapping property of free objects and that of the free product of groups (which is a coproduct in groups), it is easy to verify the mapping property for G-algebraic formulas. From the free object $F(X)$, we have that any map $\alpha : X \longrightarrow A$ for some group A leads to a unique map from $F(X) \longrightarrow A$ that agrees with α. Then since $F(X) * G$ is a coproduct, we have that given a map of the two components to A there is a unique map from the free product to A that agrees with all previous maps. Putting these together, we see that it suffices to give a map from $X \longrightarrow A$ and a map from $G \longrightarrow A$ to uniquely determine a map from the free product to A, which implies that $F(X) * G$ is the set of G-algebraic formulas. ✓

Example 4.6 *(Abelian Groups.) For an abelian group G, the set of G-algebraic formulas over n variables is simply $\mathbb{Z}^n \oplus G$.*

This follows exactly as in the preceding example since $\mathbb{Z}^n$ is a free object in the category of abelian groups, and since $\oplus$ is a coproduct in abelian groups. ✓

Recall that in our applications, we modeled abelian group algebraic formulas by affine maps, and of course, every element in $\mathbb{Z}^n \oplus G$ corresponds to an affine map. So, we've validated our initial intuition and model for this situation: Indeed, (a subset of) affine maps play the very same role for abelian groups as polynomials do for commutative rings. Of course, not every affine map comes from such a formula, so in algebraic terms, the model we analyzed is actually stronger.

Example 4.7 *(R-Modules.) More generally than abelian groups, we have R-modules. In a very analogous way, we see that for an R-module M, the set of M-algebraic formulas over n variables is $R^n \oplus M$.*

As a final example, we'll take a moment to illustrate what is meant by evaluating such a formula, which will let us speak of classes of functions say from $A^m \longrightarrow A$ which correspond to formulas in $F_A[\{x_1, ..., x_m\}]$ (which we'll often write as $F_A[x_1, ..., x_m]$, or just as $F_A[X]$ when it is clear how many variables are involved, and when it is unnecessary to explicitly name them).

Example 4.8 *Every formula $\sigma \in F_A[x_1, ..., x_m]$ can be associated with a function $f_\sigma : A^m \longrightarrow A$ as follows. Let $\varphi = \mathbf{1}_A$, the identity morphism on A, and let $y = (y_1, ..., y_m) \in A^m$. Let $\alpha : X \longrightarrow A$ be the map that sends $x_i \mapsto y_i$ for all $i \in [m]$. Then the pair (φ, α) determines a unique map $\overline{\varphi}_y : F_A[x_1, ..., x_m] \longrightarrow A$, as in Definition 4.2. We define*

$$f_\sigma(y) = \overline{\varphi}_y(\sigma)$$

It will in fact be useful to give a name to such functions that arise from some $\sigma \in F_A[X]$.

Definition 4.9 *Let A be an object in a concrete category $\mathfrak{C}$ where $F_A[X]$ exists for all finite sets X. A function $f : A^m \longrightarrow A$ is said to be* A-algebraic *if $f = f_\sigma$ for some $\sigma \in F_A[x_1, ..., x_m]$ as described above. We will denote the set of all A-algebraic functions with variables in X by $\overline{F}_A[X]$.*

Note: we may also refer to a function $f : A^m \longrightarrow A^n$ as A-algebraic, in which case it is meant that for each $i \in [n]$ there exist $\sigma_i \in F_A[x_1, ..., x_m]$ such that $f = (f_{\sigma_1}, ..., f_{\sigma_n})$. These functions will be denoted by $\overline{F_A^n}[X]$, or simply $\overline{F}_A[X]$ if the context is clear.

4.2 Towards Other Unifying Formalizations

We've now established what seems to be a very good idea of "algebraic formula". However, this is just a statement about functions from one algebraic set to another. The basic functionalities investigated in this work were not algebraic: they were just functions involving general sets. So, how do we model an arbitrary function from one set to another via some algebraic function? There are quite a few possible ways to formalize this idea, and a number of them actually turn out to be meaningful. Some quite plausible ideas however, turn out to be completely void of meaning. We will explore these ideas in some detail now, which will hopefully give us a pleasant and precise vocabulary to talk about such ideas.

In what follows, S will be a set, and A will be some algebraic structure from a concrete category $\mathfrak{C}$, e.g., groups, rings, modules, etc. We'll assume that $F_A[X]$ exists in $\mathfrak{C}$ for any finite set X, which as we've seen in the examples above, is usually the case for the algebraic objects we study. We would like to formalize the idea that a function on the set S can be somehow computed using an algebraic structure. To begin, we offer the following definition.

Definition 4.10 *Let $f : S^m \longrightarrow S^n$. We say that f is* Algebraically Representable *over A, some algebraic structure, if we have the following:*

- $\{i_k : S \longrightarrow A\}_{k=1}^m$
- $\{p_j : A \longrightarrow S\}_{j=1}^n$
- $\tilde{f} : A^m \longrightarrow A^n$

such that

$$f = p \circ \tilde{f} \circ i$$

where $\tilde{f} \in \overline{F_A^n}[x_1, ..., x_m]$.

Note that if $m = n = 1$, then this definition becomes completely meaningless. If this were the case, then all of the information of f can be encoded into i or p (or both) and have absolutely nothing to do with the algebra of A. However, with m and n greater than one, it is a very useful definition that in fact captures a great many situations.

However, if in fact such a representation *does* have meaning at the level of $m = n = 1$, then we have something very special. In this case, all information about f is represented in $\tilde{f}$ in a very strong and complete way, which gives us great flexibility. Consider the following definition.

Definition 4.11 *We say that a function $f : S^m \longrightarrow S^n$ is* Composably Representable *over A for an algebraic structure A, if there exist the following:*

- $i : S \hookrightarrow U \subseteq A$
- $p : U \longrightarrow\!\!\!\!\!\rightarrow S$
- $\tilde{f} : A^m \longrightarrow A^n$

Such that

1. $\tilde{f} \in \overline{F_A^n}[X]$ *with* $\tilde{f}(U^m) \subset U^n$
2. $i \circ p = \mathbf{1}_S$ *(extending p, i to the sum, as usual)*
3. $f = p \circ \tilde{f} \circ i$
4. $[a]_p = [a']_p \Longrightarrow [\tilde{f}(a)]_p = [\tilde{f}(a')]_p$, *where $[a]_p = \{y \in U^m \mid p(y) = a\}$ represents the equivalence class of $a \in U^m$ under the relation defined by the fibers of p (extended to sums).*

With this definition, it is easy to see that the behavior of f is found completely inside of the behavior of $\tilde{f}$, just re-labeling elements, and grouping them together with the fibers of p. If one believes the axiom of choice, then this definition is equivalent to the following condition (but in any case the above condition implies the one below):

Claim 4.12 *Suppose $f : S^m \longrightarrow S^n$ is composably representable over A via a function $p : U \subseteq A \longrightarrow\!\!\!\!\!\rightarrow S$ and an algebraic function $\tilde{f} : U^m \longrightarrow U^n$. Then,* for any *$i : S \hookrightarrow U$ with $p \circ i = \mathbf{1}_S$ we have that*

$$f = p \circ \tilde{f} \circ i$$

Proof: Let $i' : S \hookrightarrow U$ be any function such that $i' \circ p = \mathbf{1}_S$. By definition, there exists $i : S \hookrightarrow U \subseteq A$ and $\tilde{f}$ such that $f = p \circ \tilde{f} \circ i$. Let $s \in S$. Then by definition, $[i(s)]_p = [i'(s)]_p$, and then by the definition of composably representable, we have that $[i(f(s))]_p = [\tilde{f}(i(s))]_p = [\tilde{f}(i'(s))]_p$ so that indeed, $(p \circ \tilde{f} \circ i')(s) = f(s)$ as desired. ■

As mentioned above, the condition of the claim is easily seen to be equivalent to the definition of composably representable, but in the infinite case, this requires the axiom of choice to construct functions i by selecting an element from each fiber of p.

The following claim shows the motivation for our choice of nomenclature.

Claim 4.13 *If $f : S^m \longrightarrow S^m$ is composably representable over A by $(p, i, \tilde{f})$, then for any $k \in \mathbb{Z}^+$, f^k is composably representable over A by $(p, i, \tilde{f}^k)$. In particular, for any $i : S \hookrightarrow U \subseteq A$ such that $p \circ i = \mathbf{1}_S$ we have*

$$f^k = p \circ \tilde{f}^k \circ i$$

Proof: Suppose that $k = 1$. Then this is trivially true from the definition. Now, suppose that the theorem is true for all integers less than or equal to k. So, we have that $[a]_p = [a']_p \Longrightarrow [\tilde{f}^k(a)]_p = [\tilde{f}^k(a')]_p$. Next, notice that by the definition, and by our inductive hypothesis, $[\tilde{f}^r(i(s))]_p = [i(f^r(s))]_p$ for all $r \in [k]$ and hence

$$[\tilde{f}^{k+1}(i(s))]_p = [\tilde{f}^k(\tilde{f}(i(s)))]_p = [\tilde{f}^k(i(f(s)))]_p = [i(f^k(f(s)))]_p = [i(f^{k+1}(s))]_p$$

Therefore,

$$f^{k+1} = p \circ \tilde{f}^{k+1} \circ i$$

which completes the proof. ■

For simplicity, we consider the case of $f : S^m \longrightarrow S^m$, but in fact in the general case, if any function f is composably representable by $\tilde{f}$, then any composition of f's which makes sense can be computed by the analogous composition of $\tilde{f}$'s. For example, if we could *composably* represent $f : \{0,1\}^2 \longrightarrow \{0,1\}$ where $f = \text{NAND}(a, b)$, using $\tilde{f}$, then we could also composably represent $\text{OR}(a, b)$ by $\tilde{f}(\tilde{f}(a, a), \tilde{f}(b, b))$.

In what follows, we provide a few simple examples.

Example 4.14 *Let G be a group and $H < G$ such that $|G : H| = 2$. Then the function $f : \{0,1\}^2 \longrightarrow \{0,1\}$ defined by $(x, y) \mapsto x + y \bmod 2 = x$ XOR y is always* composably *representable over G as follows: let $i(0) = e, i(1) = g \in G \setminus H$, and let $p(H) = \{0\}$ and $p(G \setminus H) = \{1\}$, and finally, let $\tilde{f}(x, y) = x \cdot y$ where $\cdot$ is the operation of the group G. It is easy to verify the equality*

$$f(x, y) = x + y = p \circ \tilde{f} \circ i$$

is satisfied, no matter what representatives are chosen from the fibers of p, since subgroups if index 2 are always normal (right cosets must coincide with left cosets).

Example 4.15 *Let $X \in G^m$ and define the equality function $f_X : G^m \longrightarrow \{0,1\}$ by*

$$f_X(Y) = \begin{cases} 1 & \text{if } X = Y \\ 0 & \text{otherwise} \end{cases}$$

This function is not algebraically representable over any abelian group of size $|G|$ (e.g., G itself).

Question: for $f : \{0,1\}^m \longrightarrow \{0,1\}$, what is the minimum size (as a function of m) of an abelian group G that can algebraically represent f?

Example 4.16 *For primes p, q with $q \nmid p - 1$, we have $[\mathbf{CR}(\mathbb{Z}_p) = \mathbf{CR}(\mathbb{Z}_q)] \iff p = q$. (Here $\mathbf{CR}(G)$ denotes the set of composably representable functions over an algebraic structure G.)*

Example 4.17 *Let G be a group. Any function $f : \{0,1\}^m \longrightarrow \{0,1\}^n$ is composably representable over G if a function $A : G \times G \longrightarrow G$ such that $A(g,h) = \mathbf{1}_G \iff (g = \mathbf{1}_G$ or $h = \mathbf{1}_G)$ can be algebraically computed in G.*

Example 4.18 *Any function $f : \{0,1\}^m \longrightarrow \{0,1\}^n$ is composably representable over R, where R is any domain. Why: You already have a group (abelian in fact) and now the multiplication of R is exactly the function A from the previous example.*

Example 4.19 *(Private Information Retrieval.)*

For a PIR protocol, one can define the underlying functions for query processing as $f_X : S^q \longrightarrow S^r$, indexed by $X = S^n$ such that there exist a collection $\{Q_i\}_{i=1}^n \subset S^q$ where $f_X(Q_i)_{h(i)} = X_i$, where $Q_i \in S^q$ represents the user's query, $X \in S^n$ represents the database, and h is some function that determines which part of the response is the desired database value X_i. The original work of [14] on computational PIR (at the basic level) gives an explicit algebraic representation of such a set of functions $\{f_X\}$ over $\mathbb{Z}_N$ where each f_X is represented with the same functions (p,i) but of course with different $\tilde{f}_X$. The work presented in Section 3 shows that if $qr < n$, then no such set of functions can be algebraically represented over any abelian group (with uniform (p,i)). ✓

In the following subsection, we'll devote a little extra attention to an interesting example.

4.3 Finite Non-Abelian Simple Groups and Composable Representability

We will prove a few elementary lemmas, and then the main result (which again, uses only basic techniques from algebra). We begin, however, with some famous results that will be essential to our result. Recall that a group G is referred to as *simple* if it has no proper normal subgroups.

Theorem 4.20 *(Feit-Thompson) Every non-abelian simple group has even order.*

The proof of this theorem is on the order of 250 pages. For that, among other reasons, we will state this theorem without proof.

Theorem 4.21 *(Cauchy) Let G be a finite group, and let p be a prime integer such that $p||G|$. Then G contains an element of order p.*

Proof of this theorem can be found in virtually any undergraduate or graduate text on algebra. (See [11] or [12] for a nice proof by J.H. McKay.) Combining the above results, we see that

Fact 4.22 *Every non-abelian simple group of finite order has an element of order 2.*

This fact is the only dependence of this work on previous results.

Recall that a group G is called *perfect* if its commutator subgroup $\langle [G,G] \rangle$ is in fact all of G. Since the commutator subgroup is always normal (it is in fact fully invariant), and since the commutator subgroup of a non-abelian group G is also always non-trivial, we see

that every non-abelian simple group is perfect. In general, for any subsets $X, Y \subset G$, we will denote by $[X, Y]$ the set of all commutators in X, Y, i.e., $[X, Y] = \{xyx^{-1}y^{-1} \mid x \in X, y \in Y\}$. We will need a similar, but stronger condition than perfect, which we can derive from the property of being simple. Note that these results do not apply to all perfect groups. For example, the perfect group $SL_2(\mathbb{F}_5)$ cannot be shown to have the ability of performing all computation by the means given here.

Lemma 4.23 *Let G be a finite group and suppose that $S \subset G$ is conjugation invariant (i.e., $\forall s \in S, g \in G$ we have $gsg^{-1} \in S$). Then $\langle S \rangle \lhd G$.*

Proof: Let $x \in \langle S \rangle$. Then $x = s_1 s_2 \cdots s_k$ for some $k \in \mathbb{Z}$. Let $g \in G$ be an arbitrary element. Observe that

$$\begin{aligned} gxg^{-1} &= g(s_1 s_2 \cdots s_k)g^{-1} \\ &= gs_1(g^{-1}g)s_2(g^{-1}g)\cdots s_{k-1}(g^{-1}g)s_k g^{-1} \\ &= (gs_1g^{-1})(gs_2g^{-1})\cdots(gs_kg^{-1}) \\ &= s_1' s_2' \cdots s_k' \in \langle S \rangle \end{aligned}$$

since all $s_i' \in S$ by our assumption. Therefore, $\langle S \rangle \lhd G$ as desired. ■

Now, let us consider for a moment conjugacy classes. For an element $x \in G$, we will denote the conjugacy class by $\mathrm{Cl}_G(x)$. I.e.,

$$\mathrm{Cl}_G(x) = \{y \in G \mid y = gxg^{-1} \text{ some } g \in G\}$$

Recall that we can define a natural action of G on $\mathrm{Cl}_G(x)$ for any $x \in G$: for all $s \in \mathrm{Cl}_G(x)$, simply define $g \cdot s = gsg^{-1}$.

Now, let G be a non-abelian simple group of finite order. From Cauchy's theorem, we know that there exists $x \in G$ such that x has order 2. Consider $\mathrm{Cl}_G(x)$. Let $|\mathrm{Cl}_G(x)| = k$. It must be the case that $k > 1$. If not, then every element of G conjugates x to itself, and hence we have $x \in \mathbf{Z}(G)$, the center of G. But of course this is impossible since the center of a group is always normal and we assumed that G is simple. So, the conjugacy class of x has at least two elements. Recall next, that whenever a group acts on a set S of size k, there is an induced homomorphism,

$$\varphi : G \longrightarrow S_k$$

Since the action of G on $\mathrm{Cl}_G(x)$ is obviously transitive, and since the size k of the class of x is greater than 1, we see that φ cannot be the trivial homomorphism which sends all elements to the identity, and hence $\ker(\varphi) \neq G$. But, since G is simple, we in fact know that $\ker(\varphi)$ must be the trivial subgroup $\{e\}$, since the kernel is always normal. *Therefore, every element of G acts non-trivially on the set* $\mathrm{Cl}_G(x)$.

We will extract exactly the useful information into the following lemma which we have just now proved.

Lemma 4.24 *Let G be a finite, non-abelian simple group, and let $x \in G$ be an element of order 2. Then there exists an element $y \in \mathrm{Cl}_G(x)$ such that $yxy^{-1} \neq x$, and hence, such that $[x, y] \neq e$.*

Using these facts, we can now state and prove the following:

Theorem 4.25 *Let G be a finite non-abelian simple group. Then any function $f : \{0,1\}^m \longrightarrow \{0,1\}^n$ is composably representable over G.*

Proof: We will simply show that the function $\text{NAND}(a,b)$ is computable in this way, which suffices to prove the theorem since any such function $f : \{0,1\}^m \longrightarrow \{0,1\}^n$ can be written in terms of compositions of NAND alone. More precisely, we will show that for an element x of order 2, the set $\{e, x\}$ can be identified with $\{0,1\}$ respectively, and the operation NAND can be computed solely in terms of the group operation of G.

So, to begin, let $x \in G$ be of order 2, which as we discussed exists by Cauchy's theorem. Define $C = \text{Cl}_G(x)$. As discussed, $|C| > 1$. Consider $S = [C, C]$, the set of *commutators in* C. Note that the subset S is conjugation invariant since it is generated by $C = \text{Cl}_G(x)$, which is quite clearly conjugation-invariant. Hence by Lemma 4.23, the subgroup generated by *these specific commutators*, is a normal subgroup:

$$\langle S \rangle = \langle [C, C] \rangle \lhd G$$

However, by Lemma 4.24, we know that $|S| > 1$, as there are at least 2 non-commuting elements. But, we assumed that G was simple. Therefore, we have in fact that $\langle S \rangle = G$. So, in particular, there exists some product, $s_1 s_2 \cdots s_k$ of commutators in C such that

$$s_1 s_2 \cdots s_k = x$$

So, each $s_i = [r_i, t_i]$ where r_i and t_i are both conjugate to x. Therefore we have sequences of group elements, $\{g_i\}_{i=1}^k$ and $\{h_i\}_{i=1}^k$ such that

$$[g_i x g_i^{-1}, h_i x h_i^{-1}] = s_i$$

We are now ready to define our $\text{NAND}(a,b)$. First, define the function $\text{AND}(a,b)$ as follows:

$$\text{AND}(a,b) = \prod_{i=1}^{k} [g_i a g_i^{-1}, h_i b h_i^{-1}]$$

It is now easy to observe that it performs the appropriate function on our inputs from $\{e, x\}^2$. Whenever a or b is set to the identity, every commutator will of course be the identity since all elements commute with e. However, if both a and b are set to the group element x, the by our design, we will have $\text{AND}(x, x) = x$, exactly as desired. Now, since x has order 2, we can simply define $\text{NAND}(a,b) = \text{AND}(a,b)x$. This completes the proof. ■

Corollary 4.26 *Constructing a fully homomorphic encryption scheme over a ring with identity is equivalent to constructing a group homomorphic encryption over* any *finite non-abelian simple group. In particular, it is equivalent to constructing a homomorphic encryption scheme over A_5, the smallest such group.*

To better illustrate the proof, we will provide an explicit construction for the group A_5, which is of course the smallest non-abelian simple group.

Example 4.27 *A_5 can compute* NAND.

First, we need an element of order 2. As we've seen, any such element will suffice. For example, let $x = (1,2)(3,4)$. We know that commutators of conjugates of x will generate all of A_5, in particular, x itself. But to simplify things a bit, we'll first construct standard generators of A_5 out of such commutators, and then write down x in terms of the standard generators. For generators, we'll use $A_5 = \langle\{X, Y\}\rangle$ with $X = (1,2,3,4,5)$ and $Y = (3,4,5)$. Let's begin. Note that

$$(3,5,4)(1,2)(3,4)(3,4,5) = (1,2)(3,5)$$

$$(2,4,3)(1,2)(3,4)(2,3,4) = (1,4)(2,3)$$

and that

$$[(1,2)(3,5),(1,4)(2,3)] = ((1,2)(3,5)(1,4)(2,3))^2 = (1,2,3,4,5)$$

So, we'll set $g_1 = (3,5,4)$ and $h_1 = (2,4,3)$. Next, note that

$$(3,4,5)(1,2)(3,4)(3,5,4) = (1,2)(4,5)$$

and that

$$[(1,2)(4,5),(1,2)(3,4)] = ((1,2)(4,5)(1,2)(3,4))^2 = (3,4,5)$$

So, we'll let $g_2 = (3,4,5)$ and $h_2 = e$. Next we'll write $x = (1,2)(3,4)$ in terms of our generators:

$$(1,2)(3,4) = X^{-1}Y^{-1}X^{-1}Y^{-1}X^2$$

Finally, using the simple observations that $[x,y]^{-1} = [y,x]$, and that if a, b have order two then they are their own inverses, we can write down an explicit expression for $\mathrm{NAND}(a,b)$ in terms of the group operation alone, where $a, b \in \{e, x\}$:

$$\mathrm{NAND}(a,b) = \begin{array}{l} h_1 b h_1^{-1} g_1 a g_1^{-1} h_1 b h_1^{-1} g_1 a g_1^{-1} b g_2 a g_2^{-1} b g_2 a g_2^{-1} h_1 b h_1^{-1} g_1 a g_1^{-1} h_1 b h_1^{-1} g_1 a g_1^{-1} \backslash \\ b g_2 a g_2^{-1} b g_2 a g_2^{-1} g_1 a g_1^{-1} h_1 b h_1^{-1} g_1 a g_1^{-1} h_1 b h_1^{-1} g_1 a g_1^{-1} h_1 b h_1^{-1} g_1 a g_1^{-1} h_1 b h_1^{-1} x \end{array}$$

✓

References

[1] D. Barrington. Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC. STOC 1986: 1-5

[2] D. Boneh, G. Crescenzo, R. Ostrovsky, G. Persiano. Public Key Encryption with Keyword Search. EUROCRYPT 2004: 506-522

[3] D. Boneh, E. Goh, K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. TCC 2005: 325-341

[4] D. Boneh, R. Lipton. Searching for Elements in Black Box Fields and Applications. In Advances in Cryptology-Crypto'96, LNCS1109, pp. 283-297, Springer-Verlag, 1996.

[5] D. Boneh, E. Kushilevitz, R. Ostrovsky, W. Skeith. Public Key Encryption that Allows PIR Queries. Manuscript, 2005.

[6] Y. C. Chang. Single Database Private Information Retrieval with Logarithmic Communication. ACISP 2004

[7] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 1999.

[8] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of the 36th Annu. IEEE Symp. on Foundations of Computer Science*, pages 41–51, 1995. Journal version: *J. of the ACM*, 45:965–981, 1998.

[9] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory, v. IT-31, n. 4, 1985, pp469472 or CRYPTO 84, pp1018, Springer-Verlag.

[10] S. Goldwasser and S. Micali. Probabilistic encryption. In J. Comp. Sys. Sci, 28(1):270–299, 1984.

[11] I. N. Herstein. Abstract Algebra. Prentice-Hall, 1986, 1990, 1996.

[12] T. W. Hungerford. Algebra. Springer-Verlag, Berlin, 1984.

[13] Y. Ishai, E. Kushilevitz, R. Ostrovsky. Sufficient Conditions for Collision-Resistant Hashing. In Proceedings of the Second Theory of Cryptography Conference (TCC-2005)Springer-Verlag Lecture Notes in Computer Science, 2005.

[14] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of the 38th Annu. IEEE Symp. on Foundations of Computer Science*, pages 364–373, 1997.

[15] H. Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. IACR ePrint Cryptology Archive 2004/063

[16] U. Maurer and S. Wolf. Lower bounds on generic algorithms in groups. In Advances in Cryptology – EUROCRYPT '98, number 1403 in Lecture Notes in Computer Science, pages 72–84.

[17] W. Maurer and J. Rhodes. A property of finite non-Abelian simple groups. In proc. Am. Math. Soc., vol. 16, pages 522-554 (1965).

[18] R. Ostrovsky and W. Skeith. Private Searching on Streaming Data. In *Advances in Cryptology – CRYPTO 2005*

[19] R. Ostrovsky and W. Skeith Computational Private Information Retrieval: A Survey. Manuscript, 2006.

[20] P. Paillier. Public Key Cryptosystems based on CompositeDegree Residue Classes. Advances in Cryptology - EUROCRYPT 99, LNCS volume 1592, pp. 223-238. Springer Verlag, 1999.

[21] R. L. Rivest, L. Adleman and M. L. Dertouzos, On data banks and privacy homomorphisms, In Foundations of Secure Computation, eds. R. A. DeMillo et al., Academic Press, 1978, pp. 169-179.

[22] R. L. Rivest, A. Shamir, and L. Adleman A method for obtaining digital signatures and public key cryptosystems, Commun. ACM 21 (1978), 120126.

[23] T. Sander, A. Young, M.Yung. Non-Interactive CryptoComputing For NC1 FOCS 1999: 554-567

[24] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In Eurocrypt '97, LNCS 1233, pages 256–266. Springer-Verlag, 1997.

[25] H. Werner. Finite simple non-Abelian groups are functionally complete. In Bull. Soc. Roy. Sci. Liège, vol. 43, pp. 400, (1974)

5 Appendix

5.1 Notations

The natural numbers will be denoted $\mathbb{N}$, and the integers by $\mathbb{Z}$. For $n \in \mathbb{Z}$, the symbol $\mathbb{Z}_n$ will denote the ring $\mathbb{Z}/n\mathbb{Z}$, or the group $(\mathbb{Z}/n\mathbb{Z}, +)$. We will sometimes denote the set of integers $\{1, 2, ..., n\}$ by $[n]$ for simplicity. If G is a group, then $\mathbf{0}_G$ or $\mathbf{1}_G$ will represent the identity element of G, depending on whether additive or multiplicative notation is being used for the operation of G. The symbol $\times$ will be used to denote a direct product (in sets, groups, rings, modules, etc.), and if X is a set (or group, ring, module...) then X^n represents the direct product of n copies of X. Occasionally, if A is a subset of a group (ring, module, etc.) the symbol $\langle A \rangle$ will denote the subgroup (sub-ring, sub-module, etc.) that is generated by A. I.e., the intersection of all sub-structures containing A. However, we adhere to standard notations in more specific situations. Let R be a ring and let M be an R-module. If A and B are sub-modules of M, then we denote the sum of A and B as $A + B$. We will denote the external direct sum of any two R-modules A, B by $A \oplus B$. For any $a \in M$, Ra will denote the submodule of M defined by $Ra = \{ra \mid r \in R\}$, so that if $1 \in R$ and M is unitary then $\langle \{a\} \rangle = Ra$. The set of all R-module homomorphisms from A to B will be denoted by $\mathrm{Hom}_R(A, B)$. For an abelian group G, the ring of endomorphisms $\mathrm{Hom}_{\mathbb{Z}}(G, G)$ will be denoted by $\mathrm{End}(G)$. For any set X, $F(X)$ will denote the free group generated by X.

5.2 Algebraic PIR from Degree t Polynomials

To see how to construct an algebraic PIR with constant server-side communication given a cryptosystem that allows polynomials of total degree t to be computed on ciphertext, you can see the work of [3]. For completeness however, we sketch such a protocol below. Proceed as follows. First, organize a database of bits in t-coordinate addresses. Now to produce a query for an address $(i_1^*, i_2^*, ..., i_t^*)$, create t vectors of length $\sqrt[t]{n}$ according to the formula $(v_k)_j = \delta_{j,i_k^*}$. Encrypt these vectors and send them to the server as a query. Label the

encrypted vectors as $w_k = \mathcal{E}(v_k)$ and suppose that the bits of the database have been labeled $X = \{x_{i_1,...,i_t}\}_{i_k \in [0, \sqrt[t]{(n)}-1]}$. Then for any $X \in \{0,1\}^n$, define

$$F_X(Y_{1,1},...,Y_{1,\sqrt[t]{n}-1},...,Y_{t,1},...,Y_{t,\sqrt[t]{n}-1}) = \sum_{i_1,...,i_t \in [0,\sqrt[t]{(n)}-1]}^{n} \left[\prod_{k=1}^{t} Y_{k,i_k}\right]^{x_{i_1,...,i_t}}$$

which can of course be computed on ciphertext for any $X \in \{0,1\}^n$ since the exponents can be computed via the $\mathbb{Z}$-module action and each term has degree t. So, there exists $\widetilde{F}$, efficiently computable from public information such that if $w = \mathcal{E}(v)$ then $\mathcal{D}(\widetilde{F}(w)) = F(v)$. So, the database algorithm simply computes $\widetilde{F}((w_1)_1,...,(w_t)_{\sqrt[t]{n}-1})$ as the response to the query, which will clearly be an encryption of $x_{i_1^*,...,i_t^*}$. Under the assumption that the cryptosystem is CPA-secure, security of this PIR protocol comes from a standard hybrid argument since the only information exchanged was a few arrays of ciphertext.

5.3 Additional Definitions

5.3.1 Oblivious Modifiers

Let $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a CPA-secure cryptosystem, with plaintext set G_1 and ciphertext set G_2. We consider the following setting: A user **U** initially holds a database $\{x_i\}_{i=1}^n$ and gives to a storage provider **S** an array $X = \{c_i\}_{i=1}^n$ of ciphertexts, where $c_i = \mathcal{E}(x_i)$. Subsequently, any number of users with the public key may wish to modify one of the underlying x_i in the database, and they wish to do so without revealing any information to **S** about the modification.

Definition 5.1 *We define an* **Oblivious Database Modifier** *to be the following three algorithms:*

1. $\mathsf{Key\text{-}Gen}(k)$. *This algorithm takes a security parameter k and generates all public and private parameters for the system, including public and private keys for the underlying cryptosystem.*

2. $\mathsf{GetModifier} : \{1,2,...,n\} \times \mathbb{N} \to \mathfrak{M}$. *This takes an integer $i \in \{1,2,...,n\}$ and some integer randomness, and then outputs $\mathfrak{m}_i \in \mathfrak{M}$ which describes the modification to be done to the database.*

3. $\mathsf{Modify} : G_2^n \times \mathfrak{M} \to G_2^n$. *This takes $X \in G_2^n$ and $\mathfrak{m}_i \in \mathfrak{M}$ and outputs $X' \in G_2^n$ such that $\mathcal{D}(X_i) \neq \mathcal{D}(X_i') \iff \mathfrak{m}_i \in \mathsf{GetModifier}(\{i\} \times \mathbb{N})$.*

The above algorithms describe a one round protocol for oblivious database modification. GetModifier is executed by the various users which send the result to the database owner who executes Modify on that input and the database.

Definition 5.2 *(Correctness) If whenever $i \in \{1,2,...,n\}, \mathfrak{m}_i \in \mathsf{GetModifier}(\{i\} \times \mathbb{N}), X \in G_2^n$, and $X' = \mathsf{Modify}(X, \mathfrak{m}_i)$ it holds that*

$$(\mathcal{D}(X_j) \neq \mathcal{D}(X_j') \iff i = j)$$

with overwhelming probability (over any randomness used in GetModifier and Modify) then the Oblivious Database Modifier is said to be **correct**.

For such a system, the goal is to conceal what database element is being modified. So, we would like to have no information about the selected index to be efficiently computable from the protocol's execution with any noticeable probability. I.e., nothing about i is efficiently computable from $\mathfrak{m}_i$. Clearly, this will depend on the security of the underlying cryptosystem.

Definition 5.3 *(Privacy) We define semantic security in terms of the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The game consists of the following steps:*

1. *$\mathcal{C}$ runs* Key-Gen(k) *and sends to $\mathcal{A}$ all public parameters, including a description of the underlying cryptosystem.*
2. *$\mathcal{A}$ can ask queries of the form $i \in \{1, 2, ..., n\}$ and $\mathcal{C}$ responds with* GetModifier(i).
3. *$\mathcal{A}$ selects $i_0, i_1 \in \{1, 2, ..., n\}$ and sends both to $\mathcal{C}$.*
4. *$\mathcal{C}$ randomly chooses $b \in \{0, 1\}$ and sends* GetModifier(i_b) *to $\mathcal{A}$.*
5. *$\mathcal{A}$ may send more* GetModifier *queries to $\mathcal{C}$, and $\mathcal{C}$ will respond properly.*
6. *$\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.*

We say that $\mathcal{A}$ wins if $b' = b$ and loses otherwise. Define the adversary's advantage to be

$$\mathrm{Adv}_{\mathcal{A}}(k) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

where the probability is taken over all internal randomness of $\mathcal{A}$ and $\mathcal{C}$. We say that the Oblivious Database Modifier is **CPA-secure** *if $\mathrm{Adv}_{\mathcal{A}}(k)$ is a negligible function in k.*

5.3.2 Algebraic Oblivious Modifiers

Let $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a CPA-secure group homomorphic encryption scheme with plaintext group G. Suppose that G is an abelian group. (We will consider other structures, e.g. rings and monoids later on.) By homomorphic, we will mean that for all $a, b \in G$

$$\mathcal{D}(\mathcal{E}(a)\mathcal{E}(b)) = ab$$

Given the fact that CPA-secure ciphertexts contain no information that is computable by a user or the database owner, what could an oblivious database modification protocol's algorithm look like? To preserve privacy, the computation performed by GetModifier must involve every ciphertext in the database, and it logically must involve every underlying plaintext. Indeed, for virtually all PIR protocols derived from an encryption scheme, the only operations on the underlying plaintext are group operations. Here, we consider programs that are restricted to computing algebraic formulas on the underlying plaintext. In order to discuss lower bounds for such a system in a mathematical setting, we'll use our formalization of "arbitrary algebraic formula" developed in the preceding section. Recall that for any object A in a concrete category, we used the notation $F_A[X]$ to denote the set of all A-algebraic formulas with X as the set of variables, and the corresponding functions from $A^m \longrightarrow A^n$ are denoted by $\overline{F_A^n}[X]$, or more simply $\overline{F}_A[X]$.

Definition 5.4 *We define an* **Algebraic Oblivious Database Modifier** *to be an Oblivious Database Modifier with the following constraints:*

1. *The underlying cryptosystem is homomorphic, preserving the algebraic structure of the plaintext set, G_1 (be it a group, ring, field, et cetera).*
2. *The modification description set, $\mathfrak{M}$, is simply G_2^m, ordered m-tuples of ciphertexts.*
3. *The* Modify *protocol will compute $\psi \in \overline{F}_{G_2}[X]$, an algebraic function to determine the updated database contents. Here, the set X represents variables for every database element and for all elements of $\mathfrak{M}$.*

Clearly correctness and privacy will have the same definitions as a general modifier (see definitions 5.2 and 5.3, respectively). And as we have formalized above, the phrase "algebraic formula" now has precise meaning as well. [9]

[9] If the encryption map were deterministic, so that a homomorphic encryption function $\mathcal{E}$ was actually a homomorphism of groups, we could make a slightly more appealing definition, in which we would use $\mathcal{E} : G_1 \longrightarrow G_2$ to obtain our evaluation map. This way, we could still phrase things in terms of a G_1-algebraic formula.

ECCC ISSN 1433-8092
http://eccc.hpi-web.de/