Secure Communication: CDS, PIR, PSM

by Isaac Grosof

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfilment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2017

Author:_____
Department of Electrical Engineering and Computer Science
May 26, 2017

Certified by:_____
Professor Vinod Vaikuntanathan, Thesis Advisor
May 26, 2017

Accepted by:_____
Christopher J. Terman, Chairman,
Masters of Engineering Thesis Committee

1 Abstract

Private Information Retrieval is the problem of querying two servers to find a value in a database, while keeping the index private. We extend this problem to Generalized Wildcard PIR, where we instead query an aggregate of the entries whose indices match a pattern, called a generalized wildcard, which specifies what values each segment of the indices may take. We give a construction for this variant with similar communication to that of the best PIR protocols known.

We study information theoretic models in cryptography, namely Private Information Retrieval, Conditional Disclosure of Secrets, and Private Simultaneous Messages. We give extensions of PIR and CDS in the area of generalized wildcards, and give constructions for those extensions. We discuss directions towards more efficient protocols, and raise open questions.

Secure Communication: CDS, PSM, PIR

by Isaac Grosof

Contents

1 Abstract **2**

2 Introduction **4**

3 Organization **5**

4 Generalized Wildcard **6**

5 Matching Vector Families **7**
5.1 Matching Vector Family Construction 8
5.1.1 Low degree AND Polynomial 8
5.1.2 Boolean Polynomial Simulation 10
5.2 Generalized Wildcard Matching Vectors 13

6 Conditional Disclosure of Secrets **14**
6.1 Existing protocols 16
6.1.1 $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$ protocol 16
6.1.2 Matching Vectors-based protocol 21
6.2 Generalized Wildcard 25

7 Private Information Retrieval **30**
7.1 Existing Protocols 32
7.1.1 Cube-based $N^{1/3}$ protocol 32
7.1.2 Matching Vectors-based protocol 34
7.2 Generalized Wildcard Extension 35
7.2.1 Generalized Wildcard Cube-based protocol 37
7.2.2 Generalized Wildcard Matching Vectors protocol . . . 40

8 Private Simultaneous Messages for General Functions **43**
8.1 Inner Product 45
8.2 4-Hypercube based protocol 46
8.3 Degree 4 polynomial protocol 49

9 Conclusion **52**

2 Introduction

Cryptographers have long studied the relationship between communication and security. In particular, we often try to examine the situations under which secure communication requires more communication than insecure communication, and when it does not. Beyond that, we simply ask how we can create efficient secure protocols. These systems explore the contention between trying to communicate some information to another party, while not sending too much information and thereby violating security.

To answer these questions, cryptographers have invented simplified scenarios, with a small number of participants, well defined objectives and adversaries with specified powers. In this thesis, we shall explore three such simplified settings, each with one round and three participants in an information theoretic security context.

First, the Conditional Disclosure of Secrets model, [7] where two parties sharing only a random string must coordinate their messages to disclose or not disclose a secret based on their inputs. Second, the Private Information Retrieval model, [4] where a client queries two servers to retrieve information, while not revealing what information is desired to either server. Third, the Private Simultaneous Messages model, [6] where two parties sharing only a random string must coordinate their messages to reveal specified information, and nothing else.

In each of these models, we will present illustrative protocols from the literature, including the best known protocols. We will attempt to present both the letter and the spirit of the protocols, in as understandable and intuitive of a way possible.

The best Conditional Disclosure of Secrets and Private Information Retrieval protocols rely on a tool called a matching vector family. We present a construction of a matching vector family, appropriate for use in the CDS and PIR protocols.

We present new extensions of PIR and CDS, based on the idea of a generalized wildcard. We give constructions for these extensions. To do so, we extend the matching vector family tool.

Finally, we will discuss impedances in the way of every more efficient proto-

cols, and open problems for future lines of research.

3 Organization

In Section 4, we define the generalized wildcard concept.

In Section 5, we study matching vector families. In Section 5.1, we present an existing matching vector family construction with vectors of size $2^{O(\sqrt{\log n \log\log n})}$. [1] [8]

In Section 5.2, we extend the matching vector family tool to define a generalized wildcard matching vector family. In that section, we also present an original construction of a generalized wildcard matching vector family for wildcards with matches of size $\log\log n / \log\log\log n$, again with vectors of size $2^{O(\sqrt{\log n \log\log n})}$.

In Section 6, we study the Conditional Disclosure of Secrets problem. We present two existing CDS protocols. In Section 6.1.1, we present an existing protocol for CDS for $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$ with $O(N^{1/3})$ communication, and a consequent CDS protocol for INDEX_N with $O(N^{1/3})$ communication. [9] In Section 6.1.2, we present an existing protocol for CDS for INDEX_N with $2^{O(\sqrt{\log N \log\log N})}$ communication, which makes use of a matching vector family. [9]

In Section 6.2, we present an original protocol for $\text{AnyGW}_N^{\log\log N/\log\log\log N}$, a generalized wildcard CDS protocol that is stronger than INDEX_N. This protocol is in the Common Reference String model, and has ϵ one-sided error, with $2^{O(\sqrt{\log n \log\log n})} \log 1/\epsilon$ communication. This protocol makes use of the generalized wildcard matching vector family from Section 5.2

In Section 7, we study the Private Information Retrieval problem. We present two existing PIR protocols. In Section 7.1.1, we present an existing protocol for PIR with $O(N^{1/3})$ communication based on thinking of the database as a three-dimensional cube. [4] In Section 7.1.2, we present an existing protocol for PIR with $2^{O(\sqrt{\log n \log\log n})}$ communication, which makes use of a matching vector family. [5]

In Section 7.2, we define an pair of extensions to PIR based on generalized wildcards, namely XorGW_N^k and AnyGW_N^k, which aggregate over all

database entries whose indices match a wildcard with matches of length k. The aggregation functions are the XOR and ANY functions, respectively. In Section 7.2.1, we present an original construction for $\text{XorGW}_N^{\log N^{1/3}}$, with $O(N^{1/3})$ communication. In Section 7.2.2, we present an original construction for $\text{AnyGW}_N^{\log\log N/\log\log\log N}$. This protocol is in the Common Reference String model, and has ϵ one-sided error, with $2^{O(\sqrt{\log n \log\log n})} \log 1/\epsilon$ communication. This protocol makes use of the generalized wildcard matching vector family from Section 5.2.

In Section 8, we study the Private Simultaneous Messages problem. In Section 8.1, we present a PSM protocol for inner product, which is a special case of a protocol from [9]. In Section 8.2, we present an existing protocol for PSM for an arbitrary predicate, ALL_N, with $O(N^{1/2})$ communication, which is based on thinking of the predicate as a four-dimensional cube. [2] We give a fairly different presentation of the protocol, however. In Section 8.3, we present an existing protocol for PSM with an arbitrary predicate, ALL_N, with $O(N^{1/2})$ communication, based on a protocol for multivariate polynomials of degree 4. [9]

4 Generalized Wildcard

The idea of a generalized wildcard is used throughout this thesis to create new cryptographic constructions. A generalized wildcard encodes a pattern which a string may match.

A generalized wildcard w will match a binary string of length l, with matches of length m. w consists of a sequence of sets, where each set w_i is a subset of $\{0,1\}^m$. Think of a binary string $s \in \{0,1\}^l$ as being divided into segments of length m. Call this subdivision s', and call the ith segment s'_i A generalized wildcard w matches a string s if $s'_i \in w_i \forall i$.

In some cases, we may want to think about a string over some other alphabet matching a generalized wildcard. In this case, we can imagine converting the symbols in that alphabet to some binary encoding before applying the wildcard.

A conventional wildcard is the special case of a generalized wildcard with matches of length 1. A single index is the special case where each subset has

exactly one element. A generalized wildcard is more powerful than either of these ideas.

5 Matching Vector Families

For some of the cryptographic protocols in this thesis, we will make use of a matching vector family. A matching vector family is a set of vectors with restricted inner products, which can be useful for certain constructions.

In general, a matching vector family is a set of vectors U and some modulus m such that $u_i^T u_i \equiv 0 \mod m$, while $u_i^T u_j \not\equiv 0 \mod m \, \forall i \neq j$. These are most useful when the size of the vectors in the family is much less than the number of vectors in the family.

In this section, we will give a construction of a specific matching vector family, with some specific properties which will prove relevant. The construction will set $m = 6$, and provide the additional guarantee that $u_i^T u_j \mod 6 \in \{0, 1, 3, 4\}$, which will prove vital to its later use.

The construction will proceed in two steps: First, we will give a construction of a polynomial whose output is $0 \mod 6$ if its input is all 1s, and in the set $\{1, 3, 4\}$ for any other input in $\{0, 1\}^n$. This polynomial is due to Barrington, Beigel and Rudich. [1] Second, we will give a construction of a set of vectors whose pairwise inner products are closely related to the above polynomial. This construction is due to Grolmusz. [8]

Moreover, the polynomial will be of degree $O(n^{1/2})$, leading to vectors of length $2^{O(\sqrt{\log n \log \log n})}$ where n is the number of vectors. This will give rise to the best known cryptographic protocols for Conditional Disclosure of Secrets and Private Information Retrieval, as illustrated in Section 6.1.2 and Section 7.1.2 respectively.

We would like to create smaller matching vector families, to potentially improve our cryptographic protocols. However, lower bounds in [3] tightly bound how small matching vector families can be, at a point not much smaller than the best known constructions.

In addition to going over this existing construction, we will also present a new extension of matching vectors, namely generalized wildcard matching

vectors. This extends the capability of a matching vector family from being able to indicate index equality to being able to indicate generalized wildcard matching, a far more powerful concept. This extension builds upon both steps of the matching vector construction presented here.

Open questions:

- What other extensions of matching vectors can we create?

5.1 Matching Vector Family Construction

5.1.1 Low degree AND Polynomial

In [1], Barrington, Beigel and Rudich showed that for any composite m, there exists a specific n input polynomial OR_n such that $OR_n(0,0,\ldots) = 0 \mod m$, and OR_n does not equal $0 \mod m$ for any other input. Moreover, they showed that such a polynomial exists with degree equal to $O(n^{1/r})$, where r is the number of distinct prime factors that m has. Finally, they also showed that such a construction exists with the additional guarantee that the value of OR_n is restricted to one of 2^r values $\mod m$, regardless of the input.

The only case we will need for constructions later in this thesis is the case where $m = 6$. In addition, we will construct an AND_n polynomial, rather than an OR_n polynomial. However, we will give a more general proof, as it makes the construction no harder. This proof is based on a presentation of the proof from [8].

Theorem 1. *Given $m = p_1 p_2 \ldots p_r$ where p_i are distinct primes, there exists an explicitly constructible polynomial AND'_n such that $AND'_n(1,1,1\ldots) = 0 \mod m$ and $AND'_n(x) \in \{0,1\} \mod p_i$, for an arbitrary input x and for any prime factor p_i, and AND'_n has degree $O(n^{1/r})$.*

Let $x \in \{0,1\}^n$ be a vector of n variables, which we will consider to be the inputs. We will define $s_k(x)$ to be the kth elementary symmetric polynomial:

$$s_k(x) := \sum_{1 \leq j_1 \leq j_2 \ldots j_k \leq n} x_{j_1} x_{j_2} \ldots x_{j_k}$$

Let us also define the hamming weight of x, $wt(x) := \sum_{i \in [n]} x_i$. With this

definition

$$s_k(x) = \binom{wt(x)}{k}$$

This is due to the fact that a term of s_k is equal to 1 iff every variable in that term is 1, and there are $\binom{wt(x)}{k}$ ways to choose k variables that are set to 1 out of $wt(x)$ such variables.

Since the value of $s_k(x)$ depends only on $wt(x)$, we can prove the following lemma:

Lemma 1. *For any positive integer k, and prime p and integer e such that $k < p^e$, if $wt(x) = j$ and $wt(x') = j + p^e$, then $s_k(x) \equiv s_k(x') \mod p$.*

The statement is equivalent to the following:

$$\binom{j+p^e}{k} \equiv \binom{j}{k} \mod p$$

The following is an identity which is true for all integers u, v, t:

$$\binom{u+v}{t} \equiv \sum_{w \in [t]} \binom{u}{w}\binom{v}{t-w}$$

Making use of the fact that for any $1 \leq l \leq p^e$, $\binom{p^e}{l} \equiv 0 \mod p$, we can substitute $u = j, v = p^e, t = k$, to get

$$\binom{j+p^e}{k} = \sum_{w \in [k]} \binom{j}{w}\binom{p^e}{k-w} \equiv \binom{j}{k}\binom{p^e}{k-k} = \binom{j}{k} \mod p$$

□

Next, for $i \in [r]$, let e_i be the smallest integer such that

$$p_i^{e_i} > \lceil n^{1/r} \rceil$$

We define, for $i \in [r]$, the polynomial $G_i(x)$ to be

$$G_i(x) := \sum_{j=1}^{p_i^{e_i}-1} (-1)^{j+1} s_j(x)$$

Notice that the OR_n polynomial, the polynomial such that $OR_n(0,0,\ldots) = 0$ and for all other inputs equals 1, can be written as

$$1 - \prod_{i \in [n]} 1 - x_i = \sum_{j=1}^{n} (-1)^{j+1} s_j(x)$$

Thus, $G_i(x) = OR_n(x)$ on all inputs x with $wt(x) \leq n^{1/r}$, since $s_j(x)$ equals zero on all j such that $j > wt(x)$.

Moreover, $G_i \mod p_i$ is periodic with period $p_i^{e_i}$, due to Lemma 1.

Now, we can apply the Chinese Remainder Theorem, which shows that there exists a polynomial P which satisfies

$$P \equiv G_i \mod p_i \, \forall i \in [r]$$

In fact, since the Chinese Remainder Theorem combines the polynomials in question additively, the degree of P is the maximum of the degrees of the polynomials G_i, which is $O(n^{1/r})$.

For any $x \in \{0,1\}$, if $wt(x) \neq 0$ then there exists an $i \in [r]$ such that $wt(x) \not\equiv 0 \mod p_i^{e_i}$, since the product of the $p_i^{e_i}$ is greater than $(n^{1/r})^r = n$, and the prime powers are all relatively prime. Therefore, $G_i(x) \not\equiv 0 \mod p_i$, and so $P(x) \not\equiv 0 \mod p_1 p_2 \ldots p_r = m$. On the other hand, $P(0,0,\ldots 0) = 0$.

Thus, the polynomial $AND'_n = P(1-x_1, 1-x_2, \ldots 1-x_n)$ has the property that $AND_n(1,1,\ldots 1) \equiv 0 \mod m$, and for all other inputs, $AND'_n(x) \not\equiv 0 \mod m$. Moreover, since the result is equal to $G_i(x) \mod p_i$ for all i, and $G_i(x) \in \{0,1\}$, $AND'_n(x) \mod p_i \in \{0,1\}$. This polynomial has degree $O(n^{1/r})$, as desired. □

As a corollary, for the specific case where $m = 6$, this establishes the following:

Corollary 1. *There exists a polynomial AND'_n such that $AND'_n(1,1,\ldots 1) \equiv 0 \mod 6$, and $AND'_(x) \mod 6 \in \{1,3,4\}$ for all other inputs, with degree $O(n^{1/2})$.*

5.1.2 Boolean Polynomial Simulation

In the paper "Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs" [8], Grolmusz constructed a set of sets

whose intersections had specific sizes mod 6. When these sets are replaced with the indicator vectors of each set, the resulting vectors form a matching vector family. Here, we present a construction of that matching vector family from an vector-first perspective, as opposed to the original set intersection perspective.

We will show that given a low degree boolean polynomial for AND mod m, we can create a matching vector family with n vectors of length subpolynomial in n. The specific result of importance for later cryptographic protocols concerns the case where $m = 6$:

Theorem 2. *There exists a polynomial-time constructible set of n vectors of length $2^{O(\sqrt{\log n \log\log n})}$, which we will call U, such that $\forall u_i \in U, u_i^T u_i = 0 \mod 6$, and $\forall u_i, u_j \in U, i \neq j, u_i^T u_j \mod 6 \in \{1, 3, 4\}$.*

This result will follow from a more general lemma about simulating boolean polynomials with the inner products of vectors.

A boolean polynomial is one where the inputs are restricted to be 0 or 1.

We say that a boolean polynomial Q on b inputs is simulated by a set of vectors U of size b^b in the following scenario:

Each vector u_i is associated with an index i, which is in turn associated with a tuple of b integers in the range $[b]$. Call the tuple s^i. Given two vectors u_i, u_j, let $t_k^{ij} = 1$ if $s_k^i = s_k^j$, and 0 otherwise. t^{ij} is the element-wise equality of s^i and s^j. The vectors simulate the polynomial if for all i, j, $u_i^T u_j = Q(t_1^{ij}, t_2^{ij}, \ldots)$.

With that, we can state our lemma:

Lemma 2. *Given a boolean polynomial Q on b inputs of degree d evaluated mod m, with $b > 2d$, there exists a set of b^b vectors of length no more than $2(m-1)b^{2d}/d!$ which simulates Q.*

Let $U = \{u_i\}$ be the set of b^b vectors we are trying to construct, as described above.

We will start by simulating a single term of a boolean polynomial with coefficient 1. Without loss of generality, such a term of a boolean polynomial can be written as

$$Q_z = x_{z1} x_{z2} x_{z3}, \ldots$$

There are no exponents because a boolean variable raised to any positive

integer exponent equals the variable itself. The number of variables in this term, $|Q_z|$, is no more than d, the degree of Q.

Our objective for a single term is to map each index number s^i to a vector u_{iz} such that ${u_{iz}}^T u_{j_z} = Q_z(t_1^{ij}, t_2^{ij}, \ldots)$.

We will do so by making each u_{*z} a vector of dimension $b^{|Q_z|}$. A given u_{*z} will have a single non-zero entry, which will be a 1 in a specific location. u_{iz} has a 1 in a location uniquely determined by the indices of s^i corresponding to the variables of Q_z, $(s^i_{z1}, s^i_{z2} \ldots)$. $z1, z2, \ldots$ are the indices of the variables of the term we are simulating. Specifically, the location with a 1 will be at position $[s^i_{z1} s^i_{z2} \ldots]_b$, where the $[]_b$ notation indicates that we are interpreting the sequence of digits as a number in base b.

Since this mapping from the relevant digits of s^i to the entry of u_{iz} is bijective, ${u_{iz}}^T u_{j_z} = 1$ if and only if $s^i_{z1} = s^j_{z1} \,\Lambda\, s^i_{z2} = s^j_{z2} \,\Lambda\, \ldots$, which is equivalent to the statement $(t^{ij}_{z1} = 1) \,\Lambda\, (t^{ij}_{z2} = 1) \,\Lambda\, \ldots$. A term of a boolean polynomial is equal to one if and only if all of its variables are equal to one, so this vector perfectly simulates the term.

Next, we need to simulate the polynomial, rather than just a term. We can do this by simply concatenating the vectors corresponding to each term. If $a^T b = x$ and $c^T d = y$, then $(a|c)^T (b|d) = x + y$. Thus, let $u^j = (u^j_1 | u^j_2 | u^j_3 \ldots)$.

$$ {u_i}^T u_j = \sum_{z \in |Q|} {u_{iz}}^T u_{j_z} = \sum_{z \in [|Q|]} Q_z(t_1^{ij}, t_2^{ij}, \ldots) = Q(t_1^{ij}, t_2^{ij}, \ldots) $$

The inner product has the desired value, completing the construction.

Now, we will calculate the length of the vectors. There are no more than $\binom{b}{l}$ unique terms of degree l, and each one has a coefficient no more than $m-1$, as Q is evaluated modulo m. Thus, all of Q's degree l terms can be written as the sum of no more than $(m-1)\binom{b}{l}$ terms containing l variables each. These terms are simulated by vectors of length $(m-1)\binom{b}{l} b^l = (m-1) b^{2l}/l!$. Therefore, the overall length of the vectors is

$$ \sum_{l=1}^{d} (m-1) b^{2l}/l! \leq 2(m-1) b^{2d}/d! $$

This inequality holds because the terms on the left of the inequality grow by a multiplicative factor of b^2/l, so if $b > 2d$, this is at least a factor of 2 at

each term, and so we can upper bound the sequence by a geometric sequence ending at $(m-1)b^{2d}/d!$.

This proves Lemma 2. □

Now, let us consider the specific polynomial AND'_m from Section 5.1.1 due to [1]. $Q(1,1,\ldots) = 0 \mod 6$, and on every other input the result is in $\{1,3,4\} \mod 6$. A set of vectors that simulates this polynomial will have the desired property that $u_i^T u_i = 0 \mod 6$ and $u_i^T u_j \mod 6 \in \{1,3,4\} \forall i \neq j$.

All that is left to prove is the length of the vectors.

Substituting in the specific values of d and m for the polynomial in question, namely $d = O(b^{1/2})$ and $m = 6$, we find that the length of the vectors is $O(b^{O(b)}/O(b^{1/2})!)$

We now have $n = b^b$ matching vectors, each of length $O(b^{O(b)}/O(b^{1/2})!)$. With some tedious algebra, this length can shown to be equal to $2^{O(\sqrt{\log n \log\log n})}$.

□

5.2 Generalized Wildcard Matching Vectors

We will extend the Grolmusz-type matching vector construction in Section 5.1.2 to create a "generalized wildcard matching vector family", a new construction.

Theorem 3. *Given a matching vector family U as described in Section 5.1.2, with $|U| = n$, and a generalized wildcard w of length $\log n$ and with matches of length $\log\log n/\log\log\log n$, there exists a polynomial-time constructible vector v_w such that $\forall u_i \in U, v_w^T u_i \equiv 0 \mod 6$ if and only if w matches i, and $v_w^T u_i \mod 6 \in \{1,3,4\}$ otherwise.*

A single section of the wildcard w will match a portion of the index i corresponding to one digit of s^i, after the conversion to base b. This value is in the range $[b]$, where $b^b = n$, giving rise to a match of size $\log b = \log\log n/\log\log\log n$.

When constructing v_w, we will ensure that $v_w^T u_i = Q(t_1^{wi}, t_2^{wi}\ldots)$, where $Q = AND'_n$ is the same polynomial as in the original construction, and $t_k^{wi} = 1$ if and only if $i_k \in w_k$, e.g. if w matches i at index k.

To start constructing v_w, we will construct the vector corresponding to a single term of Q, as the construction of a given u_i is the concatenation of vectors corresponding to each term of Q. We will call the portion of the vector u_i corresponding to one term of Q a block.

The zth term of Q, Q_z, is $x_{z1}x_{z2}\ldots$. Recall from Section 5.1.2 that u_{iz}, the zth block in u_i, has a 1 in exactly one location, namely at entry $[i_{z1}i_{z2}i_{z3}\ldots]_b$, where b is the base the construction is for, with $n = b^b$. The rest of the entries of the block are 0.

When constructing v_w, we will ensure that ${v_{wz}}^T u_{iz} = Q_z(t_{z1}^{wi}, t_{z2}^{wi}, \ldots)$. To do so, we will set v_{wz} to have a 1 in every entry $[s_{z1}s_{z2}s_{z3}\ldots]_b$ such that $\forall x_{zj} \in Q_z, s_{zj} \in w_{zj}$, and 0s elsewhere. There is a 1 at every index corresponding to every index which matches the appropriate subset of w.

When we take the inner product of v_{wz} and u_{iz}, the result is equal to the value of v_{wz} at the index $[i_{z1}i_{z2}i_{z3}\ldots]_b$, since u_{iz} is 1 at that index and 0 elsewhere. That index in v_{wz} is a 1 if and only if $\forall x_{zj} \in Q_z, i_{zj} \in w_{zj}$, and is 0 elsewhere, which is exactly equal to $\prod_{x_{zj} \in Q_z} t_{zj}^{wi} = Q_z(t_{z1}^{wi}, t_{z2}^{wi}, \ldots)$. Thus, the inner product of the blocks is as desired.

We will define v_w to be the concatenation of all of these blocks.

By the same argument as in the prior construction, the inner product of the concatenations of the blocks, u_i and v_w, equals the sum of the inner products of the blocks. So $v_w^T u_i = \sum_z Q_z(t_{z1}^{wi}, t_{z2}^{wi}, \ldots) = Q(t_1^{wi}, t_2^{wi}, \ldots)$, as desired.

Since $v_w^T u_i$ is equal to the output of the polynomial Q on some input, for all u_i such that i matches w, $v_w^T u_i \equiv 0 \mod 6$. On the other hand, for all u_i such that i does not match w, $v_w^T u \mod 6 \in \{1, 3, 4\}$. □

6 Conditional Disclosure of Secrets

Two-party conditional disclosure of secrets, invented by Gertner, Ishai, Kushilevitz and Malkin [7], is a simple secure-communication problem: two parties share a secret, and want to disclose that secret to a third party if and only if their inputs satisfy some fixed, public predicate $P : X \times Y \to \{0, 1\}$.

More specifically, Alice has input x, Bob has input y, and Alice and Bob share a secret μ, and private randomness r. Charlie knows x and y but not μ or r. Alice and Bob want to disclose μ to Charlie iff $P(x, y) = 1$. Alice and Bob can each send a message to Charlie; no other communication is allowed. How many bits do Alice and Bob need to communicate to Charlie?

$O(N^{1/2})$ protocols for an arbitrary predicate are well known for this problem. Recently, better protocols have been discovered. [9]

The problem of constructing a protocol for an arbitrary predicate $P : [N] \times [N] \to \{0, 1\}$ is called ALL_N. Some protocols for solving ALL_N do so by solving a more general problem called INDEX_N, where the predicate is $f(D, i) = D_i, D \in \{0, 1\}, i \in [N]$. Any instance of ALL_N can be reduced to INDEX_N by defining $P_x(y) := P(x, y)$, and representing P_x as a truth table. Note that this transformation can introduce some inefficiency, since protocols of this form do not make use of the fact that P is public in ALL_N.

A central question in the world of CDS constructions is to try to figure out where we should look to try to find more efficient protocols. This question was partially answered by [9] who showed that any protocol for INDEX with k-degree reconstruction from the messages to Charlie's output must have $\Omega(N^{1/k+1})$ communication.

A different way to answer this question, original to this work, is to examine the additional work that a given CDS protocol performs, beyond simply solving INDEX_N. This can be made concrete by specifying a problem which the protocol can solve, which in turn can solve INDEX_N.

The INDEX_N problem's best known lower bound is $\Omega(\log N)$ communication, which is fairly trivial.

In contrast, one of the protocols for INDEX_N we will examine below will solve $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$, and use that to solve INDEX_N. $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$, which is explained in detail below, can also be used to solve inner product protocol on $N^{1/3}$ bits, as we will establish in Theorem 6. Thus, any protocol which can solve $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$ cannot be more efficient that $O(N^{1/3})$ without solving INNER_N in less than $O(N)$, which would be a major surprise in the world of CDS protocols.

The matching vector based CDS construction does not obviously solve some more difficult predicate than INDEX_N. However, in Section 6.2, we show that

replacing the matching vector family with a generalized wildcard matching vector family allows a closely related protocol to be used as a subroutine to solve AnyGWMV_N, which can in turn solve INDEX_N, up to a different model and an error term.

The construction in Section 6.2, in addition to giving evidence that the matching-vector based construction cannot be made perfectly efficient, is interesting in its own right, as it expands the scope of CDS predicates which can be performed at relatively high efficiency, and is original to this thesis.

Open problems:

- Is there a predicate which the matching vector construction can be used to solve which is more powerful than AnyGWMV_N? Is there a predicate which the matching vector construction is perfectly efficient for, up to assumptions about INNER_N or similar predicates?
- Is there a $\Omega(n)$ lower bound for INNERn?
- How can we do less work beyond INDEX_N?
- Can we construct a protocol for ALL_N without going through INDEX_N, more efficiently?

6.1 Existing protocols

We present a detailed analysis of two CDS protocols, which are both due to Liu, Vaikuntanathan and Wee. [9]

6.1.1 $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$ protocol

In this construction, we will solve the INDEX_n problem by first solving the more general problem $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$. The protocol will use $O(N^{1/3})$ communication.

MPOLY here stands for the class of boolean polynomials, polynomials over $\mathbb{F}_2$. Since each variable can only be 0 or 1, we may consider each variable to have degree at most 1 in each term.

MPOLY^3 specifically refers to degree-3 boolean polynomials, and $\text{MPOLY}^3_{n^{1/3}n^{1/3}n^{1/3}}$ refers to degree-3 boolean polynomials where the variables are divided into three sets, x_1, x_2, x_3, and each term of the polynomial consists of exactly one variable from each set.

Theorem 4. *There exists a CDS protocol for* $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$ *with* $O(N^{1/3})$ *communication.*

Since $x_1 \otimes x_2 \otimes x_3$ is the vector consisting of every possible degree-3 term with one polynomial from each set, a polynomial of the above form can be written as $\langle p, x_1 \otimes x_2 \otimes x_3 \rangle$, where p is a length N vector giving the coefficient of each term.

Our predicate will thus be the following: $P(p, (x_1, x_2, x_3)) = \langle p, x_1 \otimes x_2 \otimes x_3 \rangle$ over $\mathbb{F}_2$.

With this predicate, the setup is as follows:

Alice holds the polynomial p, as well as the secret μ and the shared randomness r. Bob holds the sets of variables (x_1, x_2, x_3), as well as the secret μ and the shared randomness r. Charlie holds the polynomial p and the variables (x_1, x_2, x_3).

At a high level, the protocol works as follows:

Let $b_1, b_2, b_3 \in \{0,1\}^{N^{1/3}}$ be random vectors sampled from r. Bob will start by sending $m_B^1 := (\mu x_1 + b_1)$ to Charlie, and m_B^2, m_B^3 defined similarly. The b vectors are used are one-time pads for the secret information μx, and ensure that no information leaks at this step.

Next, Charlie calculates

$$\langle p, (\mu x_1 + b_1) \otimes (\mu x_2 + b_2) \otimes x_3 \rangle +$$

$$\langle p, (\mu x_1 + b_1) \otimes x2 \otimes (\mu x_3 + b_3) \rangle +$$

$$\langle p, x_1 \otimes (\mu x_2 + b_2) \otimes (\mu x_3 + b_3) \rangle$$

Let us examine the value of the right side of one of these terms:

$$((\mu x_1 + b_1) \otimes (\mu x_2 + b_2) \otimes x_3)$$

$$= \mu^2(x_1 \otimes x_2 \otimes x_3) + \mu(x_1 \otimes b_2 \otimes x_3) + \mu(b_1 \otimes x_2 \otimes x_3) + b_1 \otimes b_2 \otimes x_3$$

If we sum over all three terms, the result is:

$$3\mu^2(x_1 \otimes x_2 \otimes x_3)$$
$$+2\mu(x_1 \otimes x_2 \otimes b_3 + x_1 \otimes b_2 \otimes x_3 + b_1 \otimes x_2 \otimes x_3)$$
$$+b_1 \otimes b_2 \otimes x_3 + b_1 \otimes x_2 \otimes b_3 + x_1 \otimes b_2 \otimes b_3$$

Since we are working over $\mathbb{F}_2$, the above simplifies to

$$\mu(x_1 \otimes x_2 \otimes x_3)$$
$$+b_1 \otimes b_2 \otimes x_3 + b_1 \otimes x_2 \otimes b_3 + x_1 \otimes b_2 \otimes b_3$$

Reintroducing the inner product with p, the current value is

$$\mu\langle p, x_1 \otimes x_2 \otimes x_3\rangle$$
$$+\langle p, b_1 \otimes b_2 \otimes x_3 + b_1 \otimes x_2 \otimes b_3 + x_1 \otimes b_2 \otimes b_3\rangle$$

The former term is the value we are attempting to communicate to Charlie, since it is 0 if $P(p, (x_1, x_2, x_3)) = 0$, and it is equal to μ otherwise. The latter term still needs to be removed.

If we think of the latter term not as an inner product but as a polynomial, and we think of p, b_1, b_2, b_3 as constants, it is clear that the latter term is a degree 1 polynomial. The tensor product never multiplies an x variable by another x variable.

A degree 1 polynomial can also be thought of as a simple inner product. In particular, that term can be written as an inner product between values known to Alice and values known to both Bob and Charlie.

The following general protocol allows us to communicate $\langle x, y\rangle$ to Charlie when Alice knows x and Bob and Charlie know y, and Alice and Bob share randomness r:

$$M_A = x + r$$
$$M_B = \langle y, r\rangle$$

Charlie outputs $\langle M_A, y\rangle - M_B$.

This protocol is secure because the message distribution can be perfectly simulated by a simulator only knowing y and $\langle x, y\rangle$. M_A has a uniformly

random distribution, as r serves as a one-time pad, and M_B is uniquely determined given M_A and $\langle x, y \rangle$, via the formula $M_B = \langle M_A, y \rangle + \langle x, y \rangle$.

This type of protocol, where one input party's input is secret and the other input party's input is known to the receiver, is known as 1/2 PSM, and is covered further in Section 8.

Since the term

$$+\langle p, b_1 \otimes b_2 \otimes x_3 + b_1 \otimes x_2 \otimes b_3 + x_1 \otimes b_2 \otimes b_3 \rangle$$

is a degree 1 polynomial in x_1, x_2, x_3, it can be written as

$$\langle f(p, b_1, b_2, b_3), x_1 || x_2 || x_3 \rangle$$

for some fixed function f. Now, we are ready to apply the 1/2 PSM protocol from above. Let c be sampled from the shared randomness.

$$M_A = f(p, b_1, b_2, b_3) + c$$

$$M_B = \langle x_1 || x_2 || x_3, c \rangle$$

Charlie can now learn $\langle M_A, x_1 || x_2 || x_3 \rangle = \langle f(p, b_1, b_2, b_3), x_1 || x_2 || x_3 \rangle$, while learning no other new information, thereby preserving security.

Charlie can now subtract this value from the value calculated previously, and calculate $\mu \langle p, x_1 \otimes x_2 \otimes x_3 \rangle$. If $\langle p, x_1 \otimes x_2 \otimes x_3 \rangle = 1$, Charlie learns μ, as desired.

To see that Charlie learns nothing about μ when $\langle p, x_1 \otimes x_2 \otimes x_3 \rangle = 0$, we will show that a simulator with access to p and x_1, x_2, x_3 but not μ can perfectly simulate the messages that Charlie receives. However, we will first give a formal description of the protocol:

- Alice's input: $p \in \{0,1\}^N$, the coefficients of the polynomial.
- Bob's input: $x_1, x_2, x_3 \in \{0,1\}^{N^{1/3}}$, $\mu \in \{0,1\}$.
- Charlie's output: $\mu \langle p, x_1 \otimes x_2 \otimes x_3 \rangle$
- Shared randomness: $b_1, b_2, b_3 \in \{0,1\}^{N^{1/3}}, c \in \{0,1\}^{3N^{1/3}}$
- Alice's message: Using the fixed function f which was proven to exist above, Alice sends $m_A := f(p, b_1, b_2, b_3) + c$.

- Bob's messages: Bob sends $(m_B^1, m_B^2, m_B^3, m_B^4) :=$ $(\mu x_1 + b_1, \mu x_2 + b_2, \mu x_3 + b_3, \langle c, x_1 || x_2 || x_3 \rangle)$.
- Charlie's reconstruction: Charlie outputs $\langle p, m_B^1 \otimes m_B^2 \otimes m^3 \rangle$ $-\langle m_A, x_1 || x_2 || x_3 \rangle + m_B^4$

The messages that Charlie receives can be simulated as follows: m_B^1, m_B^2, m_B^3, m_A are all sampled independently uniformly at random. This is identical to their distributions under any input, because they are each the result of a one-time pad, b_1, b_2, b_3, c respectively. $m_B^4 = \mu\langle p, x_1 \otimes x_2 \otimes x_3 \rangle - \langle p, m_B^1 \otimes m_B^2 \otimes m_B^3 \rangle +$ $\langle m_A, x_1 || x_2 || x_3 \rangle$, meaning that it is computable by the simulator from the output $(\mu\langle p, x_1 \otimes x_2 \otimes x_3 \rangle)$, the other messages, and values which Charlie starts out knowing. Thus, the protocol is secure.

We have now solved the $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$ problem. This solution uses a total amount of communication equal to $3N^{1/3}$ bits from Alice to Charlie, and $3N^{1/3} + 1$ bits from Bob to Charlie. □

This protocol is very powerful. By utilizing only a small part of its power, we will solve the INDEX_N problem, with the same amount of communication. Recall that the INDEX_n problem gives Alice a database D and Bob an index i, and the value that Charlie should learn is μD_i. To solve this, we will use the above procedure, with very simple modifications.

Theorem 5. *There is a CDS protocol for INDEX_N with $O(N^{1/3})$ communication.*

D will be used verbatim as p. We can write i as $[i_1 i_2 i_3]_{N^{1/3}}$, and $e_i = e_{i_1} \otimes e_{i_2} \otimes e_{i_3}$. Thus, we can apply the MPOLY procedure above to communicate $\mu\langle D, e_{i_1} \otimes e_{i_2} \otimes e_{i_3} \rangle = \mu D_i$, thus solving the INDEX problem in $O(N^{1/3})$ communication as well. □

Finally, we will relate the efficiency of this MPOLY protocol to the INNER_N predicate. INNER_N is the CDS problem with the predicate $p(u, v) = \langle u, v \rangle$, where the inner product is taken over $\mathbb{F}_2$.

Theorem 6. *If MPOLY^3_{nnn} can be solved in $C(n)$ communication, then INNER_n can be solved in $C(n)$ communication.*

Given a pair of vectors u, v in a INNER_n instance, we will construct an MPOLY^3_{nnn} instance with the same predicate value by setting $x_1 = v$, $x_2 = e_1$, $x_3 = e_1$, so that $x_1 \otimes x_2 \otimes x_3 = v|000\ldots$. Then, we will set $p = u|000\ldots$. Now, we are ready to use our MPOLY^3_{nnn} protocol, with p and x_1, x_2, x_3 as

the inputs. Since $\langle p, x_1 \otimes x_2 \otimes x_3 \rangle = \langle u, v \rangle$, this is a correct and secure INNER_n protocol with $C(n)$ communication. $\square$

Thus, since $\text{MPOLY}^3_{N^{1/3}N^{1/3}N^{1/3}}$ was solved with $O(N^{1/3})$ communication, that construction is asymptotically efficient, unless INNER_N can be solved with less than $O(N)$ communication.

6.1.2 Matching Vectors-based protocol

Theorem 7. *There is a CDS protocol for INDEX_N with $2^{O(\sqrt{\log N \log\log N})}$ communication.*

This protocol will directly solve the INDEX_N CDS predicate, and will do so by making use of a matching vector family of size N and vector length $2^{O(\sqrt{\log N \log\log N})}$, the protocol will use $2^{O(\sqrt{\log N \log\log N})} = N^{o(1)}$ communication.

The specific properties of the matching vector family that we will rely on are that there exists $\{u_i\}_{i\in[N]}$ such that $u_i \in \mathbb{Z}_6^l$ where $l = 2^{O(\sqrt{\log N \log\log N})}$ and

$$\langle u_i, u_i \rangle = 0$$
$$\langle u_i, u_j \rangle \in [1, 3, 4] \forall i \neq j$$

Note that the construction from Section 5.1 satisfies all of these properties.

The setup of our protocol is as follows: Alice holds D, a length N vector. Bob holds $i \in [N]$, an index into D. Alice and Bob both know μ, and wish to communicate μD_i to Charlie, who knows D and i. Alice and Bob also share randomness r.

Protocol overview: Our protocol will be based around the functions $G, G' : \{0,1\} \to \mathbb{Z}_3$, which will depend on i, D and b, a random vector in $\mathbb{Z}_6^l$:

$$G(t) := \sum_{j\in[N]} D_j (-1)^{\langle tu_i + b, u_j \rangle}$$

$$G'(t) := \sum_{j\in[N]} \langle u_i, u_j \rangle D_j (-1)^{\langle tu_i + b, u_j \rangle}$$

Our protocol will use the fact, which we will prove later, that

$$(2\mu-1)G'(0)-(2\mu-1)G(0)-G(\mu)+G'(\mu)=\mu D_i(-1)^{\langle b,u_i\rangle} \qquad (1)$$

This makes crucial use of the properties of the matching vector family, as well as of the fact that the functions are defined over $\mathbb{Z}_3$. The intuition behind these choice of functions, due to Dvir and Gopi [5], is that $G'(t)$ is the derivative of $G(t)$, thus allowing it to depend on u_i in a similar but linearly independent way from how $G(t)$ depends in u_i.

We will communicate this value to Charlie, and thereby communicate μ to Charlie if and only if $D_i = 1$, as follows:

- Bob sends $m_B^1 := \mu u_i + b$ to Charlie. Charlie can now calculate $G(\mu)$ and $G'(\mu)$.
- Alice can calculate $G(0)$, since it does not depend on i. Alice therefore sends $m_A^1 := (2\mu-1)G(0)-c'$ to Charlie. The c' is so that Charlie only learns the sum of this value and the next value, not them individually.
- $G'(0)$ can be written as a inner product between a secret value that Alice knows, and a value that Bob and Charlie both know. Communicating its value without revealing any other information is a standard 1/2 PSM protocol, the same as was used in Section 6.1.1. For more on PSM protocols, see Section 8.

  Specifically, we can write $G'(0)$ as

  $$\langle u_i, \sum_j u_j D_j(-1)^{\langle b,u_j\rangle}\rangle$$

  Thus, Alice will send $m_A^2 := c + (2\mu-1)\sum_j u_j D_j(-1)^{\langle b,u_j\rangle}$, and Bob will send $m_B^2 := \langle u_i, c\rangle + c'$.

  Charlie can now compute $(2\mu-1)G'(0)-(2\mu-1)G(0) = -m_A^1 + \langle u_i, m_A^2\rangle - m_B^2$.
- Charlie will output 1 if $(2\mu-1)G'(0)-(2\mu-1)G(0)+G'(\mu)-G(\mu)\neq 0$, and 0 otherwise.

We will also give a formal description of the algorithm:

- Public Knowledge: Matching vector family $u_1,\dots u_N \in \mathbb{Z}_6^l$.

- Alice's input: $D \in \{0,1\}^N, \mu \in \{0,1\}$.
- Bob's input: $i \in [n], \mu \in \{0,1\}$.
- Shared randomness: $b \in \mathbb{Z}_6^l, c \in \mathbb{Z}_3^l, c' \in Z_3$
- Alice's messages:
  - $m_A^1 := (2\mu - 1) \sum_j D_j (-1)^{\langle b, u_j \rangle} - c' \in \mathbb{Z}_3$
  - $m_A^2 := c + (2\mu - 1) \sum_j u_j D_j (-1)^{\langle b, u_j \rangle} \in \mathbb{Z}_3^l$
- Bob's messages:
  - $m_B^1 := \mu u_i + b \in \mathbb{Z}_6^l$
  - $m_B^2 := \langle u_i, c \rangle + c' \in \mathbb{Z}_3$
- Charlie's reconstruction: Charlie outputs 1 if

  $$\langle u_i, m_A^2 \rangle - m_A^1 - m_b^2 - \sum_{j \in [N]} D_j (-1)^{\langle m_B^1, u_j \rangle} + \sum_{j \in [N]} \langle u_i, u_j \rangle D_j (-1)^{\langle m_B^1, u_j \rangle} \neq 0$$

  and 0 otherwise.

The communication of the protocol is $O(l) = 2^{O(\sqrt{\log n \log \log n})}$.

Security follows from the following:

- The distributions of m_B^1, m_A^1, m_B^2 are all independently uniformly random, regardless of the input, as b, c', c are respectively used as one-time pads.
- If $D_i = 0$, $(2\mu - 1)G'(0) - (2\mu - 1)G(0) - G(\mu) + G'(\mu) = 0$. Therefore, in this scenario, m_B^2 is uniquely determined by the other messages and D and i, via Charlie's reconstruction function.

Thus, the messages can be simulated without μ when $D_i = 0$, as desired.

G, G' property: Now, it remains to prove the equation 1 always holds.

Lemma 3. *With G, G' defined as above, $\forall \mu \in \{0,1\}$ and $\forall b \in \mathbb{Z}_6^l$, $(2\mu - 1)G'(0) - (2\mu - 1)G(0) + G'(\mu) - G(\mu) = \mu D_i (-1)^{\langle b, u_j \rangle}$*

A given term of either $G(t)$ or $G'(t)$ is determined by three values: $\langle u_i, u_j \rangle$, D_j and $\langle b, u_j \rangle$, holding t constant. Moreover, the dependence of the term on D_j

and on $(-1)^{\langle b,u_j\rangle}$ is always linear. This ensures that $G(0), G'(0), G(1), G'(1)$ are each linear functions of $D_j(-1)^{\langle b,u_j\rangle}$. Moreover, because $G'(t)$ is the derivative of $G(t)$, these functions will also be linearly independent. See Dvir and Gopi [5] for more details.

To prove the validity of the equation, we will focus on the one part of the term that our functions have a nonlinear dependence on, namely $\langle u_i, u_j\rangle$. We will group terms with the same value of $\langle u_i, u_j\rangle$ together. Let us start by defining, for $\sigma \in \{0, 1, 3, 4\}$:

$$S_\sigma := \{j : \langle u_i, u_j\rangle = \sigma\} \subseteq [n]$$

We can then rewrite $G(t), G'(t)$ as

$$G(t) = \sum_\sigma (-1)^{t\sigma} (\sum_{j\in S_\sigma} D_j(-1)^{\langle b,u_j\rangle})$$

$$G'(t) = \sum_\sigma \sigma(-1)^{t\sigma} (\sum_{j\in S_\sigma} D_j(-1)^{\langle b,u_j\rangle})$$

To further simplify things, let us define

$$c_\sigma := \sum_{j\in S_\sigma} D_j(-1)^{\langle b,u_j\rangle}$$

Then, we can write

$$G(t) = c_0 + c_1(-1)^t + c_3(-1)^t) + c_4, G'(t) = c_1(-1)^t + c_4$$

For specific values of t, we have

$$G(0) = c_0 + c_1 + c_3 + c_4$$

$$G'(0) = c_1 + c_4$$

$$G(1) = c_0 - c_1 - c_3 + c_4$$

$$G'(1) = -c_1 + c_4$$

Now, we perform some straightforward algebra:

$$G(0) - G'(0) = c_0 + c_3$$

$$G(1) - G'(1) = c_0 - c_3$$
$$G(\mu) - G'(\mu) = c_0 + (1-2\mu)c_3$$
$$(G(\mu) - G'(\mu)) - (1-2\mu)(G(0) - G'(0)) = 2\mu c_0 = -\mu c_0$$

The equation we needed, equation 1, follows from the fact that $c_0 = D_i(-1)^{\langle b, u_i \rangle}$ □

This completes the construction. □

6.2 Generalized Wildcard

Consider a new CDS problem, which we will call AnyGW_N^k. This is the CDS problem with the predicate $P(D, w)$, where $D \in \{0,1\}^N$ and w is a generalized wildcard with matches of size k, with $(\log N)/k$ subsets. $P(D, w) = 1$ if and only if there exists and index i such that $D_i = 1$ and w matches i.

We will specifically solve $\text{AnyGW}_N^{\log\log N/\log\log\log N}$, which we will refer to a AnyGWMV_N, which refers to Any Generalized Wildcard Matching Vectors CDS.

The construction will be based off of the matching vectors based construction for INDEX_N from Section 6.1.2. However, while the prior CDS constructions have ensured perfect privacy and perfect correctness, this construction will only ensure statistical correctness. In addition, this construction will take place in the Common Reference String (CRS) model, where all parties have access to a public shared random string r. This is addition to Alice and Bob's private shared random string.

The protocol will use $2^{O(\sqrt{\log N \log\log N})} \log(1/\epsilon)$ bits of communication. This has the same asymptotic amount of communication as the previous INDEX_N protocol for any $\epsilon = n^{-O(1)}$.

This construction will make heavy use of the following lemma:

Lemma 4. *If Alice and Charlie have access to a database $D \in \{0,1\}^N$, and Bob and Charlie have access to a wildcard w with length $\log N$ and matches of size $\log\log N/\log\log\log N$, there is a protocol to securely communicate $-\mu \sum_{j \text{ matching } w} D_j(-1)^{\langle b, u_j \rangle} \mod 3$ to Charlie using $2^{O(\sqrt{\log N \log\log N})}$ communication.*

The protocol will mirror the previous matching vectors construction from Section 6.1.2, but with the matching vector family replaced by the generalized wildcard matching vector family from Section 5.2.

Our new functions G, G' will be:

$$G(t) := \sum_{j \in [N]} D_j (-1)^{\langle t v_w + n, u_j \rangle}$$

$$G'(t) := \sum_{j \in [N]} \langle v_w, u_j \rangle D_j (-1)^{\langle t v_w + n, u_j \rangle}$$

These functions are the same as previously, except that u_i is replaced by v_w. v_w is the vector whose construction is described in Section 5.2. v_w has the property that $\langle v_w, u_j \rangle = 0$ if w matches i, and $\langle v_w, u_j \rangle \in \{1, 3, 4\}$ otherwise, where $\{u_j\}$ is the same set as previously.

With this construction, the equivalent of equation 1 is

$$(2\mu - 1)G'(0) - (2\mu - 1)G(0) - G(\mu) + G'(\mu) = -\mu \sum_{j \text{ matching } w} (-1)^{\langle b, u_j \rangle}$$

To see that this equation holds, let us define S and c for $\sigma \in \{0, 1, 3, 4\}$:

$$S_\sigma := \{j : \langle v_w, u_j \rangle = \sigma\} \subseteq [n]$$

$$c_\sigma := \sum_{j \in S_\sigma} D_j (-1)^{\langle b, u_j \rangle}$$

Exactly as before, we can write

$$G(t) = c_0 + c_1(-1)^t + c_3(-1)^t) + c_4, G'(t) = c_1(-1)^t + c_4$$

Which again implies that

$$(2\mu - 1)G'(0) - (2\mu - 1)G(0) - G(\mu) + G'(\mu) = -\mu c_0$$

Which is exactly the claim, once we expand the definition of c_0.

We will communicate the value of the above expression to Charlie in exactly the same fashion as in Section 6.1.2. The same messages will be sent, except that u_i is replaced everywhere by v_w. The protocol securely transmits

$$-\mu \sum_{j \text{ matching } w} D_j (-1)^{\langle b, u_j \rangle} \mod 3$$

to Charlie. □

With that lemma proven, we are ready to prove the main theorem:

Theorem 8. *There is a CDS protocol for AnyGWMV$_N$ with ϵ one-sided error and perfect security, with $2^{O(\sqrt{\log N \log\log N})}\log(1/\epsilon)$ communication, if the parties are given access to a uniformly random Common Reference String.*

Based on the predicate, we are attempting to transmit μ to Charlie if and only if w, a wildcard with matches of length $\log\log N/\log\log\log N$, matches at least one index j such that $D_j = 1$.

Define

$$f(D, w, b) := \sum_{j \text{ matching } w} D_j(-1)^{\langle b, u_j\rangle} \mod 3$$

Simply running the protocol from Lemma 4 directly is not sufficient, as $f(D, w, b)$ may equal zero even if D_j is not uniformly zero over all matching indices. In fact, it might be possible that the above value is zero with large probability over random values of b. It is open whether that is in fact possible.

Let us call a matching index with value 1 in a given database a match.

To distinguish the case where D has no matches from the case where it has at least one match, we will create a database D', which is equal to D, except with some entries equal to 0 where the corresponding entry of D was equal to 1.

If D had no matches, then D' has no matches either, D''s entries with value 1 are a subset of D's entries with value 1. Thus, $f(D', w, b)$ will still be zero.

On the other hand, suppose there is exactly one match in D'. Then we can guarantee that $f(D', w, b)$ is nonzero for any choice of b, since the summation will contain a single nonzero element.

To construct such a D', suppose we knew that D contained k matches of w. Then, we could create $D'_{1/k}$, where each element of D that is a 1 remains a 1 with probability $1/k$. The probability that the k matches in D lead to exactly one match in $D'_{1/k}$ is

$$Pr[\text{Exactly 1 match in } D'_{1/k}] = k\frac{1}{k}\left(1 - \frac{1}{k}\right)^{k-1} = \left(1 - \frac{1}{k}\right)^{k-1}$$

$$= \left(\frac{k-1}{1+(k-1)}\right)^{k-1} = \left(\frac{1+(k-1)}{k-1}\right)^{-(k-1)} = \left(1+\frac{1}{k-1}\right)^{-(k-1)} > e^{-1}$$

Via repeated trials, we can find a $D'_{1/k}$ such that $f(D'_{1/k}, w, b') \neq 0$, thus distinguishing the case where D has k matches from the case where it has 0 matches. Since we will transmit $\mu f(D'_{1/k}, w, b')$ to Charlie via the procedure in Lemma 4, this will allow to learn Charlie when D has k matches, but not when D has 0 matches. To run the procedure, we need to ensure that both Alice and Charlie know $D'_{1/k}$, which is where we will use the CRS.

However, we don't know how many matches D has. More specifically Alice, who doesn't know w, doesn't know how many matches D has. Instead, we perform the above procedure for all k' in $1, 2, 4, \ldots 2^{\lfloor \log N \rfloor}$. Suppose again that there are k matches in D. Then, we will generate a $D'_{1/k'}$ such that $k \leq k' < 2k$. Then, the probability that there is exactly one match in $D'_{1/k'}$ is

$$Pr[\text{Exactly 1 match in } D'_{1/k}] = k\frac{1}{k'}\left(1-\frac{1}{k'}\right)^{k-1} \geq k\frac{1}{2k}\left(1-\frac{1}{k}\right)^{k-1} > \frac{1}{2e}$$

Thus, via repeated trials, we can distinguish the case where D has a nonzero number of matches from the case where D has 0 matches with high probability, solving the problem. In particular, by repeating the procedure $\log_{1-1/2e} \epsilon$ times for each k', we will guarantee an error rate of no more than ϵ.

We now present the procedure in detail, with the protocol from Lemma 4 given in detail as well:

- Public Knowledge: Matching vector family $u_1, \ldots u_N \in \mathbb{Z}_6^l$
- Public Shared Randomness: Let $p := \lceil \log_{1-1/2e} \epsilon \rceil$. Let $q := \lfloor \log_2 N \rfloor$. From the public randomness, sample $r_{st} \in \{0,1\}^N \, \forall s, t, 1 \leq s \leq p, 0 \leq t \leq q$, where $(r_{st})_j$ is 1 with probability 2^{-t}
- Alice's input: $D \in \{0,1\}^N, \mu \in \{0,1\}$
- Bob's input: w, a generalized wildcard which matches vectors of length $\log N$ with matches of length $\log \log N / \log \log \log N$, and $\mu \in \{0,1\}$
- Private Shared Randomness: $b_{st} \in \mathbb{Z}_6^l, c_{st} \in \mathbb{Z}_3^l c'_{st} \in \mathbb{Z}_3 \, \forall 1 \leq s \leq p, 0 \leq t \leq q$.

- Alice's messages: For all s, t such that $1 \leq s \leq p$ and $0 \leq t \leq q$, do the following:
  - Define $D'_{st} \in \{0,1\}^N$ such that $(D'_{st})_j = D_j \wedge (r_{st})_j$. The entry is 1 if and only if both D_j and $(r_s t)_j$ are 1.
  - $m_A^{st1} := (2\mu - 1) \sum_j (D_{st})_j (-1)^{\langle b_{st}, u_j \rangle} - c'_{st} \in \mathbb{Z}_3$
  - $m_A^{st2} := c_{st} + (2\mu - 1) \sum_j u_j (D_{st})_j (-1)^{\langle b, u_j \rangle} \in \mathbb{Z}_3^l$
- Bob's messages: For all s, t:
  - Construct v_w as explained in Section 5.2
  - $m_B^{st1} := \mu v_w + b_{st} \in \mathbb{Z}_6^l$
  - $m_B^{st2} := \mu \langle v_w, c_{st} \rangle + c'_{st} \in \mathbb{Z}_3$
- Charlie's reconstruction: With D'_{st} defined as above, Charlie calculates for every pair s, t the value

$$f(D'_{st}, w, b_{st}) = \langle v_w, m_A^{st2} \rangle - m_A^{st1} - m_B^{st2} - \sum_{j \in [N]} (D_{st})_j (-1)^{\langle m_B^{st1}, u) j \rangle}$$

$$+ \sum_{j \in [N]} \langle v_w, u_j \rangle (D_{st})_j (-1)^{\langle m_B^{st1}, u_j \rangle}$$

  Charlie outputs 1 if $f(D'_{st}, w, b_{st}) \neq 0$ for any s, t.

The communication of the protocol is $O(lpq) = 2^{O(\sqrt{\log N \log\log N})} \log \frac{N}{\epsilon}$. The $\log N$ can be subsumed into the exponent, so this is equal to $2^{O(\sqrt{\log N \log\log N})} \log \frac{1}{\epsilon}$.

The protocol is always correct if $\mu = 0$ and the predicate is true, because the value transmitted is always of the form $-\mu f(D'_{st}, w, b_{st})$, due to the use of the protocol from Lemma 4, which is always 0 if $\mu = 0$.

The protocol is correct with probability at least $1 - \epsilon$ if $\mu = 1$ and the predicate is true, e.g. if there is at least one match of w in D. Since the protocol is run with a probability of keeping each 1 in D of $1/2^t$ for every integer t such that $0 \leq t \leq \log N$, it is run with some probability k' such that $k \leq k' < 2k$, where k is the number of matches of w in D. As described above, this means that for that value of t, Charlie received a nonzero value with probability at least $1/2e$. Thus, over the $\log_{1-1/2e} \epsilon$ executions of the

subprotocol, a nonzero value was not transmitted with probability at most ϵ, as desired.

The protocol is perfectly secure when the predicate is false, e.g. if there are no matches of w in D. D'_{st} has entries which are 1 in a subset of the locations where D's entries are 1, so if D has no matches of w, then D'_{st} definitely has no matches of w, and so Charlie's result for every D'_{st} will be 0. By the security of the inner protocol, the transmitted messages of a single protocol are dependent only on the output of that protocol. The messages for different protocols are independent, because the three one time pads, $b_{st}.c_{st}, c'_{st}$, are independently randomized for each execution of that protocol. Thus, given the fact that every outputted value is 0, no other information is revealed.

We have a perfectly secure, ϵ one-sided error protocol for AnyGWMV$_N$ with $2^{O(\sqrt{\log N \log\log N})} \log \frac{1}{\epsilon}$ communication in the CRS model. $\square$

7 Private Information Retrieval

Private Information Retrieval is a related secure communication task, introduced by Chor, Goldreich, Kushilevitz, and Sudan [4]. We will be examining 1 round, 2 server PIR in particular.

In this scenario, there is a client and two servers, Server 0 and Server 1. The client knows an index $i \in [N]$, and also has access to private randomness. Both servers know a common database $D \in \{0,1\}^N$. The client wants to learn D_i, and also wants to keep i private from each server. Here we assume that the servers do not communicate.

Specifically, C composes queries q_0 and q_1, and sends each to the respective server. The servers respond with answers A_0 and A_1, and from the answers the client reconstructs D_i. There are two conditions we desire from a given construction:

- Correctness: Recon(A_0, A_1)= D_i. The reconstructed result is correct.
- Privacy: q_0 is independent of i. q_1 is also independent of i.

In designing such a protocol, our goal is to minimize the communication

between the client and the servers, for both the queries and the answers.

We will give two existing protocols for the problem: A protocol with $O(N^{1/3})$ communication, from the paper of Chor, Goldreich, Kushilevitz, and Sudan [4] which inaugurated the field, and a recent matching vectors based construction, which will be a specific instance of a general class of such protocols due to Dvir and Gopi. [5] This construction uses $2^{O(\sqrt{\log N \log\log N})}$ communication.

In addition, we will show how each of these protocols can be tweaked or used as a subroutine to solve more difficult generalizations of the private information retrieval tasks than the indexing task presented above. In particular, both can solve generalizations of PIR based on generalized wildcards.

This also opens up wider questions about solving generalized wildcard variants of PIR, which is an interesting area in its own right.

These stronger generalizations of PIR have input sizes larger than that of PIR, making it close to impossible for them to be perfectly efficient at solving PIR, without even considering privacy. This gives an indication of why these protocols are not perfectly efficient, and where to look for protocols that are perfectly efficient.

With this perspective, we can see these protocols as solving a more difficult problem, and then only using a small part of that capability to solve the indexing problem. If we want to solve PIR efficiently, we can't solve a more difficult problem along the way.

The $O(N^{1/3})$ protocol solves a generalized wildcard variant with an input size of $O(N^{1/3})$ communication, showing that it is an efficient solution to its generalized problem. On the other hand, the problem that the matching vectors protocol solves has an input size of about $O(\log^2 N)$, so it is not close to efficient, even for this generalization. This is much more than the trivial $O(\log N)$ bound for basic PIR, however, indicating that this avenue will not lead us to perfectly efficient PIR. More details are given in Section 7.2.

These observations give rise to some open questions:

- Is there a generalization of PIR which the matching vectors protocol solves efficiently?
- Can we find security-based lower bounds for PIR or any of its general-

izations?

- Can we find a PIR construction which doesn't naturally solve any PIR generalizations, even if it's not very efficient?
- Are there PIR constructions for other generalized wildcard problems?

7.1 Existing Protocols

7.1.1 Cube-based $N^{1/3}$ protocol

This protocol is due to Chor, Goldreich, Kushilevitz, and Sudan [4].
Theorem 9. *There is a PIR protocol with $O(N^{1/3})$ communication.*

Let us consider our database D as a three dimensional cube. An index j into the database is mapped to a tuple of three indices, (j_1, j_2, j_3), the indices into each dimension of the cube.

Let S_1^0 be a random subset of the indices of the first dimension, and let S_2^0 and S_3^0 be defined similarly. Let $S_1^1 = S_1^0 \oplus i_1$, where i is the client's desired index, and let S_2^1 and S_3^1 be defined similarly.

A tuple of three subsets, one for each index, can be thought of as a generalized wildcard, or equivalently as a sub-cube. Let us define the xor generalized wildcard function as the exclusive-or over all entries of the database whose indices match this generalized wildcard. Using this idea, we will define XGW based on the subsets defined above:

$$XGW_{\sigma_1\sigma_2\sigma_3} = \bigoplus_{j_1 \in S_1^{\sigma_1}, j_2 \in S_2^{\sigma_2}, j_3 \in S_3^{\sigma_3}} D_{j_1 j_2 j_3}$$

Now, notice that if we take the xor of appropriate XGW values, we will cancel out the differences between pairs of S subsets:

$$XGW_{0\sigma_2\sigma_3} \oplus XGW_{1\sigma_2\sigma_3} = \bigoplus_{j_1 \in S_1^0 \oplus S_1^1, j_2 \in S_2^{\sigma_2}, j_3 \in S_3^{\sigma_3}} D_{j_1 j_2 j_3} = \bigoplus_{j_2 \in S_2^{\sigma_2}, j_3 \in S_3^{\sigma_3}} D_{i_1 j_2 j_3}$$

$$XGW_{00\sigma_3} \oplus XGW_{01\sigma_3} \oplus XGW_{10\sigma_3} \oplus XGW_{11\sigma_3} = \bigoplus_{j_3 \in S_3^{\sigma_3}} D_{i_1 i_2 j_3}$$

$$XGW_{000} \oplus XGW_{001} \oplus XGW_{010} \oplus \ldots \oplus XGW_{111} = D_{i_1 i_2 i_3}$$

Thus, if the client can learn all of the XGW values from the servers, the client can learn the value at the desired index.

Our protocol will therefore be structured as follows:

- The client will generate a random subset S_1^0 of the indices of the first dimension, with each index having an independent $1/2$ probability of being present. S_2^0 and S_3^0 are defined similarly. The client will also create S_1^1, S_2^1, S_3^1 as defined above.
- The client will send (S_1^0, S_2^0, S_3^0) to Server 0 as q_0, and (S_1^1, S_2^1, S_3^1) to Server 1 as q_1. Each query is of size $3N^{1/3}$.
- Server σ will receive the tuple $(S_1^\sigma, S_2^\sigma, S_3^\sigma)$. It will calculate the following answers:
  - $A_0^\sigma := \bigoplus_{j_1 \in S_1^\sigma, j_2 \in S_2^\sigma, j_3 \in S_3^\sigma} D_{j_1 j_2 j_3} = XGW_{\sigma\sigma\sigma}$
  - $A_1^\sigma := \forall k_1 \in [N^{1/3}] \bigoplus_{j_1 \in (S_1^\sigma \oplus k_1), j_2 \in S_2^\sigma, j_3 \in S_3^\sigma} D_{j_1 j_2 j_3}$
  - $A_2^\sigma := \forall k_2 \in [N^{1/3}] \bigoplus_{j_1 \in S_1^\sigma, j_2 \in (S_2^\sigma \oplus k_2), j_3 \in S_3^\sigma} D_{j_1 j_2 j_3}$
  - $A_3^\sigma := \forall k_3 \in [N^{1/3}] \bigoplus_{j_1 \in S_1^\sigma, j_2 \in S_2^\sigma, j_3 \in (S_3^\sigma \oplus k_3)} D_{j_1 j_2 j_3}$

  A_0^σ is a single bit, while $A_1^\sigma, A_2^\sigma, A_3^\sigma$ are $N^{1/3}$-bit vectors. The four answers together constitute the server's response. It is of size $3N^{1/3}+1$.
- With the two answers, the client reconstructs the desired bit by deriving all of the XGW values, and xoring them together.
  - $XGW_{000} = A_0^0, XGW_{111} = A_0^1$
  - $(A_1^0)_{i_1} = \bigoplus_{j_1 \in (S_1^\sigma \oplus i_1) j_2 \in S_2^\sigma, j_3 \in S_3^\sigma} D_{j_1 j_2 j_3} = XGW_{100}$
  - Similarly, $(A_2^0)_{i_2} = XGW_{010}, (A_3^0)_{i_3} = XGW_{001}$.
  - Keeping in mind the self-inverse property of $\oplus$, we know that $(A_1^1)_{i_1} = XGW_{011}, (A_2^1)_{i_2} = XGW_{101}, (A_3^1)_{i_3} = XGW_{110}$.

  Thus, the client can calculate D_i as follows:
- $D_i = A_0^0 \oplus (A_1^0)_{i_1} \oplus (A_2^0)_{i_2} \oplus (A_3^0)_{i_3} \oplus (A_1^1)_{i_1} \oplus (A_2^1)_{i_2} \oplus (A_3^1)_{i_3} \oplus A_0^1$

With this construction, we have perfect correctness as shown above. To show privacy, we must show that the queries q_0 and q_1 are each independent of i.

q_0 is clearly independent of i, since each of S_1^0, S_2^0, S_3^0 are defined without reference to i.

q_1 is also independent of i, due to the fact that each index has an independent $1/2$ probability of being present in the initial subset, which is not changed by the $\oplus$ operation. Thus q_1 has the same distribution as q_0, and is independent of i.

The protocol is correct and private, with $O(N^{1/3})$ communication. $\square$

7.1.2 Matching Vectors-based protocol

This protocol, based on the general construction of matching vectors based PIR protocols due to Dvir and Gopi [5], is very closely related to the CDS protocol presented in Section 6.1.2. It is actually the inspiration for that CDS protocol. We will base the protocol off of the same functions G, G' as used in that construction, but we will restructure our communications to provide a different security guarantee.

Theorem 10. *There is a PIR protocol with* $2^{O(\sqrt{\log N \log\log N})}$ *communication.*

We will use a matching vector family, $u_{j\,j\in[N]}$ with $u_j \in \mathbb{Z}_6^l$, with the property that $\forall j,k \in [N]$ such that $j \neq k, \langle u_j, u_k\rangle = \{1,3,4\}$, and $\forall \langle u_j, u_j\rangle = 0$. As described in Section 5.1, such a matching vector family exists with $l = 2^{O(\sqrt{\log N \log\log N})}$. $G, G' : \{0,1\} \to \mathbb{Z}_3$ are defined as follows, where D is the database, i is the index, and b is a random vector of length l:

$$G(t) := \sum_{j\in[N]} D_j(-1)^{\langle tu_i+b, u_j\rangle}$$

$$G'(t) := \sum_{j\in[N]} \langle u_i, u_j\rangle D_j(-1)^{\langle tu_i+b, u_j\rangle}$$

Based on these functions, our protocol will make use of the following identity, very similar to to equation 1 from Section 6.1.2:

$$G'(0) - G(0) + G'(1) - G(1) = D_i(-1)^{\langle b, u_i\rangle}$$

$G(t)$ could be calculated by a server knowing only D and $tu_i + b$, which is good, because $tu_i + b$ is independent of i. $G'(t)$ cannot be calculated in the

same way, but $\sum_{j\in[N]} u_j D_j(-1)^{\langle tu_i+b,u_j\rangle}$ can be calculated with only D and $tu_i + b$, and the client can do the final step.

With this in mind, here is the formal protocol:

- Let b be a uniformly random vector from $\mathbb{Z}_6^l$
- q_0 is b, and q_1 is $u_i + b$. The client will send these queries to the appropriate servers. Each query is of size $l \log_2 6$.
- Server t receives $tu_i + b$. It will calculate the following answers:
  - $A_0^t := \sum_{j\in[N]} D_j(-1)^{\langle tu_i+b,u_j\rangle}$. This is $G(t)$.
  - $A_1^t := \sum_{j\in[N]} u_j D_j(-1)^{\langle tu_i+b,u_j\rangle}$. With this definition, $\langle u_i, A_1^t\rangle = G'(t)$

  $A_0^t \in \mathbb{Z}_3$, while $A_1^t \in \mathbb{Z}_6^l$. This response is of size $\log_2 3 + l \log_2 6$.
- The client reconstructs D_i as follows: $\langle u_i, A_1^0\rangle - A_0^0 + \langle u_i, A_1^1\rangle - A_0^1 = G'(0) - G(0) + G'(1) - G(1) = D_i(-1)^{\langle b,u_i\rangle}$. Note that the inner products are over $\mathbb{Z}_3$. The result is 0 if and only if $D_i = 0$, allowing the client to learn D_i.

The above description shows why the protocol is perfectly correct. It is also perfectly secure, because the queries are independent of i. $q_0 = b$ is clearly independent of i, while $q_1 = u_i + b$ uses b as a one-time pad, and so is independent of i as well.

The protocol is perfectly correct and secure, with communication complexity $2^{O(\sqrt{\log N \log\log N})}$. □

7.2 Generalized Wildcard Extension

PIR is defined with a database and an index in mind. The objective is to retrieve D_i privately from the pair of servers. This is in contrast to problems such as CDS (Conditional Disclosure of Secrets) when a specified function of the two inputs is communicated, and the INDEX_N problem is only one of the problems that we might analyze.

In order to gain insight into the PIR protocols from Section 7.1, it will prove useful to define generalizations of PIR, and show that those protocols can

solve these generalizations.

Our generalization will work as follows:

- The servers will have $D \in \{0,1\}^N$, as before.
- The client will have x, an arbitrary input, as opposed to the index i in PIR.
- The queries and answers will function as before.
- The client will have a desired reconstruction function $f(x, D)$. The protocol will be correct if $\text{Recon}(A_0, A_1) = f(x, D)$.
- Security is the same as before: the distributions of q_0 and q_1 must each be independent of x.

Clearly, the difficulty of this problem will vary based on the function f. We will present two functions, each of which can be solved by one of the prior constructions.

Recalling the generalized wildcard idea from Section 4, define XorGW_N^k as the PIR problem with the following function:

$$f(w, D) = \bigoplus_{j \text{ matching } w} D_j$$

w is a wildcard with with matches of length k.

We will show that the cube-based $O(N^{1/3})$ PIR construction can, with minor modification, solve the $\text{XorGW}_N^{\log N^{1/3}}$ generalization of PIR in Section 7.2.1.

We will also define AnyGW_N^k as the PIR problem with the following function:

$$f(w, D) = \bigvee_{j \text{ matching } w} D_j$$

w is a wildcard with matches of length k.

We will show that the matching vectors PIR construction can be modified for use as a subroutine to solve the $\text{AnyGW}_N^{\log \log N / \log \log \log N}$ generalization of PIR in Section 7.2.2, which we will refer to as AnyGQMV_N for brevity, albeit

with an approximation factor and an associated overhead, in the Common Reference String model.

The fact that these protocols are powerful enough to solve these generalizations gives insight into why they take certain amounts of communication. In both XorGW_N^k and Any_N^k, the size of the client's input, w is equal to $2^k(\log N)/k$. There are $(\log n)/k$ subsets, each an arbitrary subset of an alphabet of size 2^k.

A strong intuition for such problems is that they cannot be solved with less communication than the client's input. In particular, the intuition is that the queries must be as large as the client's input, unless we massively increase the answer size. While this is not true for all functions f, and it has not been proven for these specific functions, it nonetheless indicates a possible lower bound on the complexity for these functions.

Due to the fact that the normal PIR problem, the indexing task, is a special case of the generalized algorithms which will be presented, and that the privacy condition prevents the queries from being smaller when the input is in the special range of inputs, the algorithms cannot be more efficient for PIR without being unable to solve the generalized problems.

Thus, to develop more efficient PIR protocols, we should try to avoid protocols which can solve such generalizations.

In particular, the client's input size for $\text{XorGW}_N^{\log N^{1/3}}$ is $O(N^{1/3})$, so the protocol is efficient at solving this generalized task.

For AnyGWMV_N, the client's input size is $O(\log 2N)/o(\log N)$. This is not a tight lower bound, but it is stronger than the $O(\log n)$ lower bound which is the best known lower bound for PIR.

7.2.1 Generalized Wildcard Cube-based protocol

Theorem 11. *There is a PIR protocol for* $XorGW_N^{\log N^{1/3}}$ *with* $O(N^{1/3})$ *communication.*

This protocol will be based on the $O(N^{1/3})$ construction from Section 7.1.1.

The client's input will be $w = (w_1, w_2, w_3) \subseteq \{0,1\}^{\log N^{1/3}} \times \{0,1\}^{\log N^{1/3}} \times$

$\{0,1\}^{\log N^{1/3}}$

Recall that the basic identity that Section 7.1.1 used was that if S_1^0, S_2^0, S_3^0 are subsets of the indices of the three dimensions, and if $S_x^1 = S_x^0 \oplus i_x$, and if

$$XGW_{\sigma_1\sigma_2\sigma_3} := \bigoplus_{j_1 \in S_1^{\sigma_1}, j_2 \in S_2^{\sigma_2}, j_3 \in S_3 3^{\sigma}_3}$$

then we can calculate $D_{i_1 i_2 i_3}$, the desired value, as $XGW_{000} \oplus XGW_{001} \oplus \ldots \oplus XGW_{111}$.

In this construction, we'll use a similar construction. To distinguish the protocol from the previous one, let's use S' instead of S, and XGW' rather than XGW.

$S_1'^0, S_2'^0, S_3'^0$ will be defined as before, namely as uniformly random subsets of the indices of the three dimensions. $S_1'^1 := S_1'^0 \oplus w_1$, and $S_2'^1, S_3'^1$ are defined similarly. Note that when we defined S_1^1 previously, $\oplus$ was used to swap whether the single element i_1 was present in the set. Now, we are instead performing an xor between two entire sets, as both $S_1'^0$ and w_1 are subsets of the indices of the first dimension.

XGW' is defined similarly to how XGW was defined:

$$XGW'_{\sigma_1\sigma_2\sigma_3} := \bigoplus_{j_1 \in S_1'^{\sigma_1}, j_2 \in S_2'^{\sigma_2}, j_3 \in S_3'^{\sigma_3}} D_{j_1 j_2 j_3}$$

Now, just as before, we observe that the xor of two XGW instances differing in a single dimension cancels all of the subset elements common to that dimension, leaving only the xor of the two subsets:

$$XGW_{0\sigma^2\sigma^3} \oplus XGW_{1\sigma_2\sigma_3} = \bigoplus_{j_1 \in w_1, \mathrm{J}2 \in S_2'^{\sigma_2}, S_3'^{\sigma_3}} D_{j_1 j_2 j_3}$$

Combining all eight XGW values as before, we find that

$$XGW_{000} \oplus XGW_{001} \oplus \ldots \oplus XGW_{111} = \bigoplus_{j_1 \in w_1, j_2 \in w_2, j_3 \in w_3} D_{j_1 j_2 j_3} = \mathrm{XorGW}^N_{\log N^{1/3}}$$

As previously, if the client can learn all eight XGW values, the client will be able to reconstruct the output.

Our protocol will be structured as follows:

- The client will generate S'^1_0, S'^2_0, S'^3_0 as uniformly random subsets of $\{0,1\}^{(\log N)/3}$. The client will also create $S'^1_1, S'^2_1, S'^3 - 1$ as defined above.
- The client will send (S'^0_1, S'^0_2, S'^0_3) to Server 0 as q_0, and (S'^1_1, S'^1_2, S'^1_3) to Server 1 as q_1. Each query is of size $3N^{1/3}$.
- Server σ will receive the tuple $(S'^\sigma_1, S'^\sigma_2, S'^\sigma_3)$. It will calculate the following answers:
  - $A^\sigma_0 := \bigoplus_{j_1 \in S'^{\sigma_1}_1, j_2 \in S'^{\sigma_2}_2, j_3 \in S'^{\sigma_3}_3} D_{j_1 j_2 j_3} = XGW'_{\sigma\sigma\sigma}$
  - $A^\sigma_1 := \forall k_1 \in [N^{1/3}] \bigoplus_{j_2 \in S'^\sigma_2, j_3 \in S'^\sigma_3} D_{k_1 j_2 j_3}$
  - $A^\sigma_2 := \forall k_2 \in [N^{1/3}] \bigoplus_{j_1 \in S'^\sigma_1, j_3 \in S'^\sigma_3} D_{j_1 k_2 j_3}$
  - $A^\sigma_3 := \forall k_3 \in [N^{1/3}] \bigoplus_{j_1 \in S'^\sigma_1, j_2 \in S'^\sigma_2} D_{j_1 j_2 k_3}$

  These answers carry the same information as the answers in the Section 7.1.1 construction - the difference is only stylistic. In particular, $(A^\sigma_1)_{k_1} \oplus A^\sigma_0$ is equal to the other construction's $(A^\sigma_1)_{k_1}$, and the same is true of A^σ_2 and A^σ_3.

  A^σ_0 is a single bit, while $A^\sigma_1, A^\sigma_2, A^\sigma_3$ are $N^{1/3}$-bit vectors. The four answers together constitute the server's response. It is of size $3N^{1/3}+1$.
- With the two answers, the client reconstructs the desired bit by deriving all of the XGW values as follows:
  - $XGW_{000} = A^0_0, XGW_{111} = A^1_1$
  - $XGW_{100} = \bigoplus_{k_1 \in S'^1_1} (A^0_1)_{k_1}$
  - $XGW_{010} = \bigoplus_{k_2 \in S'^2_1} (A^0_2)_{k_2}$
  - $XGW_{001} = \bigoplus_{k_3 \in S'^3_1} (A^0_3)_{k_3}$
  - $XGW_{011} = \bigoplus_{k_1 \in S'^1_0} (A^1_1)_{k_1}$
  - $XGW_{101} = \bigoplus_{k_2 \in S'^2_0} (A^1_2)_{k_2}$
  - $XGW_{110} = \bigoplus_{k_3 \in S'^3_0} (A^1_3)_{k_3}$

  By xoring all of the XGW values together, the client can find $\mathrm{XorGW}^N_{\log N^{1/3}}$, as desired.

With this construction, we have perfect correctness as shown above. To show privacy, we must show that the queries q_0 and q_1 are each independent of w. q_0 is clearly independent of w, since it is defined without reference to w. q_1 is also independent of w, since the S'^0_x subsets are used as one-time pads for the w_x subsets.

The protocol is correct and private, with $O(N^{1/3})$ communication. $\square$

Notice that the prior protocol form Section 7.1.1 is a special case of this protocol. If w_1, w_2, w_3 are each restricted to be of size one, the same queries, responses, and reconstruction are performed, up to the stylistic difference in the answers. This shows that a more efficient PIR approach requires a structural change to the protocol.

7.2.2 Generalized Wildcard Matching Vectors protocol

We will construct a protocol for AnyGWMV$_N$ PIR with similar communication complexity to the PIR protocol from Section 7.1.2.

This construction is very similar to the Generalized Wildcard Conditional Disclosure of Secrets protocol in Section 6.2, in the same way that the Matching Vectors based PIR protocol was very similar to the corresponding Matching Vectors based CDS protocol.

In particular, this construction will take place in the Common Reference String model, where all parties, and in particular both servers, share access to a large uniformly random string. Also, the construction will exhibit ϵ one-sided error.

We will start with a sub-protocol, which closely mirrors the PIR protocol from Section 7.1.2.

Lemma 5. *If the servers have access to a database $D \in \{0,1\}^N$, there is a PIR protocol for the client to securely query $\sum_{j \; matching \; w} D_j(-1)^{\langle b, u_j \rangle} \mod 3$ with $2^{O(\sqrt{\log N \log\log N})}$ communication.*

The protocol will mirror the PIR protocol from Section 7.1.2, but with the matching vector family extended to a generalized wildcard matching vector family from Section 5.2.

The value that the construction in Section 7.1.2 queried from the server was

equal to

$$c_0 = \sum_{j:\langle u_i, u_j\rangle = 0} D_j(-1)^{\langle n, u_j\rangle}$$

subject to the assumption that $\forall u_j, \langle u_i, u_j\rangle \in \{0, 1, 3, 4\}$. The construction used no other property of u_i. Thus, if we replace u_i in that protocol with v_w from the generalized wildcard matching vector family construction in Section 5.2, the resulting value will be $\sum_{j \text{ matching } w} D_j(-1)^{\langle b, u_j\rangle}$, as desired.

Moreover, the security of that protocol was also not dependent on any properties of v_w, so the new protocol, with u_i replaced by v_w, is also secure. □

With that sub-construction, we are ready to construct the overall protocol:

Theorem 12. *There is a PIR protocol for AnyGWMV$_N$ with ϵ one-sided error and perfect security, with $2^{O(\sqrt{\log N \log\log N})} \log(1/\epsilon)$ communication, if the servers are given access to a uniformly random Common Reference String.*

Call a j such that j matches w and the jth entry of a given database is a 1 a match.

The protocol from Lemma 5 might give us a nonzero output when we run it on the original database D if the are a nonzero number of matches, but it might not. To ensure that it will, we will modify the database by setting some of the entries which had value 1 in D to 0. If we do so, and end up with a database D' such that D' contains exactly one match, then the protocol from Lemma 5 will be guaranteed to produce a nonzero output.

In particular, we will run the protocol from Lemma 5, on databases where each entry which was 1 in D set to 1 with probability 2^{-t}, and set to 0 otherwise. We will do this for all t such that $0 \leq t \leq \log N$. Call such a database $D'_{2^{-t}}$.

Suppose D has k matches of w, with $k \geq 1$. Then this procedure will result in running the sub-protocol with some probability $1/k'$ such that $k \leq k' < 2k$. In such a case, the probability that there is exactly one match of w in $D'_{1/k'}$ is equal to:

$$Pr[\text{Exactly 1 match in } D'_{1/k'}] = k\frac{1}{k'}\left(1 - \frac{1}{k'}\right)^{k-1} \geq k\frac{1}{2k}\left(1 - \frac{1}{k}\right)^{k-1} > \frac{1}{2e}$$

On the other hand, if D has no matches of w, the resulting value from the

sub-procedure will always be zero, because there are no non-zero values in the summation.

To ensure a probability of failure of no more than ϵ, we will run the sub-procedure $\log_{1-1/2e} \epsilon$ times for each value of t.

As an optimization, notice that we can use the share the same queries across all of the sub-protocols. The only variation between subprotocols will be D'.

With that, we are ready to give the construction in detail:

- Public Shared Randomness (CRS): Let $p := \lceil \log_{1-1/2e} \epsilon \rceil$. Let $q := \lfloor \log_2 N \rfloor$. From the public randomness, sample $r_{st} \in \{0,1\}^N \forall s, t, 1 \leq s \leq p, 0 \leq t \leq q$, where $(r_s t)_j$ is 1 with probability 2^{-t}.
- Client's input: w, a generalized wildcard which matches vectors of length $\log N$ with matches of size $\log \log N / \log \log \log N$.
- Servers' input: $D \in \{0,1\}^N$.
- Queries: Let b be a uniformly random vector from $\mathbb{Z}_6^l$. $q_0 := b$. $q_1 := v_w + b$. The client will send these queries to the appropriate servers. Each query is of size $l \log_2 6$.
- Server f receives $f v_w + b$. It will calculate the following answers, for all s, t such that $1 \leq s \leq p, 0 \leq t \leq q$:
  - Define $D'_{st} \in \{0,1\}^N$ such that $(D'_{st})_j = D_j \,\Lambda\, (r_{st})_j$
  - $A^f_{st0} := \sum_{j \in [N]} (D'_{st})_j (-1)^{\langle f v_w + b, u_j \rangle} \mathbb{Z}_3$
  - $A^f_{st1} := \sum_{j \in [N]} v_w (D'_{st})_j (-1)^{\langle f v_w + b, u_j \rangle} \in \mathbb{Z}_6^l$
- The client reconstructs whether there exists an index j such that $D_j = 1$ and j matches w as follows: For every s, t pair, the client calculates:

$$\langle v_w, A^0_{st1} \rangle - A^0_{st0} + \langle v_w, A^1_{st1} \rangle - A^1_{st0} = \sum_{j \text{ matching } w} (D'_{st})_j (-1)^{\langle b, v_w \rangle}$$

  Note that the inner products are taken over $\mathbb{Z}_3$. The client outputs 1 if any of the calculated values is nonzero.

If there are no indices j such that $D_j = 1$ and j matches w, the client always correctly outputs 0. Every D'_{st} equals one in a subset of the entries where D equals 1, so there are no indices j such that $(D_{st})_j = 1$ and j matches w, for any pair s, t. Thus, the sum which the client ends up calculating is equal to 0 in every case, and the client outputs 0.

If there is at least one index j such that $D_j = 1$ and j matches w, the client correctly outputs 1 with probability at least $1 - \epsilon$. Let k be the number of indices j such that $D_j = 1$ and j matches w. There must be a t such that $k \leq 2^t < 2k$. As established above, the probability that, for a given D'_{st}, there is exactly one index j such that $(D'_{st})_j = 1$ and j matches w, is at least $1/2e$. If there is exactly one such index, then the summation which the client calculates is over exactly one non-zero value, and hence is nonzero. The probability that this never occurs is at most $1 - 1/2e$ for each sub-protocol executed with that value of t. $\log_{1-1/2e} \epsilon$ iterations of the sub-protocol are executed with that value of t, giving a probability of no more than ϵ probability of none of the executions resulting in a nonzero value, as desired.

Thus, we have ϵ one-sided error. Perfect security follows from the fact that b is uniformly random, so both q_0 and q_1 are independent of w.

This protocol uses $2^{O(\sqrt{\log N \log\log N})} \log(N/\epsilon) = 2^{O(\sqrt{\log N \log\log N})} \log(1/\epsilon)$, as desired. □

8 Private Simultaneous Messages for General Functions

The Private Simultaneous Messages problem, due to to Feige, Killian and Naor [6], is a secure communication problem again involving three parties. It is similar to the CDS problem, with two input parties, Alice with input x and Bob with input y, who share some prior randomness. They will each send a message to Charlie, who will then output a value. In contrast to CDS, Charlie initially knows nothing about x and y. After receiving the messages, Charlie must be able to output $f(x, y)$ for some prespecified function f. Moreover, Charlie must learn nothing else.

Formal definition:

Let f be a function $X \times Y \to D$. A PSM for f consists of a randomness domain, R, and Alice and Bob's message composition functions, $f_A : X \times R \to M_A$, $f_B : X \times R \to M_B$, and Charlie's reconstruction formula, $f_C : M_A \times M_B \to D$.

For the algorithm to be perfectly secure, we require the following:

- Correctness: $\forall x \in X, y \in Y, r \in R : f(x, y) = f_C(f_A(x, r), f_B(y, r))$
- Privacy: For all pairs of inputs $(x_1, y_1), (x_2, y_2)$ such that $f(x_1, y_1) = f(x_2, y_2)$, the distributions $f_A(x_1, r), f_B(y_1, r)$ and $f_A(x_2, r), f(y_2, r)$ over a random $r \in R$ must be identical.

Alternatively, we can give a simulation based privacy definition: There must exist a simulation function $s : D \times R' \to (M_A, M_B)$ such that for any pair of inputs (x, y), the distribution of $f_A(x, r), f_B(y, r)$ over a random $r \in R$ must be the same as $s(f(x, y), r')$ over a random $r' \in R'$.

The 1/2 PSM problem, also known as PSM with one-sided privacy, can also be of interest. In particular, it can often be used as a subcomponent of CDS protocols, as seen previously.

For the 1/2 PSM problem, we modify the setup so that Charlie knows one of the inputs, but not the other. Charlie must learn nothing he did not already know, as before. We modify the formal definition as follows:

Charlie's reconstruction formula now takes an additional input: $f_C : M_A \times M_B \times Y \to D$. The other functions are defined as before. The requirements are now:

- Correctness: $\forall x \in X, y \in Y, r \in R : f(x, y) = f_C(f_A(x, r), f_B(y, r), y)$.
- Privacy: There exists a simulation function $s : D \times Y \times R' \to (M_A, M_B)$ such that for any pair of inputs (x, y), the distribution of $f_A(x, r), f_B(y, r)$ over a random $r \in R$ must be the same as $s(f(x, y), y, r')$ over a random $r' \in R'$.

While this problem can be interesting, we will focus on PSM for the rest of this section.

One common goal for creating PSM protocols is to create a protocol which can handle an arbitrary function $f : [N] \times [N] \to \{0, 1\}$. This PSM problem

is called the ALL_N problem.

We will present two protocols for ALL_N due to Beimell, Ishai, Kumaresan and Kushilevitz [2] and Liu, Vaikuntanathan and Wee [9] respectively. Both protocols have a communication complexity of $O(N^{1/2})$, which is the best known.

Both protocols also have degree-4 reconstruction. [9] has shown various relationships between the degree of reconstruction and the communication complexity, so this may be relevant.

Both protocols make use of a protocol for inner product, though neither paper gives a particularly clear presentation of such a protocol. Therefore, we give an efficient protocol for inner product, based on a more general protocol by Liu, Vaikuntanathan and Wee [9].

Open questions:

- Can we create a PSM protocol for ALL_N with less than $O(N^{1/2})$ communication?
- Can we create a PSM protocol for ALL_N with $O(N^{1/2})$ communication and less than degree 4 reconstruction?

8.1 Inner Product

In the Inner Product problem, the function in the PSM problem is $f(x, y) = \langle x, y \rangle$, where x and y are over the domain $\mathbb{Z}_q^n$. We will abbreviate this INNER_n.

This construction is based off a general PSM protocol for multilinear polynomials in [9]. They claim the resulting protocol for INNER_n as a corollary, but an explicit construction for INNER_N is easier to understand.

Theorem 13. *There is a PSM protocol for the function $f(x, y) = \langle x, y \rangle$ over $\mathbb{Z}_q$, with $|x| = |y| = n$ with $O(n \log q)$ communication.*

The protocol is as follows:

- Alice's input: $x \in \mathbb{Z}_q^n$
- Bob's input: $y \in \mathbb{Z}_q^n$

- Shared randomness: $b \in \mathbb{Z}_q^n, g \in \mathbb{Z}_q^{n+1}$
- Alice's message: $M_A := (x||(-\langle x, b\rangle)) + g$
- Bob's messages: $M_B^1 := y + b, M_B^2 := \langle g, (y + b)||1\rangle$
- Charlie's reconstruction: $\langle M_A, M_B^1||1\rangle - M_B^2$

This protocol has a total of $2n + 2\log q$ bits of communication.

Correctness:

If we substitute in the definitions of the messages, we get

$$\begin{aligned}
&\langle M_A, M_B^1||1\rangle - M_B^2 \\
&= \langle (x||(-\langle x, b\rangle)) + g, (y + b)||1\rangle - \langle g, (y + b)||1\rangle \\
&= \langle x, y + b\rangle - \langle x, b\rangle + \langle g, (y + b)||1\rangle - \langle g, (y + b)||1\rangle \\
&= \langle x, y\rangle
\end{aligned}$$

Privacy: M_A and M_B^1 are jointly independent of x and y, since g and b respectively are used as one-time pads. Given specific values of M_A and M_B^1, $M_B^2 = \langle M_A, M_B^1||1\rangle - \langle x, y\rangle$. Thus, the messages reveal no information other than $\langle x, y\rangle$, as desired. □

8.2 4-Hypercube based protocol

This construction was originally presented with "3 party decomposable PIR" as an intermediate step in [2]. We will present it without that step, to focus on the PSM protocol itself.

Theorem 14. *There is a PSM protocol for ALL_N with $O(N^{1/2})$ communication.*

This protocol will solve ALL_N directly. Thus, we have an arbitrary function $f : [N] \times [N] \to \{0, 1\}$, and we wish to output $f(x, y)$.

In this construction, we will think of f as a 4-hypercube, with a dimension corresponding to each half of x, which we will call (i_1, i_2) and to each half of y, which we will call (i_3, i_4). Then, the goal of the protocol is to output $f_{i_1 i_2 i_3 i_4}$.

We will use a very similar hypercube-sum technique as in the cube-based PIR protocol from Section 7.1.1.

From the shared randomness, we will generate a uniformly random subset of each of the 4 dimensions, $S_1^0, S_2^0, S_3^0, S_4^0 \subseteq_R [N^{1/2}]$. We will also define $S_k^1 := S_k^0 \oplus \{i_k\}\, \forall k \in \{1,2,3,4\}$.

Define T_{abcd} as follows:

$$T_{abcd} := \bigoplus_{j_1 \in S_1^a, j_2 \in S_2^b, j_3 \in S_3^c, j_4 \in S_4^d} f_{j_1 j_2 j_3 j_4}$$

Consider the sum of all 16 T values, from T_{0000} to T_{1111}. Every index appears an even number of times except for $f_{i_1 i_2 i_3 i_4}$, so that will be the result. We will communicate that sum to Charlie.

Alice and Bob both know $S_1^0, S_2^0, S_3^0, S_4^0$. Alice additionally knows S_1^1, S_2^1, since she knows i_1, i_2. Likewise, Bob knows S_3^1, S_4^1. Thus, Alice can calculate $T_{0000}, T_{1000}, T_{0100}, T_{1100}$. Likewise, Bob can calculate $T_{0000}, T_{0001}, T_{0010}, T_{0011}$

Furthermore, in order to calculate T_{1110}, Alice generate the following $N^{1/2}$ bit vector:

$$\forall k \in [N^{1/2}] \quad \bigoplus_{j_1 \in S_1^1, j_2 \in S_2^1, j_3 \in S_3^0 \oplus \{k\}, j_4 \in S_4^0} f_{j_1 j_2 j_3 j_4}$$

T_{1110} is in that vector, at index i_3. Through a similar construction, Alice can generate $N^{1/2}$-bit vectors containing any T value whose index differs in at most one bit from a T value which Alice knows. Likewise, Bob can generate $N^{1/2}$-bit vectors containing any T value which differs in at most one bit from one which Bob knows.

In each case, the above construction can be thought of as a way to frame a T value as an inner product between a vector known to A and a vector known to B. For instance, in the T_{1110} construction, A's vector would be the one described above, and B's vector would be the vector with a single 1 at index i_3. We will use the fact that we have a PSM protocol for inner product to communicate the product to Charlie.

However, we don't want Charlie to learn the individual T values, just their sum. Therefore, we will concatenate all of those vectors together, so that the resulting inner product is the sum of the T values.

This mechanism covers all T values except for T_{1111}. We will have Charlie calculate that value directly, by sending Charlie $S_1^1, S_2^1, S_3^1, S_4^1$.

The formal construction is as follows:

- Common knowledge: The function f.
- Alice's input: $x = (i_1, i_2)$
- Bob's input: $x = (i_3, i_4)$
- Shared randomness: $S_1^0, S_2^0, S_3^0, S_4^0 \subseteq_R [N^{1/2}]$ and the randomness needed for the PSM protocol for inner product, with vector length $4N^{1/2} + 2$.
- Alice:
  - Let $S_1^1 := S_1^0 \oplus \{i_1\}$. Let $S_2^1 = S_2^0 \oplus \{i_2\}$.
  - Calculate $T_{0000}, T_{1000}, T_{0100}, T_{1100}$, as defined above.
  - Let $V_{11*0} := \forall k \in [N^{1/2}] \bigoplus_{j_1 \in S_1^1, j_2 \in S_2^1, j_3 \in S_3^0 \oplus \{k\}, j_4 \in S_4^0} f_{j_1 j_2 j_3 j_4}$. This is a $N^{1/2}$ bit vector.
  - Define $V_{110*}, V_{01*0}, V_{010*}, V_{10*0}, V_{100*}$ similarly.
  - Define e_{i_1} as the vector of length $[N^{1/2}]$ with a 1 at index i^1 and 0s elsewhere. Define e_{i_2} similarly.
  - Send the message(s) corresponding to one side of the INNER protocol from Section 8.1 where Alice's input is:

    $$(T_{0000} \oplus T_{1000} \oplus T_{0100} \oplus T_{1100}) || 1 || e_{i_1} || e_{i_2} || (V_{11*0} + V_{01*0} + V_{10*0}) || (V_{110*} + V_{010*} + V_{100*})$$

    Perform the INNER protocol over $\mathbb{Z}_2$.
  - Send $M_A := (S_1^1, S_2^1)$ to Charlie.
- Bob:
  - Let $S_3^1 := S_3^0 \oplus \{i_3\}$. Let $S_4^1 = S_4^0 \oplus \{i_4\}$.
  - Calculate $T_{0001}, T_{0010}, T_{0011}$, as defined above.

- Calculate V_{*011}, V_{0*11}, as defined in Alice's section.
- Define e_{i_3}, e_{i_4} as defined in Alice's section.
- Send the message(s) corresponding to the other side of the INNER protocol from Section 8.1 where Bob's input is:

$$1||(T_{0001} \oplus T_{0010} \oplus T_{0011})||V_{*011}||V_{0*11}||e_{i_3}||e_{i_4}$$

- Send $M_B := (S_3^1, S_4^1)$ to Charlie.

- Charlie: Charlie reconstructs the inner product of the two vectors given above, and calculates T_{1111} from $S_1^1, S_2^1, S_3^1, S_4^1$. Charlie outputs the $\oplus$ of the two values.

The protocol uses $12N^{1/2} + 6$ bits of communication. It has 4th degree reconstruction, which is needed for calculating T_{1111}.

Correctness: As established above, $\langle V_{*011}, e_{i_1} \rangle = (V_{*011})_{i_1} = T_{1011}$, and the corresponding fact is true whenever a V value with a $*$ at index k is multiplied by e_{i_k}. Therefore, the result of the inner product is the sum of all 15 T values other than T_{1111}. By the correctness of the INNER protocol, this is communicated to Charlie. T_{1111} is calculated by Charlie, who has all of the necessary information, so the resulting sum is the sum of all 16 T values, which equals $f_{i_1 i_2 i_3 i_4} = f(x, y)$, as desired.

Security: We will give a simulation argument for security. First, observe that the joint distribution of $S_1^1, S_2^1, S_3^1, S_4^1$ is uniformly random over 4 subsets of $[N^{1/2}]$. Thus, the simulator will generate random subsets for each of them, S_1', S_2', S_3', S_4'. Then, the simulator will calculate

$$p = \bigoplus_{j_1 \in S_1', j_2 \in S_2', j_3 \in S_3', j_3 \in S_4'} f_{j_1 j_2 j_3 j_4}$$

We know that $f(x, y) \oplus p$ equals the result of the inner product. Thus, we use the simulator from INNER to compose the distribution of messages for the INNER communication. This completes the simulation. □

8.3 Degree 4 polynomial protocol

In this construction, from [9], we will start by constructing a PSM protocol for 4th degree functions. We will use that to construct a PSM protocol for

ALL_N.

Theorem 15. *There is a PSM protocol for any degree 4 multilinear polynomial $p(x_1, x_2, y_1, y_2)$ over $\mathbb{Z}_q$, where (x_1, x_2) is Alice's input, and (y_1, y_2) is Bob's input. This protocol has communication complexity equal to $O(n \log q)$, where $|x_1| = |x_2| = |y_1| = |y_2| = n$.*

Let p be a public degree 4 multilinear polynomial. We will represent this as a vector in $\mathbb{F}_q^{n^4}$. Alice's input is $x_1, x_2 \in \mathbb{F}_q^n$. Bob's input is $y_1, y_2 \in \mathbb{F}_q^n$. Charlie learns $\langle p, x_1 \otimes x_2 \otimes y_1 \otimes y_2 \rangle$.

Protocol overview: Alice sends $x_1 + b_1$, $x_2 + b_2$. Bob sends $y_1 + c_1$, $y_2 + c_2$. Charlie can then compute:

$$\langle p, (x_1 + b_1) \otimes (x_2 + b_2) \otimes (y_1 + c_1) \otimes (y_2 + c_2) \rangle$$

$$= \langle p, (x_1 + b_1) \otimes (x_2 + b_2) \otimes (y_1 \otimes c_2 + y_2 \otimes c_1 + c_1 \otimes c_2) \rangle$$
$$+ \langle p, (x_1 \otimes b_2 + x_2 \otimes b_1 + b_1 \otimes b_2) \otimes y_1 \otimes y_2 i \rangle$$
$$+ \langle p, x_1 \otimes x_2 \otimes y_1 \otimes y_2 \rangle$$

Note that the first term of the summation is linear in y_1 and y_2 - there is no $y_1 \otimes y_2$ term in it. Therefore, there exists a function f such that

$$\langle p, (x_1 + b_1) \otimes (x_2 + b_2) \otimes (y_1 \otimes c_2 + y_2 \otimes c_1 + c_1 \otimes c_2) \rangle =$$

$$\langle f(p, x_1, x_2, b_1, b_2, c_1, c_2), y_1 || y_2 || 1 \rangle$$

Similarly, the second term is linear in x_1 and x_2. Thus, there exists a function g such that

$$\langle p, (x_1 \otimes b_2 + x_2 \otimes b_1 + b_1 \otimes b_2) \otimes y_1 \otimes y_2 \rangle =$$

$$\langle g(p, y_1, y_2, b_1, b_2, c_1, c_2), x_1 || x_2 || 1 \rangle$$

Based on the protocol for PSM for inner product from Section 8.1, we can now construct a PSM protocol for POLY_4^q as follows:

- Public knowledge: $p \in \mathbb{F}_q^{n^4}$
- Alice's input: $x = (x_1, x_2) \in \mathbb{F}_q^n \times \mathbb{F}_q^n$
- Bob's input: $y = (y_1, y_2) \in \mathbb{F}_q^n \times \mathbb{F}_q^n$
- Shared randomness: $b_1, b_2, c_1, c_2 \in \mathbb{F}_q^n$ and the randomness for the PSM protocol for inner product.

- Alice's messages:
  - $M_A^1 := x_1 + b_1$
  - $M_A^2 := x_2 + b_2$
  - Alice's message(s) for the inner product protocol where Alice's vector is $f(p, x_1, x_2, b_1, b_2, c_1, c_2)||x_1||x_2||1$
- Bob's messages:
  - $M_B^1 := y_1 + c_1$
  - $M_B^2 := y_2 + c_2$
  - Bob's message(s) for the inner product protocol where Bob's vector is $y_1||y_2||1||g(p, y_1, y)2, b_1, b_2, c_1, c_2)$
- Charlie's reconstruction: Let the result of the inner product protocol be a. Charlie outputs $\langle p, M_A^1 \otimes M_A^2 \otimes M_B^1 \otimes M_B^2 \rangle - a$.

This protocol uses $8n \log q + 6$ bits of communication.

Correctness: Correctness follows from the fact proved about $\langle p, (x_1 + b_1) \otimes (x_2 + b_2) \otimes (y_1 + c_1) \otimes (y_2 + c_2) \rangle$ above, and the correctness of the inner product protocol.

Privacy: The messages $M_A^1, M_A^2, M_B^1, M_B^2$ are jointly independent of the input, because each involves a one-time pad. The result of the inner product, a can be reconstructed from $\langle p, M_A^1 \otimes M_A^2 \otimes M_B^1 \otimes M_B^2 \rangle$ and the output, so by the privacy of the inner product protocol, this protocol is secure as well. □

Now, we have a PSM protocol with $O(n \log q)$ communication for multilinear polynomials of degree 4. We will use this to solve the general PSM problem, ALL_N.

Theorem 16. *There is a PSM protocol for ALL_N with communication complexity $O(N^{1/2})$*

We will make use of the preceding protocol. We will encode the arbitrary function f into the polynomial p over $\mathbb{Z}_2$. p will be the truth table of f, such that $\langle p, e_x \otimes e_y \rangle = f(x, y)$. Given the natural mapping from $x \in [N]$ to $(x', x'') \in [N^{1/2}] \times [N^{1/2}]$, we will have $e_x = e_{x'} \otimes e_{x''}$, and similarly for e_y.

Thus, we will set $x_1 := e_{x'}, x_2 := e_{x''}, y_1 := e_{y'}, y_2 := e_{y''}$. With these settings of the inputs to MPOLY_4 PSM protocol, the MPOLY_4 protocol will communicate exactly $f(x, y)$. Since $|x_1| = |x_2| = |y_1| = |y_2| = n$, this reduction gives a $O(N^{1/2})$ protocol for ALL_N. □

9 Conclusion

In this thesis, we presented existing matching vectors constructions, and existing protocols for Conditional Disclosure of Secrets, Private Information Retrieval, and Private Simultaneous Messages. We gave new extensions under the framework of generalized wildcards to matching vectors, and consequently to protocols for CDS and PIR. And we gave intuitions behind lower bounds for a variety of protocols, and raised open questions for the future.

References

[1] David A Mix Barrington, Richard Beigel, and Steven Rudich. Representing boolean functions as polynomials modulo composite numbers. *Computational Complexity*, 4(4):367–382, 1994.

[2] Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In *Theory of Cryptography Conference*, pages 317–342. Springer, 2014.

[3] Abhishek Bhowmick, Zeev Dvir, and Shachar Lovett. New bounds for matching vector families. *SIAM Journal on Computing*, 43(5):1654–1683, 2014.

[4] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 41–50. IEEE, 1995.

[5] Zeev Dvir and Sivakanth Gopi. 2-server pir with subpolynomial communication. *Journal of the ACM (JACM)*, 63(4):39, 2016.

[6] Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 554–563. ACM, 1994.

[7] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 151–160. ACM, 1998.

[8] Vince Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. *Combinatorica*, 20(1):71–86, 2000.

[9] Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. New protocols for conditional disclosure of secrets (and more). 2017.