Perfect reductions

February 22, 2018

Abstract

We initiate the study of perfect (rather than merely statistical) reductions among cryptographic primitives. For simplicity, we focus on client-server functionalities. As opposed to the computational and statistical worlds, quite little is known here. 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. Furthermore [?] demonstrate a reduction reduction with improved efficiency that can be carried out in one round, making parallel calls to the OT (oracles), where the receiver plays the role of OT receiver in all instances [?].

We make first steps towards understanding perfect reductions, proving that a large class of client-server functions is perfectly reducible to 1-out-of-2 bit-OT. To the best of our knowledge. In particular, 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 constant functions are perfectly reducible to 1-out-of-2 OT. This fraction grows roughly as $1 - exp(|X| - |Y|)$ Furthermore, the reduction is 1-round, and the receiver plays the role of the receiver in all OT calls. As far as we know, before now, very few functionalities such as k-out-of-n OT were known to be computable in the 1-out-of-2 bit-OT hybrid model with perfect security.

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. Next, once could study functionalities where both parties receive inputs, and randomized functionalities.

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

TODO.

2 Preliminaries

Oblivious transfer (OT).

Theorem 2.1 ([]). *There exists a perfectly secure protocol as in Definition 2.2 of 1-out-of-2 length-h-string-OT with communication complexity $l \leq 10h$. We refer to it as the Π_{int} protocol.*

Geometry.

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. For k-out-of-n-bit-OT or k-out-of-n length-h-string-OT, it will be convenient to denote the input domain X as $X = \{c \in \{0,1,\perp\}^n | c \text{ contains exactly } k \text{ non-}\perp \text{ values}\}$.

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).[1]

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. The security notion is the standard simulation-based notion as in [?]. We consider statistical security against a malicious server or a malicious client, and perfect correctness if both parties behave honestly. If the simulation corresponding to a malicious adversary (server or client respectively) has distance $\epsilon = 0$ we say the protocol is perfectly secure against a malicious server (client).

We define ϵ-enhanced security against malicious servers, or ϵ-enhanced client correctness. Here, we assume that $f : X \cap \{\perp\} \times Y \cup \{\perp\} \to Z \cup \{\perp\}$. We have $f(x,y) = \perp$ iff. $x = \perp$ or $y = \perp$. The simulator submits a distribution D over $Y \cup \{\perp\}$ to the TP, where b. We require that the client's output distribution is $\epsilon p_\perp$-close to its output distribution in the ideal world, where $p_\perp = Pr_D(\perp)$.

may send "'abort" to the TP, which instructs it to send $\perp$ to the client.

More specifically, we focus on perfect 1-round protocols Π in the 1-out-of-2-bit-OT-hybrid model securely evaluating f, where the client plays the role of the receiver in all OT calls.

In our setting, a protocol for evaluating a client-server functionality $f : X \times Y \to Z$ is defined as follows.

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.[2] $\Pi_D(y) : Y \to \{0,1\}^{2l}$ is server's generator of OT inputs. We refer to l as the communication complexity of Π.*

For $s = (s_{1,0}, s_{1,1}, \ldots, s_{l,0}, s_{l,1}) \in \{0,1\}^{2l}$, and $c \in \{0,1\}^l$, we let $s[c]$ denote $(s_{1,c_1}, \ldots, s_{l,c_l})$. For a vector $c \in \{0,1,\perp\}^l$ $s[c]$ is defined as above, but if $c_i = \perp$ the i'th entry in $s[c]$ equals $\perp$.

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.[3] 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 (where $(s_{i,0}, s_{i,1})$ go to the i'th 1-out-of-2-bit-OT instance).

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.

Note that the above definition of a protocol is merely syntactic.

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

[2] 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.

[3] 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.

2.1.1 Restating security requirements geometrically.

We take a similar approach to that of [?] to representing protocols and their 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 distribution 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. While the first two requirements are readily expressed as a single LP (linear program) [], it is unclear how to incorporate the third requirement into a linear program.

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 $s^* \in \{0, 1\}^{2l}$[4], we 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.[5] Similarly, we consider *geometric column distributions* for Π: for a given 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 $o_{(x,i)} \geq 0$.[6] 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 $(o_{x,z})_{z \in [k-1]}$.

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

[5] 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^*.

[6] The decision to exclude 0 is merely auesthetic, intended to remain consistent with standard binary truth tables.

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).[7] Again, each row F_y is a valid geometric row distribution in the ideal world (corresponding to a server input of y).

Definition of security We require stand-alone security against a single malicious party. This requirement can be restated as three separate requirements.

Let us recall the standard definitions of security against malicious parties in the 1-out-of-2-bit-OT^l-hybrid model.

Definition 2.3. *Let Π denote a protocol for evaluating a function f as in Defintion 2.2 in the 1-out-of-2-bit-OT^l-hybrid model. Then it evaluates f with malicoius security against a non adaptive malicious adversary if it satisfies:*

Security against malicious servers: For all algorithms $\Pi^*_D(y;r)$, there exists a randomized simulator algorithm $Sim^*_D(y;r) : Y \to Y$ such that for all $x \in X, y \in Y, r_s$, the following equality of distributions holds.

$$\Pi_R(x, \Pi_Q(x;r_c), \Pi_D(y;r_s)) = f(x, Sim^*_D(y;r_s))$$

[8]

Definition 2.4. *We say that a reduction Π for evaluating $f : X \times Y \to Z$ 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.

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

We will also need a relaxed definition of client correctness, where the client may also output an error symbol $\perp$. We further relax it by allowing for a simulation error ϵ.[9]

Definition 2.5. *We say a protocol Π for computing f is secure with ϵ-relaxed client correctness, if it satisfies Definition 2.4, but client correctness is relaxed as follows. We allow the client to output*

[7] This is instead of having a single entry for each (x,y) with values in Z.

[8] Wlog. we may assume Π^*_D is deterministic. Also note that we do not require that the algorithms (as we generally consider finite functionalities). Due to the simple structure of Π the above requirement can be further simplified to requiring that for all . there exists a distribution Y^* over Y such that

$$\Pi_R(x, \Pi_Q(x;r_c), s^*) = f(x, Y^*)$$

.

[9] This is a re-interpretation of the security of [**?**]'s protocol using our language.

a special error symbol $\perp$, in particular, we reinterpret the function f as having an output domain $Z' = Z \cup \{\perp\}$. We say a row distribution o is admissible, if it is a convex combination

$$o^* = \sum_i \alpha_i o_i$$

where each o_i is either a row F_y in the truth table F, or is the all-$\perp$ vector $o_\perp$.

Then, for every deterministic server strategy $s^* \in \{0,1\}^{2l}$, the resulting row distribution o^* satisfies either: (1) There exists an admissible row distribution o where $\alpha_\perp \geq 1 - \epsilon$ such that $|o_x - o^*_x|_1 \leq \epsilon$ for all $x \in X$.[10] Or (2) There exists an admissible row distribution o, so that $o^* = o$.

For client-server functionalities and protocols Π as in Definition 2.2 as we consider, our Definitoin **??** of perfect security is equivalent to standard perfect security against malicious adversaries, and the equivalence is straight forward to prove.

Lemma 2.2. *Let $\Pi = (\Pi_Q, \Pi_R, \Pi_D)$ be a protocol for evaluating a client-server functionality $f : X \times Y \to Z$. Π is perfectly secure by Definition 2.4 iff. it is perfectly secure against a single malicious party in the 1-out-of-2-bit-OT^l-hybrid model according to standard simulation based definition [] (against a non-adaptive malicious adversary).*

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 $f(x,y)$.[11]

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.[12] 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

Our starting point is a protocol as in Definition 2.2 (1-round protocol in the (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). 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.

[10] This simply means that for every input x, a non-$\perp$ value is output with probability at most ϵ

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

[12] 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.

Theorem 4.1. *Let $f : X \times Y \to Z$ denote any (finite) 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 $l = \log(|Z|)\tilde{O}(\log(|X|)(\log(epsilon^{-1}) + h)^2) + poly(\epsilon^{-1})$.*

We transform it into a perfectly secure protocol in two steps. First, we transform it into a perfectly server-private (yet not perfectly client-correct) protocol Π'. Then we transform Π' into Π'' preserving perfect server privacy and adding perfect client correctness.

4.1 Making $\Pi_{\mathbf{IKOPSW}}$ perfectly server-private

In a nutshell, in the Π_{IKOPSW11} protocol, the server runs MPC in the head of an n-party protocol verifying that the OT input $s \in 0, 1^{2l}$ it feeds to the OT oracle is consistent with some y, r. This protocol (executed in the head) is secure against malicious corruption of δn of the n parties. The client sets up a watchlist on the view of each server with probability $p < \delta$ each (that is with probability p it will see the party's view, and $1-p$ it sees nothing). With high probability, the client will see $p/2n < cn < \delta n$ of the views, catching cheating with high probability, but not learning anyting besides $f(x, y)$. 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).

4.1.1 Setting up deterministic watchlists

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. 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 length-h-string-OT and f_2 : k_2-out-of-n length-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 length-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.*
- *Perfectl 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 above.

Proof. The parties perform n 1-out-of-2 length-nh-string-OT's in parallel (implemented via the 1-out-of-2-bit-OT^l using parallel instances of Π_{int}). We denote this construction by $\Pi_{\text{ramp-OT}}$

1. $\mathbf{\Pi_D(y)}$: In 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 $s = (y_1 \oplus r_1, s_1, y_2, s_2, \ldots, y_n \oplus r_n, s_1)$ (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. $i \in Ix = \{x_1, \ldots, x_{k_1}\}$.
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 $w_i = \Pi_{\text{int}\,R}(x, c_i, v_i)$ denote the output of the i'th instance of Π_{int}. Use the w_i's in $SR = [n] \setminus I_x$ 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 $(w_i \oplus r_i)_{i \in I_x}$.
   (b) Otherwise, output $\perp$.

□

We now prove the protocol satisfies the 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 1. *No two choices c_1, c_2 by the client yield corresponding reconstructed secret values $r_1 \neq r_2$ that are both non-$\perp$.*

Fix some server strategy $s = (s_{1,0}, s_{1,1}, \ldots, s_{n,0}, s_{n,1})$. Assume the contrary, then some pair of sets 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 $\perp$). Thus, 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 may set 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. Here we represent X, Y, Z as some $X = \{0,1\}^{l_x}, Y = \{0,1\}^{l_y}, Z = \{0,1\}^{l_z}$.

Construction 1 ([?]). *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 the function $f(x,y)$. 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 $\alpha = (\kappa+1)l_x$.*[13]

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*

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

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 length-h-string-OT protocol in $\Pi_{ramp\text{-}OT}$. [14]

   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 $(V_{i,x_1}, \ldots, V_{i,x_\alpha})$.

Theorem 4.3. *Fix some $\epsilon > 0$, and $f : X \times Y \to Z$. Then $\Pi^+_{IKOPS,f,\epsilon}$ in Construction 1 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 $\ell(\epsilon, |X|, |Y|)$.*

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).

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

(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. [15]

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).

   (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

[15] 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.

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.

Definition 5.1 ([]). *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* - $(|Z| - 1)|X|$.

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 3.3.*

As a corollary from Lemma 4.1 and Theorem 3.3, 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 $l = \ell(1/10(g-1)g!, |X|, |Z|)$, where ℓ is the function in Theorem 3.3.*

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 $\pm\epsilon$ 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 $\Pi^+_{\text{IKOPS}^+, f, \epsilon}$ satisfies ϵ-enchanced client correctness, its output distribution is of the form.

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

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 $Y_\perp$.[16].

Now, as in Π, an output of $\perp$ on input x is replaced by v_x, $\alpha_\perp Y_\perp$ in Equation 1 is replaced by $\alpha_\perp v$, resulting in a distribution $o' = \sum_i \beta_i o_i + e_r$ where $o'_1 = \sum_i \beta_i o_i$ is also a convex combination of the rows Y_i of F.

The resulting row distribution is a convex combination $\alpha_i o_i$, where o_i is either v, or a row vector in f's truth table F. As $v = \beta_i Y_i$ is a convex combination This is therefor a convex combination of the Y_i's, as required.

Otherwise, the resulting row distribution vector o_x in Π is of the form $(1-\epsilon)v + e$, where $|e|_\infty \leq \epsilon$. If $e_{x,\perp} = \alpha_{x,\perp}$, we add $\alpha_{x,\perp} v_x$ to o'_x, the row distribution vector in Π'. Thus, o'_x is a (syntactically) valid distribution for f

$$(1-\epsilon)v + e'$$

where $|e'_x|_1 \leq \epsilon$ for all x (by a simple calculation, that stems from the fact that F has 0/1 entries, and v has entries in $[0,1]$). In particular, $|e'|_\infty \leq \epsilon$ as well, leaving us inside the convex hall of the Y_i's for sufficiently small ϵ (since f is full rank).

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

It remains to prove that there is a way to pick v, ϵ so that the resulting o'_x satisfies the client correctness requirement. Since f is full-rank, let us pick $g = |X|(|Z| - 1) + 1$ 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. As F has 0-1 entries we have

Observation 2. *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 2 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

$$\sum_{i \leq g} \alpha_i Y_i = Y_1 + \sum_{i \leq g-1} \alpha_{i+1} \Delta_i \tag{2}$$

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 4.3 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 1, which falls in the required region (the coefficients of the Δ_i's are all non-negative and sum to at most 1)

6 Applications