Perfect reductions

March 26, 2018

Abstract

We initiate the study of perfect (rather than merely statistical) reductions among cryptographic primitives. For simplicity, we focus on finite client-server functionalities.[1] As opposed to the computational and statistical worlds, quite little is known for perfect security.

While 1-out-of-2 bit-OT oblivious transfer (and several other functionalities) is known to be complete for client-server functionalities since the seminal work of Killian [**?**], this work only offers a statisticaly secure with abort (efficient) protocol in the OT-hybrid model (requiring no further computational assumptions).[2]

For boolean functions a full characterization exists of 2PC functionalities that do or do not have (fully, with fairness) statistically secure efficient protocols in the OT-hybrid model. In particular, those that do not require super-polynomial in k round complexity []. This implies that functionalities not computble with fairness are also not perfectly computable in the OT-hybrid model.

Among those that are, quite surprisingly [**?**] demonstrate that all client-server functionalities can be efficiently reduced to 1-out-of-2-bit-OT evaluated with statistical full security (no abort) in only *one round*. Here a somewhat stronger oracle that securely evaluates 1-out-of-2-bit-OT^l (in particular, the client submits all inputs and recieves all outputs simultaneously) is assumed.TODO: Can we get rid of this assumption? Should verify later.

This result demonstrates that for inefficient protocols, some functions require much higher round compliexty (in the OT hybrid model) then others (for instance, client-server functionalities vs. symmetric coin tossing).

Motivated by the relative "ease" of client-server functionalities for statisticaly secure 2PC, we make first steps towards understanding perfect reductions for these calss of functions, proving that a large class of client-server functions is perfectly reducible to 1-out-of-2-bit-OT.[3] To the best of our knowledge quite few functionaliteis, such as string-OT and TO (OT with reversed roles) were previously known to be perfectly reducible to 1-out-of-2-bit-OT.

We show that for "most" finite functions of the form $f : X \times Y \to \{0, 1\}$, where server domain size $|Y|$ is larger than client domain size X, a $g(|X|, |Y|) = \Omega(1)$-fraction of functions perfectly reducible to 1-out-of-2 OT. This fraction grows roughly as $1 - exp(|X| - |Y|)$. Furthermore, our reduction is 1-round using an oracle to secure evaluation of 1-out-of-2-bit-OT^l, as in [**?**].

Our work leaves open the question of whether all finite client-server functionalities are reducible to bit-OT. In general, are there any (OT or other) complete functionalities for client server SFE, even in more then 1 round. Another open question is whether 2PC functionalities that do have perfectly secure protocols in the OT hybrid model differ in round complexity, as is the case for statistical protocols.

In addition to the obvious theoretical appeal of the question towards better understanding secure computation, perfect, as opposed to statistical reductions may be useful for designing MPC protocols with high concrete efficiency, achieved to eliminating the dependence on a security parameter.

1 Introduction

We revisit the question of the simplest idealized functionalities required for statistical fully secure (without abort) 2PC. Searching for the simplest possible setting, we focus on computationally unbounded parties. As 1-out-of-2-bit-OT is complete for several useful security notions, in this

[1] That is, protocol complexity is measureed in terms of the security parameter k.

[2] In genral, fairness can not be guaranteed, and only security with abort is achievable [**?**, **?**]. If the protocol is not required to be efficient in the security paramter k, then all 2PC functionalities can be securely evaluated [**?**].

[3] Note that here the protocol is efficient by definition, as its complexity does not depend on the security parameter.

work we focus on 1-out-of-2-bit-OT as a candidate. While the setting of statistical security is quite well understood, as we will soon see, the question of perfect security of 2PC in the OT hybrid model is almost entirely open. The focus of this work is to better understand this setting.

Briefly, in this model, additionally to sending messages among themselves, the parties are allowed to make (any number of) calls to an idelized secure implementation of 1-out-of-2-bit-OT. More precisely, we assume that in every round, one party either sends a message over the channel connecting the parties, or makes a call to an ideal OT functionality 1-out-of-2-bit-OT^l where $l \geq 1$ is some number. The party plays the sender of the OT call.[4]

That is, here we assume a somewhat stronger oracle that securely evaluates 1-out-of-2-bit-OT^l, where l may vary between rounds (in particular, the reciever submits all inputs and recieves all outputs simultaneously - he may not choose some of the inputs based on outputs of other OT calls).TODO: We can most likely get rid of this assumption - it is important to verify.

Let us briefly review some key results on statistically or computationally secure protocols in the OT-hybrid model. In this discussion we consider the more standard setting where protocols should be efficient, unless stated otherwise. In the sequel, when discussing efficient protocols from the literature, we focus on finite functionalities, so the complexity is measured in terms of a security parameter 1^k (where the simulation error is $\epsilon \leq 2^{-k}$).

Even inefficiently, statiscially secure computation is known to be impossible in the plain model even against semi-honest adversaries for some simple functions such as AND and OR []. Here even a simulation error of, say, 1/3 is not achievable. Among the functionalities that are efficiently semi-honestly computable with even perfect security, some are not computable with statistical security against malicious parties in the plain model.

In particular, Cleve [] has demonstrated that even the simple functionality of coin tossing where both parties output the same random bit (equivalently, $(x_1, x_2) \oplus (x_1 \oplus x_2, x_1 \oplus x_2)$), is not efficiently computable with statistical security. The impossibility extends to OT-hybrid model.

As a consequence, the 2PC literature (more generally, MPC without honest majority) has settled on a weaker notion of security with abort where the ideal world adversary A has the extra power allowing it to learn the output $f_A(x, y)$ first, and then decide whether to instruct the TP to send the honest party B its output $f_B(x, y)$ or to send it $\perp$ (abort). General purpose 2PC and MPC protocols that are statistically secure with abort for any 2PC functionality have been developped, such as the seminal work of [?], followed by [?], that only assumes an OT oracle (while [?] relies on additional cryptographic primitives,[?] demonstrates how to implement these and many other cryptographic primitives using OT).

A more recent alternative relaxation of statistical security [?] allows for a polynomial error $poly(k^{-1})$, while requiring the parties are still efficient. In [?], it is proved that all 2PC functionalities are computale with staistical security in the OT hybrid model according to this notion. They refer to this notion as $1/p$-security.

Furthermore, their results can be interpreted as staistically secure protocols in the OT hybrid model in our setting (when efficiency is not required). These generalize previous constructions for coin tossing, that also satisfy this security notion.

Also, quite surprisingly, it turns out that many functionalities can be evaluated with statistical security by efficient protocols [?, ?]. In particular, [] completes the characterization of symmetric boolean functions (where both parties recieve the same output)

However, all known protocols for such functions require round complexity of at least. In [?], a 1-round protocol is presented for client-server functionalities, where only the client learns an output can be implemented in a single round, by making a call to an oracle 1-out-of-2-bit-OT^l (no need to otherwise use the channel). Furthermore, the protocol's computational and communication complexity are efficient in k. So, while all 2PC functionalities can be evaluated with statistical security in the OT hybrid model (possibly inefficiently), there are differences in their round complexity.

For instance, Cleve's impossibility result, and [?]'s result for additional functions, in fact prove that certain functionalities including coin-tossing require round complexity of $\Omega(2^k)$. This puts, say coin tossing much higher in the round complexity heirarchy than all client-server functionalities which have protocols with round complexity of 1.

While the statistical setting is qualitatively understood (for unbounded parties), and even the round complexity of protocols is partially understood, much less is known in the perfect setting.

[4] We may assume wlog. that the same party plays the reciever in all OT calls, as given such an OT oracle, an OT oracle with the roles reversed can be implemented with perfect security [].

The ulitimate goal of this work is to understand which functionalities can be evaluataed with perfect security in the OT hybrid model.

On the positive side, we are aware of only a handful of perfect reductions to 1-out-of-2-bit-OT. One is from string-OT [], and one from 1-out-of-2-bit-TO, which is the same as 1-out-of-2-bit-OT where the roles of the parties are reversed. The first one sends no bits over the point to point channel, and only uses to 1-out-of-2-bit-OT^l oracle, while the latter only requires only a single bit of additional communication over the channel. On the negative side, symmetric functionalities known to not be fairly computable by an efficient protocol (in the OT hybrid model) are not perfectly computable. This holds since superpolynomial in k computational complexity is required to achieve simulation error $< 2^{-k}$, so no protocol can exist with a 0 simulation error. Most 2PC functionalities that are computable with fairness remain unclassified as to perfect security.

1.1 Our results

Motivated by the relative ease of client-server functinalities for statistical security in the OT-hybrid model, we start with client-server functionalities. *Question. Characterize the set of client-server functionalities (not necessarily boolean) which are perfectly secure in the OT-hybrid model.*

We make progress on the above question, and discover a broad class of 2PC functionalities computable with perfect security.

Theorem 1.1 (Informal)**.** *Let $f : X \times Y \to Z$ be a deterministic finite client-server functionality. If the trusth table of f is full dimensional (as defined in []), then f is computable with perfect security in the OT-hybrid model.*

We do not know whether all client-server functionalities are computable with perfect security, and leave it as an interesting open question.

In a nutshell, we start with a variant of [**?**]'s protocol, which is statistically secure with abort. We rely on a special structure of the protcol, where the probability of $\perp$ is indpendent on the input, up to negligible differences.[5]

To make the protocol perfectly secure against malicious clients, we modify the so called "watch-lists" used there (after [**?**]) to include exactly k out of n vritual servers, rather than enlist each server with a certain probability. Then, to make it perfectly secure against malicious servers, we rely on geometric interpretation of security (as previously used in the fairness literature), and come up with a way to replace $\perp$ in the ideal model simulating a corrupted server for functions of *full dimension*, which is a function of the row set of f's truth table (rows are labeled by server inputs $y \in Y$).

More on the relation to fairness. Interestingly, to obtain our results we rely on techniques from convex geometry, which are quire similar in spirit to the analysis of several protocols from the firness literature []. However, it is not clear how to obtain perfect security by tweaking known protocols (designed for the statistical setting) to obtain perfect security.

Already the work of [**?**] puts forward a class of symmetric functions that have protocols which are statistically secure with fairness, for which our protocols do not work. The condition characterizing the functions that can be handled by his protocol is identical to ours.

Theorem 1.2 (Informal [**?**])**.** *Every symmetric function $f(x, y)$ such that the truth table F of $f(x, y)$ is* full-dimensional *(when considering the set of rows of the matrix representing F), can be evaluated with full security.*

This is precisely the condition we impose, where the columns of F are labeled by X, and the rows are labeled by Y. However, if F^T rather than is full dimension, since f is symmetric, reversing the roles of the row player and the column player allows to securely compute $f'(y, x) = f(x, y)$, and thus compute $f(x, y)$.

This is interesting, since client-server functionalities are particularly easy to implement with (statistical) fairness, while symmetric functionalities are less so. However, when it comes to perfect security, the fact that the output is learned only by the client, does not allow us to transpose F, as we rely on the fact that the client is the column player. This provides candidate functions with

[5] In [**?**], they also put forward a protocol which is fully statistically secure, essentially by setting the server's input to some default value instead of outputting $\perp$. For the purpose of making the protocol perfectly seure, we make a different choice for $\perp$.

a symmetric variant which is computable with full statistical security, while their client-server variant may not be: these are f's where F is full-dimensional, while F^T is not.

Another interesting open question is *Does increasing round complexity allows to perfectly evluate more functions (which is helpful for statistical security, using GHKL-style protocols).*

One obstacle in making these protocols perfectly secure is achieving perfect semi-honest correctness.

2 Preliminaries

Convex Geometry. We will need the following geometric notion (eventually, the preresent truth tables of 2PC functions).

Definition 2.1 (Affine dimension [?])**.** *For a set of vectors $V = \{v_1, \ldots, v_t\} \subseteq \mathbb{R}^n$, we define their affine dimension $\mathcal{A}(V)$ as the dimension of the set $\{v_i - v_1\}_{i \geq 2}$.*

For a vector $v \in \mathbb{R}^m$, we let $|v|_1 = \sum_i |v_i|$ denote its ℓ_1 norm, and $|v|_\infty = max_i |v_i|$ denote its ℓ_∞ norm. For a set of vectirs $V = \{v_1, \ldots, v_t\} \subseteq \mathbb{R}^n$, a linear combination $\sum_i \alpha_i v_i$ where $\sum_i \alpha_i = 1$ and $\forall i \alpha_i \geq 0$ is a convex combination of V. The convex hall of V, $\mathrm{CH}(V) = \{u = \sigma_i \alpha_i v_i | u$ is a convex combination of $V\}$.

Algebra. For a matrix $A \in \mathbb{F}^{n \times n}$, where $\mathbb{F}$ is a field, let $|A|$ denote the determinant of A. $A^{i,j}$ denotes the (i,j)'th co factor of A, which is the $(n-1) \times (n-1)$ matrix obtained by removing the i'th row and j'th column of A. It is well known that:

Fact 1. *$A^{-1} = C$ where $|C_{i,j}| = |A_{i,j}|/|A|$ (Cramer's formula).*

For a pair of matrices $M_1 \in \mathbb{F}^{n_1 \times m}, M_2 \in \mathbb{F}^{n_2 \times m}$, we denote by $[M_1 || M_2]$ the concatenation of M_2 below M_1.

Oblivious Transfer (OT). The 1-out-of-2-bit-OT functionality is a client-server functionality, where the server inputs a pair (b_1, b_2) of bits, the cleint outputs an index $i \in \{0,1\}$. The client outputs b_i, while the server outputs $\perp$. A generalization 1-out-of-t-bit-OT of 1-out-of-2-bit-OT lets the client pick one out of t bits $b_1, b_2, \ldots, b_t$ supplied by the server. In a further generalization 1-out-of-t h-string-OT the t bits are replaced by strings $s_1, \ldots, s_t \in \{0,1\}^h$, of which on input i the client learns s_i.

A perfect reduction from 1-out-of-t h-string-OT to 1-out-of-2-bit-OT was put forward in the elegant work of [], which constitutes one of the few examples of perfect reductions to 1-out-of-2-bit-OT known so far.

Theorem 2.1. *[[[] There exists a perfectly secure protocol as in Definition 2.2 of 1-out-of-2 h-string-OT with communication complexity $l \leq (t-1)5h$. We refer to it as the Π_{int} protocol.*

2.1 Our model

We consider secure evaluation of client-server (non interactive, deterministic) functionalities $f : X \times Y \to Z$ for finite domains X, Y, Z, where the client outputs $f(x,y)$ and the server outputs $\perp$ (has no output).[6] We focus on finite functionalities f (that is X, Y, Z are finite, and independent of a security parameters k).

We consider secure evaluation of such f in the stand-alone setting, with perfect security, against a non adaptive malicious adversary corrupting a single party. We work in an "enhanced" version of the OT-hybrid model where in every round a party either sends information over the point-to-point channel, or the parties make a call to a perfectly secure implementation of 1-out-of-2-bit-OT^l for some number l that may depend on the round number (the client plays the reciever's role in all the oracle calls). We assume the order of speaking is fixed, and does not depend on the communication so far, the inputs, or the randomness. [7] For client and server inputs $c \in \{0,1\}^l$, $s = ((s_{1,1}, s_{1,2}), \ldots, (s_{l,1}, s_{l,1}))$ to 1-out-of-2-bit-OT^l, respectively, we let $s[c]$ denote $s_{1,c_1}, \ldots, s_{l,c_l}$.

[6] All the definitions below readily generalize to radomized functionalities f, but we focus on deterministic f for simplicity.

[7] The point-to-point channel is not necessary in 1-round protocols where the client does not send messages to the server, as any messages sent to the client over this channel can be emulated by the OT channel.

Our security notion is the standard simulation-based notion as in [?]. That is, we require statistical security against a malicious server or a malicious client, and perfect correctness if both parties behave honestly. We say a protocol is $\epsilon(k)$-client correct if for any malicious server the simulation error is bounded from above by ϵ. We say that the protocol is $\epsilon(k)$ server-private if the simulation error against every malicious client is bounded by $\epsilon(k)$.

For simplicity, we do not concern ourselves with the efficiency of protocols or the simulator, and allow all parties to be unbounded. The adversaries we allow are also unbounded. In fact, our simulator can be made efficient for efficient (in k) adversaries, but this is a secondary issue, and is deferred to the full version.

If the simulator's output joint with honest party's output corresponding to (every) malicious adversary (server or client respectively) has distance $\leq \epsilon$ from the real world distributions, we say the protocol is statisticall ϵ-secure. The protocol is statistically secure if $\epsilon(k)$ is negligible in k. If ϵ equals 0 for all adversaries, we say the protocol is perfectly secure against a malicious server (client).

1-round protocols In this work we further focus on 1-round protocols, where a single call to 1-out-of-2-bit-OT^l is made by the parties. In this setting, security against malicious senders reduces to requiring that the client's output distribution alone is consistent with some input distribution x^* over X, as the sender has no view and no output. For corrupted clients, we again consider only their view alone, as the sender has no output.

Definition 2.2 (1-round protocols in OT-hybrid model)**.** *A protocol for evaluating $f : X \times Y \to Z$ are tuples $\Pi = (\Pi_Q, \Pi_R, \Pi_D)$ of randomized algorithms, where $\Pi_Q(x) : X \to \{0,1\}^l$ generates client's query c. $\Pi_R(x, c, v) : X \times \{0,1\}^l \times \{0,1\}^l \to Z$ generates client's output based on x, c and OT reply v.*[8] *$\Pi_D(y) : Y \to \{0,1\}^{2l}$ is server's generator of OT inputs. We refer to l as the communication complexity of Π.*

A protocl $\Pi = (\Pi_Q, \Pi_R, \Pi_D)$ as in Definition 2.2 with CC l operates in the 1-out-of-2-bit-OT^l-hybrid model as described above.[9] That is, it is specified by a pair of randomized turing machines $\Pi_C^?, \Pi_S^?$ with oracle access to an idealized functionality 1-out-of-2-bit-OT^l operating as follows

$\Pi_C^?(x; r)$: Let $c = \Pi_Q(x; r)$. Send c as input to the 1-out-of-2-bit-OT^l oracle, and let v denote the oracle's output. Output $\Pi_R(x, c, v; r)$.

$\Pi^?(y; r)$: Let $s = \Pi_D(y; r)$, and send it to the 1-out-of-2-bit-OT^l oracle.

Both algorithms Π_C, Π_S run in parallel, in a single round. We will usually use the notation $\Pi = (\Pi_Q, \Pi_R, \Pi_D)$ to denote protocols, while Π_C, Π_S are implicit.

The above definition defines the syntax of protocols we consider. In the following (this and following section) we discuss the security requirements we pose on our protocols.

We ill usually state our protocols in a hybrid model where the parties have access to some l oracles 1-out-of-t_i h_i-string-OT in parallel, where l and (t_i, h_i) are numbers of our choice. This is potentially useful for optimizing the complexity of real-world, computationally secure protocols that are based on our construction, as OT extension allows to implement 1-out-of-2 h-string-OT using a sublinear (in k) number of calls to 1-out-of-2-bit-OT and some additional sublinear work, assuming a strong variant of PRG []. Also, this is wlog. in our perfectly secure setting and costs only linear time $O(tk)$ to convert into a perfect protocol in the 1-out-of-2-bit-OT-hybrid model. This is by using the 1-round perfect reduction of [?] from 1-out-of-t h-string-OT to 1-out-of-2-bit-OT.

From now on, all out definitions apply to 1-round protocols for client-server functionalities (in the OT hybrid), as we only consider this type of protocols.

Definition 2.3. *Here, we assume that f is defined on $f : X \times Y \to Z$, where $\perp \in Y, Z$ and $f(x, y) = \perp$ iff. $x = \perp$. We say a protocol Π as in Definition 2.2 evaluates f with ϵ-enhanced client correctness if it satisfies the same definition as ϵ-client correctness, with the following additional*

[8] For some, but not all functions f, x is not required as an input to R, as $C_x \cup C_{x'} = \phi$ for all $x \neq x'$. It is not hard to prove that a sufficient condition on f for having $C_x \cup C_{x'} = \phi$ in all secure protocols for f is the existence of a 2×2 rectangle $\{y, y'\} \times \{x, x'\}$ in which 3 of the entries are identical, and the other entry differs from these three.

[9] In particular, it first recieves all inputs and only then returns all the outputs to the client. No rushing such as sending inputs to a certain OT instance after getting outputs from other OT instances is possible.

guarantee. For any (deterministic) server strategy S^*, there exists a distribution Y^* over Y such that for all x,

$$\Delta(\Pi_R(x, c \leftarrow \Pi_Q(x), S^*[c]), f(x, Y^*)) \leq \epsilon p_\perp$$

where $p_\perp = Pr_{Y^*}(\perp)$. [10]

The second notion is a relaxation of perfect client correctness.[11]

Definition 2.4. *Here we assume $\perp \in Y$. In the model with input-dependent client-security, we modify the ideal model to allow the simulator send a single value $y^* \in Y$ to the TP. It also gives the TP a predicate P such that the TP sends the client $f(x, y)$ if $P(x) = 0$, and $f(x, \perp)$ otherwise. We say a protocol Π as in Definition 2.2 evaluates f with perfect input-dependent security, if for any adversary ,there exists a simulator with simulation error of 0.*

2.1.1 Restating security requirements geometrically.

We take a similar approach to that of [?, ?] to representing protocols' security requirements geometrically. More specifically, we represent the client's output distributions vector for a given server's strategy as a vector over $\mathbb{R}^t$ for a suitable t. This vector bundles together client output dsitributions for all client's inputs, with a subset of coordinates corresponding to each client's input. A protocol Π defines a region P_S of achievable distributions corresponding to all possible (possibly malicious) server strategies.

Similarly, for a given client's strategy, we consider the client's vector of *views*, with a subset of coordinates corresponding to each server's input y. Similarly to P_S, Π defines a region P_C of achievable distributions corresponding to all client's strategies.

On the other hand, we consider d istribution vectors achievable in the ideal model, and corresponding regions $\tilde{P}_S, \tilde{P}_C$ of all achievable distributions corresponding to possible server strategies and client's strategies respectively. Jumping ahead, our representation of distributions will be such that the regions $\tilde{P}_S, \tilde{P}_C$ will roughly correspond to convex combinations of row and column vectors of the function's truth table.

To achieve security against a malicious server and client respectively, we require that $P_S \subseteq \tilde{P}_S, P_C \subseteq \tilde{P}_C$. For honest correctness (that is, the protocol always outputs the correct value if both parties behave honesly), we make a separate requirement on distribution vectors corresponding to valid client and server strategies.

Geometric representation of client's output distributions. Fix a protocol $\Pi = (\Pi_Q, \Pi_R, \Pi_D)$ as in Defition 2.2 for evaluating a functionality $f : X \times Y \to Z$. For the sake of defining our output distributions we view the protocol as merely randomized mapping from $X \times Y$ to client outputs Z, and do not require it to securely evaluate f, or even correctly evaluate it if everyone behaves honestly.

Boolean functions. Fix $Z = \{0, 1\}$. For a given server's strategy $\Pi^*_D(x = y; r) = s^*$ for some fixed $s^* \in \{0, 1\}^{2l}$[12], 3we consider the distributions of the client's output at the end of a protocol execution $\Pi^* = (\Pi_Q, \Pi_R, \Pi^*_D)$ (that is, a protocol resulting from Π when the server runs Π^*_D instead of Π_D). We denote this set of distributions by a vector $o \in \mathbb{R}^{|X|}$ indexed by $x \in [|X|]$. Here $o_x = p$ denotes the probability of the client outputting 1 on input x That is,

$$o_x = Pr_r[\Pi_Q(x; r) = c; \Pi_R(x, c, s^*[c]) = 1].$$

We refer to such a vector corresponding to some (possibly invalid) server's strategy s^* as a *geometric row distribution* for Π. We shall also consider geometric row distributions for the ideal model evaluating f, corresponding to server's input distributions $y \in Y$, referring to them as a row distribution for f. We ommit Π, f whenever clear from the context.

Observe that the single number o_x uniquely encodes a distribution over the client's output set $\{0, 1\}$ on input x.[13] Similarly, we consider *geometric column distributions* for Π: for a given

[10] This enhances the ϵ-client correctness requiring only a distance bound of ϵ, rather than $\epsilon p_\perp$.

[11] This is a relaxation as it allows for conditional abort. It also strengthens the definition in the sense that every deterministic server strategy induces a single effective input y, rather than a distribution.

[12] This notion naturally generalizes to randomized strategies, but we do not need this extent of generality here.

[13] The vector o represents $|X|$ separate distributions, one for each client's input x. Nothing is implied about the correlation between client's outputs on different inputs for a given server's strategy s^*.

client's strategy $c^* \in \{0,1\}^l$ for its input to the OT oracle, we consider the corresponding *geometric column* distribution vector $o \in \{0,1\}^{|Y|}$ indexed by $y \in [|Y|]$, where o_y is the probability of the client outputting 1 for server input y. That is:

$$o_y = Pr_r[\Pi_R(x, c^*, \Pi_D(y; r)[c^*]) = 1].$$

General functions. Generalizing for larger $Z = \{0, 1, \ldots, k-1\}$, a (geometric) row distribution $o \in \mathbb{R}^{(k-1-1)|X|}$, has entries labeled by pairs (x, i) where $x \in [|X|], i \in Z \backslash \{0\}$, and $o_{(x,i)}$ denotes the probability of outputting i on input x. Thus, for every x, i we have $\sum_j o_{(x,j)} \leq 1$, and $o_{(x,i)} \geq 0$.[14] As in the case of $|Z| = 2$, this vector fully represents the client's output distribution for each input x. A similar extension can be made for (geometric) row distributions. For a given $x \in X$, let o_x denote the sub-vector $(o_{x,z})_{z \in [k-1]}$.

Truth tables. In the truth table F of f, we index rows by elements of Y and columns by elements of X. For $Z = \{0, 1\}$, the truth table representation we consider is just the standard one: a table where entry (y, x) equals $f(x, y)$. We use F_y to denote the row vector (X_x to denote the column) in F corresponding to y (x). We observe that each row F_y in this case is a geometric row distribution in the ideal model, where the server inputs y. We can interpret $Y_{y,x}$ as $f(x, y) = p$, where p is the probability of outputting 1 (either $p = 0$ or $p = 1$).

Let us generalize this form to larger Z. We represent the truth table in "unary", where for each y, x we have $|Z| - 1$ columns $(x, z)_{z \in [|Z|-1]}$, and we set $F((x, z), y) = 1$ if $f(x, y) = z$, and $F((x, z), y) = 0$ otherwise (if $f(x, y) = 0$, all entires $F((x, z), y))$ will be 0).[15] Again, each row F_y is a valid geometric row distribution in the ideal world (corresponding to a server input of y).

Definition 2.5 ([?]). *We say a function $f : X \times Y \to Z$ is full-dimensional if the affine dimension of the row set of its truth table F is the maximal possible* - $(|Z| - 1)|X|$.

Definition of security. As mentioned above, for cient-server functionalities, the standard requirement of perfect stand-alone security in the 1-out-of-2-bit-OT^l-hybrid model can be restated as three separate requirements, all involving only the client's output distribution. For client-server functionalities and protocols Π as in Definition 2.2 the standard definition of perfect security against malicious parties is "almost" equivalent to the following definition. It would be fully equivalent if we made sure simulation of efficient (in k) adversaries is also efficient. Our protocols in fact satisfy the stronger definition, as we prove in the full version. We ignore efficiency issues altogether in this work for the sake of simplicity.

Definition 2.6 (Perfect security of protocols as in Definition 2.2)**.** *We say that a protocol Π as in Definition 2.2 for evaluating a client-server functionality $f : X \times Y \to Z$ as above is perfectly secure against a single malicious party if it satisfies:*

1. *Client correctness: For every server's strategy $s^* \in \{0,1\}^{2l}$, the corresponding row distribution o^* of f' is in $CH(\{F_y\}_{y \in Y})$, where the F_y's are the rows of the truth table F of f.*

2. *Server privacy: Define a modified protocol Π' where the client's output is replaced by its (partial) view $s[c]$ recieved from the OT oracle. That is, $\Pi' = (\Pi_Q, \Pi'_R, \Pi_D)$, where $\Pi'_R(x, c, v) = v$. This protocol is a mapping from $X \times Y$ to $Z' = \{0,1\}^l$.*

   We say that a column distribution for Π' vector m_x is consistent with f, x, if for all y, y' such that $f(x, y) = f(x, y')$ and $i \in [2^l - 1]$, we have $m_x[(y, i)] = m_x[(y', i)]$. We require that for each client strategy $\Pi^*_Q(x; r) = c^*$ for $c^* \in \{0,1\}^l$, the resulting column distribution in Π' is a convex combination

$$\sum_{x \in X} \alpha_x m_x$$

   where each m_x is consistent for f, x.[16]

3. *Honest correctness: Let $C_x = \text{support}(\Pi_Q(x)), S_y = \text{support}(\Pi_D(y))$. Then for all $c \in C_x, s \in S_y$, $\Pi_R(x, c, s[c]) = f(x, y)$.*

[14] The decision to exclude 0 is merely auesthetic, intended to remain consistent with standard binary truth tables.
[15] This is instead of having a single entry for each (x, y) with values in Z.
[16] Here and elsewhere, we in fact satisfy the stronger standard security notion where a adversary, and corresponding simulator are given an auxiliary string $z(x, y)$. All our security proofs are easily generalized to this .

Next we restate Definition 2.3 in geometrical tems.

Observation 1. *Consider a protocol Π evaluating a function f as in Definition 2.3 with ϵ-enhanced client correctness. Then the following holds. For every server's strategy $s^* \in \{0,1\}^{2l}$, there exists an ideal world row distribution $o^* \in CH(\{F_y\}_{y\in Y})$ such that the real world row distribution o' corresponding to s^* satisfies*

$$\Delta(o', o^*) \leq \epsilon(|Z| - 1)p_\perp$$

Here $p_\perp$ is the coefficient of $Y_\perp$ in o^*.[17]

3 Warmup - $Y = \{0,1\}$

As a warmup, we observe that for all finite Z all client-server functions $f : X \times Y \to Z$ where $Y = \{0,1\}$ have a perfectly secure protocol in our model.

Theorem 3.1. *Let $f : X \times Y \to Z$ be a client server functionality. Then there exists a protocol as in Definition 2.2 evaluating f with perfect security and communicaiton complexity $l = 1$.*

Proof. First observe that wlog. we may assume $Z = \{0,1\}$, as for each input x we only need two labels for $f(x,y)$.[18]

Now, we propose the following protocol. To learn $f(x,y)$, the parties fix a column x_0 of the truth table F that dominates all columns x'. That is for all x' if $f(x_0,y) = f(x_0,y')$ then $f(x',y) = f(x',y')$. For $|Y| = 2$, such a column always exists. Furthermore, if all columns are constant, the client can just output the correct value $f(x,0) = f(x,y)$, without any communication. Thus, lets us assume the column F_{x_0} is non-constant.

In the protocol, the sender just sends the message $f(x_0,y)$ to the client. To implement this via 1-out-of-2-bit-OT, it simply puts $f(x_0,y)$ in both positions. The client considers the OT output v, and outputs the value $f(x,y)$ determined by v.

The protocol clearly satisfies honest correctness. It is also perfectly server private as the client only learns the message $f(x,y)$, which is consistent with an input x by the client.[19] The protocol is also perfectly client-correct, as for any deterministic server strategy (b_0,b_1) the client's output is consistent with $f(x,y^*)$, where y^* is a uniform distribution over the server's inputs (y_0,y_1) corresponding to $f(x_0,y)$ that equals b_0 and b_1 respectively (as F_x is non-constant, such inputs always exist).

4 A perfect reduction for any full-dimensional function

While the first two requirements are readily expressed as a single LP (linear program) [], it is unclear how to incorporate the third requirement into the same program. Thus, we do not write an explicit LP searching for a specification of (Π_Q, Π_D, Π_R). The resulting LP is not trivial to solve even for requirements 1,2 alone in 2.6. Instead, we suggest a concrete protocol, and prove it is secure for a certaon subclass of functions. For every such function, we just verify that each of conditions 1,2,3 in Definition 2.2 separately.

Our starting point is a protocol as in Definition 2.2 (1-round protocol in the $(\text{1-out-of-2-bit-OT})^l$-hybrid model for some l) by [?][Section 3]. We denote this protocol by Π_{IKOPSW}. It is stated for function with NC_0 circuits, but we restate it for general function (families), and bound its CC as a function of $|X|, |Y|, |Z|$ (which are assumed to be finite in our work).

The reason the protocol in [?] is restricted to NC_0 functionalities is for improving concrete efficiency, which is not a concern in our paper. Otherwise, their construction works for all functions, as summarized below.

Theorem 4.1. *Let $f : X \times Y \to Z$ denote any (finite) client-server function. Then for all $\epsilon \geq 0$, Π_{IKOPSW} evaluates f with ϵ-relaxed client correctness. Let h_i denote the smallest formula for evaluating the i'th bit of $f(x,y)$, and let $h = max_{i\leq \log(|Z|)} h_i$. Then, Π has communication complexity of $l = \log(|Z|)\tilde{O}(\log(|X|)(\log(\epsilon^{-1}) + h)^2) + polylog(\epsilon^{-1})$.*

[17] This is not exactly equivalent to enhanced ϵ-client correctness, as the parameters here are slightly worse, but this is all we need, and it simplifies the definition.

[18] In general, we may assume $|Z| \leq |Y|$.

[19] As it often the case, the protocol is not private against semi honest clients, as the client always learns $f(x,y)$, so the protocol is only maliciously server-private, but not semi-honestly server-private.

In a nutshell, in the Π_{IKOPSW11} protocol, reduces the evaluation of $f(x, y)$ to secure evaluation of a single instance of l parallel instances of OT 1-out-of-2 h-string-OT functionality for certain parameters h, l depending on f, ϵ. This is done using a decomposable randomized encodings of f.[?]. Here we have $l = \log |X|$ - the input to i'th OT isntance is the bit x_i. The y is mapped to a string seqeunce $(s_{1,0}, s_{1,1}, \ldots, s_{l,0}, s_{l,1})$ fed to the OT's in a certain manner.

Now, to securely evaluate the resulting functionality in the malicious setting, the OT is replaced by a COT (certified OT) functionality. The COT returns the client the output, and additionally reports whether the resulting string is consistent with some y and randomness r, as perscribed by reduction from $f(x, y)$ to the OT instances. Π_{COT} verifyies that the OT input $s \in \{0, 1\}^{2l}$ it feeds to the OT oracle is consistent with some y, r for the reduction. Along with s, the server feeds a repressentation of y, r as a witness that s is consistent. It gives client C_i the OT output along with a bit on whether it is consistent (with that witness y, r common for all indices) or not.

In the COT implementation, the server runs MPC in the head of an n'-party protocol Π for evaluating the (randomized) mapping from a pair (y, r) to a sequence s. More concretely, the protocol consists of a sender that privately sends n virtual servers its inputs y, r, and some $2 \log h$ clients that recieve corresponding OT outputs (the clients do not have inputs). There are also $2l$ clients have no inputs, but client $C_{i,b}$ gets the output of the i'th OT instance on input $x_i = b$. In Π, the virtual servers send messages to the clients in the last round of the protocol.

The COT client chooses the strings corresponding to the views seen by the virtual clients corresponding to its input x. Additionally, it asks for the entire view of the each of the virtual servers with some small constant probability δ'. We observe that these so called *watchlists* (the client watches only a small fraction of the virtual server), along with the requests for client's views can be perfectly reduced in oue round to 1-out-of-2-bit-OT oracles.

This MPC protocol Π is secure against malicious corruption of δn of the n parties. With high probability, the client will see $p/2n < \delta' n < \delta n$ of the views, catching cheating with high probability, but not learning anyting besides $f(x, y)$. For a choice of δ' as above, the IKOPS protocol already happens to be statistically private against malicious clients.

There is however (small) non-zero probability of the client learning too much (in the worst case, $f(x, y)$ for all $x \in X$), by ending up watching too many of the virtual parties' view (potentially all of them).

Thus, Π_{IKOPS} is not perfectly server-private. In the following section, we slightly tweak Π_{IKOPS}, making watchlists deterministic, thereby making it perfectly server-private. In Section [?], using geometric techniques, we demonstrate how to make the protocol from Section 4.1 perfectly client correct. Here we critically rely on the fact that Π_{IKOPS} satisfies statistical enhanced client privacy, rather than just standard statistical client privacy.

4.1 Making Π_{IKOPSW} perfectly server-private

4.1.1 Setting up deterministic watchlists

Recall the problem with client privacy was in the fact that the client may watch the internal state of too make servers, breaching security of the protocol Π, and thus of the entire construction. To solve this problem, we replace the probabilistic watchlist setup with a deterministic watchlist setup that allows the honest client to learn exactly $p_1 n$ of the veiews of its choice, while no client can learn more than $p_2 n$ of the values, where p_2 is not "much larger" than p_1. We also allow the server to input $\perp$, in which case the client should output $\perp$. Formally let k_1, k_2, n, h where $n \geq k_2 > 2k_1$, and $h \geq 1$. We define f_1 : k_1-out-of-n h-string-OT and f_2 : k_2-out-of-n h-string-OT. We extend the f_i above into $f_1' : X_1' \times Y_1' \to Z_1', f_2' : X_2' \times Y_2 \to Z_2'$, so that $Z_i = Z_i \cup \{\perp\}$, $Y_i = Y_i \cup \{\perp\}$ so that $\forall x \in X_i, f_i(x, \perp) = \perp$.

Theorem 4.2 (a (k_1, k_2)-out-of-n h-string-OT protocol)**.** *There exists a protocol Π as in Definition 2.2 for f_1' with communication complexity l. Π satisfies:*

- *Perfect honest corretness.*
- *Perfect client-correctness with input-dependent abort.*
- *Perfect server privacy relaxed to replace f_1' with f_2' (that is, malicious clients can be simulated by inputting additional, larger, sets beyond what is allowed by f_1).*

Proof. We construct a protocol as required, we refer to as $\Pi_{\text{ramp-OT}}$.

Construction 1. *The parties perform n 1-out-of-2 nh-string-OT's in parallel (implemented via the 1-out-of-2-bit-OT^l using parallel instances of Π_{int}).*

1. $\mathbf{\Pi_D(y)}$: *In the i'th instance the sender samples n random strings $r_1, \ldots, r_n \in \{0,1\}^h$. Then, it secret-shares $r = (r_1, \ldots, r_n)$ with a $(n - 2k_1)$-out-of-n threshold using Shamir's secret sharing [?]. Let $s_1, \ldots, s_n \in \{0,1\}^{hn}$ denote the resulting shares. Output $s = (y_1 \oplus r_1, s_1, y_2, s_2, \ldots, y_n \oplus r_n, s_n)$ (the y_i's are padded accordingly).*

2. $\mathbf{\Pi_Q(x = \{x_1, \ldots, x_{k_1}\})}$: *Assume wlog. that all the x_i's are distinct. Output $c = (c_1, \ldots, c_n)$, where $c_i = 0$ iff. $i \in x = \{x_1, \ldots, x_{k_1}\}$.*

3. $\mathbf{\Pi_R(x, c, v = (v_1, \ldots, v_n))}$: *Here v_i is the string returned by the i'th instance of Π_{int}. Let $w_i = \Pi_{intR}(x, c_i, v_i)$ denote the output of the i'th instance of Π_{int}. Use the w_i's in $SR = [n] \backslash x$ to recover the secret vector r as follows:*

   (a) *If all subsets of size $n - 2k_1$ of SR agree on the same secret r, output $(w_i \oplus r_i)_{i \in x}$.*

   (b) *Otherwise, output $\perp$.*

We now prove the protocol satisfies the required security guarantees. By consturction, it is not hard to see that the protocol satisfies honest correctness. As for client correctness, we observe that for any server strategy r^* (input to OT oracle) there exists a value $y^* \in \{0,1\}^{2nh}$ such that for every client's input x, either $\perp$ or $y^*[x]$ is output (with probability 1), and the choice to abort or not depends on x. To see this, it sufficies to observe the following

Observation 2. *Fix some server strategy s. Then, for server strategy s, no two choices c_1, c_2 by the (honest) client yield corresponding reconstructed secret values $r_1 \neq r_2$ that are both non-$\perp$.*

Assume the contrary, that for some server strategy $s = (s_{1,0}, s_{1,1}, \ldots, s_{n,0}, s_{n,1})$ such c_1, c_2 exist. For any pair of sets SR_1, SR_2 of size k_1, SR_1, SR_2 have at least $n - k_1 - k_1 = n - 2k_1$ coordinates in common. This set $B = SR_1 \cap SR_2$ can thus reconstruct the secret s. To lead to non-$\perp$ output by Π_R, all reconstructing subsets of SR_1, SR_2 must agree with s (if not, the corresponding output is $\perp$).

Observation 3 implies that all sets SR leading to non-$\perp$ values for some x lead to the reconstruction of the same value $r^* = s$ (if no such set exists, we fix r^* arbitrarily). Therefor, $y^* = (s_{1,0} \oplus r_1^*, \ldots, s_{n,0} \oplus r_n^*)$ is as promised above. Now, clearly, for each x $y^*[x]$ or $\perp$ is output (depending on the resulting SR_x). To see perfect server privacy holds, observe that if the client tries to learn a set S of at most k_2 of the 0-inputs, it can also reconstruct the secret mask r, and recover all values in S. It learns nothing about the other values, as it only learns one out of 2 'shares' in an additive sharing of these values (originating form r), this can be simulated by sending S in the ideal model. If it attempts to learn a set S of size $> k_2$, it learns nothing about r, as it does not have enough shares of it, so that it learns nothing about any of the server's inputs (its view consists of random field elemenets for the Shamir sharing, and random strings for each of the server inputs in S). Thus, its input can be simulated given any S (let us fix some S of size k_1) to be sent to the TP in the ideal model to simulate its view.

4.1.2 Upgrading Π_{IKOPS}

Let us recall and modify (a specific variant of) the original protocol $\Pi^{\epsilon}_{\text{IKOPSW}}$ in more detail. In particular, we change the watchlists from picking each server with a certain probability p, to picking a random subset of size exactly pn.

Here we represent X, Y, Z as some $X = \{0,1\}^{l_x}, Y = \{0,1\}^{l_y}, Z = \{0,1\}^{l_z}$.

Construction 2 ([?]). *Fix f as above, and a security parameter $\epsilon > 0$. Then $\Pi^{+}_{IKOPS,f,\epsilon}$ is as follows.*

1. *Let $\kappa = O(\epsilon^{-1})$ for a suitable constant to be determined later. Let us encode input $x = (x_1, \ldots, x_{l_x})$ via $Enc_I(x) = (x_{1,1}, \ldots, x_{1,\kappa+1}, \ldots, x_{l_x,\kappa+1})$, where each $x_{i,1}, \ldots, x_{i,\kappa+1}$ is a random additive sharing of x_i. Let $h : X^{\kappa+1} \times Y \to Z$ be defined as $h(Enc_I(x), y) = f((\oplus_{i=1}^{\kappa+1} x_{1,i}, \ldots, \oplus_{i=1}^{\kappa+1} x_{\kappa+1,i}), y)$.*

2. *Let (Enc_h, Dec_h) denote a decomposabe PRE of a function $f'(x,y)$ related to f. For instance, we could separately encode each of the bits of $f(x,y)$ using the perfect formula-based encoding from [?]. The resulting encoding has monomials of degree at most 3, and degree 1 in x, y or combinations of the form $x_i y_j$, but for a fixed value of y, all these become degree-1 in x_i (with possibly r_j's involved). Furthermore, each polynomial has a constant number of monomials, so it is easy to make the encoding (for any fixed y decomposable by introducing a constant overhead and more randomness r_s []). The complexity of this encoding is $(\kappa + |C|)^2$, where $|C|$ is the formula complexity of each bit of $f(x,y)$.*

3. *Consider a semi-honest protocol Π' for evaluating f where $c = Enc_I(x)$, and the server inputs s where $s_{i,b} = Enc_{h,i}(b, (y, r_s))$. The client then recovers $f(x,y)$ from $s[c] = s_{1,x_1}, \ldots, s_{l_x, x_{l_x}}$ by applying Dec_h to $s[c]$. We denote $\alpha = (\kappa+1)l_x$.*[20]

4. *To evaluate the protocol Π' above, we reduce it to a COT (conditional OT) functionality, similar to that of [?]. The COT protocol recieves (y, r) from the server, and evaluates the decomposable NC_0 encoding above to generate the $2l'_x$ purported output values $(v_{1,0}, v_{1,1}, \ldots, v_{\alpha,0}, v_{\alpha,1}) = Enc_{h,1}(0, (y, r_s)), \ldots, Enc_{h,\alpha}(1, (y, r_s))$. Each value is either the correct value $Enc_{h,i}(b, (y, r_s))$, or $\perp$ in case the $v_{i,b}$ is not consistent with (y, r_s). As in [?] the server performs MPC in the head of an MPC protocol Π_{HM} evaluating the COT functionality. The MPC participants are a sender with input (y, r_s), 2α clients and some n virtual servers that perform the computation. The protocol proceeds by having the sender send messages to the virtual servers, the servers running Π_{HM} among themselves, and sending the corresponding outputs to the clients during a single round at the end, where each client recieves a single message from the virtual servers, from which it recovers its output.*

   The protocol has the following security guarantees. It is perfectly correct against any adversary corrupting the sender and at most δn of the servers, and any number of the clients. Such protocols exist for all $\delta < 1/3$. Also, we will n to satisfy $2\lfloor \delta/3n \rfloor < \lfloor \delta n \rfloor$ (from now on we omit $\lfloor\ \rfloor$ for brevity). Let us fix $\delta = 1/4$, which imposes that $n > 12$.

   If the sender is honest, Π_{HM} is perfectly private against semi-honest *adversaries corrupting any subset of clients, and up to δn of the virtual servers. Consider [] as such a protocol with CC linear in the size of the encoding circuit, which is an optimized variant of [?].*

5. *The server runs Π' on (y, r_s) in his head, generating the views $V_1, \ldots, V_n$ of the virtual servers, and the purported messages $V_{1,0}, V_{1,1}, \ldots, V_{2\alpha,1}$ recieved by each of the 2α clients from all n virtual parties.*

   Recall that in [?], on input $x' = Enc(x)$, the parties run parallel instances of 1-out-of-2-bit-OTh (for the proper value of h), where in the first α instances, instance i inputs $V_{i,0}, V_{i,1}$ are used by the server. The other set of parallel OT's is dedicated to watchlists. The client picks $\delta/3n$ of the views $V_1, \ldots, V_n$. This is done using the $(\delta/3n, \delta n)$-out-of-n h-string-OT protocol in $\Pi_{ramp\text{-}OT}$. [21]

   The client sends $c = (x_1, \ldots, x_\alpha)$ as its queries for the first part, we refer to as the "keys" part, and picks a random subset S of size $\delta/3n$ of $[n]$ as its input set for the watchlist part $\Pi_{ramp-OT}$. Given the replies of both parts: if $\Pi_{ramp-OT}$ outputs $\perp$, output $\perp$ as well. Otherwise, it reconstructs the set $\{V_{s\in S}\}_S$ of virtual server views. It checks whether all the recovered V_i's are consistent among them (comparing messages sent and recieved among viewed parties, and that they are consistent with their inputs and randomness). The client outputs $\perp$ if an inconsistentcy among the watched views of the V_i's was discovered. Otherwise, it checks whether any of the messages in some V_{i,x_i} seen by the client are inconsistent with the values sent in the watched V_j's. Otherwise, it reconstructs $f(x,y) = h(Enc(x), y)$ from $(V_{i,x_1}, \ldots, V_{i,x_\alpha})$.

Lemma 4.3. *Fix some $\epsilon > 0$, and $f : X \times Y \to Z$. Then $\Pi^+_{IKOPS,f,\epsilon}$ in Construction 2 is perfectly correct for honest parties, and is perfectly server-private. It also satisfies enhanced ϵ-client correctness. Let us denote the complexity of the resulting protocol by $\ell(\epsilon, |X|, |Y|)$.*

[20] We use α to be consistent with [] to help the reader who is familiar with [].

[21] As discussed above, the watchlist implementation is where we diverge from [?], everything so far was unmodified.

Proof Sketch. It easy to see that the protocol remains correct in face of honest parties. To prove perfect server privacy, fix some client strategy c^*. The first part (virtual client views) of the protocol only discloses at most one V_{i,b_i} for every $i \in [\alpha]$, by perfect server privacy of Π_{int}. Let x^* denote the input corresponding to the b_i's learned (set x_i^* arbitrarily).

We run the simulator guaranteed by $\Pi_{\text{ramp-OT}}$ of the malicious client strategy c_2^* induced by c^* on $\Pi_{ramp-OT}$. Consider the client's induced strategy in $\Pi_{\text{ramp-OT}}$. The simulator sends some distribution M^* over $\{m \subseteq [n] || c| \leq \delta n\}$. For m^* in $support(H^*)$, by perfect server privacy of Π_{HM}, the client learns exactly what follows from $h(x^*, y)$, as it learns at most δn of the virtual server's views. Like [**?**], we also use the fact that any value $v \in \{0,1\}^\alpha$ corresponds to $Enc_I(x)$ for some x, so client's view can be perfecetly simulated given $Dec_I(x^*)$.

Enhanced ϵ-client correctness holds similarly to the original construction. On a high level, the only new case that we need to adress here is when $\Pi_{\text{ramp-OT}}$ makes the client output $\perp$.

In more detail, the simulator of the server acts as follows given the server's strategy s^*.

1. It runs the simulator for $\Pi_{\text{ramp-OT}}$ on s_2^*, the part of the server's input corresponding to the wishlist part of the protocol.

2. It picks a random subset T of size $\delta/3n$ of $[n]$ Let u_2 denote the server's effective input guaranteed by $Sim_{\text{ramp-OT}}$ for client inputs T for which the output is non-$\perp$.

   (a) If the output is $\perp$ for all T, send $\perp$ as an input to the TP (of our 2-party protocol evaluating f).

   (b) Otherwise, let u_2 denote the extracted server's input guaranteed to exist by the security definition of $\Pi_{\text{ramp-OT}}$. Consider the consistency graph G' among the virtual servers induced by u_2, where an edge (A, B) exists between a pair of virtual servers A, B iff. their views do not match. That is, some message recieved by B (A) does not match the message sent by A (B) as determined by A's (B) randomness and initial message from the sender.

      i. If the minimal VC L of the consistency graph G' is of size $t > \delta/3n$, send $\perp$ to the TP.

      ii. Otherwise, $t < \delta/3n$. Pick a subset T of size $\delta/3n$ among the virtual servers. If there is an edge of G' inside T, send $\perp$ to the TP. Otherwise, let u_1 denote the sender's input extracted for the "keys" part of the protocol - executing the α parallel instances of Π_{int}. Complete the graph G' into a graph G with all vecrtual clients and virtual servers (but not the sender). Add edges between V' and $V \setminus V'$ between client $A = (i, b)$ and server B iff. $u_1[i, b]$ is not consistent with B's message to A according to u_2. Let B_1 denote the set of client nodes that have a neighbor in T. Let $g = |B_1|$. If $g \geq \kappa + 1$ or $x_{i,j,0}, x_{i,j,1} \in B_1$ for some i, j send $\perp$ to the TP. Otherwise, send the TP $\perp$ with probability 2^{-g}, and (y, r_s) otherwise (with probability $1 - 2^{-g}$). Here (y, r_s) is obtained by extracting (y, r_s) according to the views of the sender and L, as appearing in u_2. In particular, the sender's messages are taken from all virtual server's first messages as appearing in u_2. In particular, send $\perp$ if $(y, r_s) = \perp$ is extracted. [22]

It remains to prove that the simulator indeed satisfies the client correcntness requirement. Consider several cases.

1. Assume the server's input s to the second part is such that the client outputs $\perp$ for all inputs $x \subseteq [n]$ of size $\delta/3n$. Then the client in the real world outputs $\perp$ with probability 1. This is also the case in the simulation, leading to $\epsilon = 0$ simulation error.

2. Otherwise, let u_2 denote the server's input extracted by the simulator of $\Pi_{\text{ramp-OT}}$. This value is identical to the effective input in the real worls (in fact, it is a single value, not even a general distribution).

[22] Here a note on simulator efficiency is in order. In this work we do not require efficient simulation. In particular, we compute exact minimal VC, rather than approximate VC, for the sake of simplicity. Similarly to [] we could have, if we strived for efficient simulation. The part of extracting (y, r_s) is also efficient if Π_{HM}'s simulator is efficient. To be precise, the simulator is expected to recieve the code of TM's rather than concrete views. The simulator assumes that the sender deterministically sends messages as in *all* virtual server's views, and the virtual servers in B have next-message functions consistent with the views in u_2, and arbitrary otherwise. As the protocol is Π_{HM} perfectly correct in face of an adversary corrupting $\{sender\} \cup B$, the correct distribution (y, r_s) is always extracted.

(a) If the inconsistency graph induced by u_2 is of size $t > \delta/3n$, then the smallest maximum matching in the graph is of size at least $t/2$. Thus, by a simple analysis using Chernoff bounds (see [] for details), the client picks a set T of parties covering some edge in G' with probability $2^{-\Theta(n)}$. Thus, with probability at least $1 - 2^{-\kappa}$, we the client outputs $\perp$. This probability may also rise due to an event of outputting $\perp$ for a certain choice of T due to $\Pi_{\text{ramp-OT}}$ leading to an output of $\perp$ for that value of T. The simulator otuputs $\perp$ with probability $b_\perp = 1$. Thus the security requirement is satisfied in this case for a proper choice of κ.

(b) Otherwise, for fix some choice of T by the client in the wishlist part. If $\perp$ is output due to $\Pi_{\text{ramp-OT}}$'s execution, then this is also the case when the simulator chooses T. Otherwise, extract the server's effective input $\perp$ to the "keys" part of the protocol - it is also identically distributed to the effective input of the server to the first part (in fact, by structure of Π_{int}, this is also a single value). Consider the set B_1 of nodes in $V \setminus V'$ with nieghbours in T.

i. If $g \geq \kappa + 1$, consider the maximal subset $B_1' \subseteq B_1$ where for every $\forall i \in [l_x], j \in [\kappa+1] \forall b \in \{0,1\} x_{i,j,b} \notin B_1'$. By the structure of Enc_h, at least a $(1-2\kappa^{-1})$-fraction of B_1 is in B_1'. This holds as for each i, we either keep all of the nodes in B_1, or remove at most 2 out of at least $\kappa + 1$. For all $x \in X$, the probability that $E_h(x)$ has some $x_{i,j}$ value so that node $x_{i,j,x_{i,j}}$ is in B_1' is $1 - 2^{-|B_1'|} \geq 1 - 2^{-\kappa/2}$. In this case the client outputs $\perp$. The probability of outputting $\perp$ conditioned on choosing T by the simualtor is 1.

ii. Otherwise, $g \leq \kappa$. In this case, for all x, $B_1' = B_1$ (where B_1, B_1' are defined as above). Therefore, for all x, the probability of outputting $\perp$ due to hitting an edge in $V \setminus V' \cup T$ is the same as for the simulator conditioned on choosing T. If the event of hitting an edge does not occur, the output induced by u_2 is always consistent with (y, r_s) extracted by the simulator conditioned on picking $\perp$ (in particular, it may or may not equal $\perp$).

To summarize, for every input x, the extracted distribution on server inputs induces an output distribution that is at distance at most $2^{-\kappa} p_\perp$ from the real output distribution. More concretely, the difference stems only from case 2.b.ii $\square$

5 Transforming Π_{IKOPS^+} into a perfectly correct protocol

In this section we describe the first step of the construction. Here we rely on ideas from convex geometry that are somewhat similar to those of []. In particular, the view of security definitions as in Section **??** and the following notion of full-dimensional functions is central to our construction.

Lemma 5.1. *Let $\epsilon > 0$, and let Π denote a protocol as in Definition 2.2 for evaluating f with ϵ-enhanced client security, perfect server privacy, and perfect honest correctness. Then there exists a perfectly secure protocol as in Definition **??** for evaluating f with communication complexity $\ell(\epsilon, |X|, |Y|)$, where ℓ is as in Theorem 4.3.*

As a corollary from Lemma 5.1 and Theorem 4.3, we obtain our main result.

Theorem 5.2. *Let $f : X \times Y \to Z$ denote a full-dimensional function, and $g = |X|(|Z| - 1)$. Then there exists a protocol Π' as in Definition 2.2 evaluating f with perfect security against a malicious adversary in the 1-out-of-2-bit-OT^l-hybrid model for $l = \ell(1/10(g-1)g!, |X|, |Z|) = \tilde{O}(h^2, poly(|X|, |Z|))$, where ℓ is the function in Theorem 4.3.*

Here $poly(|X|, |Z|)$ is a global polynomial, independent of f. TODO: Calculate what the polynomial's degree is. The polylog factors are more tedious to compute.

Proof sketch. We pick a sufficiently small $\epsilon > 0$, to be set later, and consider $\Pi^+_{\text{IKOPS}^+, f, \epsilon}$. Fix some vector v in the convex hall of F's (not F''s) rows F_y, which is "far enough from the edges" of that polygon. By "far enough" we mean that adding up to $\pm\epsilon p_\perp$ in every coordinate results in a point which is still inside the polygon. Our protocol Π' proceeds as follows. Whenever $\Pi^+_{\text{IKOPS}^+, f, \epsilon}$ outputs $\perp$ as an output on input x, output a distribution consistent with v_x. Otherwise, output

the value output by Π. Indeed, as $\Pi^{+}_{\mathrm{IKOPS}^{+},f,\epsilon}$ satisfies ϵ-enchanced client correctness, its output distribution is of the form.

$$o^{+} = \sum_{i} \alpha_i o_i + e_r \tag{1}$$

Here $\alpha_i o_i$ is a convex combination of the rows Y'_i of F'. The vector e_r is an error vector, $|e_r|_\infty \leq \epsilon\alpha_\perp$, where $\alpha_\perp$ is the coefficient of $Y_\perp$.[23].

Now, as in Π, an output of $\perp$ on input x is replaced by v_x, o^{+} in Equation 3 is replaced by

$$\tilde{o} = \sum_{i \in Y} \alpha_i Y_i + \alpha_\perp(v_x + \tilde{e}_r) \tag{2}$$

where $|e_r|_\infty \leq \epsilon$. It remains to show that $d = v_x + \tilde{e}_r$ is in $\mathrm{CH}((\{Y_y\}_{y\in[|Y|-1]})$ for the right choice of v, ϵ. This is the case as if $d = \beta_i Y_i$ is a convex combination of F's rows, we get that

$$\tilde{o} = \sum_{i \in Y} \alpha_i Y_i + \alpha_\perp(\beta_i Y_i) = \sum_{i \in Y} (\alpha_i + \alpha_\perp \beta_i) Y_i$$

Since $\alpha_1, \alpha_\perp, \ldots, \alpha_{|Y|}$, and (separately) the β_i's are coefficients of a convex combination, thus constituting a valid ideal-world row distribution for f.

It remains to prove that there is a way to pick v, ϵ so that the resulting $\tilde{o}$ satisfies the client correctness requirement. Since f is full-rank, let us pick a set of size $g = |X|(|Z| - 1) + 1$ of rows of F, $V = \{Y_1, \ldots, Y_g\}$, such that the dimension of $\{\Delta_{i-1} = Y_i - Y_1\}_{i\geq 2}$ is $g - 1$. We will pick v as a convex combination

$$\sum_{i \leq g} \alpha_i Y_i$$

of the Y_i's, which we determine in the following. As F has 0-1 entries we have

Observation 3. *Each Δ_j is has entires in the set $\{0, 1, -1\}$.*

Let us pick

$$v = 3/4 Y_1 + \sum_{2 \leq i \leq g} 1/4(g-1) Y_i$$

Now, we adapt ϵ to this choice based on the following observation.

Claim 5.3. *Let $u \in \mathbb{R}^{g-1}$ be a vector with $|u|_\infty \leq \epsilon$. Then, in the representation $u = \sum_{i \leq g-1} \alpha_i \Delta_i$ (it exists and is unique as the Δ_i's form a basis of $\mathbb{R}^{g-1}$)), it must be the case that $|\alpha_i| \leq \epsilon \cdot g!$.*

To prove the claim we apply Fact 1 to $M = (\Delta_1 || \Delta_2 || \ldots \Delta_{g-1})$, and use Observation 3 to bound each $|M^{-1}_{i,j}|$ as $|C_{i,j}| \leq (g-1)!/1 = (g-1)!$. Now, the solution to $Mx = u$ is $M^{-1}u$, thus we get $|x|_\infty \leq (g-1)(g-1)!\epsilon \leq g!\epsilon$, as required (here $g-1$ is the length of u).

We can rewrite a convex combination of the Y_i's as

$$\sum_{i \leq g} \alpha_i Y_i = Y_1 + \sum_{i \leq g-1} \alpha_{i+1} \Delta_i \tag{3}$$

where the α_i's on the right hand side are non-negative integers summing to *at most* 1 (and move back and forth between the two representations). In particular, we have

$$v = Y_1 + 1/4(g-1) \sum_{i \leq g-1} \Delta_i$$

Recall also that the resulting row distribution is $o' = (1-\epsilon)v + e'$, where $|e'|_\infty \leq \epsilon$. Thus, we have $e' = \sum_{i \leq g-1} \alpha_i \Delta_i$, where $|\alpha_i| \leq g!\epsilon$.

Thus, we have

$$o' = (1-\epsilon)v + e' = Y_1 + (1-\epsilon)1/g \sum_{i \leq g-1} \Delta_i + (e' - \epsilon Y_1)$$

Let us write $w = e' - \epsilon Y_1$. Clearly, $|w|_\infty \leq 2\epsilon$. Here $|\beta_i| \leq 2\epsilon$ for each i. Thus, from Claim 5.3 we have

$$o' = Y_1 + \sum_{i \leq g-1} ((1-\epsilon)1/4(g-1) + \beta_i)\Delta_i$$

where $|\beta_i| \leq 2g!\epsilon$. Thus (since $g \geq 2$), picking $\epsilon = 1/10(g-1)(g!)$, we obtain o' of the form presented in Equation 3, which falls in the required region (the coefficients of the Δ_i's are all non-negative and sum to at most 1).

[23] In fact, for every x, $|e_x|_1 \leq \epsilon$, but using this stronger property would not improve our result.

References

6 Applications

Our protocol allows evaluating functions whose truth table are full-dimensional.