On Perfectly Secure 2PC in the OT-Hybrid Model

Bar Alon(✉) and Anat Paskin-Cherniavsky

Department of Computer Science, Ariel University, Ariel, Israel
alonbar08@gmail.com, anatpc@ariel.ac.il

Abstract. A well known result by Kilian [22] (ACM 1988) asserts that general secure two computation (2PC) with statistical security, can be based on OT. Specifically, in the client-server model, where only one party – the client – receives an output, Kilian's result shows that given the ability to call an ideal oracle that computes OT, two parties can securely compute an arbitrary function of their inputs with unconditional security. Ishai et al. [19] (EUROCRYPT 2011) further showed that this can be done efficiently for every two-party functionality in NC^1 in a *single round*.

However, their results only achieve statistical security, namely, it is allowed to have some error in security. This leaves open the natural question as to which client-server functionalities can be computed with perfect security in the OT-hybrid model, and what is the round complexity of such computation. So far, only a handful of functionalities were known to have such 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 secure multiparty protocols with high concrete efficiency, achieved by eliminating the dependence on a security parameter.

In this work, we identify a large class of client-server functionalities $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0,1\}$, where the server's domain $\mathcal{X}$ is larger than the client's domain $\mathcal{Y}$, that have a perfect reduction to OT. Furthermore, our reduction is 1-round using an oracle to secure evaluation of many parallel invocations of $\binom{2}{1}$-bit-OT, as done by Ishai et al. [19] (EUROCRYPT 2011). Interestingly, the set of functions that we are able to compute was previously identified by Asharov [2] (TCC 2014) in the context of fairness in two-party computation, naming these functions *full-dimensional*. Our result also extends to randomized non-Boolean functions $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0, \ldots, k-1\}$ satisfying $|\mathcal{X}| > (k-1) \cdot |\mathcal{Y}|$.

1 Introduction

In the setting of secure two-party computation (2PC), the goal is to allow two mutually distrustful parties to compute some function of their private inputs. The computation should preserve some security properties, even in the face

B. Alon—Research supported by ISF grant 152/17.

D. Hofheinz and A. Rosen (Eds.): TCC 2019, LNCS 11891, pp. 561–595, 2019.
https://doi.org/10.1007/978-3-030-36030-6_22

of adversarial behavior by one of the parties. The two most common types of adversaries are *malicious* adversaries (which may instruct the corrupted party to deviate from the prescribed protocol in an arbitrary way), and *semi-honest* adversaries (which must follow the instructions of the protocol, but may try to infer additional information based on the view of the corrupted party).

Oblivious transfer (OT) is a two-party functionality, fundamental to 2PC and the more general secure multiparty computation (MPC). It was first introduced by Rabin [28] and Even et al. [13]. In the setting of $\binom{2}{1}$-bit-OT, there is a receiver holding a bit $b \in \{0,1\}$, and a sender holding two bit-messages $a_0, a_1 \in \{0,1\}$. At the end of the interaction, the receiver learns a_b and nothing else, and the sender learns nothing. It turns out that OT can be used in the construction of protocols, both in 2PC and MPC with various security guarantees [6,14,22,33]. Moreover, giving to the parties access to an ideal process that computes OT securely, is potentially useful. Constructing protocols in this model, called the OT-hybrid model, could be used for optimizing the complexity of real-world, computationally secure protocols for several reasons. First, using the OT-precomputation paradigm of Beaver [4], the heavy computation of OT can many times be pushed back to an off-line phase. This off-line phase is performed before the actual inputs for the computation (and possibly even the function to be computed) are known. Later, as the actual computation takes place, the precomputed OTs are very cheaply converted into actual OT interactions. Furthermore, the OT-extension paradigm of [5] offers a way to efficiently implement many OTs using a relatively small number of base OTs. This can be done using only symmetric-key primitives (e.g., one-way functions, pseudorandom generators). Furthermore, it can also be used to implement $\binom{2}{1}$-s-string-OT using a sub-linear (in the security parameter) number of calls to $\binom{2}{1}$-bit-OT and some additional sub-linear work, assuming a strong variant of PRG [17]. Additionally, there is a variety of computational assumptions that are sufficient to realize OT [27], or even with unconditional security under physical assumptions [10,11,21,26,32].

An interesting family of two-party functionalities are the client-server functionalities, where only one party – the client – receives an output. In addition to the OT functionality mentioned earlier, client-server functionalities include many other examples. Securely computing some of theses functionalities could be useful for many interesting applications, both in theory and in practice.

For client-server, a well known result due to Kilian [22], asserts that OT is complete. That is, any two-party client-server functionality can be computed with unconditional security in the OT-hybrid model. Ishai, Prabhakaran, and Sahai [18] further showed that the protocol can be made efficient. Later, it was shown by Ishai et al. [19], that in the OT-hybrid model, every client-server functionality can be computed using a *single round*. Furthermore, the protocol's computational and communication complexity are efficient for functions in NC^1. However, all of the results achieve only *statistical security*, namely, it is allowed to have some error in security.

For the case of *perfect security* in this setting much less is known. Given access to (many parallel) ideal computations for $\binom{2}{1}$-bit-OT, Brassard et al. [8] showed how to compute the functionality $\binom{n}{1}$-s-string-OT, and Wullschleger [30] showed how to compute $\binom{2}{1}$-bit-TO, which is the same as $\binom{2}{1}$-bit-OT where the roles of the parties are reversed. Furthermore, the former protocol has a single round, in which the parties invoke the OT, and with no additional bits to be sent over the channel between the parties. The latter protocol requires an additional bit to be sent by the server.

Observe that the result of [8] implies that any client-server functionality f can be computed with perfect security against *semi-honest* corruptions. Indeed, let n be the number of inputs in the client's domain, and let s be the number of bits required to represent an output of f. The server will send to the $\binom{n}{1}$-s-string-OT functionality all of the possible outputs with respect to its input, and the client will send its input. The client then outputs whatever it received from the OT. Clearly, the protocol is secure against semi-honest adversaries, however, in the malicious case, this is not true, in general. This is due to the fact that the server has complete control over the output of the client. For instance, for the "greater-than" function, the server can force the output of the client to be 1 if and only if y is even. Therefore, we are only interested in security against malicious adversaries.

Ishai et al. [20] studied perfectly secure multiparty computation in the correlated randomness model. They showed that any multiparty client-server functionality can be computed with perfect security, when the parties have access to a correlated randomness whose correlation depends on the function to be computed by the parties.

There are also various client-server functionalities that can be computed trivially (even in the plain model). For example, the XOR functionality can be computed by having the server sending its input to the client. These simple examples suggest that fairness is not a necessary condition for being able to compute a function perfectly in the client-server model.

Thus, the state of affairs is that most two-party client-server functionalities remain unclassified as to perfect security in the OT-hybrid model. In this work we address the following natural questions.

> Which client-server functionalities can be computed with *perfect security* against malicious adversaries in the OT-hybrid model? What is the round complexity of such protocols?

The questions have an obvious theoretical appeal to it, and understanding it could help us gain a better understanding of general secure computation. In addition, perfect security may be useful for designing multiparty protocols with high concrete efficiency, achieved by eliminating the dependency on a security parameter.

We stress that, under the assumption that NP $\not\subseteq$ BPP, it is impossible to achieve completeness theorems in our setting, similar to the completeness theorems of Kilian [22]. Indeed, suppose the parties want to compute an NP relation

with perfect zero-knowledge and perfect soundness. Then it is impossible even when given access to any ideal functionality with no input (distributing some kind of correlated randomness) [20]. This is due to the fact that if such a protocol does exist, then one can use the simulator to decide the relation, putting it in BPP. Since OT can be perfectly reduced to a suitable no-input functionality, this implies that no such protocol exist in the OT-hybrid model.

1.1 Our Results

Our main result is that if the parties have access to many parallel ideal computations of $\binom{2}{1}$-bit-OT, most client-server functionalities, where the server's domain is larger than the client's domain, can be computed with perfect full-security in a single round. Interestingly, the set of functions that we are able to compute was previously identified by Asharov [2] in the context of fairness in two-party computation, naming these functions as *full-dimensional*.

Let $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0,1\}$ be a function, where the server's domain size $|\mathcal{X}|$ is larger than the client's domain size $|\mathcal{Y}|$. Write $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, \ldots, y_m\}$. We consider the geometric representation of f as $|\mathcal{X}|$ points over $\mathbb{R}^{|\mathcal{Y}|}$, where the j-th coordinate of the i-th point is simply $f(x_i, y_j)$. We then consider the *convex polytope*[1] defined by these points. The function is called *full-dimensional* if the dimension of the polytope is exactly $|\mathcal{Y}|$, e.g., a triangle in the plane.[2] We prove the following theorem:

Theorem 1 (Informal). *Let $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0,1\}$ be a client-server functionality. If f is full-dimensional, then it can be computed with perfect full-security in the OT-hybrid model in a single round. Furthermore, the number of OT calls is $O(\text{poly}(|\mathcal{Y}|))$.*

In fact, we generalize the above theorem, and we give a similar criterion for randomized non-Boolean functions. The class of functions that our protocol can compute can be further extended by letting the client have inputs that fix the output. This class of functions includes many interesting examples, such as Yao's "millionaires' problem" (the "greater-than" function). Here the parties have inputs that ranges from 1 to n, and the output of the client is 1 if and only if its input is greater than or equal to the server's input. The communication complexity of our protocol is polynomial in the client's domain size, and not in its input's size. For functions with small domain, however, this does improve upon known construction that achieve statistical security (e.g., the single round protocol by Ishai et al. [19], see Sect. 7 for more details).

Its was proven by [2], that the number of full-dimensional functions tends to 1 exponentially fast as $|\mathcal{X}|$ and $|\mathcal{Y}|$ grow. Specifically, a random function with domains sizes $|\mathcal{X}| = m+1$ and $|\mathcal{Y}| = m$, will be full-dimensional with probability

[1] A polytope is a generalization in any number of dimensions of the two-dimensional polygon and the three-dimensional polyhedron.

[2] Observe that if f is full-dimensional then $|\mathcal{X}| > |\mathcal{Y}|$, since the polytope requires at least $|\mathcal{Y}| + 1$ points to be of dimension $|\mathcal{Y}|$.

at least $1-p_m$, where p_m denotes the probability that a random Boolean $m \times m$ matrix is singular. The value p_m is conjectured to be $(1/2+o(1))^m$. Currently, the best known upper bound is $(1/\sqrt{2}+o(1))^m$ proved by [31].

Theorem 1 identifies a set of client-server functionalities that are computable with perfect full-security. It does not yield a full characterization of such functions. For example, the status of the equality function $\mathsf{3EQ} : \{x_1, x_2, x_3\} \times \{y_1, y_2, y_3\} \mapsto \{0,1\}$, defined as $\mathsf{3EQ}(x,y) = 1$ if and only if $x = y$, is currently unknown. However, for the case of Boolean functions (even randomized), we are able to show that the protocol suggested in the proof of Theorem 1 computes *only* full-dimensional functions.

1.2 Our Techniques

The protocol we suggest is a variation of the protocol of Ishai et al. [19]. Viewing the protocol abstractly, in addition to the computation of some related function, the server will also send (via the OT) a proof of correct behavior. The client will use the OT functionality to learn only a few random bits from the proof so that privacy is preserved. We next give a technical overview of our construction.

In our construction, we make use of perfect randomized encoding (PRE) [1]. A PRE $\hat{f}$ of a function f is a randomized function, such that for every input x and a uniformly random choice of the randomness r, it is possible to decode $\hat{f}(x;r)$ and compute $f(x)$ with no error. In addition, the output distribution of $\hat{f}$ on input x reveals no information about x except what follows from $f(x)$. For our construction, we rely on a property called decomposability. A PRE is said to be decomposable, if it can be written as $\hat{f} = \left(\hat{f}_1, \ldots, \hat{f}_n\right)$. Here, each $\hat{f}_i$ can be written as one of two vectors that depends on the i-th bit of x, i.e., we can write it as $\mathbf{v}_{i,x_i}$, where $(\mathbf{v}_{i,0}, \mathbf{v}_{i,1})$ depends on the randomness r. This definition can be viewed as the perfect version of garbled circuits [24,33].

Our starting point is the protocol of Ishai et al. [19], which will be dubbed the IKOPS protocol. It is a single round protocol in the OT-hybrid model that achieves statistical security. It allows the parties to compute a "certified OT" functionality. We next give a brief overview of the IKOPS protocol.

The main idea behind the IKOPS protocol is to have the server run an "MPC in the head" [18]. That is, the real server locally emulates the execution of a perfectly secure protocol Π with many virtual servers performing the computation, and $2m$ virtual clients, denoted $\mathsf{C}_{1,0}, \mathsf{C}_{1,1}, \ldots, \mathsf{C}_{m,0}, \mathsf{C}_{m,1}$, receiving output, where m is the number of bits in the real client's input y. The underlying protocol Π computes (and distributes among the clients) a decomposable PRE $\hat{f} = (\hat{f}_1, \ldots, \hat{f}_m)$ of f. Specifically, the input of the virtual servers' are secret sharing of the real server's input x and randomness r. The output of the virtual client $\mathsf{C}_{j,b}$ in an execution of Π, is $\hat{f}_j(b;r)$, i.e., the part of the encoding that corresponds to the j-th bit of y being equal to b.

The real client can then use OT in order to recover the correct output of the PRE and reconstruct the output $f(x,y)$. As part of the "MPC in the head" paradigm, the client and server jointly set up a watchlist (the views of some of

the virtual servers) allowing the client to check consistency between the virtual servers' views and the virtual clients' views. If there was an inconsistency, the client outputs $f(x_0, y)$ for some default value $x_0 \in \mathcal{X}$. However, it is unclear how to have the server send only *some* of the views according to the request of the client. Ishai et al. [19] handle this by letting the client get each view with some constant probability independently of the other views.

The security of the protocol as described so far can still be breached by a malicious server. By tampering with the outputs of the virtual clients, a malicious server could force the output of the real client to be $f(x, y)$ for some inputs y and force the output to be $f(x_0, y)$ for other values of y, where the choice is completely determined by the adversary. To overcome this problem, the function f is replaced with a function f' where each bit y_i is replaced with κ random bits whose XOR equals to y_i, where κ is the security parameter.[3] This modification prevents the adversary from having complete control over which inputs the client will output $f(x_0, y)$, and for which inputs it will output $f(x, y)$.

Two problems arise when trying to use the IKOPS protocol to achieve perfect security. First, a malicious client could potentially receive the views of *all virtual servers*, and as a result, it could learn the server's input. Second, with some non-zero probability, a malicious server might still be able to have the client output be $f(x_0, y)$ for some inputs y, but output $f(x, y)$ for other inputs y.

We solve the former issue, by showing how the client can request views deterministically. We would like to have the request be made using $\binom{n}{t}$-s-string-OT, where t bounds the number of corruptions allowed in Π, namely, the client asks for exactly t views. However, it is not known if implementing it in the OT-hybrid model with perfect security is even possible. Therefore, we slightly relax the security requirement, so that a malicious client will not be able to receive *more than twice* the number of views that an honest client receives. We then let the honest client ask for exactly $t/2$ of the views. The idea in constructing such a watchlist is the following. For each view of a virtual server, the real server sends (via the OT functionality) either a masking of the view, or a share of the concatenation of the maskings. That is, the server's input to the OT is $(V_i \oplus r_i, \mathbf{r}[i])$ for every view V_i of a virtual server S_i, where $\mathbf{r} = (r_1, \ldots, r_n)$ is a vector of random strings, and $\mathbf{r}[i]$ is the i-th share of $\mathbf{r}$, for some threshold secret sharing scheme with sufficiently large threshold value.[4] As a result, in each invocation of the OT, the client will be able to learn either a masked view or a share, which bounds the number of views it can receive.

To solve the second issue, it will be convenient to represent the server security requirement from a geometric point of view. To simplify the explanation in this introduction, we only focus on deterministic Boolean functions. Recall that we can view the function f as $|\mathcal{X}|$ points over $\mathbb{R}^{|\mathcal{Y}|}$, where the j-th coordinate of the i-th point is simply $f(x_i, y_j)$. Observe that all a simulator for a malicious server can do, is to send a random input according to some distribution D.

[3] This technique for eliminating selective failure attacks was previously used in [22,23].

[4] There are additional technical subtleties, however, for this informal introduction we ignore them.

The goal of the simulator is to force the distribution of the client's output to be equivalent to the distribution in the real-world. Thus, perfect simulation of a malicious server is possible if and only if there exists such distribution D over the server's inputs in the ideal-world, such that for every input $y \in \mathcal{Y}$ of the client, $\Pr_{x \leftarrow D}[f(x,y) = 1] = q_y$, where q_y is the probability the client outputs 1 in the real-world where its input is y. Since for every $y \in \mathcal{Y}$ the value $\Pr_{x \leftarrow D}[f(x,y) = 1]$ can be written as the same *convex combination* of the points $\{f(x_i, y)\}_{i=1}^{|\mathcal{X}|}$, the point $\left(\Pr_{x \leftarrow D}[f(x,y) = 1]\right)_{y \in \mathcal{Y}}$ lie inside the *convex hull* of the points of f. Thus, we can state perfect security as follows. Simulation of an adversary is possible if and only if the *vector of outputs* $\left(q_y\right)_{y \in \mathcal{Y}}$ in the real-world is in the *convex-hull* of the points in $\mathbb{R}^{|\mathcal{Y}|}$ described by f.

Now, consider the IKOPS protocol. It could be the case that the vector of outputs has different errors in each coordinate created by an adversary, and hence is not necessarily inside the convex-hull of the points of f. To fix this issue, instead of having the client output according to a default value in case of an inconsistency, the client will now pick x_0 uniformly at random, and output $f(x_0, y)$. Stated differently, it outputs according to c_y, where $\mathbf{c}$ is the center of the polytope.[5] We next (roughly) explain why this results in a perfectly secure protocol. Let p denote the probability of detecting an inconsistency (more precisely, for each y the probability p_y of detecting an inconsistency is in $[p-\varepsilon, p+\varepsilon]$, for some small ε). Further defined the matrix $M_f(x,y) = f(x,y)$ (i.e., each row of M_f describes a point in $\mathbb{R}^{|\mathcal{Y}|}$). Thus, the output vector of the client is close to the point $\mathbf{q} = p \cdot \mathbf{c} + (1-p) \cdot M_f(x, \cdot)$, give or take $\pm\varepsilon$ in each coordinate, for some small ε. If p is close to 1, this point $\mathbf{q}$ is close to $\mathbf{c}$, and since $\mathbf{c}$ is an internal point, $\mathbf{q}$ is also internal for a sufficiently small ε. Otherwise, the point $\mathbf{q}$ will be close to the boundary. As a result, it is unclear as to why perfect security holds. Here, we utilize a special property of IKOPS protocol's security. We manage to prove that ε is bounded by $p \cdot \varepsilon'$, for some small ε'. That is, ε depends on p, unlike the standard security requirement. This property allows us to prove that perfect security holds.

1.3 Related Work

In the 2PC settings, Cleve [9] showed that the functionality of coin-tossing, where the parties output the same random bit, is impossible to compute with full-security, even in the OT-hybrid model. In spite of that, in the seminal work Gordon et al. [15], and later followed by [2,3,12,25], it was discovered that in the OT-hybrid model, most two-party functionalities can be evaluated with full security by efficient protocols. In particular, [3] completes the characterization of symmetric Boolean functions (where both parties receive the same output). However, all known general protocols for such functionalities have round complexity that is super-logarithmic in the security parameter. Moreover, this was proven to be necessary for functions with embedded XOR [15].

[5] The same construction works for any other choice of a point $\mathbf{v}$ that is *strictly* inside the convex-hull of the points.

1.4 Organization

In Sect. 2 we provide some notations and definitions that we use in this work, alongside some required mathematical background. Section 3 is dedicated to expressing security in geometrical terms and the formal statement of our result. In Sects. 4 and 5 we present the proof of the main theorem. In Sect. 6 we show that the analysis of our protocol for Boolean functions is tight.Finally, in Sect. 7 we briefly discuss the efficiency of our construction.

2 Preliminaries

2.1 Notations

We use calligraphic letters to denote sets, uppercase for random variables and matrices, lowercase for values, and we use bold characters to denote vectors and points. All logarithms are in base 2. For $n \in \mathbb{N}$, let $[n] = \{1, 2 \dots n\}$. For a set $\mathcal{S}$ we write $s \leftarrow \mathcal{S}$ to indicate that s is selected uniformly at random from $\mathcal{S}$. Given a random variable (or a distribution) X, we write $x \leftarrow X$ to indicate that x is selected according to X. We use poly to denote an unspecified polynomial, and we use polylog to denote an unspecified polylogarithmic function. For a randomized function (or an algorithm) f we write $f(x)$ to denote the random variable induced by the function on input x, and write $f(x; r)$ to denote the value when the randomness of f is fixed to r.

For a vector $\mathbf{v} \in \mathbb{R}^n$, we denote its i-th component with v_i and we let $||\mathbf{v}||_\infty = \max_i |v_i|$ denote its ℓ_∞ norm. We denote by $\mathbf{1}_n$ ($\mathbf{0}_n$) the all-ones (all-zeros) vector of dimension n. A vector $\mathbf{p} \in \mathbb{R}^n$ is called a probability vector if $\mathbf{p}_i \geq 0$ for every $i \in [n]$ and $\sum_{i=1}^n p_i = 1$.

For a matrix $M \in \mathbb{R}^{n \times m}$, we let $M(i, \cdot)$ be its i-th row, we let $M(\cdot, j)$ be its j-th column, and we denote by M^T the transpose of M. For a pair of matrices $M_1 \in \mathbb{R}^{n \times m_1}, M_2 \in \mathbb{R}^{n \times m_2}$, we denote by $[M_1||M_2]$ the concatenation of M_2 to the right of M_1.

2.2 Cryptographic Tools

Definition 1. *The statistical distance between two finite random variables X and Y is*

$$\mathrm{SD}(X, Y) = \frac{1}{2} \sum_a \left| \Pr[X = a] - \Pr[Y = a] \right|.$$

Secret Sharing Schemes. A $(t+1)$-out-of-n secret-sharing scheme is a mechanism for sharing data among a set of parties $\{\mathrm{P}_1, \dots, \mathrm{P}_n\}$, such that every set of size $t+1$ can reconstruct the secret, while any smaller set knows nothing about the secret. As a convention, for a secret s and $i \in [n]$ we let $s[i]$ be the i-th share, namely, the share received by P_i. In this work, we rely on Shamir's secret sharing scheme [29].

In a $(t+1)$-out-of-n Shamir's secret sharing scheme over a field $\mathbb{F}$, where $|\mathbb{F}| > n$, a secret $s \in \mathbb{F}$ is shared as follows: A polynomial $p(\cdot)$ of degree at most $t+1$ over $\mathbb{F}$ is picked uniformly at random, conditioned on $p(0) = s$. Each party P_i, for $1 \leq i \leq n$, receives a share $s[i] := p(i)$ (we abuse notation and let i be the element in $\mathbb{F}$ associated with P_i).

Decomposable Randomized Encoding. We recall the definition of randomized encoding [1,33]. They are known to exists unconditionally [1,16].

Definition 2 (Randomized Encoding). *Let $f : \{0,1\}^n \mapsto \mathcal{Z}$ be some function. We say that a function $\hat{f} : \{0,1\}^n \times \mathcal{R} \mapsto \mathcal{W}$ is a perfect randomized encoding (PRE) of f if the following holds.*

Correctness: *There exists a decoding algorithm* Dec *such that for every $x \in \{0,1\}^n$*

$$\Pr_{r \leftarrow \mathcal{R}}\left[\mathsf{Dec}\left(\hat{f}\left(x;r\right)\right) = f(x)\right] = 1.$$

Privacy: *There exists a randomized algorithm* Sim *such that for every $x \in \{0,1\}^n$ it holds that*

$$\mathsf{Sim}\left(f(x)\right) \equiv \hat{f}\left(x;r\right),$$

where $r \leftarrow \mathcal{R}$.

Definition 3 (Decomposable Randomized Encoding). *For every $x \in \{0,1\}^n$, we write $x = x_1, \ldots, x_n$, where x_i is the i-th bit of x. A randomized encoding $\hat{f}$ is said to be* decomposable *if it can be written as*

$$\hat{f}\left(x;r\right) = \left(\hat{f}_0\left(r\right), \hat{f}_1\left(x_1;r\right), \ldots, \hat{f}_n\left(x_n;r\right)\right),$$

where each $\hat{f}_i$, for $i \in [n]$, can be written as one of two vectors that depends on x_i, i.e., we can write it as $\mathbf{v}_{i,x_i}$, where $(\mathbf{v}_{i,0}, \mathbf{v}_{i,1})$ depends on the randomness r.

2.3 Mathematical Background

Definition 4 (Convex Combination and Convex Hull). *Let $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\} \subseteq \mathbb{R}^n$ be a set of vectors. A convex combination is a linear combination $\sum_{i=1}^{m} \alpha_i \cdot \mathbf{v}_i$ where $\sum_{i=1}^{m} \alpha_i = 1$ and $\alpha_i \geq 0$ for all $1 \leq i \leq m$. The convex hull of $\mathcal{V}$, denoted*

$$\mathbf{conv}\left(\mathcal{V}\right) = \left\{\sum_{i=1}^{m} \alpha_i \cdot \mathbf{v}_i \mid \sum_{i=1}^{m} \alpha_i = 1 \text{ and } \alpha_i \geq 0 \text{ for all } i \in [m]\right\},$$

is the set of all vectors that can be represented as a convex combination of the vectors in $\mathcal{V}$. For a matrix $M = [\mathbf{v}_1 || \ldots || \mathbf{v}_m]$ we let $\mathbf{conv}\left(M\right) = \mathbf{conv}\left(\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}\right)$.

Definition 5 (Affine Hull). *For a set of vectors* $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\} \subseteq \mathbb{R}^n$*, we define their* affine hull *to be the set*

$$\mathbf{aff}(\mathcal{V}) = \left\{ \sum_{i=1}^{m} \alpha_i \cdot \mathbf{v}_i \mid \sum_{i=1}^{m} \alpha_i = 1 \right\}.$$

For a matrix $M = [\mathbf{v}_1 || \ldots || \mathbf{v}_m]$ *we let* $\mathbf{aff}(M) = \mathbf{aff}(\{\mathbf{v}_1, \ldots, \mathbf{v}_m\})$.

Definition 6 (Affine Independence). *A set of points* $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^n$ *is said to be* affinely independent *if whenever* $\sum_{i=1}^{m} \alpha_i \cdot \mathbf{v}_i = \mathbf{0}_n$ *and* $\sum_{i=1}^{m} \alpha_i = 0$*, then* $\alpha_i = 0$ *for every* $i \in [m]$*. Observe that* $\mathbf{v}_1, \ldots, \mathbf{v}_m$ *are affinely independent if and only if* $\mathbf{v}_2 - \mathbf{v}_1, \ldots, \mathbf{v}_m - \mathbf{v}_1$ *are linearly independent.*

For a square matrix $M \in \mathbb{R}^{n \times n}$, we denote by $\det(M)$ the determinant of M, and we denote by $M_{i,j}$ the (i,j)'th cofactor of M, which is the $(n-1) \times (n-1)$ matrix obtained by removing the i'th row and j'th column of M. It is well known that:

Fact 2. *Let* $M \in \mathbb{R}^{n \times n}$ *be an invertible matrix. Then for every* $i, j \in [n]$ *it holds that* $\left|M^{-1}(i,j)\right| = \left|\det(M_{j,i}) / \det(M)\right|$.

2.4 The Model of Computation

We follow the standard *ideal vs. real* paradigm for defining security. Intuitively, the security notion is defined by describing an ideal functionality, in which both the corrupted and non-corrupted parties interact with a trusted entity. A real-world protocol is deemed secure if an adversary in the real-world cannot cause more harm than an adversary in the ideal-world. This is captured by showing that an ideal-world adversary (simulator) can simulate the full view of the real world adversary.

We focus our attention on the *client-server model*. In this model a server S holds some input x and a client C holds some input y. At the end of the interaction the client learns the output of some function of x and y, while the server learns nothing. We further restrict ourselves to allow only a *single round* of interaction between the two parties, however, as only trivial functionalities are computable in this setting, the parties interact in the $\mathcal{OT}$-hybrid model. We next formalize the interaction done in this model.

The OT Functionality. We start by formally defining the (family) of the OT functionality. The $\binom{2}{1}$-bit-OT functionality, is a two-party client-server functionality in which the server inputs a pair of bit-messages a_0 and a_1, and the client inputs a single bit b. The server receives $\perp$ and the client receives a_b. For every natural number $\ell \geq 1$, we define the functionality $\binom{2}{1}$-bit-OT^ℓ as follows. Let $\mathbf{a} = \left(a_0^i, a_1^i\right)_{i=1}^{\ell}$ and let $\mathbf{b} = (b_i)_{i=1}^{\ell}$, where $a_0^i, a_1^i, b_i \in \{0,1\}$ for every i. We let $\mathbf{a}[\mathbf{b}] := \left(a_{b_i}^i\right)_{i=1}^{\ell}$. The functionality is then defined as $(\mathbf{a}, \mathbf{b}) \mapsto (\perp, \mathbf{a}[\mathbf{b}])$. That

is, it is the equivalent to computing $\binom{2}{1}$-bit-OT ℓ times in parallel. Finally, we let $\mathcal{OT} = \left\{ \binom{2}{1}\text{-bit-OT}^{\ell} \right\}_{\ell \geq 1}$.

A generalization of $\binom{2}{1}$-bit-OT is the $\binom{n}{1}$-bit-OT functionality, which lets the client pick one out of n bits $a_1, a_2, \ldots, a_n$ supplied by the server, and on input $i \in [n]$ the client learns a_i. This can be further generalized to $\binom{n}{1}$-s-string-OT where the n bits are replaced by strings $a_1, \ldots, a_n \in \{0,1\}^s$, and this can generalized even further to $\binom{n}{k}$-s-string-OT where the input i of the client is replaced with k inputs $i_1, \ldots, i_k \in [n]$, and it receives $a_{i_1}, \ldots, a_{i_k}$.

The 1-Round $\mathcal{OT}$-Hybrid Model. We next describe the execution in the *1-round $\mathcal{OT}$-hybrid model*. In the following we fix a (possibly randomized) client-server functionality $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0, \ldots, k-1\}$. A protocol Π in the *1-round $\mathcal{OT}$-hybrid model* with security parameter κ, is a triple of randomized functions (α, β, φ). The server and client use the function α and β respectively to obtain messages to send to the OT. The client then compute some local function φ on its view to obtain an output. Formally, the computation is done as follows.

Inputs: The server S holds input $x \in \mathcal{X}$ and the client C holds input $y \in \mathcal{Y}$. In addition, both parties hold the security parameter 1^{κ}.

Parties send inputs to the OT: S samples $2\ell(\kappa)$ bits $\mathbf{a} = \alpha(x, 1^{\kappa})$, and C samples $\ell(\kappa)$ bits $\mathbf{b} = \beta(y, 1^{\kappa})$, for some $\ell(\cdot)$ determined by the protocol. S and C send $\mathbf{a}$ and $\mathbf{b}$ to the OT functionality, respectively. C then receives $\mathbf{a}[\mathbf{b}]$ from the OT.

Outputs: The server S outputs nothing, while the client C computes the local function $\varphi(y, \mathbf{b}, \mathbf{a}[\mathbf{b}], 1^{\kappa})$ and outputs its result.

We refer to the $\ell(\kappa)$ used in the protocol as the *communication complexity* (CC) of Π.

We consider an adversary $\mathcal{A}$ that controls a single party. The adversary has access to the full view of that party. We assume the adversary is malicious, that is, it may instruct the corrupted party to deviate from the protocol in any way it chooses. The adversary is non-uniform, and is given an auxiliary input aux. For simplicity we do not concern ourselves with the efficiency of the protocols or the adversaries, namely, we assume that the parties and the adversary are unbounded.

Fix inputs $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $\kappa \in \mathbb{N}$. For an adversary $\mathcal{A}$ corrupting the *server*, we let $\text{Out}^{\text{HYBRID}}_{\mathcal{A}(x,\mathsf{aux}),\Pi}(x, y, 1^{\kappa})$ denote the output of the client in a random execution of Π. For an adversary $\mathcal{A}$ corrupting the *client*, we let $\text{View}^{\text{HYBRID}}_{\mathcal{A}(y,\mathsf{aux}),\Pi}(x, y, 1^{\kappa})$ denote the adversary's view in a random execution of Π, when it corrupts the *client*. This includes its input, auxiliary input, randomness, and the output received from the OT functionality.

The Ideal Model. We now describe the interaction in the ideal model, which specifies the requirements for fully secure computation of the function f with security parameter κ. Let $\mathcal{A}$ be an adversary in the ideal-world, which is given an auxiliary input aux and corrupts one of the parties.

The Ideal Model – Full-Security

Inputs: The server S holds input $x \in \mathcal{X}$ and the client C holds input $y \in \mathcal{Y}$. The adversary is given an auxiliary input $\mathsf{aux} \in \{0,1\}^*$ and the input of the corrupted party. The trusted party T holds 1^κ.

Parties send inputs: The honest party sends its input to T. The adversary sends a value w from its domain as the input for corrupted party.

The trusted party performs computation: T selects a random string r and computes $z = f(x, w; r)$ if C is corrupted and computes $z = f(w, y; r)$ if S is corrupted. T then sends z to C (which is also given to $\mathcal{A}$ if C is corrupted).

Outputs: An honest server outputs nothing, an honest client output z, and the malicious party outputs nothing. The adversary outputs some function of its view.

Fix inputs $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $\kappa \in \mathbb{N}$. For an $\mathcal{A}$ corrupting the *server* we let $\mathrm{Out}^{\mathrm{IDEAL}}_{\mathcal{A}(x,\mathsf{aux}),f}(x, y, 1^\kappa)$ denote the output of the client in a random execution of the above ideal-world process. For an $\mathcal{A}$ corrupting the *client* we let $\mathrm{View}^{\mathrm{IDEAL}}_{\mathcal{A}(y,\mathsf{aux}),f}(x, y, 1^\kappa)$ be the view description being the *output* of $\mathcal{A}$ in such a process.

We next present the definition for security against malicious adversaries. The definition we present is tailored to the setting of the 1-round two-party client-server in the $\mathcal{OT}$-hybrid model.

Definition 7 (malicious security). *Let $\Pi = (\alpha, \beta, \varphi)$ be a protocol for computing f in the 1-round $\mathcal{OT}$-hybrid model. Let $\varepsilon(\cdot)$ be a positive function of the security parameter.*

1. **Correctness:** We say that Π is *correct* if for all $\kappa \in \mathbb{N}$, $x \in \mathcal{X}$, and $y \in \mathcal{Y}$

$$\Pr\left[\varphi(y, \mathbf{b}, \mathbf{a}[\mathbf{b}], 1^\kappa) = f(x, y)\right] = 1.$$

   Here, $\mathbf{a} = \alpha(x, 1^\kappa)$, $\mathbf{b} = \beta(y, 1^\kappa)$ and the probability is taken over the random coins of α, β, φ, and f.
2. **Server Security:** We say that Π is ε-*server secure*, if for any non-uniform adversary $\mathcal{A}$ corrupting the server in the $\mathcal{OT}$-hybrid world, there exists a non-uniform adversary $\mathsf{Sim}_{\mathcal{A}}$ (called the simulator) corrupting the server in the ideal-world, such that for all $\kappa \in \mathbb{N}$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $\mathsf{aux} \in \{0,1\}^*$ it holds that

$$\mathrm{SD}\left(\mathrm{Out}^{\mathrm{HYBRID}}_{\mathcal{A}(x,\mathsf{aux}),\Pi}(x, y, 1^\kappa),\ \mathrm{Out}^{\mathrm{IDEAL}}_{\mathsf{Sim}_{\mathcal{A}}(x,\mathsf{aux}),f}(x, y, 1^\kappa)\right) \leq \varepsilon(\kappa).$$

   We say that Π has perfect server security if it is 0-server secure.
3. **Client Security:** We say that Π is ε-*client secure*, if for any non-uniform adversary $\mathcal{A}$ corrupting the client in the $\mathcal{OT}$-hybrid world, there exists a non-uniform simulator $\mathsf{Sim}_{\mathcal{A}}$ corrupting the client in the ideal-world, such that for all $\kappa \in \mathbb{N}$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $\mathsf{aux} \in \{0,1\}^*$ it holds that

$$\mathrm{SD}\left(\mathrm{View}^{\mathrm{HYBRID}}_{\mathcal{A}(y,\mathsf{aux}),\Pi}(x, y, 1^\kappa),\ \mathrm{View}^{\mathrm{IDEAL}}_{\mathsf{Sim}_{\mathcal{A}}(y,\mathsf{aux}),f}(x, y, 1^\kappa)\right) \leq \varepsilon(\kappa).$$

   We say that Π has perfect client security if it is 0-client secure.

We say that Π computes f with ε-statistical full-security*, if Π is correct, is ε-server secure, and is ε-client secure. Finally, we say that Π computes f with* perfect full-security*, if it computes f with* 0-statistical full-security*.*

To alleviate notation, from now on we will completely remove 1^κ from the input the functions α, β, and φ, and remove κ from ℓ and ε. Statistical security will now be stated as a function of ε and the CC of the protocol as a function of ℓ. Observe that aborts in this model are irrelevant. Indeed, honest server outputs nothing, and if a malicious server aborts then the client can output $f(x_0, y)$ for some default value $x_0 \in \mathcal{X}$, which can be perfectly simulated. Therefore, throughout the paper we assume without loss of generality that the adversary does not abort the execution.

We next describe the notion of *security with input-dependent abort* [19]. Generally, it is a relaxation of the standard full-security notion, which allows an adversary to learn at most 1 bit of information by causing the protocol to abort depending on the other party's inputs. We state only perfect security. Furthermore, the security notion is written with respect only to a malicious server. Since we work in the client-server model, the trusted party *does not* send to the server any output. Therefore, in this relaxation selective abort attacks [22,23] *are simulatable.*

Definition 8. *Fix $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0, \ldots, k-1\}$. In the* input-dependent *model, we modify the ideal-world so that the malicious adversary corrupting the server, in addition to sending an input $x^* \in \mathcal{X}$, also gives the trusted party T a predicate $P : \mathcal{Y} \mapsto \{0,1\}$. T then sends to the client $f(x^*, y)$ if $P(y) = 0$, and $\perp$ otherwise. We let $\mathrm{Out}^{\mathrm{ID}}_{\mathcal{A}(x,\mathsf{aux}),f}(x,y)$ denote the output of the client in a random execution of the above ideal-world process, with $\mathcal{A}$ corrupting the server.*

Let Π be a protocol that computes f in the 1-round $\mathcal{OT}$-hybrid model. We say that Π has perfect input-dependent security*, if for every non-uniform adversary $\mathcal{A}$ corrupting the server in the $\mathcal{OT}$-hybrid world, there exists a non-uniform adversary $\mathsf{Sim}_{\mathcal{A}}$ corrupting the server in the input-dependent ideal-world, such that for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $\mathsf{aux} \in \{0,1\}^*$ it holds that*

$$\mathrm{Out}^{\mathrm{HYBRID}}_{\mathcal{A}(x,\mathsf{aux}),\Pi}(x,y) \equiv \mathrm{Out}^{\mathrm{ID}}_{\mathsf{Sim}_{\mathcal{A}}(x,\mathsf{aux}),f}(x,y)\,.$$

3 A Class of Perfectly Computable Client-Server Functions

In this section, we state the main result of this paper – presenting a large class of two-party client-server functions that are computable with perfect security. We start with presenting a geometric view of security in our model. We take a similar approach to that of [2] to representing the server-security requirement geometrically.

3.1 A Geometrical Representation of the Security Requirements

Boolean Functions. We start with giving the details for (randomized) Boolean functions. For any function $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0,1\}$ we associate an $|\mathcal{X}| \times |\mathcal{Y}|$ matrix M_f defined as $M_f(x,y) = \Pr[f(x,y) = 1]$, where the probability is taken over f's random coins (if f is deterministic, then this value is Boolean). Let $\mathcal{X} = \{x_1, \ldots, x_n\}$. Observe that in the ideal-world, every strategy that is employed by a simulator corrupting the server can be encoded with a probability vector $\mathbf{p} \in \mathbb{R}^n$, where p_i corresponds to the probability of sending x_i to T. Therefore, if the input of the client is y, then the probability that the output is 1, equals to $\mathbf{p}^T \cdot M_f(\cdot, y)$. On the other hand, in the 1-round $\mathcal{OT}$-hybrid model, a malicious server can only choose a string $\mathbf{a}^* \in \{0,1\}^{2\ell}$ and send in to the OT. Then on input $y \in \mathcal{Y}$, the probability the client outputs 1 is exactly

$$q_y^{\Pi}(\mathbf{a}^*) := \Pr[\varphi(y, \mathbf{b}, \mathbf{a}^*[\mathbf{b}]) = 1],$$

where $\mathbf{b} = \beta(y)$ and the probability is over the randomness of β and φ. This implies that an ideal-world simulator must send a random input $x^* \in \mathcal{X}$ such that the client will output 1 with probability $q_y^{\Pi}(\mathbf{a}^*)$. Thus, *perfect security* holds if and only if for every $\mathbf{a}^* \in \{0,1\}^{2\ell}$ there exists a probability vector $\mathbf{p} \in \mathbb{R}^n$ such that for every $y \in \mathcal{Y}$

$$\mathbf{p}^T \cdot M_f(\cdot, y) = q_y^{\Pi}(\mathbf{a}^*).$$

Equivalently, for every $\mathbf{a}^*$ the vector $\mathbf{q}^{\Pi}(\mathbf{a}^*) := (q_y^{\Pi}(\mathbf{a}^*))_{y \in \mathcal{Y}}$ is inside the *convex-hull* of the rows of M_f. Further observe that this holds true regardless of the auxiliary input held by a corrupt server.

General Functions. We now extend the above discussion to non-Boolean functions. For every function $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0, \ldots, k-1\}$, and every possible output $z \in \{0, \ldots, k-1\}$, we associate an $|\mathcal{X}| \times |\mathcal{Y}|$ matrix M_f^z defined as $M_f^z(x,y) = \Pr[f(x,y) = z]$. Similarly to the Boolean case, in the ideal world, every strategy that is employed by a corrupt server can be encoded with a probability vector $\mathbf{p} \in \mathbb{R}^n$, hence the probability that the client will output z, on input y, is $\mathbf{p}^T \cdot M_f^z(\cdot, y)$. In the 1-round $\mathcal{OT}$-hybrid model, for a string $\mathbf{a}^* \in \{0,1\}^{2\ell}$ chosen by a malicious server, the probability to output z equals to

$$q_{y,z}^{\Pi}(\mathbf{a}^*) := \Pr[\varphi(y, \mathbf{b}, \mathbf{a}^*[\mathbf{b}]) = z],$$

where $\mathbf{b} = \beta(y)$ and the probability is over the randomness of β and φ. Therefore, perfect security holds if and only if for every $\mathbf{a}^* \in \{0,1\}^{2\ell}$ there exists a probability vector $\mathbf{p} \in \mathbb{R}^n$ such that for every $y \in \mathcal{Y}$ and for every $z \in \{0, \ldots, k-1\}$

$$\mathbf{p}^T \cdot M_f^z(\cdot, y) = q_{y,z}^{\Pi}(\mathbf{a}^*). \tag{1}$$

Observe that since $\mathbf{p}$ is a probability vector and since $\sum_z M_f^z$ is the all-one matrix, it is equivalent to consider only $k-1$ possible values for z instead of all k values considered in Eq. (1). We next write the perfect security formulation more succinctly.

Let $M_f = \left[M_f^1 || \ldots || M_f^{k-1}\right]$ be the concatenation of the matrices by columns, and let $\mathbf{q}^{\Pi}(\mathbf{a}^*) := \left((q_{y,z}^{\Pi}(\mathbf{a}^*))_{y\in\mathcal{Y}}\right)_{z\in[k-1]}$. Then Eq. (1) is equivalent to saying that for every $\mathbf{a}^*$ the vector $\mathbf{q}^{\Pi}(\mathbf{a}^*)$ belongs to the *convex-hull* of the rows of M_f. It will be convenient to index the columns of M_f with (y,z), i.e., we let $M_f(x,(y,z)) = M_f^z(x,y)$.[6] We now have an equivalent definition of perfect server security.

Lemma 1. *Let Π be a protocol for computing some function $f : \mathcal{X}\times\mathcal{Y} \mapsto \{0,\ldots,k-1\}$ in the 1-round $\mathcal{OT}$-hybrid model with* CC *of ℓ. Then Π has perfect server security if and only if for every $\mathbf{a}^* \in \{0,1\}^{2\ell}$ it holds that*

$$\mathbf{q}^{\Pi}(\mathbf{a}^*) \in \mathbf{conv}\left(M_f^T\right).$$

We next describe another for security against a corrupt server. Intuitively, it states that for a malicious server, the *less* it deviates from the prescribed protocol, the *better* it can be simulated. Moreover, instead of using the traditional ℓ_1 distance (i.e., statistical distance) we phrase the security in terms of the ℓ_∞ norm. This, somewhat non-standard definition will later act as a sufficient condition for reducing perfect server-security to perfect client-security.

Definition 9. *Let $f : (\mathcal{X}\cup\{\perp\})\times\mathcal{Y} \mapsto \{\perp,0,\ldots,k-1\}$. Assume that $f(x,y)=\perp$ if and only if $x=\perp$. Let $\Pi=(\alpha,\beta,\varphi)$ be a protocol for computing f in the 1-round $\mathcal{OT}$-hybrid model. We say that Π is* strong ε-server secure[7] *if the following holds. For every message $\mathbf{a}^*$ sent by a malicious server in the $\mathcal{OT}$-hybrid world, there exists a probability vector $\mathbf{p} = (p_x)_{x\in(\mathcal{X}\cup\{\perp\})} \in \mathbb{R}^{|\mathcal{X}|+1}$ such that*

$$\left|\left|\mathbf{q}^{\Pi}(\mathbf{a}^*) - M_f^T\cdot\mathbf{p}\right|\right|_\infty \le \varepsilon\cdot p_\perp.$$

3.2 Stating the Main Result

With the above representation in mind, we are now ready to state our main result. We first recall the definition of a full-dimensional function, as stated in [2].

[6] We may view the above presentation differently. We can apply the presentation discussed for *Boolean* functions, to the function $f' : \mathcal{X}\times(\mathcal{Y}\times[k-1]) \mapsto \{0,1\}$, defined as $f'(x,(y,z)) = \Pr[f(x,y)=z]$.

[7] Although this definition as stated is not actually stronger than the standard server security definition, we decide to keep this name because of the intuition behind it. Furthermore, stating simulation error with respect to the ℓ_1 norm instead of the ℓ_∞ norm, is in fact stronger.

Definition 10 (full-dimensional function). *We say that a function* $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0, \ldots, k-1\}$ *is* full-dimensional *if*

$$\dim\left(\mathbf{aff}\left(M_f^T\right)\right) = (k-1) \cdot |\mathcal{Y}|,$$

namely, the affine-hull defined by the rows of M_f *spans the entire vector space.*

Recall that a basis for an affine space of dimension n has cardinality $n+1$, and therefore it must holds that $|\mathcal{X}| > (k-1) \cdot |\mathcal{Y}|$. Thus, the assumption that f is full-dimensional implies this condition. We are now ready to state our main result.

Theorem 3. *Let* $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0, \ldots, k-1\}$ *be a full-dimensional function. Then there exists a protocol* Π *in the 1-round* $\mathcal{OT}$*-hybrid model, that computes* f *with perfect full-security. Furthermore, if* f *is deterministic the* CC *is the following. Let* γ_i *denote the size of the smallest formula for evaluating the* i*'th bit of* $f(x, y)$*, and let* $\gamma = \max_i \gamma_i$*. Then* Π *has* CC *at most*

$$\xi \cdot \gamma^2 \cdot \log k \cdot \log |\mathcal{Y}| \cdot \mathrm{poly}\left(k \cdot |\mathcal{Y}|\right),$$

where $\xi \in \mathbb{R}^+$ *is some global constant independent of the function* f*.*

Although the communication complexity of our protocol is roughly $\mathrm{poly}\left(k \cdot |\mathcal{Y}|\right)$, for functions with small client-domain, it does yield a concrete improvement upon known protocols such as the protocol proposed by [19].

A simple corollary of Theorem 3 is that adding constant columns to a full-dimensional function, results in a functions that can still be computed with perfect security.

Corollary 1. *Let* $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0, \ldots, k-1\}$ *be some function. Assume that there exists a subset* $\mathcal{Y}' \subseteq \mathcal{Y}$ *that fixes the output distribution of* f*, i.e., for all* $y \in \mathcal{Y}'$ *there exists a distribution* D_y *over* $\{0, \ldots, k-1\}$ *such that* $f(x, y) \equiv D_y$ *for every* $x \in \mathcal{X}$*. Then if the function* $f' : \mathcal{X} \times (\mathcal{Y} \setminus \mathcal{Y}') \mapsto \{0, \ldots, k-1\}$*, defined as* $f'(x, y) = f(x, y)$*, is full-dimensional, then* f *can be computed the 1-round* $\mathcal{OT}$*-hybrid model with perfect full-security and with the same communication complexity as* f'*.*

Many interesting examples of functionalities that satisfy the constraints in Theorem 3 and Corollary 1 exists. Yao's millionaires' problem is an example for such a function. Here, the server and the client each hold a number from 1 to n. The output is 1 if and only if the client's input is greater than or equal to the server's input. The matrix for this function has a constant column of 1's (when taking the client's input to be n). After removing it, the last row of the matrix will be the all 0 vector, and the other rows are linearly independent. Therefore the function satisfies the constraints in Corollary 1.

Theorem 3 clearly follows from the following two lemmata. The first lemma reduces the problem of constructing a perfectly secure protocol, to the task of constructing a protocol with perfect client security and strong statistical server security. The second lemma states that such a protocol exists.

Lemma 2. *Let $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0, \ldots, k-1\}$ be some function. Define the function $g : (\mathcal{X} \cup \{\perp\}) \times \mathcal{Y} \mapsto \{\perp, 0, \ldots, k-1\}$ as $g(x, y) = f(x, y)$ if $x \neq \perp$ and $g(\perp, y) = \perp$, for every $y \in \mathcal{Y}$. Assume that for every $\varepsilon > 0$, there exists a protocol $\Pi_g(\varepsilon)$ in the 1-round $\mathcal{OT}$-hybrid model that computes g with correctness, is strong ε-server secure, has perfect client security, and has* CC *at most $\ell(\varepsilon, |\mathcal{X}|, |\mathcal{Y}|, k)$. Then, if f is full-dimensional, there exists a protocol Π_f in the 1-round $\mathcal{OT}$-hybrid model, that computes f with perfect full-security. Moreover, if f is deterministic then Π_f has* CC *at most*

$$\ell\left(\frac{1}{2n(n+1)!},\ |\mathcal{X}|,\ |\mathcal{Y}|,\ k\right),$$

where $n = (k-1) \cdot |\mathcal{Y}|$.

Lemma 3. *Let $g : (\mathcal{X} \cup \{\perp\}) \times \mathcal{Y} \mapsto \{\perp, 0, \ldots, k-1\}$ be a function such that $g(x, y) = \perp$ if and only if $x = \perp$. Then for every $\varepsilon > 0$, there exists a protocol $\Pi_g(\varepsilon)$ in the 1-round $\mathcal{OT}$-hybrid model that computes g with correctness, is strong ε-server secure, and has perfect client security. Furthermore, its communication complexity is the following. Let γ_i denote the size of the smallest formula for evaluating the i-th bit of $g(x, y)$, and let $\gamma = \max_i \gamma_i$. Then $\Pi_g(\varepsilon)$ has* CC *at most*

$$\xi \cdot \gamma^2 \cdot \log k \cdot \log |\mathcal{Y}| \cdot \mathrm{polylog}\left(\varepsilon^{-1}\right),$$

where $\xi \in \mathbb{R}^+$ is some global constant independent of the function g and of ε.

We prove Lemma 2 in Sect. 4 and we prove Lemma 3 in Sect. 5.

4 Proof of Lemma 2

In this section, we reduce the problem of constructing a perfectly secure protocol, to the problem of constructing a protocol that has perfect client security and has strong statistical server security. The idea is to wrap the given protocol for computing g. Whenever the output of $\Pi_g(\varepsilon)$ is $\perp$ (for small enough ε), the client will choose $x_0 \in \mathcal{X}$ at random and output $f(x_0, y)$. Stated from a geometric point of view, the client outputs according to a distribution that is consistent with some point that is *strictly inside* the convex-hull of the rows of M_f (e.g., the center).

Proof (of Lemma 2). It is easy to see that if the probability that the output of $\Pi_g(\varepsilon)$ equals $\perp$ is 0 for every $y \in \mathcal{Y}$ for some $\varepsilon > 0$, then $\Pi_g(\varepsilon)$ computes f with perfect security.

Assume otherwise. Let $n = (k-1) \cdot |\mathcal{Y}|$. Since f is full-dimensional there exists a subset $\mathcal{S} = \{\mathbf{x}_0, \ldots, \mathbf{x}_n\} \subseteq \mathbb{R}^n$ of the rows of M_f, that is affinely independent. Let $\mathbf{u}_\mathcal{S} \in \mathbb{R}^n$ be the vector associated with uniform distribution over $\mathcal{S}$ (i.e., $u_i = 1/|\mathcal{S}|$ if $i \in \mathcal{S}$ and $u_i = 0$ otherwise), and let $\mathbf{c} = (c_{y,z})_{y \in \mathcal{Y}, z \in [k-1]} := M_f^T \cdot \mathbf{u}_\mathcal{S}$ be the center of the simplex[8] defined by the points in $\mathcal{S}$. The protocol Π_f is described as follows.

[8] A simplex is the convex-hull of an affinely independent set of points.

Protocol 4 (Π_f)

Input: Server S *has input* $x \in \mathcal{X}$ *and client* C *has input* $y \in \mathcal{Y}$.

1. *The parties execute protocol* $\Pi_g(\varepsilon)$ *with small enough* $\varepsilon > 0$ *to be determined by the analysis. Let* z *be the output* C *receive.*
2. *If* $z \neq \perp$, *then* C *output* z. *Otherwise, output* $z' \in [k-1]$ *with probability* $c_{y,z'}$ *(and output 0 with the complement probability).*

Correctness and perfect client-security follows from the fact that Π_g satisfies these properties. It remains to show that perfect server-security holds. By Lemma 1, it suffices to show that for every $\mathbf{a}^* \in \{0,1\}^{2\ell}$ sent to the OT by a malicious server, it holds that

$$\mathbf{q}^{\Pi_f}(\mathbf{a}^*) \in \mathbf{conv}\left(M_f^T\right). \tag{2}$$

Fix $\mathbf{a}^* \in \{0,1\}^{2\ell}$. For brevity, we write $\mathbf{q}^f$ and $\mathbf{q}^g$ instead of $\mathbf{q}^{\Pi_f}(\mathbf{a}^*)$ and $\mathbf{q}^{\Pi_g(\varepsilon)}(\mathbf{a}^*)$ respectively. Since $\Pi_g(\varepsilon)$ is strong ε-server secure, it follows that there exists a probability vector $\mathbf{p}^g \in \mathbb{R}^{|\mathcal{X}|+1}$ such that

$$\mathbf{q}^g = M_g^T \cdot \mathbf{p}^g + \mathbf{err}, \tag{3}$$

where $\mathbf{err} \in \mathbb{R}^{k\cdot|\mathcal{Y}|}$ satisfies $||\mathbf{err}||_\infty \leq \varepsilon \cdot p_\perp^g$. Let $\overline{\mathbf{p}}^g = (p_x^g)_{x\in\mathcal{X}}$ be the vector $\mathbf{p}$ with $p_\perp$ removed. We first show that Eq. (2) follows from the following two claims.

Claim 5. *There exists a vector* $\widehat{\mathbf{err}} \in \mathbb{R}^{k\cdot|\mathcal{Y}|}$ *satisfying* $||\widehat{\mathbf{err}}||_\infty \leq 2\varepsilon$, *such that*

$$\mathbf{q}^f = M_f^T \cdot \overline{\mathbf{p}}^g + p_\perp \cdot (\mathbf{c} + \widehat{\mathbf{err}}).$$

Claim 6. *There exists a small enough* $\varepsilon > 0$ *such that*

$$\mathbf{c} + \widehat{\mathbf{err}} \in \mathbf{conv}\left(M_f^T\right),$$

where $\widehat{\mathbf{err}}$ *is the same as in Claim 5.*

Indeed, by Claim 6 there exists a probability vector $\widehat{\mathbf{p}} \in \mathbb{R}^{|\mathcal{X}|}$ such that

$$\mathbf{c} + \widehat{\mathbf{err}} = M_f^T \cdot \widehat{\mathbf{p}}.$$

Thus, by Claim 5

$$\mathbf{q}^f = M_f^T \cdot \overline{\mathbf{p}}^g + p_\perp \cdot (\mathbf{c} + \widehat{\mathbf{err}}) = M_f^T \cdot (\overline{\mathbf{p}}^g + p_\perp \cdot \widehat{\mathbf{p}}).$$

Recall that the entries of $\overline{\mathbf{p}}$ sum up to $1 - p_\perp$. Therefore $\overline{\mathbf{p}}^g + p_\perp \cdot \widehat{\mathbf{p}}$ is a probability vector, hence Eq. (2) holds.

To conclude the proof, we next prove Claims 5 and 6.

Proof (of Claim 5). Let $\mathbf{err}' = \frac{1}{p_\perp} \cdot \mathbf{err}$. Observe that for every $y \in \mathcal{Y}$ and $z \in [k-1]$ it holds that

$$\begin{aligned} q^f_{y,z} &= q^g_{y,z} + q^g_{y,\perp} \cdot c_{y,z} \\ &= M_g^T(\cdot,(y,z)) \cdot \mathbf{p}^g + \mathrm{err}_{y,z} + \left(M_g^T(\cdot,(y,\perp)) \cdot \mathbf{p}^g + \mathrm{err}_{y,\perp}\right) \cdot c_{y,z} \\ &= M_f^T(\cdot,(y,z)) \cdot \overline{\mathbf{p}}^g + \mathrm{err}_{y,z} + (p_\perp + \mathrm{err}_{y,\perp}) \cdot c_{y,z} \\ &= M_f^T(\cdot,(y,z)) \cdot \overline{\mathbf{p}}^g + p_\perp \cdot (c_{y,z} + \mathrm{err}'_{y,z} + \mathrm{err}'_{y,\perp} \cdot c_{y,z}), \end{aligned}$$

where the first equality is by the description of Π_f, the second is by Eq. (3), and the third follows from the definition of g. Define the vector $\widehat{\mathbf{err}}$ as follows. For every $y \in \mathcal{Y}$ and $z \in [k-1]$ let $\widehat{\mathrm{err}}_{y,z} = \mathrm{err}'_{y,z} + \mathrm{err}'_{y,\perp} \cdot c_{y,z}$. Then

$$\mathbf{q}^f = M_f^T \cdot \overline{\mathbf{p}}^g + p_\perp \cdot (\mathbf{c} + \widehat{\mathbf{err}}).$$

To conclude the proof, we upper-bound $||\widehat{\mathbf{err}}||_\infty$. It holds that

$$||\widehat{\mathbf{err}}||_\infty \leq ||\mathbf{err}'||_\infty \cdot (1 + ||\mathbf{c}||_\infty) = \frac{1}{p_\perp} \cdot ||\mathbf{err}||_\infty \cdot (1 + ||\mathbf{c}||_\infty) \leq 2\varepsilon.$$

Proof (of Claim 6). One approach would be to use similar techniques as in [2], namely, take a "small enough" Euclidean ball around $\mathbf{c}$ and take ε to be small enough so that $\mathbf{c} + \widehat{\mathbf{err}}$ is contained inside the ball. This approach, however, only proves the existence of such an ε. We take a slightly different approach, which would also provide an explicit upper bound on ε for deterministic functions.

For every $i \in [n]$ let $\overline{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{x}_0$, let $\overline{\mathcal{S}} = \{\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n\}$ be a basis for $\mathbb{R}^n$, and let $A = [\overline{\mathbf{x}}_1 || \ldots || \overline{\mathbf{x}}_n]$ be the corresponding change of basis matrix. Then

$$\mathbf{c} = M_f^T \cdot \mathbf{u}_{\mathcal{S}} = \sum_{i=0}^{n} \frac{1}{n+1} \cdot \mathbf{x}_i = \mathbf{x}_0 + \sum_{i=1}^{n} \frac{1}{n+1} \cdot \overline{\mathbf{x}}_i = \mathbf{x}_0 + \frac{1}{n+1} \cdot A \cdot \mathbf{1}_n. \quad (4)$$

Observe that a point $\mathbf{v}$ is in the convex-hull of $\mathcal{S}$ if and only if it can be written as $\mathbf{x}_0 + \sum_{i=1}^{n} p_i \cdot \overline{\mathbf{x}}_i$, where the p_i's are non-negative real numbers that sum up to *at most* 1. Indeed, we can write

$$\mathbf{x}_0 + \sum_{i=1}^{n} p_i \cdot \overline{\mathbf{x}}_i = \left(1 - \sum_{i=1}^{n} p_i\right) \cdot \mathbf{x}_0 + \sum_{i=1}^{n} p_i \cdot \mathbf{x}_i.$$

Next, as $\overline{\mathcal{S}}$ forms a basis, there exists a vector $\widetilde{\mathbf{err}} \in \mathbb{R}^n$ such that $\widehat{\mathbf{err}} = A \cdot \widetilde{\mathbf{err}}$. Then, if $||\widehat{\mathbf{err}}||_\infty \leq \frac{1}{n(n+1)}$, by Eq. (4) it follows that

$$\mathbf{c} + \widehat{\mathbf{err}} = \mathbf{x}_0 + A \cdot \left(\frac{1}{n+1} \cdot \mathbf{1}_n + \widetilde{\mathbf{err}}\right) = \mathbf{x}_0 + \sum_{i=1}^{n} p_i \cdot \overline{\mathbf{x}}_i,$$

where $0 \leq p_i \leq 1/n$ for every $i \in [n]$, implying that the point is inside $\mathbf{conv}(\mathcal{S})$. Thus, it suffices to find ε for which $||\widetilde{\mathbf{err}}||_\infty \leq \frac{1}{n(n+1)}$. It holds that

$$\begin{aligned}
||\widetilde{\mathbf{err}}||_\infty &= \left|\left|A^{-1} \cdot \widehat{\mathbf{err}}\right|\right|_\infty \\
&= \max_{i\in[n]} \left\{\left|A^{-1}(i,\cdot) \cdot \widehat{\mathbf{err}}\right|\right\} \\
&\leq \max_{i\in[n]} \left\{\sum_{j=1}^{n} \left|A^{-1}(i,j) \cdot \widehat{\mathrm{err}}_j\right|\right\} \\
&= \max_{i\in[n]} \left\{\sum_{j=1}^{n} \left|\frac{\det(A_{j,i})}{\det(A)}\right| \cdot |\widehat{\mathrm{err}}_j|\right\} \\
&\leq n \cdot \frac{(n-1)!}{|\det(A)|} \cdot 2\varepsilon \\
&= \frac{2n!}{|\det(A)|} \cdot \varepsilon,
\end{aligned}$$

where the third equality is by Fact 2, and the second inequality is due to the fact that each entry in A is a real number between -1 and 1. Therefore, by taking $\varepsilon = \frac{|\det(A)|}{2n(n+1)!}$ the claim will follow. Observe that if the function f is deterministic, then the entries of A are in $\{-1, 1\}$ implying that $|\det(A)| \geq 1$, and hence taking $\varepsilon = \frac{1}{2n(n+1)!}$ suffices. Therefore the communication complexity will be at most $\ell\left(\frac{1}{2n(n+1)!},\ |\mathcal{X}|,\ |\mathcal{Y}|,\ k\right)$ in this case.

5 Proof of Lemma 3

In this section we fix a function $g : (\mathcal{X} \cup \{\perp\}) \times \mathcal{Y} \mapsto \{\perp, 0, \ldots, k-1\}$ satisfying $g(x, y) = \perp$ if and only if $x = \perp$. We show how to construct a protocol for computing the function g in the 1-round $\mathcal{OT}$-hybrid model. The protocol we construct has perfect client security, and has strong statistical server security. Our protocol is a modified version of the protocol by Ishai et al. [19], which we shall next give an overview of. Their protocol is parametrized with ε, and we denote this protocol by $\Pi_{\mathrm{IKOPS}}(\varepsilon)$. It is a single round protocol in the $\mathcal{OT}$-hybrid model, that has ε-statistical full-security. It is stated for functions computable by NC^1 circuits, however, this is only done for improving concrete efficiency, which is not a concern in our paper. We therefore restate it for general functions, and bound its communication complexity as a function of $|\mathcal{X}|$, $|\mathcal{Y}|$, and k (which are assumed to be finite in our work).

5.1 The Protocol Π_{IKOPS}The IKOPS Protocol

We next give the rough idea of Π_{IKOPS}. First, we view the inputs x and y as a binary strings.[9] The parties will compute a "certified OT" functionality. We next give a brief overview of the IKOPS protocol.

The main idea behind the Π_{IKOPS} is to have the server run an "MPC in the head" [18]. That is, the real server locally emulates the execution of a perfectly secure protocol Π with many virtual servers performing the computation, and $2m$ virtual clients, denoted $\mathsf{C}_{1,0}, \mathsf{C}_{1,1}, \ldots, \mathsf{C}_{m,0}, \mathsf{C}_{m,1}$, receiving output, where m is the number of bits in the client's input y. The underlying protocol Π computes a decomposable PRE $\hat{g} = (\hat{g}_0, \hat{g}_1, \ldots, \hat{g}_m)$ of g. Specifically, the output of client $\mathsf{C}_{j,b}$ in an execution of Π is the corresponds to the j-th bit of y, when the bit equals to b.

The real client can then use OT in order to recover the correct output of the PRE and reconstruct the output $g(x, y)$. As part of the "MPC in the head" paradigm, the client further ask the server to send a watchlist (the views of some of the virtual servers) and check consistency. If there was an inconsistency, then the client outputs $\perp$. To make sure that the client will not receive too large of a watchlist and break the privacy requirement, it will get each view with some (constant) probability independently of the other views.

Observe that although the client can use OT in order to receive the correct output from the virtual clients, the two real parties need to use string-OT, while they only have access to bit-OT. This technicality can be overcome using the perfect reduction from $\binom{n}{1}$-s-string-OT to OT that was put forward in the elegant work of Brassard et al. [8], which also constitutes one of the few examples of perfect reductions to $\binom{2}{1}$-bit-OT known so far. They proved the following theorem.

Theorem 7. *There exists a protocol $\Pi_{\text{BCS}} = (\alpha_{\text{BCS}}, \beta_{\text{BCS}}, \varphi_{\text{BCS}})$ in the 1-round $\mathcal{OT}$-hybrid world that computes $\binom{n}{1}$-s-string-OT with perfect full-security. Furthermore, its communication complexity is at most $5s(n-1)$.*

The security of the protocol described so far can still be breached by a malicious server. By tampering with the outputs of the virtual clients, a malicious server could force the output of the real client to be $g(x, y)$ for some inputs y and force the output to be $\perp$ for other values of y, where the choice is *completely determined* by the adversary. To overcome this problem, we replace g with a function g' where each bit y_i is replaced with m' random bit whose XOR equals

[9] We can assume without loss of generality that the size of $\mathcal{X}$ and $\mathcal{Y}$ are a power of 2. This is due to the fact that we can add new elements to $\mathcal{X}$ such that the new rows in M_f are duplicates of existing rows. We then do the same to $\mathcal{Y}$. It is easy to see that the new function is computable with statistical (perfect) full-security if and only if the previous function is computable with statistical (perfect) full-security.

to y_i, for some large m'.[10] Here, the adversary does not have complete control over which inputs the client will output $\perp$, and for which inputs it will output $g(x, y)$. We next describe the protocol formally. We start with some notations.

Notation. Throughout the following section, client's input are now binary strings $\mathbf{y}$ of length m. Let $m' = m'(\varepsilon) = \left\lceil \log\left(\varepsilon^{-1}\right)\right\rceil + 1$ and let $\mathsf{Enc} : \{0,1\}^m \mapsto \left(\{0,1\}^{m'}\right)^m$ be a randomized function that on input m bits $y_1, \ldots, y_m$, outputs $m \cdot m'$ random bits $\left(y_i^1, \ldots, y_i^{m'}\right)_{i\in[m]}$ conditioned on $\oplus_{j=1}^{m'} y_i^j = y_i$ for every $i \in [m]$. We also let $\mathsf{Dec} : \left(\{0,1\}^{m'}\right)^m \mapsto \{0,1\}^m$ be the inverse of Enc, namely,

$$\mathsf{Dec}\left(\left(y_i^1, \ldots, y_i^{m'}\right)_{i\in[m]}\right) = \left(y_i^1 \oplus \ldots \oplus y_i^{m'}\right)_{i\in[m]}.$$

Finally, we let $g' : (\mathcal{X} \cup \{\perp\}) \times \left(\{0,1\}^{m'}\right)^m \mapsto \{\perp, 0, \ldots, k-1\}$ be defined as

$$g'\left(x, \left(y_i^1, \ldots, y_i^{m'}\right)_{i\in[m]}\right) = g\left(x, \mathsf{Dec}\left(\left(y_i^1, \ldots, y_i^{m'}\right)_{i\in[m]}\right)\right),$$

and let $\hat{g}$ be a decomposable PRE of g'.

Protocol 8 ($\Pi_{\mathrm{IKOPS}}(\varepsilon)$)

Input: Server has input $x \in (\mathcal{X} \cup \{\perp\})$ and client has input $\mathbf{y} \in \{0,1\}^m$.

– $\alpha(x)$:

1. *The server S runs "MPC in the head" for the following functionality. There are $n = \Theta\left(\log\left(\varepsilon^{-1}\right)\right)$ virtual servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$ with inputs and $2m \cdot m'$ virtual clients $\mathsf{C}_{1,0}, \mathsf{C}_{1,1}, \ldots, \mathsf{C}_{m\cdot m',0}, \mathsf{C}_{m\cdot m',1}$ receiving outputs. Each virtual server holds a share of the S's input and randomness, where the shares are in an n-out-of-n secret sharing scheme. Each virtual client $\mathsf{C}_{j,b}$ will receive $\hat{g}_{j,b}(x)$, namely, it will receive the (j,b)-th component of the decomposable PRE where the first part of the input is fixed to x. In addition every virtual client will hold $\hat{g}_0(x)$ which is the value of $\hat{g}$ that depends only on x and the randomness.*
2. *The virtual parties execute a multiparty protocol in order to compute $\hat{g}$. The protocol used has perfect full-security against $t = \lceil n/3\rceil - 1$ corrupted virtual servers and any number of corrupted virtual clients. We also assume that the virtual clients receive messages at the last round of the protocol. (e.g., the BGW protocol [7]).*

[10] This method has the disadvantage of increasing the length of the client's input and as a result increase the communication complexity, so [19] suggested a different approach. We stick with the presented approach, as we prefer simplicity over concrete efficiency.

3. *Let* $V_{j,b}$ *be the view of* $\mathsf{C}_{j,b}$*, and let* $\mathbf{a}_1 = (\alpha_{\mathrm{BCS}}(V_{j,0}, V_{j,1}))_{j\in[m\cdot m']}$.
4. *Let* V_i *be the view of* S_i*. For each* $i \in [n]$ *the server creates* $\tilde{\mathbf{a}}_i$ *of length* $\lceil 2n/t \rceil$*, where* V_i *is located in a randomly chosen entry, while the other entries are* $\perp$ *(this allows the server to send each* V_i *with probability* $t/2n$*). Let* $\mathbf{a}_2 = (\alpha_{\mathrm{BCS}}(\tilde{\mathbf{a}}_i))_{i\in[n]}$.
5. *Output* $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2)$.

– $\beta(\mathbf{y})$:

1. *The client computes* $\left(y_i^1, \ldots, y_i^{m'}\right)_{i\in[m]} = \mathsf{Enc}(\mathbf{y})$.
2. *Let* $\mathbf{b}_1 = \left(\beta_{\mathrm{BCS}}\left(y_j^{j'}\right)\right)_{j\in[m],j\in[m']}$.
3. *Let* $\mathbf{b}_2 = (\beta_{\mathrm{BCS}}(1))_{i\in[\lceil 2n/t\rceil]}$ *(i.e., a constant vector of length* $\lceil 2n/t \rceil$*).*
4. *Output* $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2)$.

– $\varphi(\mathbf{y}, \mathbf{b}, \mathbf{c}')$:

1. *Let* $\mathbf{c} = (\varphi_{\mathrm{BCS}}(c_i'))_i$[11]*. Write* $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$*, where* $\mathbf{c}_1$ *corresponds to the outputs and* $\mathbf{c}_2$ *corresponds to the watchlist.*
2. *For every* $V_{j,b}$ *in* $\mathbf{c}_1$*, we may write without loss of generality that* $V_{j,b} = \left(V_{j,b}^i\right)_{i\in[n]}$*, where* $V_{j,b}^i$ *is the message that* V_i *sends to* $V_{j,b}$.
3. *If there exists* $V_{i_1}, V_{i_2} \in \mathbf{c}_2$ *or* $V_i \in \mathbf{c}_2$ *and* $V_{j,b}^i \in \mathbf{c}_1$ *that are inconsistent, output* $\perp$.
4. *Otherwise, apply the PRE decoder on* $\mathbf{c}_1$ *to recover the output* z.

We summarize the properties of the protocol below.

Theorem 9 ([19, Theorem 1]). *For every* $\varepsilon > 0$*,* $\Pi_{\mathrm{IKOPS}}(\varepsilon)$ *computes* g *with* ε*-statistical full security.*[12] *Furthermore, using the PRE from [1,16] and the BGW protocol, the* CC *will be the following. Let* γ_i *denote the size of the smallest formula for evaluating the* i*'th bit of* $g(x, y)$*, and let* $\gamma = \max_i \gamma_i$*. Then,* Π_{IKOPS} *has* CC *at most*

$$\ell_{\mathrm{IKOPS}} = \xi_{\mathrm{IKOPS}} \cdot \gamma^2 \cdot \log k \cdot \log|\mathcal{Y}| \cdot \mathrm{polylog}\left(\varepsilon^{-1}\right),$$

where $\xi_{\mathrm{IKOPS}} \in \mathbb{R}^+$ *is some global constant independent of the function* g *and of* ε.

Observe that Π_{IKOPS} has a (small) non-zero probability of the client seeing to many views of the virtual servers (in the worst case all of them which gives him the knowledge of x). Thus, Π_{IKOPS} is not perfectly client secure.

In the following section, we slightly tweak Π_{IKOPS}, making the watchlists deterministic, thereby making it perfectly client secure. The new protocol will have the desired properties as stated in Lemma 3.

[11] The function φ is different when applying to recover $\mathbf{a}_1[\mathbf{b}_1]$ from when applying to recover $\mathbf{a}_2[\mathbf{b}_2]$. To keep the presentation simple we will abuse notation and write as if they are the same function.

[12] In fact, the protocol even admits strong ε-server security.

5.2 Setting up Fixed-Size Watchlists

Recall the problem with client privacy was in the fact that the client may watch the internal state of too many servers, breaching perfect security of the protocol Π_{IKOPS}, and thus of the entire construction. To solve this problem, we replace the current watchlist setup with a *fixed-size watchlist* setup.

In order to achieve the fixed-size watchlist, the parties will use a perfectly secure protocol for computing $\binom{n}{t/2}$-s-string-OT. We do not know, however, if such a protocol even exists in the $\mathcal{OT}$-hybrid model. Instead, we relax the security notion a bit, so that we will be able to construct the protocol, and its security guarantees still suffice for the main protocol. Specifically, we show how in the $\mathcal{OT}$-hybrid model, the parties can compute $\binom{n}{t/2}$-s-string-OT in a single round, where a malicious client will only be able to learn at most t strings rather than $t/2$. We stress that the construction we suggest *does not* achieve perfect server security. Instead, it admits perfect *input-dependent security*. As we show in Sect. 5.3, this will not affect the security properties of our final construction.

Let $t, n, s \in \mathbb{N}$ where $t < n$, and $s \geq 1$. For simplicity, we assume that t is even. Let f_1 and f_2 be the $\binom{n}{t/2}$-s-string-OT and $\binom{n}{t}$-s-string-OT functionalities respectively. We next briefly explain the ideas behind the construction. The parties will use protocol Π_{BCS} in order to simulate computation of n instances of $\binom{2}{1}$-sn-string-OT in parallel. On input $(x_1, \ldots, x_n)$, the i-th pair of strings the server will send (by first applying α_{BCS}) consist of a masking of the i-th string x_i, and a Shamir share of the concatenation of all of the maskings, that is, the pair will be $(x_i \oplus r_i, \mathbf{r}[i])$, where $\mathbf{r} = (r_1, \ldots, r_n)$. The client will then recover the maskings of the correct outputs alongside the shares, which will help him to reconstruct the outputs. Since for each i the client will learn either a share or a masked string, a malicious client will not be able to learn to many masked strings. The protocol $\Pi_{\text{ROT}} = (\alpha_{\text{ROT}}, \beta_{\text{ROT}}, \varphi_{\text{ROT}})$ for computing f_1 in the 1-round $\mathcal{OT}$-hybrid model is formally described as follows.

Construction 10 (Π_{ROT})
Input: Server S *holds* $\mathbf{x} = (x_1, \ldots, x_n) \in (\{0,1\}^s)^n$, *and the client* C *holds* $\mathbf{y} = \{y_1, \ldots, y_{t/2}\} \subseteq [n]$.

- $\alpha_{\text{ROT}}(\mathbf{x})$: *Samples* n *random strings* $r_1, \ldots, r_n \leftarrow \{0,1\}^s$ *independently. For every* $i \in [n]$, *let* $\mathbf{r}[i] \in \{0,1\}^{sn}$ *be a share of* $\mathbf{r} = (r_1, \ldots, r_n)$ *in an* $(n-t)$*-out-of-*n *Shamir's secret sharing (we pad* $\mathbf{r}[i]$ *if needed). Output* $\mathbf{a} = \left(\alpha_{\text{BCS}}\left((x_i \oplus r_i, \mathbf{r}[i])\right)\right)_{i \in [n]}$ *(the* $x_i \oplus r_i$*'s are also padded accordingly).*
- $\beta_{\text{ROT}}(\mathbf{y})$: *Output* $\mathbf{b} = (\beta_{\text{BCS}}(b_1), \ldots, \beta_{\text{BCS}}(b_n))$, *where* $b_i = 0$ *if and only if* $i \in \mathbf{y}$.
- $\varphi_{\text{ROT}}(\mathbf{y}, \mathbf{b}, \mathbf{c}')$: *Let* $\mathbf{c} = (\varphi_{\text{BCS}}(c'_i))_{i=1}^n$, *let* $\mathbf{c}_1 = (c_i)_{i \in \mathbf{y}}$, *and let* $\mathbf{c}_2 = (c_i)_{i \notin \mathbf{y}}$. *If the elements in* $\mathbf{c}_2$ *agree on a common secret* $\mathbf{r} \in \{0,1\}^{sn}$, *then output* $\mathbf{c}_1 \oplus (r_i)_{i \in \mathbf{y}}$. *Otherwise, output* $\perp$.

Lemma 4. *Π_{ROT} computes f_1 with* CC *at most $5 \cdot sn^2$, such that the following holds:*

- *Π_{ROT} is correct.*
- *Π_{ROT} has perfect input-dependent security.*
- *For any non-uniform adversary $\mathcal{A}$ corrupting the client in the $\mathcal{OT}$-hybrid world, there exists a non-uniform simulator $\mathsf{Sim}_{\mathcal{A}}$ corrupting the client in the ideal-world of f_2, such that for all $\mathbf{x} \in (\{0,1\}^s)^n$, $\mathbf{y} \subseteq [n]$ of size $t/2$, and $\mathsf{aux} \in \{0,1\}^*$ it holds that*

$$\text{View}^{\text{HYBRID}}_{\mathcal{A}(\mathbf{y},\mathsf{aux}),\Pi_{\text{ROT}}}(x,\mathbf{y}) \equiv \text{View}^{\text{IDEAL}}_{\mathsf{Sim}_{\mathcal{A}}(\mathbf{y},\mathsf{aux}),f_2}(x,\mathbf{y}).$$

In other words, although the simulator receives $t/2$ indexes as inputs, it is allowed to ask for t strings from the server's input.

Intuitively, a malicious server cannot force the client to reconstruct two different secrets $\mathbf{r}$ for two different inputs. This is due to the fact that for every two different inputs the set of common b_i's that are 1 (i.e., the number of common shares the client will receive for both inputs) is of size at least $n-t$. This implies that up to a certain set of client-inputs that the adversary can choose, the client will receive a correct output. As for a malicious client, observe that it can ask for at most t masked values, as otherwise it will not have enough shares to recover the secret $\mathbf{r}$.

We next incorporate Π_{ROT} into Π_{IKOPS} to get a protocol that is perfectly client-secure. The full proof of Lemma 4 is deferred to Sect. 5.4.

5.3 Upgrading Π_{IKOPS}

We are finally ready to prove Lemma 3. As stated in Sect. 5.2, we replace the randomly chosen watchlist with a deterministic one using Π_{ROT}. Formally, the protocol, denoted Π^{+}_{IKOPS}, is described as follows.

Protocol 11 ($\Pi^{+}_{\text{IKOPS}}(\varepsilon)$)
Input: Server has input $x \in (\mathcal{X} \cup \{\perp\})$ and client has input $\mathbf{y} \in \{0,1\}^m$.

- $\alpha^{+}(x)$: *Output $(\mathbf{a}_1, \mathbf{a}_2)$ as in Π_{IKOPS}, with the exception of $\mathbf{a}_2$ being equal to $\alpha_{\text{ROT}}(V_1, \ldots, V_n)$ (recall that V_i is the view of the virtual server S_i).*
- $\beta^{+}(\mathbf{y})$: *Output $(\mathbf{b}_1, \mathbf{b}_2)$ as in Π_{IKOPS}, with the exception of $\mathbf{b}_2$ being equal to $\beta_{\text{ROT}}(\mathcal{W})$, where $\mathcal{W} \subseteq [n]$ is of size $t/2$ chosen uniformly at random (recall that $t = \lceil n/3 \rceil - 1$ bounds the number of corrupted parties in the MPC protocol).*
- $\varphi^{+}(\mathbf{y}, \mathbf{b}, \mathbf{c}')$: *Output same as $\varphi(\mathbf{y}, \mathbf{b}, \mathbf{c}')$, with the exception that we apply φ_{ROT} to recover the outputs and watchlist.*

Clearly, Lemma 3 follows from the following lemma, asserting the security of Π^{+}_{IKOPS}.

Lemma 5. *For every $\varepsilon > 0$, $\Pi^{+}_{\mathrm{IKOPS}}(\varepsilon)$ computes g with correctness, it is strong ε-server secure, and has perfect client security. Furthermore, using the PRE from* [1,16] *and the BGW protocol, the* CC *will be the following. Let γ_i denote the size of the smallest formula for evaluating the i'th bit of $g(x,y)$, and let $\gamma = \max_i \gamma_i$. Then, Π^{+}_{IKOPS} has* CC *at most*

$$\ell^{+}_{\mathrm{IKOPS}} = \xi^{+}_{\mathrm{IKOPS}} \cdot \gamma^2 \cdot \log k \cdot \log |\mathcal{Y}| \cdot \mathrm{polylog}\left(\varepsilon^{-1}\right),$$

where $\xi^{+}_{\mathrm{IKOPS}} \in \mathbb{R}^{+}$ is some global constant independent of the function and of ε. In comparison to Π_{IKOPS}, the only difference in the CC *is in the constant and the exponent of $\log\left(\varepsilon^{-1}\right)$ taken. Specifically, it holds that*

$$\frac{\ell^{+}_{\mathrm{IKOPS}}}{\ell_{\mathrm{IKOPS}}} = \frac{\xi^{+}_{\mathrm{IKOPS}}}{\xi_{\mathrm{IKOPS}}} \cdot \log^2\left(\varepsilon^{-1}\right).$$

Proof. Correctness trivially holds. We next prove that the protocol is strong ε-server secure. Consider a message $\mathbf{a}^*$ sent by a malicious server holding $x \in (\mathcal{X} \cup \perp)$ and an auxiliary input $\mathsf{aux} \in \{0,1\}^*$ in the $\mathcal{OT}$-hybrid world. We need to show the existence of a certain probability vector $\mathbf{p} \in \mathbb{R}^{|\mathcal{X}|+1}$. It will be convenient to describe the vector $\mathbf{p}$ using a simulator Sim that will describe the probability of sending x^* to T as an input.

The idea is to have the simulator check the inconsistencies made by the adversary. This is done via an *inconsistency graph*, where each vertex corresponds to a virtual party, and each edge corresponds to an inconsistency. There are three cases in which the simulator will send $\perp$ to T. The first case, is when there is a large vertex cover among the servers. In the $\mathcal{OT}$-hybrid world, the client will see an inconsistency with high probability, and hence output $\perp$. The second case, is when there are two virtual clients $\mathsf{C}_{j,0}$ and $\mathsf{C}_{j,1}$, corresponding to the same bit of $\mathsf{Enc}(\mathbf{y})$ that are both inconsistent with the same server. Observe that the real client will always see an inconsistency, regardless of its input or randomness. The final case remaining, is when for each $j \in [m \cdot m']$, the adversary tampered with exactly one of $\mathsf{C}_{j,0}$ or $\mathsf{C}_{j,1}$. Here the real client will not notice the inconsistency only if asked for the virtual clients the adversary did not tamper with, which happens with low probability. For all other cases, the probability that the real client will see an inconsistency is *independent of its input*. Therefore the simulator can compute it and send $\perp$ with this probability. When the simulator does not send $\perp$ as its input, it uses the MPC simulator to reconstruct an effective input.

We next formalize the description of the simulator. The simulator holds $\mathbf{a}^*$ and aux as an input.

1. Write $\mathbf{a}^* = (\mathbf{a}^*_1, \mathbf{a}^*_2)$, where $\mathbf{a}^*_1$ corresponds to the outputs and $\mathbf{a}^*_2$ corresponds to the watchlist.
2. Apply the simulator guaranteed by the security of Π_{BCS} to each pair of messages in $\mathbf{a}^*_1$ to obtain $V_{1,0}, V_{1,1}, \ldots, V_{m\cdot m',0}, V_{m\cdot m',1}$, and apply the simulator guaranteed by Π_{ROT} for each pair in $\mathbf{a}^*_2$ to obtain $V_1, \ldots, V_n$ and a predicate P (if the output of the simulator is $\perp$ instead of views, then send $\perp$ to T).

3. Generate an inconsistency graph G', with $[n]$ as vertices, and where $\{i_1, i_2\}$ is an edge if and only if V_{i_1} and V_{i_2} are inconsistent. Let VC be a minimum vertex cover of G'.[13] If $|\mathsf{VC}| > t$ then send $\perp$ to T.
4. Otherwise, pick a subset $\mathcal{W} \subseteq [n]$ of size $t/2$ uniformly at random. If there exist $i_1, i_2 \in \mathcal{W}$ with an edge between them in G or $P(\mathcal{W}) = 1$, then send $\perp$ to T.
5. Otherwise, extend G' into an inconsistency graph G, where there are new vertices $(j, b) \in [m \cdot m'] \times \{0, 1\}$, and $\{i, (j, b)\}$ is an edge if and only if $V^i_{j,b}$ is inconsistent with V_i (i.e., the view $\mathsf{C}_{j,b}$ received from S_i is inconsistent with the view of S_i).
6. Let $\mathcal{S} \subseteq [m \cdot m'] \times \{0, 1\}$ be the set of vertices corresponding to the virtual clients, that have an edge with a vertex in $\mathcal{W}$. If there exists $j \in [m]$ such that either
   - $(m'(j-1) + j', 0), (m'(j-1) + j', 1) \in \mathcal{S}$ for some $j' \in [m']$, or
   - for every $j' \in [m']$ exactly one the vertices $(m'(j-1) + j', 0), (m'(j-1) + j', 1)$ is in $\mathcal{S}$,

   then send $\perp$ to T.
7. Otherwise, send $\perp$ with probability $1 - 2^{-e(\mathcal{S})}$, where $e(\mathcal{S})$ is the number of edges coming out of $\mathcal{S}$. With the complement probability, apply the (malicious) MPC simulator on the virtual servers S_i, where $i \in \mathsf{VC}$, to get an input for each of virtual servers in VC. The simulator Sim can then use the inputs of the other virtual servers to get an effective input $x^* \in (\mathcal{X} \cup \{\perp\})$, and send it to T.

The vector $\mathbf{p}$ is then defined as $p_{x^*} = \Pr[\mathsf{Sim} \text{ sends } x^* \text{ to } \mathsf{T}]$. Recall that for every $\mathbf{y} \in \mathcal{Y}$ and $z \in \{\perp, 0, \ldots, k-1\}$ we denote

$$q^{\Pi^+_{\text{IKOPS}}}_{y,z}(\mathbf{a}^*) = \Pr\left[\varphi^+(\mathbf{y}, \mathbf{b}, \mathbf{a}^*[\mathbf{b}]) = z\right],$$

where $\mathbf{b} = \beta(\mathbf{y})$ and the probability is over the randomness of β and φ. To alleviate notations, we will write $\mathbf{q} = \mathbf{q}^{\Pi^+_{\text{IKOPS}}(\varepsilon)}(\mathbf{a}^*)$. Fix $\mathbf{y} \in \{0,1\}^m$ and $z \in \{\perp, 0, \ldots, k-1\}$. We show that[14]

$$\left|q_{\mathbf{y},z} - M_g^T(\cdot, (\mathbf{y}, z)) \cdot \mathbf{p}\right| \leq \varepsilon \cdot p_\perp. \tag{5}$$

Observe that since Π_{BCS} and Π_{ROT} has perfect server-security, each $V_{m'(j-1)+j',b}$ and each V_i in the $\mathcal{OT}$-hybrid world is distributed exactly the same as its counterpart in the ideal world. Therefore, we may condition on the event that they are indeed the same. Furthermore, by the security of Π_{ROT}, we may also assume that the watchlist $\mathcal{W}$ is distributed the same, and that $P(\mathcal{W}) = 0$,

[13] Recall that we do not care about the efficiency of simulator. We stress that it is also suffices to use a 2-approximation to compute the minimum vertex-cover, while slightly tweaking t.

[14] In fact we can show something stronger – that the ℓ_1 distance (i.e., statistical distance) is smaller than $\varepsilon \cdot p_\perp$, implying that the protocol has standard ε-server security. However, this does not improve our result and the proof is therefore omitted.

as otherwise in both worlds the client will output $\perp$. In the following we fix the views and $\mathcal{W}$. We next separate into four cases, stated in the following claims (proven below). These claims together immediately imply Eq. (5).

Claim 12. *If $|\mathsf{VC}| > t$ then Eq. (5) holds.*

Claim 13. *Assume that $|\mathsf{VC}| \leq t$ and that for every $i \in \mathcal{W}$ and every $j \in [m]$, there exists $j' \in [m']$ such that either both $V_{m'(j-1)+j',0}$ and $V_{m'(j-1)+j',1}$ are consistent with V_i, or both are inconsistent with V_i. Then Eq. (5) holds. Moreover, the simulation is perfect.*

Claim 14. *Assume that $|\mathsf{VC}| \leq t$ and that there exists $i \in \mathcal{W}$ and $j \in [m]$, such that for every $j' \in [m']$ exactly one of the views $V_{m'(j-1)+j',0}$ and $V_{m'(j-1)+j',1}$ are inconsistent with V_i, then Eq. (5) holds.*

Proof (of Claim 12). Intuitively, the vertex cover of the graph G gives us information on which servers "misbehaved". A large vertex cover means that a lot of servers have inconsistent views, implying that there are many edges in the graph. Therefore, a random subset of the vertices would contain at least one edge with high probability. We next formalize this intuition.

Since $|\mathsf{VC}| > t$ then the maximum matching in G' is of size at least $(t+1)/2$. Therefore, in the $\mathcal{OT}$-hybrid world, the expected number of edges that the client will have in its watchlist is at least $\frac{t+1}{2} \cdot \frac{\binom{n-2}{t/2-2}}{\binom{n}{t/2}} = \Theta(n)$. By applying Hoeffding's inequality,[15] with probability at least $1 - 2^{-\Theta(n)} = 1 - \varepsilon$ the client will output $\perp$. As in the ideal-world the simulator sends $\perp$ to T with probability 1, Eq. (5) follows.

Proof (of Claim 13). We separate into two cases. For the first case, assume that there exist $i \in \mathcal{W}$, $j \in [m]$, and $j' \in [m']$ such that both $V_{m'(j-1)+j',0}$ and $V_{m'(j-1)+j',1}$ are inconsistent with V_i. Then $(m'(j-1)+j', 0), (m'(j-1)+j', 1) \in \mathcal{S}$, hence the simulator always sends $\perp$ in this case. Furthermore, in the $\mathcal{OT}$-hybrid world, for every input $\mathbf{y} \in \{0,1\}^m$ the client will see an inconsistency between either $V_{m'(j-1)+j',0}$ and V_i, or an inconsistency between either $V_{m'(j-1)+j',1}$ and V_i. Thus, Eq. (5) holds with no error.

By the assumptions of the claim, for the second case we may assume that for every $i \in \mathcal{W}$ and every $j \in [m]$, there exists $j' \in [m']$ such that both $V_{m'(j-1)+j',0}$ and $V_{m'(j-1)+j',1}$ are consistent with V_i. In this case, in the $\mathcal{OT}$-hybrid world, the client will see an inconsistency with probability $1-2^{-e(\mathcal{S})}$. With the complement probability, its output is determined by whatever the virtual servers computed. The output of the client in the ideal-world is either $\perp$ with probability $1-2^{-e(\mathcal{S})}$ or it is determined by the MPC simulator. Since it is assumed to be perfect and $|\mathsf{VC}| \leq t$ bound from above the number of corrupted servers, it follows that Eq. (5) holds with no error.

[15] Although Hoeffding's inequality is stated for the sum of independent random variables, it still works in our case since the sampled can be modeled as if we are picking vertices without repetitions. Sampling with repetitions only decreases the probability for an edge.

Proof (of Claim 14). By construction, the ideal-world simulator always sends $\perp$ in this case. Additionally, in the $\mathcal{OT}$-hybrid world, the client uses Enc on its input $\mathbf{y}$ to receive $m \cdot m'$ random bits $\left(y_j^{j'}\right)_{j\in[m],j'\in[m']}$ conditioned on $\oplus_{j'=1}^{m'} y_j^{j'} = y_j$ for every $j \in [m]$. Since we assume that exactly m' virtual clients, corresponding to the same input bit y_j, where tampered by the adversary, it follows that with probability $2^{-(m'-1)} \leq \varepsilon$ the client will see only consistent views. Therefore, for every $\mathbf{y} \in \{0,1\}^m$ it holds that

$$\left|q_{\mathbf{y},\perp} - M_g^T(\cdot,(\mathbf{y},\perp)) \cdot \mathbf{p}\right| = \left|q_{\mathbf{y},\perp} - p_\perp\right| = \Pr\left[\varphi^+(\mathbf{y},\mathbf{b},\mathbf{a}^*[\mathbf{b}]) \neq \perp\right] \leq \varepsilon,$$

and for every $z \neq \perp$

$$\left|q_{\mathbf{y},z} - M_g^T(\cdot,(\mathbf{y},z)) \cdot \mathbf{p}\right| = \left|\Pr\left[\varphi^+(\mathbf{y},\mathbf{b},\mathbf{a}^*[\mathbf{b}]) = z\right] - 0\right| \leq \varepsilon.$$

Equation (5) follows.

We next show that the protocol has perfect client-security. Consider an adversary $\mathcal{A}$ corrupting the client. We construct the simulator $\mathsf{Sim}_{\mathcal{A}}$. The construction of the simulator is done in the natural way, namely, it will apply the simulators of Π_{BCS} and Π_{ROT}, and then the decoding of the PRE, to receive an output. It can then use the MPC simulator to simulate the views of the virtual servers in its watchlist. Formally, the simulator operates as follows.

1. On input $\mathbf{y} \in \{0,1\}^m$ and auxiliary input $\mathsf{aux} \in \{0,1\}^*$, query $\mathcal{A}$ to receive a message $\mathbf{b}^*$ to be sent to the OT.
2. Write $\mathbf{b}^* = (\mathbf{b}_1^*, \mathbf{b}_2^*)$, where $\mathbf{b}_1^*$ corresponds to the outputs and $\mathbf{b}_2^*$ corresponds to the watchlist.
3. Apply the simulator guaranteed by the security of Π_{BCS} to each pair of messages in $\mathbf{b}_1^*$ to obtain $(b_j)_{j\in[m\cdot m']}$ for some $b_j \in \{0,1\}$, and apply the simulator $\mathsf{Sim}_{\mathrm{ROT}}$, guaranteed by the security of Π_{ROT}, for each pair in $\mathbf{b}_2^*$ to obtain a set $\mathcal{W} \subseteq [n]$.
4. Send $\mathsf{Dec}\left((b_j)_{j\in[m\cdot m']}\right)$ to T to obtain an output z.
5. Apply the PRE simulator on z to obtain outputs $(z_j)_{j\in[m\cdot m']}$ for each virtual client.
6. If $|\mathcal{W}| > t$ then output $(z_j)_{j\in[m\cdot m']}$ alongside whatever $\mathsf{Sim}_{\mathrm{ROT}}$ outputs and halt.
7. Otherwise, apply the (semi-honest) MPC simulator on the parties $\{\mathsf{S}_i\}_{i\in\mathcal{W}}$ with random strings as inputs, and on $\left\{\mathsf{C}_{j,b_j}\right\}_{j\in[m\cdot m']}$ with z_j as the output respectively. Send the output of the MPC simulator to $\mathsf{Sim}_{\mathrm{ROT}}$, outputs whatever it outputs and halt.

The security of Π_{BCS} and Π_{ROT} implies that $(b_j)_{j\in[m\cdot m']}$ and $\mathcal{W}$ are distributed exactly the same in both worlds. Therefore, the output $z = g\left(x, \mathsf{Dec}\left((b_j)_{j\in[m\cdot m']}\right)\right)$ is distributed the same, hence applying the PRE simulator on z will also result in the same distribution. Now, if $|\mathcal{W}| > t$ then $\mathsf{Sim}_{\mathrm{ROT}}$

is guaranteed to produce a correct view as an output. If $|\mathcal{W}| \leq t$, then the MPC simulator will perfectly generate $|\mathcal{W}|$ virtual views. Handing them over to $\mathsf{Sim}_{\mathrm{ROT}}$ would result in the view that is distributed the same as in the $\mathcal{OT}$-hybrid world.

5.4 Proof of Lemma 4

We first prove the following simple claim, stating that the client will always reconstruct a unique secret (if its not outputting $\perp$).

Claim 15. *Consider a message* $\mathbf{a}^* = \left((a^*_{1,0}, a^*_{1,1}), \ldots, (a^*_{n,0}, a^*_{n,1})\right) \in \{0,1\}^{2sn}$ *sent to the OT by a malicious server. Then for any different inputs* $\mathbf{y}_1 \neq \mathbf{y}_2$ *for the client, either it will output* $\perp$ *for at least one of the inputs, or there exists a common secret* $\mathbf{r}$ *that will be reconstructed.*

Proof. Let $\mathcal{B} = \{i \in [n] : i \notin \mathbf{y}_1 \wedge i \notin \mathbf{y}_2\}$. Then $|\mathcal{B}| \geq n - t$, hence the client – who receives the shares $\left(a^*_{i,1}\right)_{i \in \mathcal{B}}$ – can reconstruct a secret $\mathbf{r}$ in case the share are consistent. This secret will be the same for both $\mathbf{y}_1$ and $\mathbf{y}_2$.

We now prove the lemma.

Proof (of Lemma 4). By construction, it is not hard to see that the protocol is correct. We next prove that the protocol has perfect input-dependent security. Consider a adversary $\mathcal{A}$ corrupting the server. We construct a simulator $\mathsf{Sim}_{\mathcal{A}}$ as follows. On input $\mathbf{x}$ and auxiliary input $\mathsf{aux} \in \{0,1\}^*$, query $\mathcal{A}$ to receive a message $\mathbf{a}^* = \left((a^*_{1,0}, a^*_{1,1}), \ldots, (a^*_{n,0}, a^*_{n,1})\right) \in \{0,1\}^{2sn}$. If there are no $n - t$ shares from $\left(a^*_{i,1}\right)_{i \in [n]}$ that are consistent, then $\mathsf{Sim}_{\mathcal{A}}$ will send the constant 1 predicate alongside some arbitrary input $\mathbf{x}_0$ to the trusted party T. Otherwise, let $\mathcal{B}$ be the maximum set of indexes $i \in [n]$ such that the $a^*_{i,1}$ are shares consistent with single value $\mathbf{r} \in (\{0,1\}^s)^n$. Then $\mathsf{Sim}_{\mathcal{A}}$ will send to T the input $\left(\left(a^*_{i,0} \oplus r_i\right)_{i \notin \mathcal{B}}, (0^s)_{i \in \mathcal{B}}\right)$ with the predicate $P_{\mathcal{B}}(\mathbf{y}) = 1$ if and only if $\mathbf{y} \cap \mathcal{B} \neq \emptyset$.

To see why the simulator works, observe that $\mathsf{Sim}_{\mathcal{A}}$ sends constant 1 predicate if and only if $\mathcal{A}$ sent at most t consistent shares, forcing C to output $\perp$ in the ideal-world. Since this happens if there are too many inconsistencies, C will output $\perp$ in the $\mathcal{OT}$-hybrid world as well. Furthermore, if there are at least $n - t$ shares that are consistent, then by Claim 15, there is a unique secret $\mathbf{r}$ that can be reconstructed. Therefore, in the $\mathcal{OT}$-hybrid world, on input $\mathbf{y}$, C will output $\perp$ if $\mathbf{y} \cap \mathcal{B} \neq \emptyset$, and output $\left(\mathbf{a}^*[\beta_{\mathrm{ROT}}(\mathbf{y})]_i \oplus r_i\right)_{i \in \mathbf{y}}$ otherwise. Since $\mathcal{B}$ was chosen to be the maximum set of indexes, the same holds in the input-dependent ideal-world.

We next show that the relaxed security requirement against malicious clients holds. Let $\mathcal{A}$ be an adversary corrupting the client. The simulator $\mathsf{Sim}_{\mathcal{A}}$ works as follows. On input $\mathbf{y}$ and auxiliary input $\mathsf{aux} \in \{0,1\}^*$, query $\mathcal{A}$ to receive $\mathbf{b}^* \in \{0,1\}^n$. If there are strictly more than t 0's in $\mathbf{b}^*$ then output n random strings, each of length s. Otherwise, send $\{i \in [n] : b^*_i = 0\}$ to T to receive output $(x_i)_{i : b^*_i = 0}$. $\mathsf{Sim}_{\mathcal{A}}$ samples n random strings $r_1, \ldots, r_n \leftarrow \{0,1\}^s$. For $i \in [n]$, let

$\mathbf{r}[i] \in \{0,1\}^{sn}$ be a share of $\mathbf{r} = (r_1, \ldots, r_n)$ in an $(n-t)$-out-of-n Shamir's secret sharing (pad $\mathbf{r}[i]$ if needed). The simulator then generates the values

$$\mathbf{a} := \left(\left(\alpha_{\text{BCS}}(x_i \oplus r_i, \mathbf{r}[i])\right)_{i:b_i^*=0}, \left(\alpha_{\text{BCS}}\left(0^{sn}, \mathbf{r}[i]\right)\right)_{i:b_i^*=1} \right),$$

where the $x_i \oplus r_i$'s are padded accordingly. $\mathsf{Sim}_{\mathcal{A}}$ will then compute and output

$$\binom{2}{1}\text{-bit-OT}^{\ell}\left(\mathbf{a}, \left(\beta_{\text{BCS}}\left(b_1^*\right), \ldots, \beta_{\text{BCS}}\left(b_n^*\right)\right)\right).$$

$\mathsf{Sim}_{\mathcal{A}}$ works since in the case where there are more than t 0's in $\mathbf{b}^*$, by the properties the sharing scheme, the view of C in the $\mathcal{OT}$-hybrid world consist only of random values. Otherwise, C will receive the masked x_i for the indexes i on which $b_i^* = 0$, and shares of the maskings for the indexes i on which $b_i^* = 1$.

6 Tightness of the Analysis

Recall that our final protocol is a "wrapper" for an upgraded version of the protocol by Ishai et al. [19], namely, protocol Π^{+}_{IKOPS} from Sect. 5.3. In the following section, we prove that for any (randomized) *Boolean* function f that is *not* full-dimensional, and does not satisfy the constraints in Corollary 1, no "wrapper" protocol for Π^{+}_{IKOPS} will compute f with perfect full-security. Here, the "wrapper" protocol simply replaces the output $\perp$ that the client receive from Π^{+}_{IKOPS} with a random bit. Formally, any "wrapper" protocol is parametrized with a vector $\mathbf{v} \in [0,1]^{|\mathcal{Y}|}$ and an $\varepsilon > 0$, and is denoted by $\Pi_f^{\mathbf{v}}(\varepsilon)$. Let $g : (\mathcal{X} \cup \{\perp\}) \times \mathcal{Y} \mapsto \{\perp, 0, 1\}$ be defined as $g(x,y) = f(x,y)$ if $x \neq \perp$ and $g(\perp, y) = \perp$. The "wrapper" protocol $\Pi_f^{\mathbf{v}}$ is described as follows.

Protocol 16 ($\Pi_f^{\mathbf{v}}(\varepsilon)$)

Input: Server S *has input* $x \in \mathcal{X}$ *and client* C *has input* $y \in \mathcal{Y}$.

1. *The parties execute protocol* $\Pi^{+}_{\text{IKOPS}}(\varepsilon)$ *in order to compute* g. *Let* z *be the output* C *receive.*
2. *If* $z \neq \perp$, *then* C *output* z. *Otherwise, output 1 with probability* v_y.

We next claim that the protocol cannot compute Boolean functions that are not full-dimensional with perfect full-security.

Theorem 17. *Let* $f : \mathcal{X} \times \mathcal{Y} \mapsto \{0,1\}$ *denote a (possibly randomized) Boolean function that has* no *constant columns, i.e.,* $M_f(\cdot, y)$ *is not constant for every* $y \in \mathcal{Y}$, *and is* not *full-dimensional, i.e.,* $\dim\left(\mathbf{aff}\left(M_f^T\right)\right) < |\mathcal{Y}|$. *Then for every* $\mathbf{v} \in [0,1]^{|\mathcal{Y}|}$ *and every* $\varepsilon > 0$, $\Pi_f^{\mathbf{v}}(\varepsilon)$ *does not compute* f *with perfect full-security.*

Proof. Assume towards contradiction that $\Pi_f^{\mathbf{v}}(\varepsilon)$ has perfect server security, for some $\mathbf{v}$ and ε. We next construct $|\mathcal{Y}|+1$ adversaries, such that each adversary forces the vector of outputs of the client $\mathbf{q}^{\Pi_f^{\mathbf{v}}(\varepsilon)}$, to be a different point inside the convex-hull of the rows of M_f. We then show that these points are affinely independent, giving us a contradiction. First, write each input of the client as a binary string $\mathbf{y}$ of length m. For every $\mathbf{y} \in \mathcal{Y}$ define the adversary $\mathcal{A}_{\mathbf{y}}$ as follows.

1. Fix an encoding $\mathbf{y}' = \left(y_j^1, \ldots, y_j^{m'}\right)_{j\in[m]} \in \mathrm{Supp}\left(\mathsf{Enc}(\mathbf{y})\right)$, and fix some $x_{\mathbf{y}}^* \in \mathcal{X}$ such that $f(x_{\mathbf{y}}^*, \mathbf{y}) \neq v_y$. (such an $x_{\mathbf{y}}^*$ exists, since M_f does not have constant columns).
2. Execute $\Pi_{\mathrm{IKOPS}}^{+}(\varepsilon)$ honestly with input $x_{\mathbf{y}}^*$, as fixed above, with the following one exception. For every $i \in [n]$, $j \in [m]$, and $j' \in [m']$, modify $V^i_{m'(j-1)+j', 1-y_j^{j'}}$ such that it is inconsistent with V_i.

Finally, define the adversary $\mathcal{A}_0$ who picks an arbitrary $x_0^* \in \mathcal{X}$ as an input, and acts honestly with the exception that it tampers with all $V^i_{j,b}$'s, making them inconsistent with the corresponding V_i. Let $\mathbf{a}^*\left(x_{\mathbf{y}}^*\right)$ be the message $\mathcal{A}_{\mathbf{y}}$ sends to the OT.

Let us analyze the client's vector of outputs $\mathbf{q}^{\Pi_f^{\mathbf{v}}(\varepsilon)}\left(\mathbf{a}^*\left(x_{\mathbf{y}}^*\right)\right)$, for any adversary $\mathcal{A}_{\mathbf{y}}$, for $\mathbf{y} \in \mathcal{Y} \cup \{0\}$. For brevity, we write $\mathbf{q}\left(x_{\mathbf{y}}^*\right)$ instead. By definition, $\mathcal{A}_0$ forces the client to sample its output according to $\mathbf{v}$, hence $\mathbf{q}\left(x_0^*\right) = \mathbf{v}$. Next, fix $\mathbf{y} \in \mathcal{Y}$. Observe that for every $\hat{\mathbf{y}} \neq \mathbf{y}$, any of their encodings will differ on at least one bit, i.e., $\hat{y}_j^{j'} \neq y_j^{j'}$ for some $j \in [m]$ and $j' \in [m']$, hence on input $\hat{\mathbf{y}}$, the client will see $V^i_{m'(j-1)+j', 1-y_j^{j'}}$ for every i. Since the inconsistency is made with every virtual server, on input $\hat{\mathbf{y}}$, the client will notice it and output $\perp$ with probability 1. By the description of $\Pi_f^{\mathbf{v}}$, it follows that

$$q_{\hat{\mathbf{y}}}\left(x_{\mathbf{y}}^*\right) = v_{\hat{\mathbf{y}}}, \tag{6}$$

for every $\hat{\mathbf{y}} \neq \mathbf{y}$. On the other hand, on input $\mathbf{y}$, the client outputs $\perp$ if and only if the event $\mathsf{Enc}(\mathbf{y}) = \mathbf{y}'$ occurs, which happens with probability $1 - 2^{-m(m'-1)}$. With the complement probability $2^{-m(m'-1)}$ it does not detect an inconsistency, and outputs $f(x_{\mathbf{y}}^*, \mathbf{y})$. Therefore

$$q_{\mathbf{y}}\left(x_{\mathbf{y}}^*\right) = 2^{-m(m'-1)} \cdot f(x_{\mathbf{y}}^*, \mathbf{y}) + (1 - 2^{-m(m'-1)}) \cdot v_{\mathbf{y}}. \tag{7}$$

Thus, Eqs. (6) and (7) yield that

$$\mathbf{q}\left(x_{\mathbf{y}}^*\right) = \mathbf{v} - 2^{-m(m'-1)}(v_{\mathbf{y}} - f(x_{\mathbf{y}}^*, \mathbf{y})) \cdot \mathbf{e}_{\mathbf{y}},$$

where $\mathbf{e}_{\mathbf{y}}$ is the $\mathbf{y}$-th unit vector in $\mathbb{R}^{|\mathcal{Y}|}$.

To conclude the proof, observe that $x_{\mathbf{y}}^*$ was chosen so that $f(x_{\mathbf{y}}^*, \mathbf{y}) \neq v_{\mathbf{y}}$, implying that the set of points $\left\{\mathbf{q}\left(\mathbf{x}_{\mathbf{y}}^*\right)\right\}_{\mathbf{y}\in\mathcal{Y}\cup\{0\}}$ are affinely independent. Furthermore, since $\Pi_f^{\mathbf{v}}$ is assumed to have perfect server security, Lemma 1 implies that all of theses points lie inside $\mathbf{conv}\left(M_f^T\right)$. Therefore, $\mathbf{aff}\left(M_f^T\right) = \mathbb{R}^{|\mathcal{Y}|}$ contradicting the assumption that f is not full-dimensional.

7 A Note on Efficiency

While our main goal is to understand the feasibility of perfectly secure 2PC, our construction does confer concrete efficiency benefits for certain parameter ranges. It is instructive to compare our construction with the IKOPS protocol, for deterministic functions (from the right class). Here we focus on the number of OT calls, which are the most expensive part to implement in practice (usually with computational security). Specifically, for simplicity, we consider the number of calls to a $\binom{2}{1}$-s-string-OT oracle, of any length s, rather than $\binom{2}{1}$-bit-OT. We note that the strings' length of our OT's is quite a bit larger than in IKOPS (due to the step ensuring perfect client security, where the length is multiplied by the number of servers). However, we claim that this comparison is somewhat justified when having practical efficiency in mind, since for particularly long strings, a string-OT oracle can be used to pick *short PRG seeds* instead of the strings themselves during a preprocessing phase. This will be done by having the server send s_0 and s_1 to the "short" string-OT functionality, and the client will receive s_b, where $b \in \{0,1\}$ as its input. Then, to implement the "long" string-OT during the protocol execution, the sender sends to the client $\mathsf{G}(s_a) \oplus m_a$, for $a \in \{0,1\}$, where G is a PRG and m_0 and m_1 are the "long" messages.

Fix a deterministic function $f : \mathcal{X} \times \mathcal{Y} \to \{0, \ldots, k-1\}$ satisfying the conditions of Corollary 1. The number of calls to string-OT in $\Pi^{+}_{\text{IKOPS}}(\varepsilon)$ and $\Pi_{\text{IKOPS}}(\varepsilon)$ is $\log\left(\varepsilon^{-1}\right)(\log|\mathcal{Y}| + c)$, where c is a constant circa 1400 (c is roughly the same in both protocols). When considering our perfectly secure protocol, we set $\varepsilon = \frac{1}{2n(n+1)!}$, where $n = (k-1) \cdot |\mathcal{Y}|$. On the one hand, this results in communication complexity that is polynomial in $|\mathcal{Y}|$ and k, which may be prohibitive for functions with large client-domain or range sizes. On the other hand, for functions with a small client-domain and range sizes, we do better than IKOPS even for real-world error ranges, and the advantage grows as the allowed error ε decreases. For instance, consider the greater than function $3 >: \{0,1,2\} \times \{0,1,2\} \to \{0,1\}$, with an error of $\varepsilon = 2^{-40}$. The communication complexity we obtain is bounded by a factor smaller than $40/\log(24) \approx 8.724$ than that of the IKOPS protocol.

Acknowledgements. We are very grateful to Yuval Ishai for suggesting this question, and for many helpful discussions. We also want to thank Eran Omri for many helpful comments.

References

1. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in nc^0. SIAM J. Comput. **36**(4), 845–888 (2006)
2. Asharov, G.: Towards characterizing complete fairness in secure two-party computation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 291–316. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_13

3. Asharov, G., Beimel, A., Makriyannis, N., Omri, E.: Complete characterization of fairness in secure two-party computation of boolean functions. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 199–228. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_10
4. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 97–109. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_8
5. Beaver, D.: Correlated pseudo randomness and the complexity of private computations. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 479–488. ACM (1996)
6. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols. In: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, pp. 503–513. ACM (1990)
7. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS), pp. 1–10 (1988)
8. Brassard, G., Crépeau, C., Santha, M.: Oblivious transfers and intersecting codes. IACR Cryptology ePrint Archive, 1996:10 (1996). http://eprint.iacr.org/1996/010
9. Cleve, R.: Limits on the security of coin flips when half the processors are faulty. In: Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC), pp. 364–369 (1986)
10. Crépeau, C.: Efficient cryptographic protocols based on noisy channels. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 306–317. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_21
11. Crépeau, C., Morozov, K., Wolf, S.: Efficient unconditional oblivious transfer from almost any noisy channel. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 47–59. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30598-9_4
12. Daza, V., Makriyannis, N.: Designing fully secure protocols for secure two-party computation of constant-domain functions. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 581–611. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_20
13. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM **28**(6), 637–647 (1985)
14. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC19, pp. 218–229 (1987)
15. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC), pp. 413–422 (2008)
16. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45465-9_22
17. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, pp. 433–442. ACM (2008)
18. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_32

19. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_23
20. Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the power of correlated randomness in secure computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 600–620. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_34
21. Khurana, D., Maji, H.K., Sahai, A.: Secure computation from elastic noisy channels. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 184–212. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_7
22. Kilian, J.: Founding cryptography on oblivious transfer. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC), pp. 20–31 (1988)
23. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_4
24. Lindell, Y., Pinkas, B.: A proof of security of yao's protocol for two-party computation. J. Cryptol. **22**(2), 161–188 (2009)
25. Makriyannis, N.: On the classification of finite boolean functions up to fairness. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 135–154. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_9
26. Nascimento, A.C., Winter, A.: On the oblivious transfer capacity of noisy correlations. In: 2006 IEEE International Symposium on Information Theory, pp. 1871–1875. IEEE (2006)
27. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_31
28. Rabin, M.O.: How to exchange secrets with oblivious transfer (2005). http://eprint.iacr.org/2005/187. Harvard University Technical Report 81 talr@watson.ibm.com 12955. Accessed 21 Jun 2005
29. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
30. Wolf, S., Wullschleger, J.: Oblivious transfer is symmetric. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 222–232. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_14
31. Wood, P.J.: On the probability that a discrete complex random matrix is singular. Ph.D. thesis, Rutgers University-Graduate School-New Brunswick (2009)
32. Wullschleger, J.: Oblivious transfer from weak noisy channels. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 332–349. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_20
33. Yao, A.C.: Protocols for secure computations. In Proceedings of the 23th Annual Symposium on Foundations of Computer Science (FOCS), pp. 160–164 (1982)