On the Black-box Use of Somewhat Homomorphic Encryption in Non-Interactive Two-Party Protocols.*

Nirattaya Khamsemanan † Rafail Ostrovsky‡ William E. Skeith III§

Abstract

In this work, we develop a methodology for determining the communication required to implement various two-party functionalities non-interactively. In the particular setting on which we focus, the protocols are based upon somewhat homomorphic encryption, and furthermore, they treat the homomorphic properties as a *black box*. In this setting, we develop lower bounds which give a smooth trade-off between the communication complexity and the "expressiveness" of the cryptosystem—the latter being measured in terms of the depth of the arithmetic circuits that can be evaluated on ciphertext. Given the current state of the art in homomorphic encryption, this trade-off may also be viewed as one between communication and *computation*, since more expressive cryptosystems are presently less efficient. We then apply this methodology to place lower bounds on a number of cryptographic protocols including PIR-writing and private keyword search.

While the hypotheses of our results preclude them from having universal applicability, we hope that this work will nevertheless provide a valuable "litmus test" of feasibility for use by other cryptographic researchers attempting to develop new protocols that require certain levels of communication efficiency.

Lastly, we also answer an open question from the thesis of Rappe [Rap06] regarding the construction of fully homomorphic encryption from group homomorphic encryption.

1 Introduction

1.1 Background and Motivation

Homomorphic encryption schemes offer an intriguing compromise between *functionality* and *security*, and as such have played a vital role in the design of many cryptographic protocols

*Abridged version appeared at CRYPTO 2008.

†School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology (SIIT), Thammasat University, Thailand. E-mail: `nirattaya@siit.tu.ac.th`. This author was supported by the DPST Research Fund Grant number 041/2555..

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

§Department of Computer Science, City College of New York. E-mail: `wes@cs.ccny.cuny.edu`. This author was supported by the NSF grant CNS 1117675

over the years. To name but a few of the success stories, we mention oblivious transfer and private information retrieval (PIR), private keyword search, voting protocols, and collision-resistant hashing. Indeed, many novel protocols have been developed using homomorphic encryption as building block, and in numerous instances, it is in fact the sole cryptographic ingredient.

Motivated by these successes, many recent research efforts have focused on improving homomorphic encryption. In recent literature, we have seen numerous proposals of schemes supporting increasingly rich homomorphic properties. In addition to schemes that are homomorphic over abelian groups (*e.g.,* [GM84, Gam85, Pai99, DJ03, KTX07]) there have also been proposals to evaluate polynomials of small total degree on ciphertext (*e.g.,* [BGN05, MGH08, GHV10]) as well as a large number of proposals for fully homomorphic encryption [Gen09, vDGHV10, SV10, GH11, BV11, Gen11]. We now have public-key cryptosystems which cover a broad spectrum of homomorphic properties—the simple XOR operation lying on one end and arbitrary circuits on the other. Cryptosystems on the latter end of the spectrum provide a powerful tool for manipulating encrypted data, however, this added functionality does not come without a cost. As functionality requirements increase, the richness of options is naturally diminished. Current techniques in fully homomorphic encryption suffer from a number of drawbacks. Perhaps the most severe is that of computational complexity. With the exception of [Gen11], all known FHE schemes require a costly "bootstrapping" step, in which the scheme's own decryption circuit is evaluated on ciphertext. One such scheme has been implemented in [GH10] who report that on a modern machine, the bootstrapping operation alone requires anywhere from 30 seconds to 30 minutes, depending on the security parameters, which seems to be a major obstacle for most practical applications. Furthermore, most FHE schemes have relied on somewhat non-standard intractability assumptions. Progress in this direction has been made in the recent works of [Gen11, BV11] which present schemes based on LWE, rather than on ideal lattice problems. In the work of [Gen11], the bootstrapping step has also been removed, but not under the standard LWE assumption. For many situations, the scheme of [Gen11] presents better asymptotic efficiency than previously known FHE schemes, yet the practical significance of these improvements has yet to be studied, and the authors warn that the constants hidden the asymptotic notation are large. Other work in this direction includes that of [NLV11], which considers the practicability of *somewhat* homomorphic encryption. A brief summary of various homomorphic cryptosystems can be found in Table 1.

This situation presents a number of interesting trade offs between the various parameters and assumptions for researchers and practitioners to consider when designing and implementing protocols. One recurring question is of the following form. If functionality X is needed, but also with computational efficiency requirements A and communication complexity requirements B, what options are available? Furthermore, on what intractability assumptions can the protocol be founded? In this work, we take steps toward answering this type of question. In particular, we consider the case of *black-box usage* of homomorphic encryption, and provide a simple "litmus test" for the feasibility of constructing protocols in this setting.

Black-box usage of homomorphic encryption. Many useful protocols and primitives have been derived from homomorphic schemes in a "black box" way, by manipulating the homomorphic properties to construct various systems, but without regard to any particular details of the representation of the plaintext or ciphertext elements. Prominent examples include private keyword search protocols [OS07], private database writing [BKOS07], as well as certain implementations of single-database private information retrieval and collision-resistant hashing (see [KO97, Cha04, IKO05]). Black-box use of homomorphic encryption has also been utilized in more general secure function evaluation settings, *e.g.,* in the work of [KSS09]. In this work, we demonstrate lower bounds for a variety of natural tasks when constructed in a strictly black-box manner; in particular, we demonstrate precise trade-offs between the completeness of the homomorphic properties of the cryptosystem and the communication complexity. This is accomplished by analyzing one simple task, and showing communication bounds for realizing this task via a restricted set of algebraic operations, which correspond to the homomorphic properties of the cryptosystem. As we demonstrate, this task is inherent in a number of the protocols listed above (*e.g.,* private database writing) and as such, the bounds for the simple task also apply to the protocol. This result yields a simple "litmus test" for determining the feasibility of constructing black-box protocols from homomorphic encryption under certain demands on the communication and computation efficiency, and on the required hardness assumptions.

1.2 Our Results

Our results fall into two main categories. The first is a study of cryptographic protocols based on the *black box use* of somewhat homomorphic encryption. In particular, we develop techniques for lower bounding the communication complexity in such settings, where the protocol messages are the result of some *formulaic* computation, rather than arbitrary boolean functions.[1] After developing lower bound criteria for generic, formula-based protocols (Proposition 4.6), we then explore a number of applications, including PIR-writing, private keyword searching, and *homomorphic* PIR (Corollaries 5.1, 5.4, and 5.3).

The second category of our results concerns equivalences among various flavors of homomorphic encryption. We study in detail equivalences between homomorphic encryption over certain non-abelian groups, and fully homomorphic encryption (over $\mathbb{F}_2$). We give an original proof that homomorphic encryption over any finite non-abelian simple group is equivalent to fully homomorphic encryption (Theorem 6.5 and Corollary 6.9), thereby answering an open question posed by Rappe [Rap06]. We also revive an interesting piece of group theory history, showing that several lesser-known results of Maurer *et al.* [MR65] not only have the aforementioned equivalence among homomorphic encryption flavors as a consequence, but also have the main results of Barrington [Bar86] as a trivial consequence. (The work of [KMR66] even goes so far as to define Barrington's *permutation branching programs*, albeit with different notation and terminology.)

[1] Of course, the cases of interest are formulas over algebraic structures other than $\mathbb{F}_2$, since any boolean function can be computed by an arithmetic circuit over a non-trivial ring.

Cryptosystem	Homomorphic properties	Practical space requirements?	Practical computational requirements?	Assumptions
[GM84]	$(\mathbb{Z}_2, +)$	$\approx \checkmark$ (Message space is $\{0,1\}$)	$\checkmark$	Quadratic Residuosity
[Pai99]	$(\mathbb{Z}_n, +)$	$\checkmark$	$\checkmark$	DCRA
[Gam85]	$(\mathbb{Z}_p, \cdot)$	$\checkmark$	$\checkmark$	DDH
[BGN05]	degree 2 polynomials over $\mathbb{Z}_{\mathcal{O}(\text{poly})}$	$\approx \checkmark$ ($\mathcal{O}(1)$-sized message space)	$\checkmark$	Bilinear Subgroup Decision Problem
[GHV10]	degree 2 polynomials over $\mathbb{Z}_n$	$\checkmark$	$\checkmark$	LWE
[MGH08] instantiated with [KTX07]	degree d polynomials over $\mathbb{Z}_n$	Only if d is small	Only if d is small	uSVP over integer lattices
[Gen09]	$(\mathbb{Z}_2, +, \cdot)$	No	No	SVP over Ideal Lattices; Sparse Subset Sum

Table 1: List of homomorphic cryptosystems along with their functionality, efficiency and required assumptions. When mentioned, κ denotes the security parameter.

1.3 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 [BL96] examine the computational difficulty breaking any algebraically homomorphic (over a field) cryptosystem. Other related works are those of Shoup [Sho97] and Maurer and Wolf [MW98], 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. In contrast, our lower bounds are geared towards communication complexity and program size, and furthermore, they apply to a wider variety of algebraic structures (including arbitrary abelian groups and bounded degree polynomials over rings). Our re-

sults are also related in some ways to the classic results of Yao [Yao79] and Abelson [Abe80]. Building upon their foundation, we develop and analyze yet another model of communication complexity for protocols in which the messages between parties are the results of formulaic computations.

1.4 Techniques

In order to study what can be accomplished with black box usage of somewhat homomorphic encryption, we first develop a new model for *algebraic* communication complexity. The model is based upon the classic works of Abelson [Abe80] and Yao [Yao79], which can both be seen as specializations of a generic protocol pattern. [Abe80] uses continuous functions to compute the protocol messages, while [Yao79] uses boolean functions; our instantiation of the pattern restricts the computation of protocol messages to functions which arise from *algebraic formulas*. We define such formulas precisely in Definition 4.1, but there is a simple intuition behind the formalism: essentially, we are considering a straightforward generalization of arithmetic circuits to algebraic structures other than $\mathbb{F}_2$ (*e.g.,* non-commutative groups).

We then pursue lower bounds in this new model. In place of Yao's combinatorial rectangles and Abelson's partial derivative matrix, we have the notion of "G-independent elements" over an abelian group G. The notion of G-independence can be thought of as a natural analog of linear independence over vector spaces, and although many vector space niceties are absent, we are nevertheless able to establish a number of analogous results. For example, we show in Theorem 3.6 that if a set of n, G-independent elements are in the range of an affine[2] map, then the domain must be of size exponential in n. Using these results on G-independent sets, we then show (Proposition 4.6) that if a protocol is restricted to abelian group formulas, then the communication complexity of any function with n G-independent elements in the range is at least $n - 1$ bits.

We also generalize these basic results in a number of ways. We consider a common setting in which the group in question is a direct product G^n, and the independent elements are "characteristic vectors," meaning that only a small number of coordinates contain non-identity elements of G. In this setting, we prove a smooth trade-off in communication complexity as the number of non-identity elements in the characteristic vectors increases (Corollary 3.11).

Lastly, we show (using a fairly straightforward linearization technique) that if we allow the protocol messages to be total degree t polynomials over any ring R (instead of abelian group formulas), then the communication is bounded by $\Omega(\sqrt[t]{n})$ (*cf.* Corollary 3.12). Hence the results we derive are applicable to a wide variety of homomorphic cryptosystems, including [RSA78, GM84, Gam85, Pai99, BGN05, GHV10, MGH08]. See Table 1 for a brief summary.

[2] Affine maps over groups are simply translated homomorphisms, analogous to the case of linear spaces. See Definition 3.3.

1.5 Remarks

First, we note that while the premise of black box usage of homomorphic encryption may seem restrictive, the results in this model are fairly complete: they apply to *all* finite abelian groups, and to polynomials of total degree t over *arbitrary* finite rings. Furthermore, our results apply to arbitrary "affine maps," (see Definition 3.3) which include functions that may not be realized by formulaic means (*e.g.,* arbitrary endomorphisms of an abelian group). Additionally, we remark that there are numerous examples in the literature of effective cryptocomputing protocols constructed in exactly this way.

Secondly, we note that in some sense, our results state the impossibility of representing certain functions using abelian group formulas. From this perspective, it is not surprising that many functions cannot be represented by such formulas: a simple counting argument reveals that the number of m-variable formulas over G is far less than the number of G-valued functions (as set maps) which depend on m variables. But what are these functions? Furthermore, in what sense can they not be represented? The surprising aspect of our results is that we find simple and natural classes of functions which cannot be represented with classes of formulas over groups and rings, and furthermore that the non-existence of these functions has impact on protocol design.

2 Background and Notation

2.1 Mathematical Notation

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. For an abelian group G, we will generally use additive notation, and denote the identity element by $\mathbf{0}_G$. If G is not necessarily abelian, multiplicative notation will be used and $\mathbf{1}_G$ will represent the identity element. 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. $X \amalg Y$ denotes the coproduct. 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.

2.2 Private Information Retrieval

Recall the setting of ***private information retrieval***, in which a user **U** wishes to retrieve the i-th element of a database X, without revealing the index i to the database owner (denoted **DB**). More formally, it is a two-party protocol consisting of a triple of algorithms $(\mathsf{Query}, \mathsf{Reply}, \mathsf{Reconstruct})$, such that

1. *(Correctness.)* For any database X and index i, $\mathsf{Reconstruct}(\mathsf{Reply}(\mathsf{Query}(i))) = X_i$.
2. *(Privacy.)* For all $i, j \in [n]$, the distributions of $\mathsf{Query}(i), \mathsf{Query}(j)$ are computationally indistinguishable.

Since the above can trivially be accomplished by setting $\mathsf{Query}(i) = \perp$ for all i, $\mathsf{Reply}(\perp) = X$, and $\mathsf{Reconstruct}(X, i) = X_i$ (that is, by simply communicating the entire database), a third constraint is generally implicit:

3. *(Communication efficiency.)* The sum of the length of the messages passed between **U** and **DB** is strictly less than[3] n bits, where n is the size of X.

Definition 2.1. *Let $\sigma \in \mathbb{Z}^+$ be a security parameter. A* **Encryption Scheme***, or* **Cryptosystem** *with message space M and ciphertext space C is a tuple of* PPT *algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ *with the following descriptions.*

- KeyGen *takes a security parameter 1^σ and outputs* $(\mathsf{PK}, \mathsf{SK})$*, encryption and decryption keys.*
- Enc *takes input $1^\sigma, \mathsf{PK}$ and $m \in M$ and outputs $c \in C$, an element of the ciphertext space.*
- Dec *takes as input $1^\sigma, \mathsf{SK}$ and a ciphertext $c \in C$, and outputs an element $m' \in M$ such that if $c = \mathsf{Enc}(1^\sigma, \mathsf{PK}, m)$ and $(\mathsf{PK}, \mathsf{SK}) = \mathsf{KeyGen}(1^\sigma)$, then $m = m'$ with all but negligible probability (as a function of σ, where the probability ranges over the internal coins of the algorithms).*

3 Independent Vectors Over Groups

3.1 Motivation: Generating Encryptions of Characteristic Vectors

This section provides a simple example of a task that cannot be implemented with small communication using only abelian group algebra. Later, we show a variety of problems and cryptographic protocols (usually related to PIR or PIR-writing) which would imply a low-communication solution to this type of task. Hence, we will derive bounds on how well such protocols can be implemented with "black-box" usage of somewhat homomorphic encryption. First we will need a simple definition.

[3] Typically the communication is $o(n)$, but there are theoretical results deriving PIR from minimal assumptions which save only a constant number of bits (*e.g.,* [KO00]).

Definition 3.1. *Let G be an abelian group and let $\{v_i\}_{i=1}^k$ be elements of G. Analogously to the case of linear spaces, we say that the v_i are G-***dependent** *(or simply* **dependent** *when G is clear) if any of the $v_i = \mathbf{0}_G$, or if there exist $\alpha_i \in \mathbb{Z}$ such that*

1. $\sum_{i=1}^k \alpha_i v_i = \mathbf{0}_G$
2. $\exists i \in [k]$ *such that* $\alpha_i v_i \neq \mathbf{0}_G$.

*We say that the v_i are G-***independent** *if they are not dependent.*

Example 3.2. Again, borrowing terminology from linear algebra, we say that an $n \times m$ matrix $D = (d_{ij})$ of elements of G is ***diagonal*** if $d_{ij} \neq \mathbf{0}_G \iff i = j$. (Note that we demand *all* of the on-diagonal elements are not the identity.) It is easy to see that the n rows (resp., m columns) of such a matrix will be G-independent if $n \leq m$ (resp., $m \leq n$). Moreover, any ***diagonalizable*** matrix (one which is diagonal upon left / right multiplication by permutation matrices) will similarly have G-independent rows and columns. $\diamondsuit$

Of particular interest to us will be "formulas" over algebraic structures. A detailed treatment can be found in Section 4, but we take note of the case of abelian groups here. For abelian groups, any formula (see Definition 4.1) can be expressed via an *affine map*:

Definition 3.3. *Let A, G be abelian groups. An* **affine map** *is a function of the form $F = f + c$ where $F : A \longrightarrow G$ is a homomorphism of groups, and $c \in G$ is a constant. That is, F maps $x \mapsto f(x) + c$.*

An answer to the following question regarding such affine maps will provide us with a number of the lower bounds we desire regarding protocols based on homomorphic encryption.

Question 3.4. (Informal) Let $F : A \longrightarrow G^m$ be an affine map. If the image of F contains a set of k independent elements in G^m, what can be said regarding the size of A? In particular, how small can the domain of F be?

We will soon answer this question in a variety of contexts, but first we give a motivational example which illustrates some of the complexities of the problem, as well as the non-applicability of several straightforward (but naïve) approaches toward a solution.

One case of special interest is when $F : G^\ell \longrightarrow G^m$ for some group G. Given the apparent similarity to linear spaces over a field, it is tempting to push the analogy further and attempt an argument for lower bounding ℓ based on "dimension." However, we cannot bound ℓ in a non-trivial way—even if we restrict G to be a cyclic group. As the following example demonstrates, it is possible to have a large set of independent vectors in the range with $\ell = 1$. The key point, however, is that this necessarily comes at the cost of increasing the size of G.

Example 3.5. 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

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

Consider the homomorphism $f = (f_1, ..., f_n)$ from $G \longrightarrow G^n$ defined by $f_i(x) = z_i \cdot x$, and construct a matrix M having its i-th row equal to $f(z_i)$. It is easy to see that M is diagonal, and hence each of the n rows are independent. $\diamondsuit$

However, notice that n different primes had to divide the order of G in the preceding example. Hence, $|G| > 2^n$ is of exponential size in n. We will show that even using affine maps, this is always the case: if an affine map is to have n independent elements in its range, then the domain must be of exponential size in n.

3.2 A Basic Algebraic Result

Here we will make precise the relationship regarding the number of independent elements in the range of an affine map and the size of its domain. Although the details differ, the results are analogous to the case of vector spaces: a set of independent vectors will span a large space, and translating such a set along a fixed vector cannot introduce too many dependencies. Thus, the span of the translates is also large. Working with G-independent sets instead of linear spaces introduces some complications, but we can nevertheless achieve results that deliver the desired effect: if an affine map has n independent elements in the image, then the domain must be of size exponential in n.

Theorem 3.6. *Let A, G be abelian groups, and let $V = \{v_i\}_{i=1}^n \subset G$ be any collection of independent elements. If $F = f + c$ is an affine map from $A \longrightarrow G$ such that $V \subset F(A)$, then $|A| \geq 2^{n-1}$.*

We will factor the proof into two simple, but useful observations about G-independent sets which show that even after translation by a constant, a G-independent set will always generate an exponentially large subgroup.

Observation 3.7. Let G be a finite abelian group, and suppose that $V = \{v_i\}_{i=1}^n$ are G-independent elements. Then $|\langle V \rangle| \geq 2^n$.

Proof. Let $V = \{v_i\}_{i=1}^n$ satisfy the hypothesis, and consider the set of all binary vectors $(\alpha_1, \ldots, \alpha_n)$. It is easy to see that each $\sum_{i=1}^n \alpha_i v_i$ must be unique: if $\sum_{i=1}^n \alpha_i v_i = \sum_{i=1}^n \alpha_i' v_i$ then $\sum_{i=1}^n (\alpha_i - \alpha_i') v_i$ yields a non-trivial dependence unless $\alpha_i = \alpha_i'$ for all i (recall that independence precludes $v_i = 0$). This completes the proof. ■

This observation shows that a set of n independent elements in the image of a *homomorphism* would require the domain to be large, but we need a similar result to hold for affine maps. For this, we would like to analyze translates of an independent set by some fixed constant c. The number of independent elements in such a translate is not as easy to reason about as in the case of linear spaces over a field, yet we can still derive strong lower bounds on the size.

Lemma 3.8. *Let G be a finite abelian group, $c \in G$ an arbitrary element, and suppose that $\{v_i\}_{i=1}^n$ are G-independent. Then the set $\{v_i + c\}_{i=1}^n$ will generate a subgroup of size at least 2^{n-1}.*

Proof. Let $V = \{v_i\}_{i=1}^n$ be independent, and denote by $\overline{v}_i$ the translated vectors $v_i + c$. Consider the set of $(\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n$ satisfying $\sum_{i=1}^n \alpha_i = 0$. Notice that

$$\sum_{i=1}^n \alpha_i \overline{v}_i = \sum_{i=1}^n \alpha_i v_i + c \sum_{i=1}^n \alpha_i = \sum_{i=1}^n \alpha_i v_i.$$

But just as we have seen in Observation 3.7, each such *binary* vector must yield a unique element. Constraining the first $n-1$ coordinates to $\{0, 1\}$, and adjusting $\alpha_n = -\sum_{i=1}^{n-1} \alpha_i$ so that the sum remains 0, we see at once that $|\langle \overline{v}_1, \ldots, \overline{v}_n \rangle| \geq 2^{n-1}$ as desired. ■

The proof of 3.6 now follows readily.

Proof. **(Theorem 3.6)** Recall that $F = f + c$ was an affine map, and that $V = \{v_i\}_{i=1}^n$ was a set of G-independent elements. Consider the translates of V by $-c$: $\overline{v}_i = v_i - c$. By Lemma 3.8, it follows that the $\{\overline{v}_i\}_{i=1}^n$ will generate a subgroup of order at least 2^{n-1}. However, each $\overline{v}_i \in f(A)$, and f is a homomorphism. Hence $|f(A)| \geq 2^{n-1}$, which gives $|A| \geq 2^{n-1}$ as desired. ■

In our applications, we will generally be concerned with functions that have a range in G^m, corresponding to a set of ciphertexts under a homomorphic encryption scheme. We now establish a few corollaries specific to this case. First and foremost, we note the following immediate consequence of Theorem 3.6 regarding diagonalizable matrices (see Example 3.2).

Corollary 3.9. *Let $F : A \longrightarrow G^m$ be an affine map. If $\{v_i\}_{i=1}^n \subseteq F(A)$ are diagonalizable, then $|A| \geq 2^{n-1}$.*

In the diagonal case, independence is clear, but in our applications, diagonalizability often proves too restrictive a criterion. We establish in what follows a simple technique to lower bound the size of an independent set in G^m given only that each element of the set does not have "too many" non-identity coordinates.

Lemma 3.10. *Let G be an abelian group and let $D = (d_{ij}) \in G^{n \times m}$ such that*

1. *none of the columns d_{*j} are the zero vector $\begin{pmatrix} \mathbf{0}_G \\ \vdots \\ \mathbf{0}_G \end{pmatrix}$, and*
2. *no row d_{i*} has more than ℓ non-identity elements*

then the there exists a subset of at least $\lceil m/\ell \rceil$ rows which are G-independent.

Proof. Let R be a minimal subset of the rows for which no column is the zero vector and consider the subset $X \subseteq [m]$ of column indexes j for which precisely one $r \in R$ has a non-identity element at position j. This produces a surjective mapping from $X \longrightarrow R$ (else the minimality of R would be contradicted). Choose a right inverse $r \mapsto j_r$ to this mapping so that row r is the unique member of R having a non-identity element in column j_r. But now it is easy to see R is diagonalizable: left multiplying by a permutation matrix P_1 can sort the rows of R according to the values of j_r, and right multiplying by a permutation matrix P_2 can arrange $X = \{1, \ldots, |R|\}$, resulting in a diagonal matrix:

$$P_1 R P_2 = \begin{pmatrix} g_1 & \mathbf{0}_G & \cdots & \mathbf{0}_G & * & \cdots & * \\ \mathbf{0}_G & g_2 & \cdots & \mathbf{0}_G & * & \cdots & * \\ \vdots & & \ddots & & & & \\ \mathbf{0}_G & \cdots & \mathbf{0}_G & g_{|R|} & * & \cdots & * \end{pmatrix}.$$

As seen in Example 3.2, a nontrivial dependence (Definition 3.1) among the rows is clearly impossible. By simply counting, we see that $|R| \geq \lceil m/\ell \rceil$, which proves the result. ■

Corollary 3.11. *Let G be an abelian group and consider a matrix $V = (v_{ij}) \in G^{n \times m}$. For $\ell \in \mathbb{Z}^+$, suppose V is such that at most ℓ coordinates of each row differ from $\mathbf{0}_G$, and that V has k non-zero columns. Then for any affine map $F : A \longrightarrow G^m$, $V \subseteq F(A)$ implies $|A| \geq 2^{\lceil k/\ell \rceil - 1}$.*

Proof. By Lemma 3.10, we see that the sub-matrix consisting of the k non-zero columns of V will satisfy the hypothesis. Thus, V contains at least $\lceil k/\ell \rceil$ independent elements of G^m. Direct application of Theorem 3.6 then yields the result. ■

3.3 Arithmetic Circuits of Bounded Multiplicative Depth

There are a number of cryptosystems in the literature that provide homomorphic properties which allow the computation of arithmetic circuits of bounded multiplicative depth on ciphertext (*e.g.,* [BGN05, GHV10, MGH08, Gen09]). Here, we extend Theorem 3.6 to handle these situations, providing a smooth trade-off between communication and circuit depth for the characteristic vector problem. For convenience, we work with an equivalent formulation, and consider polynomials with bounded total degree t over a ring R. Although the following result will apply to the ring of polynomials over arbitrary 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" (*cf.* Example 4.2).

Corollary 3.12. *Let R, S be rings, and let $t \in \mathbb{N}$. Let $V \subseteq S^r$ be a set of n independent elements over $(S^r, +)$. Suppose also that $\varphi : R \longrightarrow S$ is a ring homomorphism*[4] *and $F : S^m \longrightarrow S^r$ is such that $F = (F_1, ..., F_r)$ with each $F_i \in S[X_1, ..., X_m]$ of total degree less than or equal to t. Then $V \subset F \circ \varphi(R^m)$ implies*

$$\sqrt[t]{n} \leq m \sqrt[t]{2 \log(|R|)} + 1.$$

[4]We will denote the product map $\varphi^m : R^m \longrightarrow S^m$ also by φ when there is no risk of confusion.

Proof. Let $f \in S[X_1, ..., X_m]$ be of total degree t. Let $N_{m,t}$ denote the number of monomials of f. Since we do not assume that S is commutative, our only bound is

$$N_{m,t} \leq \sum_{k=0}^{t} m^k = \begin{cases} \frac{m^{t+1}-1}{m-1}, & \text{if } m \geq 2 \\ t+1 & \text{if } m = 1 \end{cases} \tag{1}$$

Next observe that by increasing the number of variables, we can simulate the functionality of f with a polynomial $\overline{f} \in S[Y_1, ..., Y_{N_{m,t}}]$ of total degree 1 (*i.e.,* an affine map abelian groups). More precisely, if

$$f = s_0 + \sum_{k=1}^{N} \left[s_k \prod_j X_{\alpha_k(j)} \right] \tag{2}$$

where $\alpha_k : [t] \longrightarrow [m]$, then we set

$$\overline{f} = s_0 + \sum_{k=1}^{N} s_k Y_k. \tag{3}$$

Finally, observe that

$$f(X_1, ..., X_m) = \overline{f}(\prod_j X_{\alpha_1(j)}, ..., \prod_j X_{\alpha_N(j)}). \tag{4}$$

Hence any function on S^m which is realized via f, can also be realized by $\overline{f}$. Observe also that $\overline{f}$ is an affine map from $(S,+)^{N_{m,t}} \longrightarrow (S,+)$. Let $F = (F_1, ..., F_r)$ be as in the theorem statement. Component-wise applying the transformation in (3) and composing with φ yields an affine map $\overline{F} : (R,+)^{N_{m,t}} \longrightarrow (S,+)^r$ which can simulate the functionality of F. Hence $V \subset \overline{F}((R,+)^{N_{m,t}})$, and since $(R,+), (S,+)$ are abelian groups, we can directly apply Theorem 3.6 which yields

$$|R|^{N_{m,t}} \geq 2^{n-1}. \tag{5}$$

For the case of $m \geq 2$, note that $\frac{2}{m} \geq \frac{1}{m-1}$, which combined with subadditivity of $\sqrt[t]{\cdot}$ gives

$$\begin{aligned} n - 1 &\leq \frac{m^{t+1}-1}{m-1} \log(|R|) \\ n &\leq 2m^t \log(|R|) + 1 \\ \sqrt[t]{n} &\leq m\sqrt[t]{2\log(|R|)} + 1 \end{aligned}$$

as desired. For $m = 1$, since $\sqrt[t]{t+1} \leq 2$ for $t \geq 1$, Equation (5) and subadditivity again yield

$$\begin{aligned} n - 1 &\leq (t+1)\log(|R|) \\ \sqrt[t]{n} &\leq \sqrt[t]{2\log(|R|)} + 1 \end{aligned} \tag{6}$$

so that indeed for every integer $m > 0$, $\sqrt[t]{n} \leq m\sqrt[t]{2\log(|R|)} + 1$. ■

4 Algebraic Two-Party Protocols

Here, we establish formal definitions for "black-box use of homomorphic encryption," expressed as a certain type of two-party protocol. We derive our notion from the classic descriptions of two-party protocols used in the study of communication complexity [Abe80, Yao79]. The work of [Abe80] and [Yao79] can be seen, respectively, as analytic and combinatorial instantiations of a more generic protocol for computing a function for which the inputs are spread across two disjoint sets of processors. We describe another natural variation here: that of an algebraic protocol. To begin, we will first need to lay down some basic definitions for what it means for a function to be "algebraic" in a sense that is aligned with our purpose of studying black-box homomorphic encryption.

4.1 Algebraic Formulas as our "Black-Box"

Intuitively, our notion of "black-box" usage of homomorphic encryption is that given some number of ciphertexts, the only meaningful way to combine them is via the homomorphic properties of the cryptosystem. In most situations the homomorphic properties are naturally expressed in some algebraic system, *e.g.,* that of a group, or polynomials of bounded total degree (arithmetic circuits of bounded depth) over a ring. In such cases, an arbitrary "black-box" manipulation of the homomorphic property amounts to a "formula" in the algebraic system. In this section, we provide a formal definition of this idea and show that it matches our intuition in familiar settings. Furthermore, we show that the abstract definition specializes to the formulas that were considered in the previous sections, thereby justifying our choices.

For intuition, consider first the ring of polynomials $R[x_1, \ldots, x_n]$ over a commutative ring R with identity. This seems to match our intuition of a generic algebraic formula in the operations $(+, \cdot)$. The key concept is that of an *evaluation map*, which takes an assignment of elements to variables and yields a homomorphism from $R[x_1, \ldots, x_n] \longrightarrow R$ that "evaluates" each polynomial using that assignment. The following encapsulates these ideas in an abstract setting.[5]

Let $\mathfrak{A}$ be a concrete category with $U : \mathfrak{A} \longrightarrow \mathbf{Set}$ the underlying (forgetful) functor. Consider the following functor:

$$G = (I_{\mathfrak{A}} \times U) : \mathfrak{A} \longrightarrow \mathfrak{A} \times \mathbf{Set}$$

where $I_{\mathfrak{A}}$ is the identity functor. If G has a left adjoint F, we call the category $\mathfrak{A}$ ***Admissible***, and make the following definition.

[5]The rationale for employing elementary ideas from category theory is twofold. First, it is a rather economical way to obtain the desired results—sophisticated machinery sweeps away many of the tedious details. Second (and somewhat more subjective), we feel that the very fact that the definition is expressible as a universal indicates that it is approaching "the right" definition, or at the very least, it is a natural one. Indeed, a great many interesting mathematical constructions (especially *formulaic* constructions) are expressible in terms of adjoints.

Definition 4.1. *For any pair of objects* $(A, X) \in \mathfrak{A} \times \mathbf{Set}$*, we call* $F(A, X)$ *the set of all* **$\mathfrak{A}$-algebraic formulas** *with scalars in* A *and variables in* X*. We denote* $F(A, X)$ *by the symbol* $A[X]$*.*

Most all of the categories we are interested in are admissible. As we show below, it suffices for the category to have coproducts and free objects.

Existence and Uniqueness. More generally, it is easily shown that if $H : \mathfrak{A} \longrightarrow \mathfrak{B}, K : \mathfrak{A} \longrightarrow \mathfrak{C}$ are functors having left adjoints $\bar{H}, \bar{K}$ and if $\mathfrak{A}$ has coproducts, then the product functor $H \times K$ has a left adjoint with object function $(b, c) \mapsto \bar{F}b \amalg \bar{G}c$. As shown in [ML98, pp. 124], U will have a left adjoint for any category of small algebraic systems (including commutative and non-commutative groups and rings, R-modules, and many others) which sends a set X to the free object having X as generators. Applying this to our case of interest (Def. 4.1), we see that $A[X]$ will almost always exist, and that $A[X] \cong \mathbf{Free}(X) \amalg A$. (Note that the left adjoint of the identity functor is again the identity.) Since we have expressed $A[X]$ as an adjoint, the uniqueness (up to isomorphism) comes for free via the usual "abstract nonsense."

Evaluation Maps. It will often be convenient to view a formula $f \in A[X]$ as a function from $A^X \longrightarrow A'$, corresponding to the *evaluation of* f at some tuple of points $k \in A^X$. This simple concept of evaluating a formula has a somewhat hard-to-parse description in terms of our current notation; we thus introduce the following shorthand. For a fixed morphism $\psi : A \longrightarrow A'$, and for $f \in A[X]$, denote by $\overline{f} : A^X \longrightarrow A'$ the map sending

$$\begin{aligned} k &\mapsto \varphi^{-1}((G\psi)^*(\mathrm{Id}_A, k))(f) \\ &= \psi(\varphi^{-1}(\mathrm{Id}_A, k)(f)) \end{aligned} \tag{7}$$

where φ denotes the natural bijection $A'^{F(A,X)} \cong (GA')^{(A,X)}$ from the adjunction. Simply put, $\overline{f}$ plugs a tuple of elements of A into the variables of f, reduces the expression in the algebra, and composes with ψ to arrive at the value in A'. The following diagram may also be helpful in visualizing the situation.

$$\begin{tikzcd} & A[X] \arrow[dl, "\overline{f}"'] \arrow[d, dashed] \\ A' & A'[UA'] \arrow[l, "\xi_{A'}"'] \end{tikzcd} \qquad \begin{tikzcd} (A, X) \arrow[d, dashed, shift right] \arrow[d, dashed, shift left] \\ (A', UA') \end{tikzcd} \quad \langle h, k \rangle = \varphi(f) \tag{8}$$

As the diagram shows, specifying a $\mathfrak{A}$-morphism $h : A \longrightarrow A'$ and a set map $k : X \longrightarrow UA'$ is both necessary and sufficient for specifying a $\mathfrak{A}$-morphism from $A[X] \longrightarrow A'$.

4.2 Discussion and Examples

Although the existence and uniqueness for our settings of interest follows easily from general categorical facts, we believe it is worthwhile to concretely present a few examples (and non-examples). A brief list follows, demonstrating (hopefully) that the definition is tightly aligned with our intuition.

Example 4.2 (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 $R[x_1, \ldots, x_n]$. $\diamondsuit$

It is well known that $R[x_1, \ldots, x_n]$ satisfies the required universal mapping property depicted in (8). This is not terribly surprising, considering that it was polynomials which guided us toward definition 4.1 to begin with.

It is also interesting to consider the case of a field $\mathbb{F}$. This is of course a commutative ring with identity, so the formulas described by $F[x_1, \ldots, x_n]$ all make sense, but it seems like we could accomplish more with division, so that perhaps rational functions would correspond 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. Thinking categorically, indeed, there is no "free field." This could have implications in cryptocomputing protocols as well: for example, what if some computation on ciphertext corresponded to a division by zero? The cryptocomputer should not be able to differentiate from division by a unit, yet the former raises an exception that ought to prevent any further computation.

Note that in non-commutative rings, the set of formulas is more complex than just $R[x_1, \ldots, x_n]$ since coefficients and variables cannot always be written so concisely. Section 6 provides a (somewhat extreme) example of this complexity.

Example 4.3 (Groups). For a group G, the set of all G-algebraic formulas on a set of variables X is $\mathbf{Free}(X) * G$, the free product of the free group on X with G. The elements are arbitrary words in X, mixed with elements of G, which indeed embodies the most generic formulas or straight-line programs that could be carried out using only the group operation. $\diamondsuit$

Example 4.4 (Abelian Groups). If G is an abelian group, the set of G-algebraic formulas over n variables is simply $\mathbb{Z}^n \oplus G$. Note that every element in $\mathbb{Z}^n \oplus G$ corresponds to an affine map, which was the focus of a number of results in Section 3. This again validates our initial intuition: a subset of affine maps play the very same role for abelian groups as polynomials do for commutative rings. However, not every affine map comes from such a formula: for a simple example, consider $G = \mathbb{Z}_p \times \mathbb{Z}_p$ and $\varphi \in \hom_{\mathbb{Z}}(G, G)$ by $(a, b) \mapsto (b, a)$. Thinking again of cryptographic application, such a non-formulaic affine map could still be computed on ciphertext, simply by shuffling ciphertexts around. This highlights the importance of the more general formulation of Theorem 3.6. $\diamondsuit$

4.3 Algebraic Protocols

With Definition 4.1 in hand, we may now return attention to the main point of defining an algebraic protocol. We introduce here yet another specialization of a "generic" model underlying the works of [Abe80, Yao79] which restricts to algebraic functions.

A Generic Protocol. We first describe an abstract "protocol pattern" which can be seen as a generalization of both [Abe80] and [Yao79]. Let X, Y be disjoint sets (not necessarily finite), and consider a function $\Phi : X \times Y \longrightarrow Z$. Two players A and B will take turns communicating information to each other to compute Φ. The protocol proceeds in rounds, where each party computes a function of her input and the communication history from prior rounds, and then sends the result to the other. We require that the output of these intermediate functions all lie in Z, which serves as a unit of communication. More formally, we have round functions a^r, b^r:

$$a^r : X \times Z^{r-1} \longrightarrow Z, \qquad b^r : Y \times Z^r \longrightarrow Z.$$

Define functions $\alpha^r(x, y), \beta^r(x, y)$ to be the values at the internal nodes after r rounds of the protocol. More formally, we can define $\alpha^1(x, y) = a^1(x), \beta^1(x, y) = b^1(y, a^1(x))$, and then recursively define the others via

$$\begin{aligned} \alpha^r(x, y) &= a^r(x, \beta^1(x, y), \ldots, \beta^{r-1}(x, y)) \\ \beta^r(x, y) &= b^r(y, \alpha^1(x, y), \ldots, \alpha^r(x, y)). \end{aligned} \tag{9}$$

We say that the protocol ***implements*** Φ if $\Phi = \alpha^r$ or $\Phi = \beta^r$ for some r.[6] Note that we are assuming synchronous communication, and that we assume A speaks first. The ***cost with respect to*** Z of the protocol for the function Φ is defined to be the number of elements of Z that have been transmitted, which is $2r - 2$ in the case that A computes the final output, and $2r - 1$ if the computation ends with B. If Z is finite, then we define the ***concrete cost*** to be $\log |Z|$ times the cost with respect to Z. The ***communication complexity*** is defined to be the minimum cost ranging over all protocols implementing Φ, and the ***concrete communication complexity*** is defined similarly. We remark that rather than forcing the parties to alternate communicating single elements of Z, we could allow for an arbitrary communication pattern. However, it is easy to see that this will reduce the communication of any protocol by at most a factor of 2.

Adaptation to the Algebraic Setting. In the generic protocol above, Yao's model sets $X = \{0, 1\}^m, Y = \{0, 1\}^n, Z = \{0, 1\}$, and all the functions involved are boolean functions. In Abelson's model, $X = \mathbb{R}^m, Y = \mathbb{R}^n, Z = \mathbb{R}$, and all functions involved are continuous (usually in C^2). For the purposes of studying the black box use of homomorphic encryption, we present yet another adaptation.

[6] In many instantiations, we can force the worst-case behavior if the number of rounds is data-dependent by simply echoing the output for the remaining rounds. However, for most of our situations of interest, the number of rounds will be independent of the inputs.

Definition 4.5. *Let X, Y be sets, and let H, Z be a objects in an admissible category Ω. An Ω-**Algebraic** two party protocol for a function $\Phi : X \times Y \longrightarrow Z$ consists of two sets of functions $\{a^i\}_{i=1}^r, \{b^i\}_{i=1}^r$, an Ω-morphism $\psi : H \longrightarrow Z$, and a single bit q, such that*

1. *$\forall i \in [r]$, $a^i : X \longrightarrow H[x_1, \ldots, x_{i-1}]$ and $b^i : Y \longrightarrow H[x_1, \ldots, x_i]$, with $H[x_1, \ldots, x_i]$ as in Definition 4.1;*

2. *with α, β the iterated applications of the round functions as in (9),*

$$\Phi = \begin{cases} \psi \circ \alpha^r, & \text{if } q = 0 \\ \psi \circ \beta^r, & \text{if } q = 1. \end{cases}$$

Remarks and Discussion. Conceptually, our model of "black box" use of homomorphic encryption is as follows: we suppose that the messages from the client consist of values encrypted under some homomorphic scheme, and furthermore, that there is nothing "useful" that can be done with these encrypted values beyond the provisions offered by the homomorphic properties of the cryptosystem. Hence, the functions computed by the parties in the protocol are restricted to formulas in the algebra. A straightforward analog of [Abe80] would suggest that we require Z to be an object of an admissible category Ω, set $X = Z^m, Y = Z^n$, and lastly force the function Φ to be an algebraic formula in $Z[x_1, \ldots, x_k]$ (see 4.1). However, we contend that such a model would be too restrictive for our setting. Thinking of cryptocomputing protocols, focus in particular on the functions that a party computes *based on her own inputs*. If these inputs are not encrypted, It seems unjustified to restrict the computation to something algebraic. Moreover, it may be the case that the inputs do not have a natural representation as elements of an algebraic structure to begin with (think for example the database in a PIR protocol, or the list of keywords in a private searching protocol). To remedy this situation, we allow any function to be computed on a party's own input, and only restrict the functions *of the history* to be algebraic. As a final relaxation, we allow the intermediate values to take on values in a different domain than Z. In particular, we fix an Ω-morphism $\psi : H \longrightarrow Z$, so that H will be the unit of communication for the protocol. We also remark that while it seems that a more general definition might include different domains for the history at each stage, this is not a useful modification for most settings. For example in the case of affine maps (the setting of our lower bounds), this would be superfluous, as any such protocol implies a protocol as in Definition 4.5 with communication complexity bounded above by the original. Moreover, in the case of finite groups, the order of the homomorphic image of an element will divide the order of the original element. Thus in information theoretic terms,[7] when computing the k-th map as a function of the history, it would have been be no less efficient to have communicated each element as coming from a subgroup of A_k^k than to communicate an element of $A_1 \times \cdots \times A_k$.

We may now state and prove a general lower bound for algebraic protocols based the results of Theorem 3.6.

[7] That is, assuming that we are afforded a dense representation for elements of any subgroup of A_k^k.

Proposition 4.6. *Let G be a finite abelian group, and let $\Phi : X \times Y \longrightarrow G$ be such that $\exists y \in Y$ for which $\Phi(X \times \{y\})$ contains a set of n G-independent elements. Then any group-algebraic two-party protocol (Definition 4.5) implementing Φ necessarily has concrete communication complexity of $n-1$ bits.*

Proof. The proof is a straightforward consequence of Theorem 3.6. Let Φ satisfy the hypotheses of the proposition, and let $\{a^i\}_{i=1}^r, \{b^i\}_{i=1}^r$ be the round functions of a group-algebraic protocol implementing Φ. Consider the function b_y^r, viewed as a function $H^r \longrightarrow Z$ via the correspondence in Equation (7). Since the protocol's output factors through b_y^r, it must be the case that $b_y^r(H^r)$ also contains a n independent elements. Furthermore, note that in the case of groups, any $b_y^r \in H[x_1, \ldots, x_r]$ can be represented by an affine map. Theorem 3.6 then implies that $|H^r| \geq 2^{n-1}$ so that $r \log |H| \geq n-1$. This proves the result, since $r \log |H|$ of course lower bounds the concrete complexity of the protocol for Φ, which must be at least $(2r-1) \log |H|$. ■

Remarks. Note that a symmetric argument may be made for the case of $\Phi(\{x\} \times Y)$ containing a complete set of characteristic vectors. It may also be of interest to note that the lower bounds in this model are actually above the trivial *upper bounds* shown for the classical setting.

5 Applications

We begin with the observation that numerous cryptocomputing functionalities have natural solutions in terms of algebraic protocols (Definition 4.5), citing a number of examples in the literature. We then reason about the limitations of these natural, algebraic approaches, deriving lower bounds on the communication complexity from Proposition 4.6. We hope that these techniques will provide researchers and practitioners with a useful sanity check for a protocol's feasibility. 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 reasonable approaches to a problem.

5.1 Private Database Modification (PIR Writing)

As seen in [BKOS07], the ability to privately modify an encrypted database in a communication efficient way could provide a valuable tool for private computation. One natural approach to such a problem (and indeed, the one taken by [BKOS07]), is to proceed in a manner analogous to many PIR protocols, and use homomorphic encryption as a building block. Such a protocol could communicate encrypted values which encode the modification to take place, and then the database owner would perform algebraic manipulations on the encrypted database and the values from the client to update the database contents. Since the communication consists solely of encrypted values, CPA-type security comes easily.

Ignoring security for a moment and focusing solely on a functional description, we can model PIR writing as a two party protocol for $\Phi : X \times Y \longrightarrow Y$ in which the client holds

an index $x \in X = \{1, \ldots, n\}$, the server holds a database $y \in Y = Z^n$, and Φ satisfies $(\Phi(x, y))_i \neq y_i \iff i = x$. That is, the protocol produces a new database which differs only at position x. It is easy to see that Proposition 4.6 has the following consequence.

Corollary 5.1. *The concrete communication complexity for any abelian-group-algebraic protocol for PIR writing is at least $n - 1$ bits.*

Proof. Using the notation established above, we have the client's input domain as $X = \{1, \ldots, n\}$, and the server's input domain is $Y = Z^n$. By our hypothesis, Z is an abelian group (which would be natural if the protocol were implemented using homomorphic encryption). The functionality of the protocol may now be expressed as $\Phi : X \times Z^n \longrightarrow Z^n$ such that $\Phi(x, y) = y + \delta_{i,x}$ where $\delta_{i,x} \neq 0_Z \iff i = x$. Now fix a database $y = (0_Z, \ldots, 0_Z)$ which has the identity in every position. We see at once that $\Phi(X \times \{y\})$ is diagonalizable, and thus contains a set of n independent elements. Hence, by Proposition 4.6, any abelian group algebraic protocol implementing Φ must have concrete communication complexity $n - 1$. ■

5.2 Algebraic and Homomorphic Protocols for PIR

As a second corollary, we consider "homomorphic" protocols for private information retrieval. Roughly speaking, we call a PIR protocol "homomorphic" if the responses to queries are encryptions of the desired database elements under a homomorphic encryption scheme.

Motivation. Such a protocol could be used to reduce communication complexity in a number of non-interactive cryptocomputing settings. Consider a protocol between parties A and B in which encrypted tags are associated to each element of a universe set X, and in which only a small subset $S \subset X$ are given non-identity tags. So that B may perform computations on the tags, we would like them to be homomorphically encrypted. For example, in a natural PIR-writing scheme based on homomorphic encryption (see 5.1), $X = [n]$ and S is the small set of indexes that A would like to update. To reduce communication, A can transmit *homomorphic PIR queries* corresponding to S rather than the tags for the entire set X. B will then construct databases "on the fly" against which the queries will be run. For PIR-writing, the databases would be characteristic vectors for each index $i \in [n]$, leaving B with a homomorphically encrypted characteristic vector for S after executing the queries. A similar technique was used in [OS07] for reducing the program size in private keyword search.

Examples and Intuition. There are a number of examples of homomorphic PIR in the literature. Consider the PIR protocol of [KO97]. In their initial scheme, the database is represented as a $\sqrt{n} \times \sqrt{n}$ matrix X of bits, and a user's query is a column vector of $\sqrt{n}$ ciphertexts (in a homomorphic encryption scheme over an abelian group). The server responds to a query by computing the product[8] $X \cdot v$ and returning the resulting $\sqrt{n}$ values. In more detail, to query an index (i, j), the user supplies a vector which is an encryption

[8]To parse this statement algebraically, treat the bits of X as integers, and the ciphertexts as (abelian) group elements. Then $X_{i,j} \cdot v_j$ refers to the $\mathbb{Z}$-module action.

of a non-identity element at position j and the identity in all other locations. The vector returned by the server is then an encryption of column j of the database, from which entry (i, j) is easily retrieved. Notice that the response to a query contains an encryption, under a homomorphic scheme, of the value $X_{i,j}$. However, it is important to note that the response contains *multiple* encrypted values in addition to the desired encryption of $X_{i,j}$. To further reduce the communication complexity, the authors then apply the scheme recursively, treating the batches of results as databases for further queries. Note that responses in this modified scheme are no longer homomorphic encryptions, but *nested encryptions*. Thus, we are faced with a trade-off between having a succinct response (as in the recursive scheme with nested encryption), or having a response in which algebraic structure is retained (as in the initial scheme, where the response contains $E(X_{i,j})$, along with $\sqrt{n} - 1$ other "uninteresting" values). As discussed in the paragraph above, if we could find a PIR protocol which combined both of these elements, it would have a number of applications in non-interactive cryptocomputing protocols. In what follows, we present some evidence for the difficulty in constructing such schemes with somewhat homomorphic encryption, showing for example that if using somewhat homomorphic encryption in a black-box manner, the product of the client's and the server's communication must be linear in the database size.

Rather than arguing the case of homomorphic PIR directly, we consider a weaker notion regarding simple information retrieval, and prove bounds for any algebraic protocol implementing this functionality, thus lower-bounding the efficacy of "natural" solutions to homomorphic PIR, including all those presented in the literature. Background and notation for general PIR schemes can be found in Section 2.2. Again, we let $(\mathsf{Query}, \mathsf{Reply}, \mathsf{Reconstruct})$ denote the 3 algorithms that comprise a PIR protocol.

In what follows we follow the notation established in Section 2.2.

Definition 5.2. *A* **Multi-Information-Retrieval** *protocol (abbr. MIR) between a client* $\mathbf{U}$ *and database* $\mathbf{DB}$ *consists of two algorithms,* $(\mathsf{Query}, \mathsf{Reply})$ *such that* $\mathsf{Reply}(\mathsf{Query}(i))$ *returns a subset* $\{X_\alpha\}_{\alpha \in S(i)}$ *where* $S : [n] \longrightarrow \mathcal{P}([n])$ *is such that* $i \in S(i)$.

A few remarks are in order. First, note that an *algebraic* multi-information-retrieval protocol together with a homomorphic encryption scheme for the given algebra gives rise to a PIR protocol by simply encrypting the query, performing the actions of Reply via the homomorphic properties, and setting $\mathsf{Reconstruct}$ to perform decryption and select the appropriate element from the resulting list.[9] However, note that one must know the *sequence* in which elements are returned to allow retrieval of the appropriate element, yet this was not specified in the definition, nor was the sequence even required to be a function of i alone. The definition is left in this weak form primarily because the proof of Corollary 5.3 goes through without additional constraints. Note also that for a PIR protocol constructed in this manner, $\mathsf{Reconstruct}$ amounts only to decryption, and thus functions of PIR queries can be computed *without the secret key* by simply multiplying the vector of ciphertexts. As discussed above, this can be a useful tool in cryptocomputing protocols.

[9] Of course, the resulting PIR protocol may be trivial if the communication is large.

Corollary 5.3. *Let P be a* non-interactive *abelian-group-algebraic two party protocol which implements multi-information-retrieval, and suppose that each query returns at most ℓ elements of the database. Then the concrete communication complexity of P will be at least $\lceil n/\ell \rceil - 1$ bits.*

Proof. Since Definition 5.2 mandates that the output is a subset of one of the inputs, we see that in the algebraic setting, P implements a function $\Phi : X \times G^n \longrightarrow G^\ell$ where G is an abelian group. For a particular database y, denote the database's function in the protocol by b_y. The b_y will always be affine maps which functionally take the history H^r to an output element $\mathbf{v} \in G^\ell$. We now "open the box" of P, and construct a modified protocol P' which outputs elements of G^n rather than G^ℓ. Denote by $\sigma : G^\ell \longrightarrow G$ the coproduct map, which simply sums all ℓ coordinates together. Consider databases $y^i \in G^n$ defined by $y^i_j \neq 0_G \iff i = j$. For protocol P', we leave the client's function(s) identical to those of P, but for the database, we ignore the input, and define its output function b'_* as

$$b'_* = (\sigma \circ b_{y^1}, \ldots, \sigma \circ b_{y^n}).$$

That is, the database takes a query for j and then ranges over all y^i, $i \in [n]$, computing for each, the output that P would have produced. It then collapses each output to a single value of G by summing the coordinates, and finally outputs a single vector $\mathbf{v}_j$. By hypothesis, each of the $\mathbf{v}_j$ can have at most ℓ coordinates which are non-identity. But now by Corollary 3.11, we see that P' is such that $\Phi(X \times \{y\})$ contains $\lceil n/\ell \rceil$ independent elements, and so by Proposition 4.6, the concrete communication complexity of P' must be $\lceil n/\ell \rceil - 1$ bits. Noting that the messages from the client were identical in both P' and P, we similarly have bounded the concrete communication complexity of P, completing the proof. ■

Using Corollary 3.12, we can generalize this result to cryptosystems that may have additional homomorphic properties, showing $\Omega(\sqrt[t]{n})$ bounds if total degree t polynomials over a ring R can be evaluated 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 § 8 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 3.12.

5.3 Private Keyword Searching

As another relatively simple corollary, we partially resolve an open problem posed in [OS07] regarding extending the query semantics for private searching on streaming data. In particular, we demonstrate lower bounds for a class of techniques built upon somewhat homomorphic encryption, which includes the method of [OS07].

The original protocol provided a means for "conditionally" encrypting documents matching a query into a buffer, making use of additively homomorphic encryption. As it turns out, we do not need many of the details of [OS07] in what follows, as a simple functional skeleton of the protocol will suffice for our analysis. More specifically, we analyze a simplified version of the protocol (which in fact was used as a subroutine of [OS07]) that outputs a non-identity element for matching documents, and an identity element otherwise. We can phrase this simplified protocol in terms of Definition 4.5 as follows. For a dictionary $D \subseteq \{0,1\}^*$, we define ***disjunctive keyword search functionality*** as a function $\Phi : \mathcal{P}(D) \times \mathcal{P}(D) \longrightarrow \{0,1\}$ such that

$$\Phi(x,y) = 0 \iff x \cap y = \varnothing. \tag{10}$$

Similarly, we define ***conjunctive keyword search functionality*** as a function $\Phi : \mathcal{P}(D) \times \mathcal{P}(D) \longrightarrow \{0,1\}$ such that

$$\Phi(x,y) \neq 0 \iff x \subseteq y. \tag{11}$$

To make the protocols algebraic, say over an abelian group G, we replace 0 with $\mathbf{0}_G$ in (10) and (11).[10] Using the cryptosystem of [BGN05], it was shown how to perform a query consisting of a single conjunction, but no general solution was known. Two of the primary open questions from [OS07] were to

1. reduce the dictionary size;
2. perform conjunctive queries.

First, we note that these are equivalent questions to some extent: one can trivially perform k-ary conjunctive queries by expanding the dictionary to $\binom{|D|}{k}$ elements, consisting of all (lexicographically sorted) k-tuples of D, and performing a disjunctive search for the singleton consisting of the original subset x. (The server of course must also be modified to parse its document as a list of all subsets of size k.) Hence, if it were possible to reduce the dictionary to $\sqrt[k]{|D|}$, then k-ary conjunctions could be evaluated without increasing the space requirements beyond the original $\widetilde{\mathcal{O}}(|D|)$ (although not without some performance cost in parsing for the server). On the other hand, if one could perform conjunctive queries, then the dictionary size could be reduced as well: consider a simplified setting in which every word has exactly 2 syllables. In this case, a search for a single word could be implemented as a search for a conjunction of the syllables (processing each word of the document separately). The lack of entropy of most spoken languages makes the actual savings difficult to estimate, but one could reasonably expect a reduction in the dictionary size whenever many words have common syllables, or if a dense encoding is available.

We now show a partial negative answer to both questions, demonstrating that no algebraic protocol over an abelian group can perform conjunctions or reduce dictionary size. Then, by applying Corollary 3.12, we can show more generally that evaluating polynomials of total degree t can at best perform t-ary conjunctions, or reduce the dictionary size to $\sqrt[t]{|D|}$.

[10] Note that 0 could be replaced with any other element of G, and Corollary 5.4 would still hold with only minor modifications to the proof.

Corollary 5.4. *Any* non-interactive *abelian-group-algebraic protocol for k-ary conjunctive keyword search (Equation* (11)*) relative to a dictionary D necessarily has concrete communication complexity $\binom{|D|}{k} - 1$, where $|D|$ denotes the number of words in D.*

Proof. Similar to the proof of Corollary 5.3, we can open the box, and run a single query, say for a subset $S \subseteq D$ of size k, against many documents which range over all subsets of D of size k. We then output the resulting vector, which by definition (see Eq. (11)) will have a single non-identity coordinate. Now ranging the query over all $\binom{|D|}{k}$ possibilities manifestly produces a diagonal matrix over $G^{\binom{|D|}{k}}$, so that $\Phi(X \times \{y\})$ contains $\binom{|D|}{k}$ independent elements. Proposition 4.6 now yields the result. ■

We believe that this example illustrates particularly well the utility of our results. The method of [OS07] critically depends on the ability to produce encryptions of non-identity elements conditioned upon whether or not a document matches some set of keywords. The elementary observation that this functionality gives rise to many algebraically independent elements at once gives insight as to how well it can be implemented with somewhat homomorphic encryption, and in fairly general terms. Indeed, it seems that improving the work of [OS07] would require either a radically different approach, or the usage of homomorphic encryption supporting a broader set of operations, which currently comes at a steep cost. It is this type of information that we hope will save researchers time and effort in the future. The hypothesis for bounds (large independent sets) are generally easy to recognize, and thus can help quickly eliminate a large space of what might otherwise seem to be feasible approaches to a problem.

6 FHE and Finite Non-Abelian Simple Groups

In this section, we show an equivalence between fully homomorphic encryption over a ring (say $\mathbb{Z}_2$) and *group homomorphic* encryption for finite non-abelian simple groups. This result resolves an open question posed by Rappe [Rap06]. Before getting into the details, we take a moment to review the history of this, and related problems, and how they have been rediscovered and recast over the years. The origins of the study appear to be a lesser known result by Maurer and Rhodes [MR65], which actually gives the main result of Barrington [Bar86]—20 years earlier. Although the terminology of "branching programs" and NC^1 circuits were not yet in use, [MR65, KMR66] has, as an immediate consequence, the fact that every fan-in 2 circuit of depth d can be represented with a width 5 permutation branching program of length ℓ^d for a constant ℓ. Conversely (with sufficient massaging), [Bar86] can be seen to give a result similar to [KMR66].

6.1 Representing Functions with Non-abelian Simple Groups

Krohn, Maurer and Rhodes. Let G be a finite non-abelian simple group, let $F : G^n \longrightarrow G$ be an *arbitrary* function, and let $X = \{x_1, \ldots, x_n\}$ be a set of variables. In a nutshell, the

work of [MR65] states that

$$\exists f \in G[X] \ \mid \ \overline{f} = F \tag{12}$$

where $G[X]$ is as in Definition 4.1 and $\overline{f}$ is the induced function of the formula f (see Eq (7)). The 1966 work of Krohn, Maurer and Rhodes [KMR66] takes this somewhat abstract result and gives it a more computational flavor. They describe a special type of DFA which is easily seen to be precisely the *permutation branching programs* of [Bar86]. In more modern language, their main result is that any boolean function can be computed by a width 5 permutation branching program. Although bounds on length for certain circuit classes were not explicitly stated, it is easy to see that the result implies that for a fan-in 2 circuit of depth d, the branching program needs only length ℓ^d for some constant ℓ, which is essentially the main result of [Bar86]. The details are as follows. Let C be a circuit of depth d with fan-in bounded by a constant c, and pick an encoding of $\{0,1\} \leftrightarrow \{\alpha, \beta\} \in G$. Using this encoding we can view the standard arithmetic gates for $\wedge, \vee, \neg$ as functions $\{\alpha, \beta\}^c \longrightarrow \{\alpha, \beta\}$, and thus by (12), we can implement each of them as formulas in $G[X]$. Since the fan-in is bounded, each of these formulas will have some constant length. Let ℓ denote the maximum. Then by starting at the output gate and recursively filling in the variables with the formulas corresponding to the input wires, we can construct a branching program which has length at most ℓ^d, and can compute F. The width w of the branching program corresponds to the smallest w for which G has a faithful representation in the symmetric group S_w. Instantiating the scheme with A_5 of course gives the result for width 5.

Barrington's Results. Barrington [Bar86] shows that permutation branching programs can be used to represent boolean functions. In particular, let $f : \{0,1\}^n \longrightarrow \{0,1\}$ be any boolean function. Then f can be modeled as a function $\alpha : [\ell] \longrightarrow [n]$ and a sequence $\{p_i^0, p_i^1\}_{i=1}^{\ell}$ of S_5 permutations as follows. $f(x_1, \ldots, x_n) = 0$ if and only if

$$\prod_{i=1}^{\ell} p_i^{x_{\alpha(i)}} = e$$

where e denotes the identity element. Furthermore, it can be arranged so that $f(x_1, \ldots, x_n) = 1$ is represented by the product evaluating to a specific 5-cycle. Some important remarks:

- In the construction, ℓ depends on the representation of f; in particular, $\ell = 4^d$ if f is represented as a boolean circuit of depth d.
- The "internal wires" of the branching program do not have a uniform representation of 0/1. As such, the construction requires that the entire branching program is known in advance; one cannot concatenate additional instructions ad-hoc.
- From this result, it is not clear that an S_5-homomorphic encryption would give rise to FHE. On the surface, it seems that it would allow one to compute encrypted branching programs (or NC^1 circuits) on plaintext, a problem which already has a number of solutions in the literature, *e.g.,* [SYY99, Bea00, Rap06]. (However, with sufficient massaging, [Bar86] can be converted into similar statements to [Rap06, MR65].)

In contrast, our result shows at once that S_5-homomorphic encryption (or homomorphic encryption for any other group containing a finite non-abelian simple group) would give the very same benefits as FHE. Moreover, it has as a simple corollary the main result of [Bar86] (although with a few different constants).

Our Approach. We follow the approach of [Rap06] (which utilizes some techniques from [BOC92]), but rather than focusing on an explicit group ([Rap06] makes use of $SL_3(\mathbb{F}_2)$ and S_7), we extend the techniques to any finite non-abelian simple group. Both our result and that of [Rap06] can be thought of as specializations of [MR65]. Nevertheless, there are some advantages of our work in contrast to [MR65]. The most prominent, perhaps, is that our results provide a more direct, simple, and intuitive proof. Rather than attacking an arbitrary boolean function head-on as was done in [MR65], we instead focus on a universal set of gates—in particular, NAND. The following observation illustrates the high-level approach.

Observation 6.1 (Informal). Let G be a group and suppose that $\beta, \beta^{-1} \in G$ can be expressed as a products

$$\beta = \prod_{i=1}^{k} f_i(\beta, \beta), \quad \beta^{-1} = \prod_{i=1}^{\ell} h_i(\beta, \beta) \tag{13}$$

where each f_i, h_i is a commutator of conjugates of its inputs. That is, each function maps $(x, y) \mapsto [gxg^{-1}, hyh^{-1}]$ for some $g, h \in G$. Then, identifying $0 \leftrightarrow e$ and $1 \leftrightarrow \beta$, the formula

$$\mathsf{AND}(x, y) = \prod_{i=1}^{k} f_i(x, y) \tag{14}$$

behaves precisely like AND, and the formula

$$\mathsf{NEG}(x) = \beta \prod_{i=1}^{k} h_i(x, x) \tag{15}$$

behaves precisely like boolean negation.

As we demonstrate below, *any* element β in a finite non-abelian simple group satisfies the condition in (13). Note also that by the celebrated Feit-Thompson theorem [FT63], we can choose β to have order 2. In this case, we can replace (15) with $\mathsf{NEG}(x) = \beta x$. We also note that the collapsing of certain commutator formulas when a single variable is the identity is something of a common thread between all of the main results on this topic. The role of simplicity is to ensure that commutator formulas are "expressive enough" to represent many functions, and for this, simple groups seem ideal—intuitively, one could think of simple groups (which are always perfect) as being as "non-commutative as possible."

Lemma 6.2. *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. ■

For an element $x \in G$, we denote the conjugacy class by $\mathrm{Cl}_G(x) = \{gxg^{-1} \mid g \in G\}$. Recall that G acts naturally on $\mathrm{Cl}_G(x)$ by $g \cdot s = gsg^{-1}$, and note that the action is transitive.

Lemma 6.3. *Let G be a finite non-abelian simple group, and let $\alpha, \beta \in G$ with $\beta \neq e$. Then there exist functions $\{f_i(x,y)\}_{i=1}^k$, each of the form $(x,y) \mapsto [gxg^{-1}, hyh^{-1}]$ such that $\alpha = \prod_{i=1}^k f_i(\beta, \beta)$.*

Proof. Let G be as in the lemma, and let $\beta \in G$. Let $S = [\mathrm{Cl}_G(\beta), \mathrm{Cl}_G(\beta)]$, and consider $\langle S \rangle$. First note that since the set S is conjugation-invariant, $\langle S \rangle \lhd G$ by Lemma 6.2. We will furthermore show that $|S| > 1$. The action of G on $\mathrm{Cl}_G(\beta)$ induces a homomorphism $\varphi : G \longrightarrow S_m$. φ is easily seen to be injective: $m > 1$ since $Z(G)$ must be trivial; and since the action is transitive, $\ker(\varphi) \neq G$, and thus is trivial as G simple. Therefore, every element of G acts non-trivially (by conjugation) on the set $\mathrm{Cl}_G(\beta)$. In particular, this shows that at least two different conjugates of β do not commute, and hence $S \neq \{e\}$. Since G is simple, we conclude that $\langle S \rangle = G$, and the lemma now follows. ■

We now formalize Observation 6.1 and apply it to finite non-abelian simple groups via Lemma 6.3. First, we must make precise the notion of representing a boolean function $f : \{0,1\}^m \longrightarrow \{0,1\}$ via a group formula $\widetilde{f} \in G[x_1, \ldots, x_m]$, as alluded to in the observation. There are a number of seemingly natural choices as to how one represents binary input as group elements, but some are too weak to be of much use. For example, if we allow any embedding $\iota : \{0,1\}^m \longrightarrow G^m$ for the representation, then it may be possible to push the majority of the work of f into ι rather than $\widetilde{f}$. So at a minimum, we would like to fix a single pair of group elements (g_0, g_1) which will represent $(0,1)$ in every coordinate. If we furthermore use the same (g_0, g_1) to represent bits in the output, then we gain additional advantage in terms of modularity: we can then *compose* representations of different boolean functions to obtain a representation of the composition. This is not only important for applications to homomorphic encryption, but it also allows us to prove functional completeness results by arguing about a small set of universal functions, *e.g.,* NAND. Hence we make the following definition.

Definition 6.4. *A formula $\widetilde{f} \in G[x_1, \ldots, x_m]$* **represents** *a boolean function $f : \{0,1\}^m \longrightarrow \{0,1\}$ if there exists a pair of elements $g_0, g_1 \in G$ such that $f = \iota^{-1} \circ \widetilde{f} \circ \iota^m$, where ι denotes the map sending $i \mapsto g_i$ for $i \in \{0,1\}$.*

Note that in the above definition, $\widetilde{f}$ is used synonymously with the function from $G^m \longrightarrow G$ coming from the evaluation map, as in Equation (7), and that ι^m denotes the m-fold product of the map ι, rather than repeated composition. We are now ready to state and prove the following:

Theorem 6.5. *Let G be a finite non-abelian simple group. Then any boolean function*

$$f : \{0,1\}^m \longrightarrow \{0,1\}$$

can be represented by a formula $\widetilde{f} \in G[x_1, \ldots, x_m]$ in the sense of Definition 6.4. Moreover, the elements g_0, g_1 used to represent bits can be chosen independently of the function f.

Remarks. Note that as a simple corollary, any function $f : \{0,1\}^m \longrightarrow \{0,1\}^n$ can also be represented by a group formula by breaking f into component maps $f = (f_1, \ldots, f_n)$ and applying the result to each component function.

To prove the theorem, we will show that the function $\mathsf{NAND}(a, b)$ can be represented by a formula in G as described above, 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. Since our input/output representation is uniform (we always identify the same elements g_0, g_1 with 0 and 1) the composition will be consistent, letting us represent any function via a composition of our representation of NAND.

Proof. Set $g_0 = e$, the identity element of G, and let $g_1 \in G$ be of order 2 (such an element is guaranteed to exist by the Feit-Thompson theorem). Denote by ι the map sending $i \mapsto g_i$ for $i \in \{0,1\}$. By Lemma 6.3, we can find $f_i(x, y)$ as in Observation 6.1 such that $g_1 = \prod_{i=1}^{k} f_i(g_1, g_1)$. But now one may easily verify that for any $a, b \in \{0,1\}$, we have

$$\mathsf{NAND}(a, b) = \iota^{-1}\left(g_1 \prod_{i=1}^{k} f_i(\iota(a), \iota(b))\right).$$

■

6.2 Homomorphic Encryption

We now prove a straightforward corollary of Theorem 6.5, which asserts that constructing a homomorphic encryption scheme over any finite non-abelian simple group is equivalent to constructing a fully homomorphic encryption scheme. First, we need to establish a few definitions.

At a high level, homomorphic encryption schemes provide facility for computing on encrypted data. That is, given encryptions of values $x_1, \ldots, x_n$, and *the public key alone*, one can compute encryptions of non-trivial functions of $x_1, \ldots, x_n$. In most cases, the domain for the plaintext and ciphertext are endowed with an algebraic structure, and the functions that can be computed on encrypted data are algebraic formulas. If the binary operation of

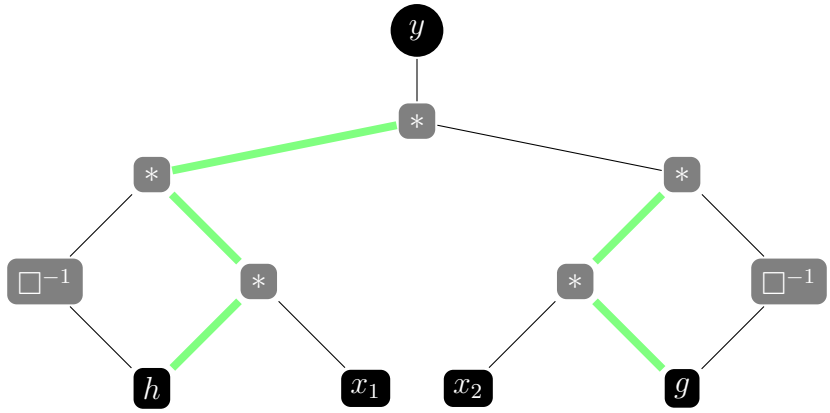

Figure 1: Possible G-circuit for the formula $y(x_1, x_2) = hx_1h^{-1}gx_2g^{-1}$. Bold edges correspond to $\omega(e) = 0$; *i.e.,* they are the left inputs to a multiplication gate.

a group can be computed on encrypted data, we call the system "group homomorphic", or more simply, "homomorphic". If both addition and multiplication for a ring with identity can be computed, the system is called "fully homomorphic." Intuitively what we would like from our definition is that any *formula* can be computed on encrypted data, with only a small additional cost in contrast with the effort required to compute the same function directly on the plaintext. For commutative rings, we can use traditional arithmetic circuits to measure of the cost, but the possible non-commutativity of groups introduces a slight complication into our circuit definition. To define a ***G-Circuit***, we begin as usual with a directed, acyclic graph V, E in which every node has indegree less than or equal to 2. Nodes with indegree 0 are either variables or constants, nodes with indegree 1 correspond to inverse gates, and nodes with indegree 2 are product gates. To specify the order of the operands to a multiplication gate, we add to our definition a function $\omega : E \longrightarrow \{0, 1\}$ such that for any edges with common destination $(v, w), (v', w)$, it holds that $\omega(v, w) \neq \omega(v', w)$. Edges labeled 0 (resp. 1) will correspond to the left (resp. right) input of the gate, and naturally, $\omega(v, w)$ has no meaning of w is an inverse gate (indegree 1). A G-circuit then computes a formula (see Definition 4.1 and Equation (7)) in the same way as a classical arithmetic circuit, with each gate's output wire set to the product (or inverse) of its children. The size of a G-circuit is defined to be the number of gates. An example G-circuit is depicted in Figure 1.

Lastly, we will be interested in functional equivalences between G-circuits and boolean circuits. Suppose that a boolean circuit C has variable input gates $x_1, \ldots, x_n$, and output gates $y_1, \ldots, y_m$. What is meant by a *functional equivalence* between C and a G-circuit $\widetilde{C}$, is that $\widetilde{C}$ also has n input variables and m outputs, and that there exists a bijection $\iota : \{0, 1\} \longrightarrow \{g_0, g_1\} \subset G$ such that for any assignment $a : \{x_1, \ldots, x_n\} \longrightarrow \{0, 1\}$, the assignment $\iota \circ a$ in $\widetilde{C}$ will yield outputs $\widetilde{y}_1, \ldots, \widetilde{y}_m$ such that $\iota^{-1}(\widetilde{y}_1), \ldots, \iota^{-1}(\widetilde{y}_m)$ are the outputs of C. That is, modulo relabeling $\{0, 1\}$ with elements of G, both $C, \widetilde{C}$ compute the same function.

Lemma 6.6. *For every finite non-abelian simple group G, there exists a constant γ_G such*

that for any boolean circuit C in $\neg, \wedge, \vee$ of size N, there exists a functionally equivalent G-circuit of size $\gamma_G N$. Moreover, the correspondence $\iota : \{0,1\} \leftrightarrow \{g_0, g_1\}$ for the equivalence can be chosen independently of the circuit C.

Proof. From Theorem 6.5, there exist $f_\wedge, f_\vee \in G[x,y]$ and $f_\neg \in G[x]$ such that $\iota^{-1} \circ f_* \circ \iota = *$ for $* \in \{\neg, \wedge, \vee\}$, where ι denotes a bijection $\{0,1\} \leftrightarrow \{g_0, g_1\}$. Note that each f_* is simply a word in G and $\{x, y\}$ (see Example 4.3). Let ℓ be the maximal length of the words $f_\neg, f_\wedge, f_\vee$. Then there exist G-circuits $A_\neg, A_\wedge, A_\vee$, each of size at most $3\ell - 1$ computing the same functions, since each formula f_* (without any optimizations) could be implemented with at most $\ell - 1$ multiplication gates, ℓ inverse gates, and ℓ input gates. Now let C be a boolean circuit of size N. Since each A_* expects the same representation (g_0, g_1) for bits, we can compose copies of $A_\neg, A_\wedge, A_\vee$ in precisely the same way as C to produce a G-circuit with the same functionality as C only with g_0, g_1 in place of 0 and 1. This circuit has size at most $N(3\ell - 1)$. Since $\gamma_G = 3\ell - 1$ is a constant which only depends on G, this completes the proof. ■

We now define group homomorphic encryption schemes and fully homomorphic encryption schemes in terms of G-circuits and boolean circuits, and prove their equivalence when G is finite, non-abelian and simple.

Definition 6.7. *Let G be a finite group and let $\sigma \in \mathbb{Z}^+$ be a security parameter. A* **Group-Homomorphic Cryptosystem** *over G is a cryptosystem (Definition 2.1) with plaintext space G, and the following additional property. Suppose that $(\mathsf{PK}, \mathsf{SK}) = \mathsf{KeyGen}(1^\sigma)$. Then for any G-circuit A of size N with k input gates, there exists a boolean circuit $\widetilde{A}$ of size at most $\sigma K N$ for a constant K such that for all $h_1, \ldots, h_k \in G$,*

$$\mathsf{Dec}(\widetilde{A}(\mathsf{PK}, \mathsf{Enc}_{\mathsf{PK}}(h_1), \ldots, \mathsf{Enc}_{\mathsf{PK}}(h_k))) = A(h_1, \ldots, h_k)$$

holds with all but negligible probability, ranging over the coins of KeyGen *and* Enc.

Definition 6.8. *Let let $\sigma \in \mathbb{Z}^+$ be a security parameter. A* **Fully Homomorphic Cryptosystem** *is a cryptosystem (Definition 2.1) with plaintext space $\{0,1\}$, and the following additional property. Suppose that $(\mathsf{PK}, \mathsf{SK}) = \mathsf{KeyGen}(1^\sigma)$. Then for any boolean circuit A of size N with k input gates, there exists a boolean circuit $\widetilde{A}$ of size at most $\sigma K N$ for a constant K such that for any inputs $x_1, \ldots, x_k$,*

$$\mathsf{Dec}(\widetilde{A}(\mathsf{PK}, \mathsf{Enc}_{\mathsf{PK}}(x_1), \ldots, \mathsf{Enc}_{\mathsf{PK}}(x_k))) = A(x_1, \ldots, x_k)$$

holds with all but negligible probability, ranging over the coins of KeyGen *and* Enc.

Remarks. Classical definitions of homomorphic encryption define additional algorithms which compute each of the operations of the algebraic structure (usually a group or a ring) on encrypted data. However, such definitions have the following drawback: there is nothing to prevent the ciphertexts which are the result of the computations on encrypted data from growing significantly in size, thus pushing most of the "real work" into the decryption algorithm in some sense. See for example [SYY99, MGH08]. Nevertheless, if appropriate

constraints on ciphertext compactness are put in place, the definitions are easily seen to be equivalent. Note also that since the same boolean circuit must work for all possible encryption keys, trivial solutions, such as hard-wiring the decryption key into the circuit, do not satisfy the definition.

Corollary 6.9. *Constructing a fully homomorphic encryption scheme (Definition 6.7) is equivalent to constructing a group homomorphic encryption (Definition 6.8) 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.*

Proof. This follows directly from Lemma 6.6, which shows that (up to constant factors) G-circuits for a finite non-abelian simple group G are equivalent to boolean circuits. Let G be such a group, and suppose that $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a group homomorphic encryption scheme over G. We define a fully homomorphic cryptosystem $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ as follows. $\overline{\mathsf{KeyGen}}$ is precisely the same as KeyGen. To define $\overline{\mathsf{Enc}}, \overline{\mathsf{Dec}}$, we simply right (resp. left) compose Enc (resp. Dec) with ι (resp. ι^{-1}), where $\iota : \{0,1\} \leftrightarrow \{g_0, g_1\}$ is the correspondence guaranteed by Lemma 6.6. Now let A be a boolean circuit of size N which is to be evaluated on encrypted data. Using Lemma 6.6, we can first convert the circuit to a G-circuit C, which will be at most size $\gamma_G N$. Now, using the homomorphic property of the scheme over G, there exists a boolean circuit $\widetilde{C}$ of size at most $\sigma\gamma_G KN$ such that for any $h_1, \ldots, h_k \in G$,

$$\mathsf{Dec}(\widetilde{C}(\mathsf{PK}, \mathsf{Enc}_{\mathsf{PK}}(h_1), \ldots, \mathsf{Enc}_{\mathsf{PK}}(h_k))) = C(h_1, \ldots, h_k)$$

holds with all but negligible probability, ranging over the coins of KeyGen and Enc. Hence by construction,

$$\overline{\mathsf{Dec}}(\widetilde{C}(\mathsf{PK}, \overline{\mathsf{Enc}}_{\mathsf{PK}}(x_1), \ldots, \overline{\mathsf{Enc}}_{\mathsf{PK}}(x_k))) = A(x_1, \ldots, x_k)$$

also holds with all but negligible probability for any $x_1, \ldots, x_k \in \{0,1\}^k$. Conversely, suppose we are given a fully homomorphic encryption scheme and a binary representation of elements of G. We encrypt elements of G bitwise. We can evaluate any circuit computing the group operation of G on ciphertext.

■

Remarks. Note that we are actually free to choose any representation $\{0,1\} \hookrightarrow G$: for $\alpha \neq \beta \in G$, we can simply perform a change of variables $x \mapsto \alpha^{-1}x, y \mapsto \alpha^{-1}y$ and use the formula $\alpha \prod_{i=1}^{k} f_i(\alpha^{-1}x, \alpha^{-1}y)$, where $\beta = \prod_{i=1}^{k} f_i(\alpha^{-1}\beta, \alpha^{-1}\beta)$, which is possible by Lemma 6.3.

To better illustrate the proof, we provide an explicit construction for the smallest non-abelian simple group, A_5. So that inversion may be accomplished by a single multiplication, we use an element β of order 2. Any element of order 2 can be used in the construction, *e.g.,* $\beta = (1,2)(3,4)$. We know that commutators of conjugates of β will generate all of A_5, in particular, β itself. But to simplify things a bit, we first construct standard generators of A_5

out of such commutators, and then write down β in terms of the standard generators. For generators, we use $A_5 = \langle\{X, Y\}\rangle$ with $X = (1, 2, 3, 4, 5)$ and $Y = (3, 4, 5)$. 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 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 let $g_2 = (3, 4, 5)$ and $h_2 = e$. Next we write $\beta = (1, 2)(3, 4)$ in terms of our generators:

$$\beta = X^{-1}Y^{-1}X^{-1}Y^{-1}X^2$$

Finally, using the simple observations that $[x, y]^{-1} = [y, x]$, and that if group elements a, b have order dividing 2 then they are their own inverses, we can write down an explicit expression for a group formula representing NAND in terms of the group operation alone. Again, identifying $0 \leftrightarrow e$ and $1 \leftrightarrow \beta$, then (with a slight abuse of notation) we have the following expression, where $a, b \in \{e, \beta\}$:

$$\mathsf{NAND}(a, b) = \begin{array}{l} h_1bh_1^{-1}g_1ag_1^{-1}h_1bh_1^{-1}g_1ag_1^{-1}bg_2ag_2^{-1}bg_2ag_2^{-1}h_1bh_1^{-1}g_1ag_1^{-1}h_1bh_1^{-1}g_1ag_1^{-1}\backslash \\ bg_2ag_2^{-1}bg_2ag_2^{-1}g_1ag_1^{-1}h_1bh_1^{-1}g_1ag_1^{-1}h_1bh_1^{-1}g_1ag_1^{-1}h_1bh_1^{-1}g_1ag_1^{-1}h_1bh_1^{-1}\beta. \end{array}$$

7 Acknowledgments

This work was supported in part by NSF grant CNS 1117675 and DPST Research Fund Grant number 041/2555. The authors would also like to thank Associate Professor Dr. Bunyarit Uyyanonvara who has served as a mentor under DPST Research Fund.

References

[Abe80] Harold Abelson. Lower bounds on information transfer in distributed computations. *Journal of the ACM (JACM)*, 27(2):384–392, 1980.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In *STOC*, pages 1–5, 1986.

[Bea00] Donald Beaver. Minimal-latency secure function evaluation. In *EUROCRYPT*, pages 335–350, 2000.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC'05*, pages 325–341, 2005.

[BKOS07] Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and William E. Skeith. Public key encryption that allows PIR queries. In *CRYPTO'07*, pages 50–67, 2007.

[BL96] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In *CRYPTO*, pages 283–297, 1996.

[BOC92] Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM Journal on Computing*, 21(1):54–58, 1992.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. Preliminary result., 2011.

[Cha04] Y. C. Chang. Single database private information retrieval with logarithmic communication. In *Austrailian Conference on Information Security and Privacy*, 2004.

[DJ03] Ivan Damgård and Mads Jurik. A length-flexible threshold cryptosystem with applications. In *ACISP*, pages 350–364, 2003.

[Fre64] Peter Freyd. *Abelian categories*, volume 1964. Harper & Row New York, 1964.

[FT63] Walter Feit and John G. Thompson. Solvability of groups of odd order. *Pacific J. Math.*, 13:775–1029, 1963.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178, New York, NY, USA, 2009. ACM.

[Gen11] Craig Gentry. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. `http://eprint.iacr.org/`.

[GH10] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. Cryptology ePrint Archive, Report 2010/520, 2010. `http://eprint.iacr.org/`.

[GH11] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. Cryptology ePrint Archive, Report 2011/279, 2011. `http://eprint.iacr.org/`.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple bgn-type cryptosystem from lwe. Cryptology ePrint Archive, Report 2010/182, 2010. `http://eprint.iacr.org/`.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS*, 28(2):270–299, 1984.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In *TCC*, pages 445–456, 2005.

[KMR66] Kenneth Krohn, WD Maurer, and John Rhodes. Realizing complex boolean functions with simple groups. *Information and Control*, 9(2):190–195, 1966.

[KN06] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge university press, 2006.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.

[KO00] Eyal Kushilevitz and Rafail Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In *Advances in Cryptology2̆014EUROCRYPT 2000*, pages 104–121. Springer, 2000.

[KSS09] V. Kolesnikov, A.R. Sadeghi, and T. Schneider. How to combine homomorphic encryption and garbled circuits. In *Signal Processing in the Encrypted Domain–First SPEED Workshop–Lousanne*, page 100, 2009.

[KTX07] A. Kawachi, K. Tanaka, and K. Xagawa. Multi-bit cryptosystems based on lattice problems. In *In Public Key Cryptography PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 315–329, Berlin, 2007. Springer.

[MGH08] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with d-operand multiplications. Cryptology ePrint Archive, Report 2008/378, 2008. `http://eprint.iacr.org/`.

[ML98] Saunders Mac Lane. *Categories for the Working Mathematician (Graduate Texts in Mathematics)*. Springer, September 1998.

[MR65] W. D. Maurer and John L. Rhodes. A property of finite simple non-abelian groups. *Proceedings of The American Mathematical Society*, 16:552–552, 1965.

[MW98] Ueli M. Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In *EUROCRYPT*, pages 72–84, 1998.

[NLV11] M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.

[OS07] R. Ostrovsky and W.E. Skeith. Private searching on streaming data. *Journal of Cryptology*, 20(4):397–430, 2007.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[Rap06] Doerte K. Rappe. Homomorphic cryptosystems and their applications. Cryptology ePrint Archive, Report 2006/001, 2006. `http://eprint.iacr.org/`.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography*, pages 420–443, 2010.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC^1. In *FOCS*, pages 554–567, 1999.

[vDGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 209–213. ACM, 1979.

8 Appendix

8.1 Algebraic PIR from Degree t Polynomials

To see how to construct an algebraic PIR with constant[11] server-side communication given a cryptosystem that allows polynomials of total degree t to be computed on ciphertext, you can see the work of [BGN05]. 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

[11] "Constant" refers to the fact that a single, compact ciphertext is returned by the server; there is of course logarithmic dependence of the security parameter on the size of the database.

labeled $X = \{x_{i_1,\ldots,i_t}\}_{i_k \in [0, \sqrt[t]{(n)}-1]}$. Then for any $X \in \{0,1\}^n$, define

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

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 $\widetilde{F}((w_1)_1, \ldots, (w_t)_{\sqrt[t]{n}-1})$ as the response to the query, which will clearly be an encryption of $x_{i_1^*,\ldots,i_t^*}$. 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.