Private Computation Using a PEZ Dispenser

József Balogh, János A. Csirik, Yuval Ishai, Eyal Kushilevitz

November 6, 2001

Abstract

We show how two or more *deterministic* players can privately compute a function of their inputs using a (big) PEZ dispenser.

Key words: secure multiparty computation, recreational cryptography, combinatorial constructions, deterministic protocols.

1 Introduction

The story. Consider a group of kids who wish to vote between two candidates without revealing anything except the final outcome. To their aid is a PEZ dispenser, which may be pre-loaded with some publicly known sequence of red and yellow candies.[1] The kids take turns, where in each turn one of the kids, based on her vote, decides how many candies to pop out of the dispenser. (No other kid can see the number or the colors of these candies; neither can a kid weigh the dispenser and deduce the number of remaining candies.) The pre-loaded color sequence and the kids' instructions must be arranged so that: (1) at the end of the above process, the color of the candy at the top of the dispenser corresponds to the correct majority vote; (2) no kid can tell *anything* about the remaining kids' votes from the colors of the candies she took out of the dispenser. The voting process is completed by having one of the kids pop an additional candy and announce its color. The above two requirements guarantee that only the correct outcome is learned.

The problem. A more general and precise formulation of the problem is the following. There are n *deterministic* players $P_1, \ldots, P_n$, where $n \geq 2$. Each player P_i holds an input $x_i \in \Sigma$, and together they wish to compute some function $f : \Sigma^n \to \Gamma$ of their inputs while hiding their inputs from each other. The players are assumed to be *honest-but-curious*: they follow their instructions, but may try to deduce information about the other players' inputs from what they see.[2] The players are allowed to interact with a very restricted trusted party, which implements the functionality of a PEZ dispenser. This interaction is prescribed by a protocol of the following form. The trusted party is initialized with a fixed string $\alpha \in \Gamma^*$, depending only on f. The protocol then proceeds by a sequence of *moves*. At each move, some predetermined player P_i asks the trusted party to see the next k characters of α, where k depends deterministically on P_i's input x_i. Upon termination, the protocol's output is defined to be

[1]The reader who is not familiar with PEZ dispensers is referred to various online information about them (e.g. `www.pez.com` and `alanb.com/pez`).

[2]This assumption also models an attack by a passive adversary, who has access to all of the information available to the players it corrupts but cannot control their behavior.

the first untouched character of α. (We may assume that one of the players reads this character and announces it to whoever is supposed to learn the output.) A protocol as above is said to be a *private PEZ protocol computing* f if it satisfies the following two requirements:

CORRECTNESS. On any input $x = (x_1, \ldots, x_n) \in \Sigma^n$ the protocol's output is equal to the correct value $f(x)$.

PRIVACY. No player or set of players can learn any information about the other players' inputs from the characters they read (excluding the last character, which contains the protocol's output). Equivalently, the substring of α read by each player P_i in each of its moves is completely determined by x_i, independently of all other inputs.[3]

A simple observation is that if we remove the privacy requirement, then any function f can be correctly computed as follows. For simplicity, assume that $\Sigma = \Gamma = \{0, 1\}$. The initial string α will contain the truth-table of f, where the inputs are ordered by lexicographic order (i.e., the 2^n-bit string $f(000\ldots00), f(000\ldots01), \ldots, f(111\ldots11)$). In move j, $1 \leq j \leq n$, player P_j will ask to read $2^{n-j} \cdot x_j$ bits (i.e., either 2^{n-j} bits or none). At the end of this protocol, the first unread bit of α is the bit corresponding to the input x, whose value is $f(x)$. Unfortunately, this solution completely fails to satisfy the privacy requirement: a player who reads a long sequence of bits from the truth-table of f will typically be able to infer the location of this sequence, thereby learning information about the inputs of other players. This illustrates the apparent difficulty of designing *private* PEZ protocols for non-trivial functions. We refer the reader to Appendix A for a further discussion of the model, its robustness, and some of its natural variants.

Before presenting some examples of private PEZ protocols and describing our results we shall try to motivate the problem and put it in context.

Recreational cryptography. Our problem is best viewed as a "recreational crypto" problem. By this we mean that we are mainly motivated by its sheer aesthetic qualities and are not aware of any real-world application of any sort.[4] Saying that, however, recreational crypto problems can still yield interesting insights and techniques that might be employed in other contexts, as well as a conceptual understanding of the underlying problems. Indeed, several problems of this type were already studied in the Crypto community and attracted a significant amount of interest. Examples include *Visual Cryptography*, introduced by Naor and Shamir [8], and cryptographic protocols based on a deal of cards, studied by Fischer and Wright [4]. For other recreational cryptography work see [7] and references therein. Similarly to the above works, we also suggest a physically realizable implementation of a cryptographic primitive. In contrast, however, our implementation does not require any use of randomness.

Secure multiparty computation. The PEZ model may be viewed as a specific model for secure computation. While our problem is very different in nature from the mainstream line of work in this area, initiated in [9, 5, 2, 3], it is still instructive to put it in this more general context. The

[3]This privacy requirement may seem stronger than needed: it does not allow players to learn information that follows from the value $f(x)$ before the last character is read. The stricter definition is useful for making the model more robust. For instance, it provides maximal privacy even in a case where $f(x)$ should be learned only by some subset of the players or by an external party.

[4]Other than, of course, playing with your kids...

original motivation of our research was a study of the power of trusted parties in secure protocols in general, and in particular in the context of the amount of randomness used by the players. In what follows we explain how the PEZ problem is related to these issues. It is well known that (without a trusted party) randomness is essential for secure computation (see, e.g., [6]). Obviously, when a trusted party is available there is a simple *deterministic* solution for the problem: each player sends its input to the trusted party, who computes the desired value $f(x)$ by itself and announces the result. This solution, however, puts all the computational burden on the trusted party and so the question is raised whether a similar deterministic solution is still possible using a weaker trusted party. The main conclusion of our work is a surprising affirmative answer to an instance of this question: even a severely handicapped, physically realizable, and inherently "leaky" trusted party (as the one described above) allows nontrivial deterministic secure computation.

Two examples. As a first and simple example of a private PEZ protocol, consider the following protocol for the function $\mathrm{AND}_n : \{0,1\}^n \to \{0,1\}$, computing the product of the n input bits. We use the initial string

$$\alpha = \underbrace{0\,0\,\ldots\,0}_{n}\,1.$$

Each player P_i in its turn reads a single bit of α if its input x_i is 1 and otherwise it reads nothing. It is clear that if all players have the input 1 (i.e., $\mathrm{AND}_n(x) = 1$) then the output will be 1, while otherwise the output will be 0. Moreover, each player when reading a bit from the string α always gets a 0 bit (independently of the inputs of other players) and hence privacy is maintained. In general, however, things are not as simple. For most functions it is not easy to find PEZ protocols and it does not even seem clear that such protocols exist. Even for relatively simple functions, constructing private PEZ protocols does not seem to be trivial. As a second example, let MAJ_3 be the majority function on $n = 3$ bits. A private PEZ protocol for this function that uses the following 13-bit string $\alpha = 1001010010101$ and has 7 moves proceeds as follows:[5]

Move	Player	#bits read on input 0	#bits read on input 1	substring read on input 0	substring read on input 1
1	P_1	5	0	10010	-
2	P_2	0	3	-	100
3	P_3	0	2	-	10
4	P_2	0	1	-	1
5	P_3	0	1	-	0
6	P_2	1	0	1	-
7	P_1	0	1	-	0

Figure 1: A private PEZ protocol for MAJ_3 using initial string $\alpha = 1001010010101$

It can be manually verified that for each of the 8 possible inputs in $\{0,1\}^3$, the substring of α that is supposed to be read in each move is consistent with the number of bits read in previous moves, and that the protocol has the correct output on each input.

[5]For each move we specify which player is reading from the PEZ string, how many bits she is reading given each of his two possible inputs (0 or 1) and what substring she is expected to see in each case, where '-' denotes the empty string.

Our results. We begin by studying the class of one-round PEZ protocols, i.e., protocols in which each player makes at most one move (according to some predetermined order). We show that the functions computable by these protocols are exactly those admitting a *decision list* representation. Consequently, even simple functions such as the exclusive-or of two bits cannot be computed by a one-round private PEZ protocol. We then proceed to show our main result: if no restriction on the number of moves is made, then, quite surprisingly, *every* function can be computed by a private PEZ protocol.

The length of the initial string α used by our general construction is doubly exponential in the number of players n, while that of the protocols for decision lists is "only" exponential. Hence, these constructions are feasible only when n is small. It is important to note in this context that even without the privacy requirement, most functions f (even explicit and simple ones) require exponential string length.[6]

Organization. The rest of this paper is organized as follows. In Section 2 we formally define the model and study some of its basic properties. Section 3 studies the power of one-round protocols, and Section 4 includes our general construction. In Section 5 we mention some open problems. Finally, Appendix A discusses some issues related to the robustness of the PEZ model.

2 Preliminaries

In this section we define the private PEZ model and prove two basic closure properties which will be useful in the sequel. Some natural variants of the model are discussed in Appendix A.

2.1 Definitions

We start by formally defining the PEZ model and some notation related to PEZ protocols.

Definition 1 *A PEZ protocol Π for a function $f : \Sigma^n \to \Gamma$ is defined by an initial string $\alpha \in \Gamma^*$ and a sequence of m moves $(M_1, \ldots, M_m)$. Each move M_j consists of a pair (i_j, μ_j), where i_j is a player index (between 1 and n) and $\mu_j : \Sigma \to \{0, 1, \ldots, |\alpha| - 1\}$ specifies the number of characters to read, for each possible value $\sigma \in \Sigma$ that input x_{i_j} may have. We will refer to the index sequence $i_1, i_2, \ldots, i_m$ as the move order of Π. We assume that for any $x \in \Sigma^n$, $\sum_{j=1}^{m} \mu_j(x_{i_j}) < |\alpha|$ (so that the PEZ dispenser never becomes empty during the execution of a protocol). A protocol Π as above is said to privately compute f if there exists a mapping $\beta : [m] \times \Sigma \to \Gamma^*$ (specifying the string read in each move on each possible input), such that:*

1. *For any $j \in [m]$ and $\sigma \in \Sigma$, $|\beta(j, \sigma)| = \mu_j(\sigma)$.*
2. *For any $x \in \Sigma^n$,*
$$\beta(1, x_{i_1}) \circ \beta(2, x_{i_2}) \circ \cdots \circ \beta(m, x_{i_m}) \circ f(x) \preceq \alpha \tag{1}$$
where $\circ$ denotes string concatenation and $A \preceq B$ signifies that the string A is a prefix of the string B.

[6]A large class of Boolean functions requiring exponential complexity in the non-private PEZ model can be obtained by communication complexity arguments. For instance, we show a simple decision list for which the exponential complexity of our protocol is provably optimal, even in the non-private variant of the PEZ model. An important exception is the class of *symmetric* functions, which do admit efficient protocols in the non-private model.

We note that Definition 1 is somewhat redundant, in the sense that the strings $\beta(j,\sigma)$ are uniquely defined by α and the moves M_j. Conversely, the function β and the move order $i_1,\ldots,i_m$ alone are sufficient to uniquely define a protocol Π computing f. However, the above form of the definition will be convenient for our purposes.

Complexity. With a PEZ protocol Π we associate two complexity measures: its *string length*, denoted by $\mathsf{length}(\Pi)$ and defined as $|\alpha|$ (the number of characters in α), and the number of moves m, denoted by $\mathsf{moves}(\Pi)$. We say that Π is a *one-round* protocol if each player makes at most one move.

2.2 Two Closure Properties

Towards understanding the power of PEZ computation, it is natural to ask whether a protocol computing a function f can be efficiently converted into protocols for related functions f'. We show two simple transformations that may be applied to arbitrary PEZ protocols, preserving their privacy and complexity. The first transformation allows to manipulate the output of a PEZ protocol.

Lemma 2 (Output substitution lemma) *Let Π be a private PEZ protocol computing $f:\Sigma^n\to\Gamma$, and let $g:\Gamma\to\Gamma'$ be an arbitrary character substitution function. Then, there is a private PEZ protocol Π' computing the function $f'(x)\overset{def}{=}g(f(x))$ with $\mathsf{length}(\Pi')\le\mathsf{length}(\Pi)$ and $\mathsf{moves}(\Pi')\le\mathsf{moves}(\Pi)$.*

Proof: A protocol Π' as required may be obtained from Π by applying the output substitution g to each character of α. It is easy to see that the new protocol correctly computes f' and that the privacy of Π is not violated by the character substitution. $\square$

In the case of Boolean functions ($\Gamma=\{0,1\}$), Lemma 2 implies that computing a function in the PEZ model is as hard as computing its negation. A somewhat more surprising corollary is that the identity function is the hardest to compute in the PEZ model. More precisely:

Corollary 3 *Suppose that Π_I computes the identity function over Σ^n (that is, the function $I:\Sigma^n\to\Sigma^n$ defined by $I(x)=x$). Then, for every output alphabet Γ and every function $f:\Sigma^n\to\Gamma$ there is a PEZ protocol Π_f computing f with $\mathsf{length}(\Pi_f)\le\mathsf{length}(\Pi_I)$ and $\mathsf{moves}(\Pi_f)\le\mathsf{moves}(\Pi_I)$.*[7]

The second transformation, described next, allows to manipulate the inputs to a PEZ protocol in a restricted way.

Lemma 4 (Projection lemma) *Let Π be a private PEZ protocol computing $f:\Sigma^n\to\Gamma$. Let $\phi:(\Sigma')^{n'}\to\Sigma^n$ be such that each of the n outputs of ϕ depends on at most a single input, and let $f':(\Sigma')^{n'}\to\Gamma$ be the function defined by $f'(x')\overset{def}{=}f(\phi(x'))$. Then, there is a PEZ protocol Π' computing f' with $\mathsf{length}(\Pi')\le\mathsf{length}(\Pi)$ and $\mathsf{moves}(\Pi')\le\mathsf{moves}(\Pi)$.*

Proof sketch: The idea is to let each player in Π' simulate the players in Π whose outputs of ϕ are affected by its input. Formally, let $\Pi=(\alpha,M_1,M_2,\ldots,M_m)$. A protocol $\Pi'=(\alpha',M'_1,\ldots,M'_m)$ as required may be defined as follows. The initial string α' is equal to α. For each move $M_j=(i_j,\mu_j)$, let i'_j be the index of the input of ϕ on which the i_j-th output of ϕ depends (or 1 if the i_j-th output is a constant), and let $\phi_{i_j}:\Sigma'\to\Sigma$ be a mapping defining the i_j-th output of ϕ as a function of its i'_j-th input. The corresponding move M'_j is defined by $M'_j=(i'_j,\ell'_j)$, where $\ell'_j(\sigma')\overset{\text{def}}{=}\mu_j(\phi_{i_j}(\sigma'))$ for each $\sigma'\in\Sigma'$. $\square$

[7]Recall that $\mathsf{length}(\Pi_I)$ is measured in terms of the number of characters of the output alphabet, which is Σ^n in this case.

In the case of a binary input alphabet, Lemma 4 allows to restrict input variables, permute them, and replace variables by their negations. Moreover, Lemma 4 implies that, in some sense, it suffices to consider functions and protocols over a binary input alphabet. This is formalized by the following corollary.

Corollary 5 *Let $f : \Sigma^n \to \Gamma$ be a function over an arbitrary alphabet Σ, and let $k = \lceil \log_2 |\Sigma| \rceil$. Let $E : \Sigma \to \{0,1\}^k$ be a bijection, mapping each character in Σ to a distinct binary string. Let $f' : \{0,1\}^{nk} \to \Gamma$ be the translation of f to a binary alphabet, i.e., f' is such that $f'(E(x_1) \circ \cdots \circ E(x_n)) = f(x_1, \ldots, x_n)$ for all $(x_1, \ldots, x_n) \in \Sigma^n$. Then, any protocol Π' computing f' can be converted into a protocol Π computing f such that $\mathsf{length}(\Pi) \le \mathsf{length}(\Pi')$ and $\mathsf{moves}(\Pi) \le \mathsf{moves}(\Pi')$.*

Proof: The protocol Π is obtained by applying Lemma 4 with $\Pi(x_1, \ldots, x_n) \overset{\text{def}}{=} E(x_1) \circ \cdots \circ E(x_n)$ to the protocol Π'. $\square$

3 One-Round Protocols

In this section we attempt to answer the following questions: (1) which functions can be computed by a one-round private PEZ protocol? (2) what is the cost (in terms of string length) required by such a computation?

We start by defining the *decision list* computational model.

Definition 6 *A decision list with input alphabet Σ and output alphabet Γ is defined by a list*

$$L = \langle (i_1, D_1), (i_2, D_2), \ldots, (i_\ell, D_\ell), \mathsf{def} \rangle,$$

where each $i_j \in [n]$ is an index of an input variable, each $D_j : \Sigma \to \Gamma \cup \{\bot\}$ decides, based on x_{i_j}, whether to output a specified value or to continue, and $\mathsf{def} \in \Gamma$ is a default output value. The value $L(x)$ computed by L on input x is the output of the following program:

```
if $(D_1(x_{i_1}) \neq \bot)$ output $D_1(x_{i_1})$; else
    if $(D_2(x_{i_2}) \neq \bot)$ output $D_2(x_{i_2})$; else
            $\vdots$
        if $(D_\ell(x_{i_\ell}) \neq \bot)$ output $D_\ell(x_{i_\ell})$; else
            output $\mathsf{def}$;
```

We refer to ℓ as the length of L. We say that L is read-once if each input variable x_i appears at most once in L.

Note that in the case of a binary input alphabet, any decision list L can be assumed to be read-once (otherwise there exists a shorter decision list L' computing the same function). However, this is not true for general input alphabets.[8]

We now show that any read-once decision list can be computed by a one-round protocol. We start with the following composition lemma.

[8]Corollary 5 reduces the task of computing a general function f to that of computing a function f' over a binary input alphabet. However, it does not guarantee that if f can be computed in one round then so can f'. We will therefore address the general case in this section.

Lemma 7 *Let Π be a private PEZ protocol computing $f : \Sigma^n \to \Gamma$ with an initial string α of length l and with move order $i_1, i_2, \ldots, i_m$ (i.e., player i_1 plays in move 1, i_2 in move 2, etc.). For $i \in [n]$ and $D : \Sigma \to \Gamma \cup \{\perp\}$, define a function $f_{D,i} : \Sigma^n \to \Gamma$ by:*

$$f_{D,i}(x) = \begin{cases} f(x), & D(x_i) = \perp \\ D(x_i), & \text{otherwise} \end{cases}$$

Then, $f_{D,i}$ admits a protocol Π' with string length $|D(\Sigma)| \cdot l$ and player move order $i_1, i_2, \ldots, i_m, i$.

Proof: We may assume that $\perp \in D(\Sigma)$: otherwise $f_{D,i}$ depends only on x_i, in which case the lemma easily follows. Let $D(\Sigma) = \{\perp, \gamma_0, \ldots, \gamma_{k-1}\}$, and let $\gamma = \gamma_0 \circ \gamma_1 \circ \cdots \circ \gamma_{k-1}$. A protocol Π' as required can be constructed as follows. Its initial string α' is defined by

$$\alpha' = \gamma \circ \alpha_1 \circ \gamma \circ \alpha_2 \circ \cdots \circ \gamma \circ \alpha_l.$$

The move structure of Π' is as follows. First, the m moves of Π are executed, where the number of characters read at each move is multiplied by $k + 1$. By the correctness of Π, at the end of these moves the first $k + 1$ unread characters of α are $\gamma \circ f(x)$. In the final move of Π', player P_i will read k characters if $D(x_i) = \perp$ and k' characters, $0 \le k' < k$, if $D(x_i) = \gamma_{k'}$. Thus, the output of Π' is equal to $f(x)$ if $D(x_i) = \perp$ and is equal to $D(x_i)$ otherwise. The privacy of Π' follows from the privacy of Π and from the fact that the additional characters read by P_i in the last move are determined by x_i. □

Theorem 8 *Let $L = \langle (i_1, D_1), (i_2, D_2), \ldots, (i_\ell, D_\ell), \mathsf{def} \rangle$ be a read-once decision list with input alphabet Σ and output alphabet Γ. Then, L can be computed by a one-round private PEZ protocol with move order $i_\ell, i_{\ell-1}, \ldots, i_1$ and string length k^ℓ, where $k = \min(|\Sigma|, |\Gamma| + 1)$.*

Proof: We use induction on ℓ. If $\ell = 0$, then $L = \langle \mathsf{def} \rangle$ computes a constant function, which admits a trivial one-round protocol with string length 1 and no moves. Now, consider a length-ℓ read-once decision list $L = \langle (i_1, D_1), (i_2, D_2), \ldots, (i_\ell, D_\ell), \mathsf{def} \rangle$. By the induction's hypothesis, the sub-list $L' = \langle (i_2, D_2), \ldots, (i_\ell, D_\ell), \mathsf{def} \rangle$ has a one-round protocol $\Pi_{L'}$ with a string α' of length $k^{\ell-1}$ and move order $i_\ell, i_{\ell-1}, \ldots, i_2$. (Note that if L is read-once then so is L'.) A protocol Π_L computing L can now be obtained by applying Lemma 7 to $\Pi = \Pi_{L'}$ with $i = i_1$ and $D = D_1$. Since $|D(\Sigma)| \le k$, the string length of Π_L is at most $k \cdot k^{\ell-1} = k^\ell$. □

To get a complete characterization of the class of functions computable by one-round protocols, we prove the following converse of Theorem 8.

Theorem 9 *Suppose that $f : \Sigma^n \to \Gamma$ can be computed by a one-round private PEZ protocol. Then, f can be computed by a read-once decision list.*

Proof: We prove by induction on m that if f is computed by an m-move one-round protocol Π, then f can also be computed by a read-once decision list L_Π of length m whose variables correspond to the active players in Π. If $m = 0$, then f is a constant function, which can be computed by a length-0 decision list. Now, suppose that f is computed an m-move one-round protocol Π with move order $i_1, \ldots, i_m$ and last move $M_m = (i_m, \mu_m)$. We will use this last move to define the first decision of L_Π.

Let $k = \max(\mu_m(\Sigma))$, and let $\hat{\sigma} \in \Sigma$ be some character on which this maximum is attained (that is, $x_{i_m} = \hat{\sigma}$ maximizes the number of characters read by P_{i_m} on move m). Let f' denote the function f restricted by $x_{i_m} = \hat{\sigma}$. The correctness of Π implies that the same f' is obtained if $\hat{\sigma}$ is replaced by any other $\hat{\sigma}'$ such that $\mu_m(\hat{\sigma}') = k$. Let β be as promised in Definition 1, i.e., $\beta(j, \sigma)$ is the substring of α read on move j when $x_{i_j} = \sigma$, and let $\gamma = \beta(m, \hat{\sigma})$. We define a function $D : \Sigma \to \Gamma \cup \{\perp\}$ as follows:

$$D(\sigma) = \begin{cases} \perp, & \mu(\sigma) = k \\ \gamma_{\mu(\sigma)+1}, & \text{otherwise} \end{cases}$$

We will now argue that for any $x \in \Sigma^n$,

$$f(x) = \begin{cases} f'(x), & D(x_{i_m}) = \perp \\ D(x_{i_m}), & \text{otherwise} \end{cases} \tag{2}$$

Let $\sigma = x_{i_m}$. It follows directly from the definition of f' that if $\mu(\sigma) = k$ then $f(x) = f'(x)$. It remains to show that if $\mu(\sigma) < k$ then $f(x) = \gamma_{\mu(\sigma)+1} (= D(x_{i_m}))$. By Eq. (1) we have:

$$\beta(1, x_{i_1}) \circ \cdots \circ \beta(m-1, x_{i_{m-1}}) \circ \beta(m, \sigma) \circ f(x) \preceq \alpha \tag{3}$$

Letting x' be the input obtained from x by changing position i_m to $\hat{\sigma}$, we also have:

$$\beta(1, x_{i_1}) \circ \cdots \circ \beta(m-1, x_{i_{m-1}}) \circ \gamma \circ f(x') \preceq \alpha \tag{4}$$

From (3),(4), and the fact that $\beta(m, \sigma) < k = |\gamma|$, we may conclude that

$$\beta(m, \sigma) \circ f(x) \preceq \gamma,$$

implying that $f(x) = \gamma_{|\beta(m,\sigma)|+1}$, as required.

Finally, observe that the restriction f' can be privately computed by a protocol Π' with move order $i_1, \ldots, i_{m-1}$. Such a Π' can be obtained from Π by letting $P_{i_{m-1}}$ play the role of P_{i_m} on input $\hat{\sigma}$. Formally, the first $m-2$ moves of Π' are as in Π, and its last move is $(i_{m-1}, \mu_{m-1} + k)$. By the induction's hypothesis, f' admits a read-once decision list $L_{\Pi'}$ over the variables $i_1, \ldots, i_{m-1}$. It follows from Eq. (2) that $L \stackrel{\text{def}}{=} (i_m, D) \circ L_{\Pi'}$ is a read-once decision list for f over the variables $i_1, \ldots, i_m$, as required. $\square$

Theorem 9 proves that privacy comes at a price. Indeed, while any function can be computed in one round in the non-private variant of the PEZ model, for most functions this is not the case in the private model. The simplest example of a function which cannot be computed by a decision list is the exclusive-or of 2 bits. In the next section we will see that 3 moves ("a round and a half") are already sufficient to privately compute this function.

Combining Theorems 8, 9, we get:

Corollary 10 *A function $f : \Sigma^n \to \Gamma$ admits a one-round private PEZ protocol if and only if it can be computed by a read-once decision list.*

We conclude this section with a brief discussion of the *string length* of decision lists in the PEZ model. The upper bound given by Theorem 8 is exponential in the decision list length. While it is possible to do much better for some special classes of decision lists (generalizing the AND_n example from the introduction), we show that this is not the case in general.

Theorem 11 *Let $L = \langle (1, D_1), (2, D_2), \ldots, (\ell, D_\ell), \ell + 1 \rangle$ be a decision list over input alphabet $\Sigma = \{0, 1\}$ and output alphabet $\Gamma = [\ell + 1]$, such that $D_i(0) = \perp$ and $D_i(1) = i$ for $1 \leq i \leq \ell$. Then, even if the privacy requirement is dropped, the string length of every PEZ protocol computing L is at least $2^\ell + 1$.*

Proof: Let Π be a correct (but not necessarily private) PEZ protocol for L, and for $x \in \{0,1\}^n$ let w_x be the total number of bits read on input x. Assume towards a contradiction that $\mathsf{length}(\Pi) \leq 2^\ell$. It follows that $w_x < 2^\ell$ for every x, and so there must exist distinct x', x'' such that $w_{x'} = w_{x''}$. Moreover, we can assume that there is no position i such that $x'_i = x''_i = 1$; otherwise, both x and x' can be changed to 0 in every such position i without violating the equality $w_{x'} = w_{x''}$. But this implies that $L(x') \neq L(x'')$, since the smallest position i such that $x'_i = 1$ must differ from the smallest position i such that $x''_i = 1$. This contradicts the assumption that $w_{x'} = w_{x''}$. $\square$

4 A General Construction

The goal of this section is to construct a private PEZ protocol for *every* function $f : \Sigma^n \to \Gamma$. By Corollary 5, we may restrict our attention to the case of a binary input alphabet. Furthermore, by Corollary 3 it suffices to specifically consider the *identity* function $I : \{0,1\}^n \to \{0,1\}^n$. Our construction may indeed be viewed as applying to this specific function. However, to make its presentation more explicit we will directly apply our construction to a general function $f : \{0,1\}^n \to \Gamma$.

Notation. Throughout this section, we will index the players and the input variables by $0, 1, \ldots, n-1$ (rather than $1, 2, \ldots, n$). For an integer m, the *ith bit of* m is the coefficient of 2^i in the binary expansion of m. We will view an input x as an integer, between 0 and $2^n - 1$, where the input of P_i is the ith bit of (the integer) x.

Before describing our general construction we will need the following two definitions.

Definition 12 *For any non-negative integer j, define the* **level** *of j to be the number of consecutive 1s that appear at the end of the binary expansion of j. This is the unique non-negative integer i such that $j \equiv 2^i - 1 \pmod{2^{i+1}}$.*

Definition 13 *For any non-negative integers x and j we say that x* **allows** *j if j is of level i and the ith bit of x is 1 (in other words, $x \bmod 2^{i+1} \geq 2^i$). That is, the binary expansion of x contains 1 in the position corresponding to the rightmost 0 of the binary expansion of j.*

We are now ready to describe our protocol. Consider an arbitrary function $f : \{0,1\}^n \to \Gamma$. We define a protocol Π as follows. The protocol has $2^n - 1$ moves, which are executed in descending order: $(M_{2^n-2}, M_{2^n-3}, \ldots, M_0)$. Move M_j is played by player P_{i_j}, where i_j is defined as the level of j. The initial string is defined as $\alpha \overset{\text{def}}{=} S(2^n - 1)$, where the function $S(\cdot)$ (induced by f) will be defined below. In move M_j, player P_{i_j} will read nothing if $x_{i_j} = 0$, or the string $S(j)$ if $x_{i_j} = 1$.

Definition 14 *For any non-negative integer x, define $S(x)$ recursively as follows:*

(a) $S(0) = f(0)$*;*

(b) for $x > 0$, $S(x)$ is the concatenation of those $S(j)$ (in decreasing order of indices), where $j < x$ and x allows j, followed by the symbol $f(x)$.

Examples: Here are the first values of $S(\cdot)$:

$$\begin{aligned}
S(0) &= f(0) \\
S(1) &= S(0) \circ f(1) \\
S(2) &= S(1) \circ f(2) \\
S(3) &= S(2) \circ S(1) \circ S(0) \circ f(3) \\
S(4) &= S(3) \circ f(4) \\
S(5) &= S(4) \circ S(3) \circ S(2) \circ S(0) \circ f(5) \\
&\vdots
\end{aligned}$$

In order to show the correctness of our construction we need two additional definitions.

Definition 15 *For any non-negative integers x and c, $B(x,c)$ is the concatenation of those $S(j)$ (in decreasing order of indices) where x allows j and $j < c$.*

Lemma 16 *$B(x,c)$ is the substring of α that will be read after Move M_c under the input x.*

Proof: Consider Move M_j for some $j < c$. This move is played by P_i, where i is the level of j. By Definition 13, x allows j iff the ith bit of x, or P_i's input, is 1. It follows from the description of Π that the substring $S(j)$ will be read at Move M_j iff x allows j. $\square$

Definition 17 *For any non-negative integers x and c, we write $x \subseteq c$ if every bit in the binary expansion of c is no less than the corresponding bit in x. (In other words, the set naturally encoded by x is a subset of the set encoded by c.) We write $x \subset c$ if $x \subseteq c$ and $x \neq c$.*

Observe that $x \subseteq c$ implies $x \leq c$.

Our main theorem is the following:

Theorem 18 *For any non-negative integers x and c, $x \subseteq c$ implies that*

$$B(x,c) \circ f(x) \preceq S(c).$$

Substituting $c = 2^n - 1$, we obtain the following corollary, which establishes the correctness of our protocol.

Corollary 19 *For any input $0 \leq x \leq 2^n - 1$,*

$$B(x, 2^n - 1) \circ f(x) \preceq S(2^n - 1).$$

Since $\alpha = S(2^n - 1)$ and (by Lemma 16) $B(x, 2^n - 1)$ is the prefix of α read on input x, this means that our substrings $S(j)$ are consistent with α and f.

Before we can prove Theorem 18, we need another technical lemma.

Lemma 20 *If $x \subset c$, then there is a k such that $x \leq k < c$, k is allowed by c, and k is not allowed by x.*

Proof: Let i be one of the integers such that the ith bit of c is 1 and the ith bit of x is 0. Then $c \bmod 2^{i+1} \geq 2^i$ and $x \bmod 2^{i+1} < 2^i$, thus there is at least one integer k in the range $x \leq k < c$ such that $k \equiv 2^i - 1 \pmod{2^{i+1}}$. This k is allowed by c, but not by x. $\square$

Proof of Theorem 18: This proof proceeds by induction on $c - x \geq 0$. The statement is true by definition if $c = x$: $S(x) = B(x, x) \circ f(x)$.

Let us now consider $x \subset c$. By Lemma 20, there exist k such that $x \leq k < c$ and k is allowed by c but not by x. Let k_0 be the largest such k.

Since the string $S(c)$ does not occur in the definition of $B(x, c)$, the strings defining $S(c)$ and $B(x, c)$ match up exactly up till $S(k_0)$ occurs in $S(c)$, but not in $B(x, c)$. Thus, in order to prove the theorem, it suffices to show that $B(x, k_0) \circ f(x) \preceq S(k_0)$. Since k_0 is smaller than c, this would follow by induction, if it could be established that $x \subseteq k_0$.

Let i be the level of k_0. Then k_0 is the largest integer of level i that is no greater than c (otherwise, that bigger number would have been selected as k_0). Since $k_0 \bmod 2^{i+1} = 2^i - 1$ and $c \bmod 2^{i+1} \geq 2^i$, the binary digits in k_0 and c that correspond to powers of 2 over 2^i are equal. (Thus they dominate the corresponding bits of x.) The ith bit of x is 0, since x does not allow k_0. Thus there is no condition on the corresponding bit in k_0.[9] Finally, the bits in k_0 that correspond to powers of 2 below 2^i are all 1, thus dominating whatever bits x might have there. $\square$

For a crude analysis of $\mathsf{length}(\Pi)$, note that $|S(j)| = O(j!)$, implying that $\mathsf{length}(\Pi) = O(2^n!) = 2^{2^{\tilde{O}(n)}}$. We conclude this section with the following theorem:

Theorem 21 *Any function $f : \{0,1\}^n \to \Gamma$ admits a private PEZ protocol with $2^n - 1$ moves and string length $2^{2^{\tilde{O}(n)}}$.*

5 Open Questions

Having established the universality of private PEZ computation, it remains to better understand its complexity. Some of the more interesting open questions are:

- What are the minimal string length and number of moves required for computing the *worst* functions? Note that, by Corollary 3, it suffices to consider the identity function. Our general upper bound leaves room for an exponential improvement.

- Do the answers to the above question change if one considers natural special classes of functions, such as (from large to small): (1) Boolean functions, (2) symmetric functions, or (3) threshold functions? Note that for classes (2) and (3) there seems to be no inherent reason precluding protocols with poly(n) string length. Moreover, the specific protocol for MAJ_3 described in Figure 1 provides some positive evidence: its string length is 13, in comparison to 72 in the general construction for $n = 3$.

References

[1] D. Beaver. Precomputing oblivious transfer. In *Proc. of Crypto '95*, pages 97-109.

[9]Which is just as well: that bit is 0 too.

[2] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of 20th STOC*, pages 1–10, 1988.

[3] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. of 20th STOC*, pages 11–19, 1988.

[4] M. J. Fischer and R. N. Wright. Multiparty Secret Key Exchange Using a Random Deal of Cards. In *Proc. of Crypto '91*, LNCS 576, pages 141–155.

[5] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game (extended abstract). In *Proc. of 19th STOC*, pages 218–229, 1987.

[6] E. Kushilevitz, and A. Rosén, A Randomness-Rounds Tradeoff in Private Computation. *SIAM Jour. on Discrete Math.*, Vol. 11(1), pages 61–80, 1998.

[7] M. Naor, Y. Naor and O. Reingold. Applied Kid Cryptography or How to convince your children you are not cheating. *J. of Craptology*, vol. 0(1), April 1999. Available online at http://www.wisdom.weizmann.ac.il/~naor/onpub.html.

[8] M. Naor and A. Shamir. Visual Cryptography. In *Proc. of Eurocrypt '94*, pages 1–12.

[9] A. C. Yao. How to generate and exchange secrets. In *Proc. of 27th FOCS*, pages 162–167, 1986.

A Variants of the PEZ Model

In this section we discuss several natural variants of the definition of the private PEZ model. Our main conclusion is that this model is quite robust: most natural changes to its definition either leave it (essentially) the same, or make it (essentially) uninteresting.

A.1 (Essentially) Equivalent Variants

We start with modifications of the problem which do not really make a difference.

General privacy thresholds. In our definition, as described in the Introduction, we require that no subset of players can learn information about the inputs of other players from the substrings of α which they read. A natural relaxation is a t-*privacy* requirement, which should hold only for subsets of at most t players. However, since our model is deterministic, the weakest requirement of 1-privacy is already sufficient to imply our default requirement of n-privacy.

Adaptivity and additional communication. Our definition requires players to be non-adaptive: their choice of how many characters to read cannot depend on what they have previously seen. However, since by our privacy requirement the bits read by P_i depend only on x_i, the notion of adaptivity becomes void. Similarly, allowing the players to directly communicate with each other over private channels would not have made a difference (unless randomization is allowed). These issues will be later revisited.

Incorporating multi-output functionalities. As far as our definitions are concerned, the execution of a PEZ protocol ends when the first *unread* character is the output of the computation. The issue of how this output is learned by the players (e.g., by having one of the players read it and announce it to the other players) is external to our model. For this reason, we also do not explicitly specify who should learn the final output: whether it should be learned by all players, by some strict subset of the players, or by an external party. An even more general specification of a functionality f to be computed by the protocol can assign an arbitrary function of the input x to each player. We now show how to naturally incorporate an output-learning stage in our model, such that computing an arbitrary multi-output functionality $\vec{f}: \Sigma^n \to \Gamma^n$ in the new model reduces to computing some related single-output functionality f' in the original model. In the new model (which incorporates output learning), a special *output round* is added to the end of the protocol, where each player reads one character. The character read by P_i at the output round should be equal to its output. The privacy requirement remains unchanged: players should learn nothing until the output round (a modification of this requirement is discussed in the next paragraph). To implement $\vec{f}$ in the new model, let f' be a single-output function whose single output is a concatenation of the n outputs of $\vec{f}$ (so that the output alphabet of f' is $\Gamma' = \Gamma^n$. Now, suppose that Π' computes f' in the standard model, with initial string α'. A protocol Π computing $\vec{f}$ in the new model may proceed as follows. The output alphabet of Π is Γ. Its string α will be the natural length-$n|\alpha|$ conversion of α' to the alphabet Γ. The moves of Π will be the same as in Π', except that the number of bits read in each move is multiplied by n. Finally, at the output round, each player in its turn reads a *single* character. The protocol Π clearly computes $\vec{f}$, and its privacy follows from that of Π'.

Allowing players to learn the function value. In our definition we make the strongest privacy requirement possible: no player or set of players is allowed to learn any information, even if such information would follow from the output of the function. This convention has several important advantages, most notably the fact that it allows various manipulations of the protocols (see, e.g., Lemma 2). But it is still natural to ask how the power of the model would change if we allow the players to learn information that follows from their inputs and the value of the function f. (We are assuming here that the value of f should be learned by all players; however, a similar discussion applies to general functionalities.) For this modification of the model, it is not clear that its power remains the same. For instance, computing the function XOR_n (exclusive-or of n bits) in the more liberal model efficiently reduces to computing the function XOR_{n-1} in the original model, which may as well be a significantly easier task. However, the extra power of the more liberal model does not extend much farther than that. Specifically, consider a Boolean function f. We argue that from a protocol for f in the liberal model, one can get a protocol in the standard model for any function f' obtained by restricting a single variable of f. Indeed, let f' denote the function f with x_i restricted to some value c. A standard protocol for f' can be obtained by simulating the liberal model with the value of P_i fixed to c. By the (relaxed) privacy requirement of the protocol for f, player P_i should see at most two possible transcripts: one for all inputs x such that $x_i = c$ and $f(x) = c$, and one for all x such that $x_i = c$ and $f(x) = 1$. If these two transcripts are identical, then the protocol already satisfies the default privacy requirement. Otherwise, the protocol can be truncated just before the first point of difference, resulting (again) in a protocol for f' in the standard model. A similar argument applies even if the model is further relaxed by adding adaptivity and direct communication between players.[10]

[10]In fact, adaptivity alone is as strong: point-to-point communication channels can be simulated via an adaptive access to the PEZ dispenser at a moderate complexity cost.

A physically motivated variant. In the physical PEZ model we assume that it is impossible for a player to see anything but the candies it takes out of the dispenser. In practice, however, additional one or two candies can be seen. This concern also arises when physically implementing PEZ protocols using a deck of cards, where the card at the bottom of the deck is the analogue of the candy at the top of the dispenser. A private PEZ protocol in our setting can be converted into a private PEZ protocol in this more leaky setting by inserting a small number of "dummy" candies, of some default color, between any two original candies.

A.2 "Uninteresting" Variants

We now discuss some modifications of the default PEZ model which make it either useless or essentially equivalent to other standard models.

Randomized PEZ. A seemingly interesting modification of the PEZ model allows the initial string α to be randomized. (This also allows to simulate private sources of randomness.) The adaptive version of this model is very powerful, since it allows to privately distribute arbitrarily correlated random sources to the players. Such resources can be used, for instance, as one-time pads for performing subsequent oblivious transfer protocols [1], which in turn can be used for privately computing any circuit of size s with an initial string of length $O(n^2 s)$. However, this model is not very interesting, in the sense that it *equivalent* to standard secure computation with initialization by a trusted party. Indeed, the "interactive" functionality of the PEZ dispenser can be simulated in the latter model by standard general techniques for secure computation.

Malicious PEZ. Throughout this work, we assume that the players are honest-but-curious. It is not hard to see that there is not much to do against malicious players (if all other aspects of the definition remain the same). For instance, the player who makes the second move can attempt to read all the remainder of α and see how many characters the first player took.

Non-private PEZ. We already mentioned in the introduction the non-private version of the PEZ model and showed how any function can be easily implemented in that model (by implementing its truth-table). Besides being unrelated to cryptography, this model also is not very interesting from a complexity theoretic point of view. Moreover, by allowing this model to be adaptive, it becomes as strong as the general communication complexity model (where the PEZ dispenser can be used to implement communication between parties).