Efficient and Generic Methods to Achieve Active Security in Private Information Retrieval and More Advanced Database Search

Reo Eriguchi[1], Kaoru Kurosawa[2,1], and Koji Nuida[3,1]

[1] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
eriguchi-reo@aist.go.jp
[2] Research and Development Initiative, Chuo University, Tokyo, Japan
kaoru.kurosawa.kk@vc.ibaraki.ac.jp
[3] Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan
nuida@imi.kyushu-u.ac.jp

Abstract. Motivated by secure database search, we present secure computation protocols for a function f in the client-servers setting, where a client can obtain $f(x)$ on a private input x by communicating with multiple servers each holding f. Specifically, we propose generic compilers from passively secure protocols, which only keep security against servers following the protocols, to actively secure protocols, which guarantee privacy and correctness even against malicious servers. Our compilers are applied to protocols computing any class of functions, and are efficient in that the overheads in communication and computational complexity are only polynomial in the number of servers, independent of the complexity of functions. We then apply our compilers to obtain concrete actively secure protocols for various functions including private information retrieval (PIR), bounded-degree multivariate polynomials and constant-depth circuits. For example, our actively secure PIR protocols achieve exponentially better computational complexity in the number of servers than the currently best-known protocols. Furthermore, our protocols for polynomials and constant-depth circuits reduce the required number of servers compared to the previous actively secure protocols. In particular, our protocol instantiated from the sparse Learning Parity with Noise (LPN) assumption is the first actively secure protocol for multivariate polynomials which has the minimum number of servers, without assuming fully homomorphic encryption.

1 Introduction

Client-server outsourcing is a central problem in secure computation. In particular, there are a wide variety of deployed systems which allow a client to search a database stored in one or more servers for desired contents. Since a client's query may contain sensitive information, it is important to realize *secure database search*, enabling a client to search a database without revealing his or her query to the servers. A trivial solution is downloading the whole database and searching it locally. However, since the database size is typically very large, we need to construct protocols whose communication and client-side computational complexity is sublinear in the database size.

Traditionally, the problem of secure database search has been considered in two types of setting. In the *single-server* setting, there is only one server storing a database who may be corrupted; In the *multi-server* setting, there are m servers storing copies of a database and any t of them are corrupted. In this work, we focus on the multi-server setting since it is known to be impossible to efficiently achieve information-theoretic security in the single-server setting [24] and even in computational settings, the bounded collusion of servers allows better efficiency and weaker cryptographic assumptions than single-server protocols [30, 14, 15].

Private information retrieval (PIR) is a fundamental cryptographic primitive to realize the most basic database search. The goal of PIR is to enable an honest client to retrieve a data item a_i from a database $\boldsymbol{a} = (a_1, \ldots, a_N)$ hiding the index i from the servers[4]. To allow more complex queries such as partial match search, Barkol and Ishai [5] considered a more general setting in which a client has a private input x and servers share a function f, and the goal of the client is to obtain $f(x)$ by communicating with the servers. A rich line of works proposed secure protocols computing various classes of functions f including PIR [9, 8, 32, 10, 31, 16, 19], bounded-degree multivariate polynomials [50, 6, 41, 28], and bounded-depth circuits [5, 18, 44, 46]. Note that the communication complexity of these protocols is much smaller than the size of circuits computing f, which is the main advantages over usual multiparty computation protocols [27, 39, 40, 17].

We note that the above-mentioned protocols are *passively secure*, i.e., the privacy and correctness are guaranteed only if servers follow the protocol specifications. On the other hand, it is desirable to achieve *active security* in real-world scenarios. Namely, protocols should not only protect the privacy of queries but also guarantee the correctness of results even if some servers deviate from the protocols arbitrarily. For example, servers may try to let a client accept an incorrect result, or compute responses from an out-of-date copy of a database. This paper concerns a fundamental problem of constructing an efficient compiler from passively secure protocols to actively secure protocols. Given such a compiler, existing passively secure protocols can be directly upgraded into actively secure protocols with small overheads. Prior to our work, however, the only known passive-to-active compilers are the inefficient ones applied to PIR [11, 34], which incurs exponentially large computational overheads in the number of servers (see Section 1.2 for more related works on compilers in different settings including GMW-style compilers).

1.1 Our Results

In this paper, we study the problem of secure computation in the client-servers setting, where a client can obtain $f(x)$ on a private input x by communicating

[4] Throughout the paper, a client is assumed to be honest.

with multiple servers each holding f. We demonstrate theoretical feasibility of compilers that upgrade passively secure k-server protocols into actively secure m-server protocols with $m > k$. We present two such compilers: The first one upgrades a one-round passively secure protocol into a multi-round actively secure protocol. It increases the number of servers by a corruption threshold t, which seems the best possible (see Remark 1). The second one upgrades a one-round passively secure protocol into a one-round actively secure protocol while increasing the number of servers by a larger factor. More specifically,

- Our first compiler transforms a one-round passively secure k-server protocol into an $O(m^2)$-round actively secure m-server protocol such that $m = k + t$.
- Our second compiler transforms a one-round passively secure k-server protocol into a one-round actively secure m-server protocol such that $m = O(k \log k) + 2t$.

Our compilers are generic and efficient in the sense that they are applied to protocols computing any class of functions f and the overheads in communication and computational complexity are only polynomial in m, independent of the complexity of f. Furthermore, our compilers are unconditional, i.e., requires no additional assumptions, which allows us to obtain actively secure protocols from various assumptions or even information-theoretically as shown below.

Along the way, we introduce two novel notions, *conflict-finding protocols* and *locally surjective map families*. The former is an intermediate notion between passively secure and actively secure protocols, which is used in our first compiler. The latter is a variant of perfect hash families with a stronger property, which is used in our second compiler. A key observation behind our techniques is that if a pair of servers return different answers to the same query, then a client finds that at least one of them is malicious. A difficulty is that we have to carefully design such a strategy, since just disclosing a query for one server to another may reveal his private input. See Section 2 for details on our techniques.

Remark 1. Our first compiler increases the number of servers by t but this seems the best possible. Indeed, the existence of a generic compiler for an actively secure protocol with $m' < t + k$ servers implies a compiler from a k-server protocol to a k'-server protocol for $k' := m' - t < k$ since an actively secure m'-server protocol implies a passively secure $(m' - t)$-server protocol[5]. Thus, the increase in the number of servers is optimal unless there is a generic method to reduce the number of servers. Such a method has not been found up until now.

Instantiations. Based on our compilers, we show concrete actively secure protocols for PIR, bounded-degree multivariate polynomials and constant-depth circuits. Remarkably, our protocol instantiated from the sparse LPN assumption is the first actively secure protocol for multivariate polynomials which has the minimum number of servers, without assuming fully homomorphic encryption.

[5] The active security enables a client to obtain a correct result by interacting with $m' - t$ servers and computing the responses of the other t servers arbitrarily.

PIR. There are compilers from a passively secure k-server PIR protocol to an actively secure m-server protocol for $m = k + 2t$ [11] and $m = k + t$ [34]. However, these compilers incur exponentially large multiplicative overheads $m^{O(t)}$ in client-side computational complexity. On the other hand, our first compiler gives an actively secure m-server protocol such that $m = k+t$ with a polynomial computational overhead $m^{O(1)}$. The only cost is that it requires $O(m^2)$ rounds of interaction between a client and servers. Our second compiler gives a one-round actively secure protocol with a polynomial computational overhead at the cost of a larger number of servers $m = O(k \log k) + 2t$. A detailed comparison is shown in Table 1.

In the information-theoretic setting, the currently most communication-efficient passively secure PIR protocol for $t \geq 2$ is the 3^t-server protocol in [10], which has sub-polynomial communication and computational complexity $N^{o(1)} \cdot 3^{t+o(t)}$ in the database size N. (Although the original protocols in [10, 16] assume non-colluding servers, i.e., $t = 1$, the corruption threshold t can be amplified by using the technique in [7] as pointed out in [37].) By applying our compilers, we obtain actively secure $3^{t+o(t)}$-server PIR protocols whose computational complexity is $N^{o(1)} \cdot 2^{O(t)}$. It exponentially (in t) improves the complexities $N^{o(1)} \cdot 3^{t+o(t)} \cdot m^{O(t)} = N^{o(1)} \cdot 2^{O(t^2)}$ of actively secure protocols that are obtained from the previous compilers [11, 34]. In the computational setting, if we apply our compilers to the protocol assuming one-way functions [16], we can achieve logarithmic communication and computational complexity in N and reduce the number of servers. A detailed description is shown in Table 2.

Table 1. Comparison of passive-to-active compilers for PIR

Method	Multiplicative overhead to client-side computation	# servers	# rounds
[11]	$m^{O(t)}$	$k + 2t$	1
[34]	$m^{O(t)}\lambda$	$k + t$	1
Ours (Thm. 3)	$O(m^5\lambda)$	$k + t$	$O(m^2)$
Ours (Thm. 5)	$O(tm^2)$	$O(k \log k) + 2t$	1

k and m denote the numbers of servers in passively secure and actively secure protocols, respectively, t denotes a corruption threshold, and λ denotes a security parameter.

Bounded-degree Multivariate Polynomials. In the information-theoretic setting, there is a passively secure protocol for polynomials in [50], which can be made actively secure by using the technique in [43]. In the computational setting, a passively secure protocol is given in [28], which can be made actively secure by the standard error correction algorithm [47] (see Appendix C for details). Now, by applying our compilers, we can reduce the required number of servers of these protocols by t. Based on the passively secure protocol in [41], we can

Table 2. Comparison of PIR protocols with sub-polynomial communication and computational complexity in the database size N for a corruption threshold $t \geq 2$

	Method	Client-side computation	# servers	# rounds	Assumption
Passive	[10] + [7]	$N^{o(1)}2^{O(t)}$	3^t	1	IT
	[16] + [7]	$\log N \cdot 2^{O(t)}\lambda^{O(1)}$	2^t	1	OWF
Active	[11] + [7] + [10]	$N^{o(1)} \cdot 2^{O(t^2)}$	$3^t + 2t$	1	IT
	[11] + [7] + [16]	$\log N \cdot 2^{O(t^2)}$	$2^t + 2t$	1	OWF
	[34] + [7] + [10]	$N^{o(1)} \cdot 2^{O(t^2)}\lambda$	$3^t + t$	1	IT
	[34] + [7] + [16]	$\log N \cdot 2^{O(t^2)}\lambda^{O(1)}$	$2^t + t$	1	OWF
	Thm. 3 + [7] + [10]	$N^{o(1)} \cdot 2^{O(t)}\lambda$	$3^t + t$	$O(3^{2t})$	IT
	Thm. 3 + [7] + [16]	$\log N \cdot 2^{O(t)}\lambda^{O(1)}$	$2^t + t$	$O(2^{2t})$	OWF
	Thm. 5 + [7] + [10]	$N^{o(1)} \cdot 2^{O(t)}$	$O(t3^t)$	1	IT
	Thm. 5 + [7] + [16]	$\log N \cdot 2^{O(t)}$	$O(t2^t)$	1	OWF

λ denotes a security parameter. "IT" stands for "information-theoretic" (i.e., no cryptographic assumption is necessary) and "OWF" stands for "one-way functions."

further reduce the number of servers by a factor of d assuming homomorphic encryption for degree-d polynomials. Notably, our protocol instantiated from [28] achieves the minimum number of servers $2t + 1$.[6] A detailed comparison is shown in Table 3.

Constant-depth Circuits. Barkol and Ishai [5] proposed a passively secure protocol for unbounded fan-in constant-depth circuits (i.e., the complexity class AC^0). It can be made actively secure by applying the error correction algorithm [47], and the resulting protocol needs at least $(\frac{1}{2}(\log M + O(1))^{D-1} + 2)t$ servers, where M and $D = O(1)$ are the size and depth of circuits, respectively. On the other hand, if we apply our first compiler, we need only $(\frac{1}{2}(\log M + O(1))^{D-1} + 1)t$ servers, which decreases the number of servers of [5] by t. For example, for the partial match problem on an M-sized database (which can be captured by depth-2 circuits of size M), our protocol requires only $(\log M + 2.5)t$ servers while the protocol obtained from [5] requires $(\log M + 3.5)t$ servers.

A beneficial consequence is that our compilers can be directly applied to future developments in passively secure protocols in the client-servers scenario and may yield new efficient constructions of actively secure protocols.

[6] The active security is impossible if the majority of servers are corrupted, i.e., $m \leq 2t$, since there is an attack for corrupted servers to replace their input f with another function f'.

Table 3. Comparison of actively secure protocols for multivariate polynomials

Method	# servers	# rounds	Assumption
[43] + [50]	$\left(\frac{D}{2}+2\right)t+1$	1	IT
Thm. 3 + [50]	$\left(\frac{D}{2}+1\right)t+1$	$O((Dt)^2)$	IT
Thm. 3 + [41]	$\left(\frac{D}{d+1}+1\right)t+1$	$O((Dt)^2)$	d-HE
[47] + [28]	$3t+1$	1	sparse LPN
Thm. 3 + [28]	$2t+1$	$O(t^2)$	sparse LPN

D denotes the degree of the polynomials and t denotes a corruption threshold. "IT" stands for "information-theoretic" and "d-HE" stands for homomorphic encryption for polynomials of degree d.

1.2 Related Work

Passive-to-Active Compilers. Within the context of PIR, there are compilers from a passively secure k-server protocol to an actively secure m-server protocol for $m = k + 2t$ [11] and $m = k + t$ [34]. As said above, however, these compilers are not only less generic in that they are applied only to PIR, but also inefficient since they incur exponential overhead $\binom{m}{t} = m^{O(t)}$ in computational complexity. There are also passive-to-active compilers in a more general multi-client setting where a private input is arbitrarily distributed among multiple clients [27, 39, 40, 17]. However, in actively secure protocols resulting from these compilers, servers need to interactively evaluate a circuit gate by gate. Consequently, protocols require communication and computational complexity that is proportional to the size of a function, and do not work efficiently if the function encodes a large database.

PIR. There are direct constructions of m-server PIR protocols in a malicious setting [38, 29, 3, 43, 54]. However, the communication complexities of [38, 29, 3, 43] are all polynomial $N^{O(t/m)}$ in the database size N, while those of ours are $N^{o(1)}$, i.e., smaller than any polynomial function in N. The protocol in [54] does not guarantee privacy if malicious servers collude, and thus does not satisfy active t-security for $t > 1$ in our sense[7]. There are also constructions of PIR with a weaker security guarantee [25, 22], which can only tell a client the existence of malicious servers. Actively secure PIR is also considered in a special setting where the length of each entry of a database is sufficiently large (e.g., [4, 51] and references therein). The protocols in [4, 51] assume that the length of each entry of a database is at least exponential in N and hence result in exponentially large communication complexity in N.

Protocols in the Single-server Setting. Generally, if we have a passively secure single-server protocol, then we can obtain an actively secure protocol with the minimum number of servers $2t + 1$ since a client just runs the passively secure

[7] We mean by t-security that at most t servers are corrupted.

protocol with each server and computes the majority of $2t+1$ outputs. However, there is an impossibility result on efficient single-server protocols for PIR in the information-theoretic setting [24]. Even if we go for computational security, it seems to be impossible to construct single-server PIR protocols from the minimal assumption of one-way functions [30], and for a general function, we currently need to assume fully homomorphic encryption, which is only instantiated from a narrow class of assumptions [36, 49].

Verifiable Computation. The problem of dealing with malicious servers has also been considered within the context of verifiable computation in the single-server setting [35, 23, 2] and in the multi-server setting [1, 52, 53]. However, these verifiable computation protocols only detect malicious behavior of servers and cannot achieve active security in our sense. The protocol in [20] uses a similar idea that a client compares answers from one server with those from another. However, it does not consider the setting where a client's input x should be private and also assumes that all parties agree on a function f in advance, while in our setting the client does not know f since it corresponds to an unknown database.

1.3 Publication Note

This paper is the full version of a paper at EUROCRYPT 2024 with the same title. A preliminary version of the paper has previously appeared in a preprint by the same authors (ePrint report 2023/210). The current version generalizes part of the techniques and presents more formal security proofs.

2 Technical Overview

In this section, we provide an overview of our compilers to construct an actively t-secure m-server protocol Π' from any one-round passively t-secure k-server protocol Π such that $k < m$. Let $V = \{\mathsf{S}_1, \ldots, \mathsf{S}_m\}$ be the set of m servers of Π'. A key observation behind our constructions is that if a pair of servers in V return different answers to the same query of Π, then at least one of them is malicious[8]. We call such two servers a *conflicting pair*. The client continues to remove conflicting pairs from V in an appropriate way. Finally, the client executes a protocol only with remaining honest servers and obtains a correct result. For ease of exposition, we first explain our non-interactive actively secure protocol and then explain our interactive protocol with fewer servers.

2.1 Non-interactive Actively Secure Protocols

As a first attempt, we consider the following basic construction.

1. A client C partitions $V = \{\mathsf{S}_1, \ldots, \mathsf{S}_m\}$ into k groups $V = G_1 \cup \ldots \cup G_k$ in such a way that each G_j contains at least one honest server.

[8] Here, we assume that the server-side computation is deterministic.

2. C computes k queries of Π on his private input and sends the j-th query to all servers in the j-th group G_j.

If every group contains no conflicting pair (i.e., all servers in each G_j return the same answer), then C can compute the correct result from the k answers of the k groups. Otherwise, C removes a conflicting pair from V, and repeats the above process at most t times to remove all malicious servers. This method, however, requires a large number of servers $m = \Omega(kt)$ since the size of each group G_j needs to be larger than t.

We reduce the number of servers to $m \approx 2t + k$ by introducing a novel notion of locally surjective map families. Technically, we consider a family $\mathcal{F}$ of maps from the set $V = \{\mathsf{S}_1, \dots, \mathsf{S}_m\}$ of m servers to $[k] := \{1, 2, \dots, k\}$. Each map $f \in \mathcal{F}$ defines a partition $V = G_{f,1} \cup \cdots \cup G_{f,k}$, where $G_{f,j} = \{\mathsf{S}_i : f(\mathsf{S}_i) = j\}$. For each map $f \in \mathcal{F}$, the client C computes k queries of Π on his private input and sends the j-th query to all servers in the j-th group $G_{f,j}$. Our strategy is that C proceeds in t steps to detect and remove at least one new malicious server per step. In each step,

- If for every (f, j), all the remaining servers in $G_{f,j}$ return the same answer $\mathsf{ans}_{f,j}$, then C computes an output x_f of Π from $(\mathsf{ans}_{f,1}, \dots, \mathsf{ans}_{f,k})$ for each f and decides the final output by the majority vote over the x_f's;
- Otherwise, i.e., if two remaining servers in some $G_{f,j}$ give different answers, then C removes this conflicting pair and proceeds to the next step.

Observe that in the latter case, at least one of the two servers is malicious and hence at least one malicious server is always removed. The requirement for C to succeed is that in the former case, more than half of the x_f's are correct. A sufficient condition is that for more than half of the f's, there remains at least one honest server in each of $G_{f,1}, \dots, G_{f,k}$. Indeed, for such f's, C receives the correct answer from servers in each of $G_{f,1}, \dots, G_{f,k}$, or proceeds to the latter case and removes a conflicting pair. Since there remains at least $m - 2t$ honest servers at every step, the condition can be formulated as the family $\mathcal{F}$ of maps satisfying that for any subset $H \subseteq V$ of size $m - 2t$, there exist more than half of the f's such that $f(H) = [k]$. We name such a family as a *locally surjective map family.*

We can prove by a probabilistic argument the existence of a locally surjective map family $\mathcal{F}$ of size $O(m)$ if $k = O((m - 2t)/\log(m - 2t))$. Therefore, we can obtain an actively t-secure m-server protocol Π' from a passively t-secure k-server protocol Π if $m = O(k \log k) + 2t$. Since the client can run all instances of Π in parallel, the resulting protocol Π' is one-round and only incurs a $O(tm|\mathcal{F}|) = O(tm^2)$ multiplicative overhead to communication and computational complexity.

2.2 Interactive Actively Secure Protocols

We further reduce the number of servers from $m = O(k \log k) + 2t$ to $m = t + k$. In our first construction, if a client C finds a conflicting pair of servers, then he removes both servers from the set V. After eliminating all t malicious servers, the number of remaining servers is reduced to $m - 2t$ in the worst case. Therefore, as long as this approach is used, the number of servers must be $m \geq 2t + k$ since it should hold that $m - 2t \geq k$.

Our second construction reduces the required number of servers to $m = t + k$ by introducing a notion of t-*conflict-finding* protocols, which is an intermediate notion between passively t-secure protocols and actively t-secure ones. Intuitively, in a conflict-finding protocol, a client C obtains a correct result or a non-trivial partition (G_0, G_1) of the set V of servers such that all honest servers are included in G_0 or G_1 (and hence the other group consists of malicious servers only)[9]. A pair of servers crossing the partition (G_0, G_1) is supposed to be conflicting.

More concretely, we consider a graph $\mathcal{G}$ with m vertices each of which represents a server. Our protocol starts with $\mathcal{G}$ being a complete graph, and repeats the following steps:

1. The client C executes a conflict-finding protocol Π_{CF} with some subset $V' \subseteq V$ which forms a connected subgraph of size $k = m - t$ in $\mathcal{G}$ (which can be efficiently found).
2. If all servers in V' behave honestly, then C obtains the correct output.
3. Otherwise C can find a partition (G_0, G_1) of V' thanks to the conflict-finding property of Π_{CF}. Note that there is always an edge $e = (\mathsf{S}_i, \mathsf{S}_j)$ between G_0 and G_1 since $G_0 \cup G_1 = V'$ is connected. Furthermore, since all honest servers in V' are included in G_0 or G_1, at least one of S_i and S_j is malicious. Now, C removes the edge e from $\mathcal{G}$ instead of eliminating the two servers, and goes back to the first step.

Since all edges among honest servers remain unremoved (and hence the set of all honest servers remains connected), C can successfully find a set of $k = m - t$ honest servers within $O(m^2)$ rounds. Note that in the above construction, C chooses a set of servers with which he executes Π_{CF}, depending on the answers that are maliciously computed in the previous rounds. Thus servers may learn some information on the client input x by seeing which servers C removes. To address this problem, we impose an additional property that the distribution of the partition (G_0, G_1) is independent of x regardless of how malicious servers behave. Then, an edge removed in each round leaks no information on x and hence the privacy of x is preserved.

Two-round Conflict-finding Protocols. The remaining problem is how to construct conflict-finding protocols. We show a construction of a two-round t-conflict-finding k-server protocol Π_{CF} from a passively t-secure k-server one Π. For simplicity, let $V' = (\mathsf{S}_1, \ldots, \mathsf{S}_k)$. In the first round, a client computes real queries $(\mathsf{que}_i)_{i\in[k]}$ on his private input x according to Π as usual. He also computes dummy queries $(\mathsf{que}'_i)_{i\in[k]}$ on a default input x_{def} which is independent

[9] We say that a partition (G_0, G_1) is non-trivial if neither of G_0 nor G_1 is empty.

of x. He then sends a random permutation of $(\mathsf{que}_i, \mathsf{que}'_i)$ to each server S_i. Note that the privacy of Π and the random permutation ensure that servers cannot distinguish which queries are computed on x or x_{def}. Each server S_i returns answers $(\mathsf{ans}_i, \mathsf{ans}'_i)$ to the two queries as usual. In the second round, the client sends all the dummy queries $(\mathsf{que}'_i)_{i\in[k]}$ on x_{def} to all servers in V', which does not affect privacy since x_{def} is independent of x. In response, each server S_j returns $v_j := (\mathsf{ans}'_i(j))_{i\in[k]}$ to $(\mathsf{que}'_i)_{i\in[k]}$, where $\mathsf{ans}'_i(j)$ is the answer which S_i would compute to the dummy query que'_i.

For simplicity, suppose that S_1 is the only malicious server. If S_1 behaved honestly in the first round, it holds that $\mathsf{ans}'_1 = \mathsf{ans}'_1(2) = \cdots = \mathsf{ans}'_1(k)$. If S_1 returned an incorrect answer to que'_1, it is different from any of $\mathsf{ans}'_1(2), \dots, \mathsf{ans}'_1(k)$. From this observation, the client C trusts the answer ans_1 of S_1 to the main query que_1 in the first round if and only if $\mathsf{ans}'_1 = \mathsf{ans}'_1(2) = \cdots = \mathsf{ans}'_1(k)$. Generalizing this, we let C compute an output based on $(\mathsf{ans}_1, \mathsf{ans}_2, \dots, \mathsf{ans}_k)$ if all the v_j's take the same value. Otherwise, he partitions the set of servers into equivalence classes by placing S_i and S_j into the same class if and only if $v_i = v_j$, and outputs a non-trivial partition (G_0, G_1) in some way. Note that any pair of honest servers $\mathsf{S}_i, \mathsf{S}_j$ return the same answer in the second round, i.e., $v_i = v_j$, and hence they are placed in the same class. A malicious server successfully submits an incorrect answer without being detected only if it guesses correctly which query encodes the client's true input x. As we said above, it happens with probability $1/2$. More generally, if the client prepares $M - 1$ sets of dummy queries, the cheating probability of malicious servers can be reduced to $1/M$. This can be made even negligible by executing sufficiently many (say, κ) instances in parallel. If a conflict is found in some instance, C outputs a non-trivial partition (G_0, G_1) obtained in that instance. Otherwise, he outputs the majority of the κ outputs if it exists. To let this protocol fail, malicious servers need to let the client output valid but incorrect outputs in at least $\kappa/2$ instances. The cheating probability is thus $O(M^{-\kappa/2})$, which is negligible.

To see that the partition (G_0, G_1) leaks no information on the client's input x, observe that (G_0, G_1) is determined by answers $(v_j)_{j\in[k]}$. These answers are independent of x since they can be simulated from dummy queries and t queries for x that malicious servers see. The former is independent of x in the first place and the latter leaks no information on x due to the privacy of Π.

To summarize, we obtain an $O(m^2)$-round actively t-secure m-server protocol from a passively t-secure k-server protocol if $k \le m - t$. The communication and computational overhead is a multiplicative polynomial factor in m.

3 Preliminaries

Notations. For $m \in \mathbb{N}$, define $[m] = \{1, 2, \dots, m\}$. Let X, Y be sets. If $X \subseteq Y$, we define $Y \setminus X = \{y \in Y : y \notin X\}$ and simply denote it by $\overline{X}$ if Y is clear from the context. We write $u \leftarrow_{\$} X$ if u is chosen uniformly at random from X. Define $\binom{X}{k}$ as the set of all subsets of X of size k. Define $\mathrm{Map}(X, Y)$ as the set of all maps from X to Y. If $X = [m]$ and $Y = [k]$, we simply denote it by $\mathrm{Map}(m, k)$.

Let $\log x$ denote the base-2 logarithm of x and $\ln x$ denote the base-e logarithm of x, where e is the Napier's constant. We call a function $f : \mathbb{N} \ni \lambda \mapsto f(\lambda) \in \mathbb{R}$ *negligible* if for any $c > 0$, there exists $\lambda_0 \in \mathbb{N}$ such that $0 \leq f(\lambda) < \lambda^{-c}$ for any $\lambda > \lambda_0$. We call f *polynomial* there exists $c > 0$ such that $0 \leq f(\lambda) < \lambda^c$ for all λ. Throughout the paper, we use the following notations:

- m denotes the total number of servers, which is polynomial in a security parameter λ.
- t denotes the number of corrupted servers.
- C denotes a client and S_i denotes the i-th server.

The notation $\widetilde{O}(\cdot)$ hides a polylogarithmic factor in a security parameter λ.

3.1 Secure Computation in the Client-Servers Setting

We follow the client-servers model used in [5]. In this model, there is an honest client C who holds a private input x, and m servers $\mathsf{S}_1, \ldots, \mathsf{S}_m$ who all hold the same input p. The goal is:

Byzantine-robustness. The client learns the value $F(p, x)$ for a publicly known function F even if t servers behave maliciously;

Privacy. The client keeps his input x hidden from any collusion of t servers.

We do not assume any interaction between servers. We call a message from a client to servers a *query* and a message from servers to the client an *answer*.

In the above setting, we assume that the function F takes a common input p from servers. Typically, the input p will be a description of a function f applied to the input x of a client (e.g., a description of a circuit or a polynomial) and F is the universal function defined by $F(p, x) = f(x)$.

If $m \geq 2t + 1$, there is a trivial 1-round protocol achieving the above goal: C downloads p from all servers, finds the correct p by the majority vote, and computes $F(p, x)$ by himself. However, this protocol results in large communication and client-side computational complexity that is linear in the description length of p. In applications to database search, p encodes a large database and its size is proportional to the database size. From this point of view, we say that a protocol is *efficient* if its communication and client-side computational complexity is sublinear in the description length of p and linear in that of x.

More formally, we define a secure computation protocol in the client-servers setting as an abstract primitive. First, we show the syntax and correctness.

Definition 1. *Let $\mathcal{P} = (P_\lambda)_{\lambda \in \mathbb{N}}$, $\mathcal{X} = (X_\lambda)_{\lambda \in \mathbb{N}}$, and $\mathcal{Y} = (Y_\lambda)_{\lambda \in \mathbb{N}}$ be sequences of sets with polynomial-size descriptions and $\mathcal{F} = (F_\lambda : P_\lambda \times X_\lambda \to Y_\lambda)_{\lambda \in \mathbb{N}}$ be sequences of functions with polynomial-size descriptions. An* ℓ-round m-server protocol for *$\mathcal{F}$ is a tuple of three polynomial-time algorithms $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$, where:*

– $\mathsf{Query}(1^\lambda, x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]}) \to ((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)})$: Query *is a possibly randomized algorithm that takes* $x \in X_\lambda$, *a state* $\mathsf{st}^{(j-1)}$ *and answers* $(\mathsf{ans}_i^{(j-1)})_{i\in[m]}$ *in round* $j-1$ *as input, and outputs queries* $(\mathsf{que}_i^{(j)})_{i\in[m]}$ *and a state* $\mathsf{st}^{(j)}$ *in round* j, *where we define* $\mathsf{st}^{(0)}$, $\mathsf{ans}_i^{(0)}$ *as the empty string;*
– $\mathsf{Answer}(1^\lambda, p, \mathsf{que}_i^{(j)}) \to \mathsf{ans}_i^{(j)}$: Answer *is a deterministic algorithm that takes* $p \in P_\lambda$ *and a query* $\mathsf{que}_i^{(j)}$ *in round* j *as input, and outputs an answer* $\mathsf{ans}_i^{(j)}$ *in round* j*;*
– $\mathsf{Output}(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}) \to y$: Output *is a possibly randomized algorithm that takes a state* $\mathsf{st}^{(\ell)}$ *and answers* $(\mathsf{ans}_i^{(\ell)})_{i\in[m]}$ *in round* ℓ *as input, and outputs* $y \in Y_\lambda$*;*

satisfying the following property:

Correctness. *There exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any* $\lambda \in \mathbb{N}$ *and any* $(p, x) \in P_\lambda \times X_\lambda$,

$$\Pr\left[\mathsf{Output}(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}) = F_\lambda(p, x)\right] \geq 1 - \mathsf{negl}(\lambda),$$

where

$$((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]}),$$
$$\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(1^\lambda, p, \mathsf{que}_i^{(j)})$$

for all $j \in [\ell]$ *and* $i \in [m]$.

We note that an answer algorithm Answer is not always defined to be deterministic in the literature but all the instantiations considered in this paper actually have deterministic answer algorithms. We omit a security parameter 1^λ from inputs if it is clear from the context.

An abstract primitive $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ immediately implies an ℓ-round protocol in the above client-servers setting. Indeed, a client has a private input x and m servers have a common input p. In each round, the client runs Query, sends queries to servers and stores a state in his memory. In response, servers run Answer on the queries that they receive, and send answers back to the client. In the final round, the client runs Output on his state and servers' answers, and obtains $y = F_\lambda(p, x)$. Due to this correspondence, we will use the terminologies interchangeably for the sake of readability.

The above-mentioned trivial 1-round protocol corresponds to the scheme in which Query outputs nothing, Answer outputs p and then Output computes $y = F_\lambda(p, x)$. To rule out this, we define the efficiency measures of Π as follows. Let $\mathsf{que}_i^{(j)}$ and $\mathsf{ans}_i^{(j)}$ be queries and answers computed by Π and denote their bit-lengths by $|\mathsf{que}_i^{(j)}|$ and $|\mathsf{ans}_i^{(j)}|$, respectively. Define the *communication complexity* $\mathrm{Comm}_\lambda(\Pi)$ as

$$\mathrm{Comm}_\lambda(\Pi) = \sup_{(p,x)\in P_\lambda\times X_\lambda} \sum_{i\in[m], j\in[\ell]} (|\mathsf{que}_i^{(j)}| + |\mathsf{ans}_i^{(j)}|).$$

Define the *client-side computational complexity* $\text{c-Comp}_\lambda(\Pi)$ as the sum of the running time of $\mathsf{Query}(1^\lambda, x, \cdot, \cdot)$ and $\mathsf{Output}(1^\lambda, \cdot, \cdot)$ with worst-case inputs $(p, x) \in P_\lambda \times X_\lambda$. Let $\text{Comm}(\Pi) = (\text{Comm}_\lambda(\Pi))_{\lambda\in\mathbb{N}}$ and $\text{c-Comp}(\Pi) = (\text{c-Comp}_\lambda(\Pi))_{\lambda\in\mathbb{N}}$. We say that Π is *efficient* if there exists a sublinear function $g(\ell) = o(\ell)$ such that

$$\max\{\text{Comm}(\Pi), \text{c-Comp}(\Pi)\} \in g(|p|) \cdot |x| \cdot \mathsf{poly}(m, \lambda)\,, \tag{1}$$

where $|p|$ and $|x|$ are the description lengths of elements of P_λ and X_λ, respectively. One can also define the *server-side computational complexity* $\text{s-Comp}_\lambda(\Pi)$ as the running time of $\mathsf{Answer}(1^\lambda, p, \cdot)$, and define $\text{s-Comp}(\Pi) = (\text{s-Comp}_\lambda(\Pi))_{\lambda\in\mathbb{N}}$. We see the communication and client-side complexity as a primary efficiency measure and the server-side computational complexity as a secondary measure.

Next, we show the security requirements.

Definition 2. *Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be an ℓ-round m-server protocol for $\mathcal{F} = (F_\lambda : P_\lambda \times X_\lambda \to Y_\lambda)_{\lambda\in\mathbb{N}}$. We say that Π is* actively t-secure *if it satisfies the following requirements:*

Privacy. *There exists a negligible function $\mathsf{negl}(\lambda)$ such that for any stateful algorithm $\mathcal{A}$ and any $\lambda \in \mathbb{N}$,*

$$\mathsf{Adv}_{\Pi,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Priv}^0_{\Pi,\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{Priv}^1_{\Pi,\mathcal{A}}(\lambda) = 0\right]\right| < \mathsf{negl}(\lambda)\,,$$

where for $b \in \{0, 1\}$, $\mathsf{Priv}^b_{\Pi,\mathcal{A}}(\lambda)$ is the output b' of $\mathcal{A}$ in the following experiment:

1. *$(x_0, x_1, p, T) \leftarrow \mathcal{A}(1^\lambda)$, where $x_0, x_1 \in X_\lambda$, $p \in P_\lambda$ and $T \subseteq [m]$ is of size at most t.*
2. *For each $j = 1, 2, \ldots, \ell$,*
   - *(a) Let $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x_b, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ and give $(\mathsf{que}_i^{(j)})_{i\in T}$ to $\mathcal{A}$.*
   - *(b) If $j < \ell$, $\mathcal{A}$ outputs $(\mathsf{ans}_i^{(j)})_{i\in T}$. If $j = \ell$, $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.*

Byzantine-robustness. *There exists a negligible function $\mathsf{negl}(\lambda)$ such that for any stateful algorithm $\mathcal{A}$ and any $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{BR}_{\Pi,\mathcal{A}}(\lambda) = 1] < \mathsf{negl}(\lambda)\,,$$

where $\mathsf{BR}_{\Pi,\mathcal{A}}(\lambda)$ is the output of the following experiment:

1. *$(x, p, T) \leftarrow \mathcal{A}(1^\lambda)$, where $x \in X_\lambda$, $p \in P_\lambda$ and $T \subseteq [m]$ is of size at most t.*
2. *For each $j = 1, 2, \ldots, \ell$,*
   - *(a) Let $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x_b, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ and give $(\mathsf{que}_i^{(j)})_{i\in T}$ to $\mathcal{A}$.*
   - *(b) $\mathcal{A}$ outputs $(\mathsf{ans}_i^{(j)})_{i\in T}$.*
3. *Return 1 if $\mathsf{Output}(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}) \neq F_\lambda(p, x)$, and otherwise return 0.*

We say that Π is *passively t-secure* if it satisfies the above requirements for *semi-honest* adversaries $\mathcal{A}$, i.e., those following the instructions of Π. Note that for semi-honest adversaries, the Byzantine-robustness of Π immediately follows from the correctness of Π. We say that Π is *computationally actively t-secure* (resp. *computationally passively t-secure*) if it satisfies the above requirements for probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ (resp. semi-honest PPT adversaries $\mathcal{A}$).

3.2 Existing Passively Secure Protocols

Private Information Retrieval. Let $N = N(\lambda)$ be a polynomial function. Define $\textsc{Index}_N = (F_\lambda : \{0,1\}^N \times [N] \to \{0,1\})_{\lambda \in \mathbb{N}}$ as a sequence of functions such that for each $\lambda \in \mathbb{N}$,

$$F_\lambda((a_1, \ldots, a_N), x) = a_x,\ \forall (a_1, \ldots, a_N) \in \{0,1\}^N, \forall x \in [N].$$

An m-server protocol for $\textsc{Index}_N$ is called an m-server private information retrieval (PIR) protocol for N-sized databases. In the information-theoretic setting, the most communication-efficient passively secure 3-server PIR protocol was given by [10] and in the computational setting, the passively secure 2-server PIR protocol was given by [16] assuming the existence of one-way functions. Although the original protocols in [10, 16] assume $t = 1$, the corruption threshold t can be amplified by using the technique in [7] as pointed out in [37] (see Appendix A for details). More specifically, the following propositions hold.

Proposition 1. *There exists a passively t-secure 1-round 3^t-server protocol Π for $\textsc{Index}_N$ such that*

- $\mathrm{Comm}(\Pi) = \exp(O(\sqrt{\log N \log\log N})) \cdot t3^t = N^{o(1)} \cdot 2^{O(t)}$;
- $\text{c-Comp}(\Pi) = \exp(O(\sqrt{\log N \log\log N})) \cdot t3^t = N^{o(1)} \cdot 2^{O(t)}$;
- $\text{s-Comp}(\Pi) = N^2 \cdot \exp(O(\sqrt{\log N \log\log N})) \cdot 2^t = N^{2+o(1)} \cdot 2^t$.

Note that the above protocol satisfies the efficiency requirement (1) since $\mathrm{Comm}(\Pi)$ and $\text{c-Comp}(\Pi)$ are sub-polynomial (i.e., less than any polynomial) in the description length N of elements of $P_\lambda = \{0,1\}^N$.

Proposition 2. *Assume a pseudorandom generator $G : \{0,1\}^\lambda \to \{0,1\}^{2(\lambda+1)}$. There exists a computationally passively t-secure 1-round 2^t-server protocol Π for $\textsc{Index}_N$ such that*

- $\mathrm{Comm}(\Pi) = O(\log N \cdot \lambda \cdot t2^t)$;
- $\text{c-Comp}(\Pi)$ *is* $O(\log N \cdot t2^t)$ *invocations of* G;
- $\text{s-Comp}(\Pi)$ *is* $O(N^2 \log N \cdot t)$ *invocations of* G.

Remark 2. Dvir and Gopi [31] devised a technique to optimize the 3-server protocol in [32] and obtained a 2-server PIR protocol with $N^{o(1)}$ communication. However, since the answer length is not constant, the passively t-secure protocol obtained by applying the amplification technique of [7] has larger communication complexity $\exp(O(t\sqrt{\log N \log\log N}))$ and does not satisfy the efficiency requirement (1).

Bounded-Degree Polynomials. Let $N = N(\lambda)$, $D = D(\lambda)$ and $M = M(\lambda)$ be polynomial functions. We define $\textsc{Poly}_{N,D,M}(R) = (F_\lambda)_{\lambda\in\mathbb{N}}$ as a sequence of functions such that $F_\lambda(p, \mathbf{x}) = p(\mathbf{x})$ for any N-variate polynomial p over a ring R with degree D and number of monomials M, and for any $\mathbf{x} \in R^N$. The following is implicit in [50].

Proposition 3. *Let $N, D, M \in \mathsf{poly}(\lambda)$. Let R be a ring such that for any $a \in \{1, 2, \ldots, m-1\}$, an element $a \cdot 1_R$ has an inverse in R, where 1_R is the multiplicative identity of R. Suppose that $m > Dt/2$. Then, there exists a passively t-secure 1-round m-server protocol Π for $\textsc{Poly}_{N,D,M}(R)$ such that*

- $\mathrm{Comm}(\Pi)$ *is $O(Nm)$ ring elements;*
- $\text{c-Comp}(\Pi)$ *is $O(Ntm)$ ring operations;*
- $\text{s-Comp}(\Pi)$ *is $O(NMD)$ ring operations.*

Since the description length of a polynomial with M monomials is $\widetilde{O}(MD \log |R|)$, the above protocol satisfies the efficiency requirement (1) if $MD = \omega(N)$.

We also have passively secure protocols proposed within the context of homomorphic secret sharing [15]. Roughly speaking, a homomorphic secret sharing scheme for a function f is an advanced variant of secret sharing, which has an evaluation algorithm such that shares for a secret x can be locally converted into shares for a secret $f(x)$. An m-server homomorphic secret sharing scheme for a function f implies an m-server protocol for $F(f, x) = f(x)$. We formally explain the relation in Appendix B.

Ishai, Lai and Malavolta [41] showed that assuming homomorphic encryption for degree-d polynomials, the number of servers in Proposition 3 can be decreased by a factor of d. See Appendix D for the definition of homomorphic encryption.

Proposition 4. *Let $d = O(1)$ and R be a ring such that for any $a \in \{1, 2, \ldots, \max\{d, m-1\}\}$, an element $a \cdot 1_R$ has an inverse in R, where 1_R is the multiplicative identity of R. Assume a homomorphic encryption scheme HE for degree-d polynomials over R. Let $M, N \in \mathsf{poly}(\lambda)$ and $D = O(1)$. Suppose that $m > Dt/(d+1)$. Then there exists a computationally passively t-secure 1-round m-server protocol Π for $\textsc{Poly}_{N,D,M}(R)$ such that*

- $\mathrm{Comm}(\Pi) = O(Nm \cdot \ell_{\mathsf{ct}})$*, where ℓ_{ct} is the description length of ciphertexts of HE;*
- $\text{c-Comp}(\Pi) = O((Nt \cdot \tau_{\mathsf{Enc}} + \tau_{\mathsf{Dec}})m)$*, where τ_{Enc} and τ_{Dec} are the running time of the encryption and decryption algorithms of HE, respectively;*
- $\text{s-Comp}(\Pi) = O(MN \cdot \tau_{\mathsf{Eval}})$*, where τ_{Eval} is the running time of the evaluation algorithm of HE per operation.*

Note that we have $\max\{d, m-1\} = \mathsf{poly}(\lambda)$ since $d = O(1)$ and $m = \mathsf{poly}(\lambda)$. On the other hand, homomorphic encryption schemes mentioned in [41] assume that R is a prime field of size q or a ring of integers modulo $n = q_1q_2$ for exponentially large primes q, q_1, q_2. In these cases, $a \cdot 1_R$ has an inverse in R if $a \in \{1, 2, \ldots, \max\{d, m-1\}\}$.

Under the sparse Learning Parity with Noise (LPN) assumption over a field $\mathbb{F}_q$, Dao et al. [28] proposed a passively t-secure $(t+1)$-server protocol for polynomials of degree $D = O(\log\lambda/\log\log\lambda)$. Although the original protocol does not have sublinear-size upload cost when evaluating a single polynomial, it can be seen that the upload cost is amortized if sufficiently many polynomials are evaluated on the same input. Specifically, let $N = N(\lambda)$, $D = D(\lambda)$, $M = M(\lambda)$, and $L = L(\lambda)$ be polynomial functions. We define $\textsc{Poly}^L_{N,D,M}(R) = (F_\lambda)_{\lambda\in\mathbb{N}}$ as a sequence of functions such that $F_\lambda((p_1,\ldots,p_L),\mathbf{x}) = (p_1(\mathbf{x}),\ldots,p_L(\mathbf{x}))$ for any N-variate polynomials $p_1,\ldots,p_L$ over a ring R with degree D and number of monomials M, and for any $\mathbf{x}\in R^N$.

Proposition 5. *Assume that the (δ,q)-sLPN assumption holds for a constant $0\le\delta\le 1$ and a sequence $q = (q(\lambda))_{\lambda\in\mathbb{N}}$ of prime powers that are computable in polynomial time in λ. Let $L,M,N\in\mathsf{poly}(\lambda)$ and $D = O(\log\lambda/\log\log\lambda)$. Then, there exists a computationally passively t-secure 1-round $(t+1)$-server protocol Π for $\textsc{Poly}^L_{N,D,M}(\mathbb{F}_q)$ such that*

- $\mathrm{Comm}(\Pi) = \widetilde{O}((M^{2/\delta}N + L)(\log q)m\lambda)$;
- $\text{c-Comp}(\Pi) = \widetilde{O}((M^{2/\delta}N + L)(\log q)m\lambda)$;
- $\text{s-Comp}(\Pi) = \widetilde{O}(M^{1/\delta+1}L(\log q)\lambda)$.

Note that the description length of L polynomials each with M monomials is $\widetilde{O}(ML\log q)$ if the degree is $D = o(\log\lambda)$. Thus, if $L = \omega(M^{2/\delta-1})$, the above protocol satisfies the efficiency requirement (1). See [28] or Appendix C for the details including the definition of the sparse LPN assumption.

Constant-Depth Circuits. We consider Boolean circuits of constant depth with unbounded fan-in and fan-out. Formally, a Boolean circuit C is a labelled directed acyclic graph. The nodes with no incoming edges are labelled with input variables, their negations, or constants. The other nodes are called gates and are labelled with one of operators in $\{\mathsf{AND},\mathsf{OR},\mathsf{NOT}\}$. Nodes with no outgoing edges are called output nodes. We only consider a circuit with a single output node. The size of a circuit is the number of edges and its depth is the length of the longest path from an input node to the output node. We define the output of C on input x, which we denote by $C(x)$, as the value of the output node after input values proceed through a sequence of gates.

Let $N = N(\lambda)$, $D = D(\lambda)$ and $M = M(\lambda)$ be polynomial functions. We define $\textsc{Circ}_{N,D,M} = (F_\lambda)_{\lambda\in\mathbb{N}}$ as a sequence of functions such that $F_\lambda(C,x) = C(x)$ for any Boolean circuit C with N input variables, depth D and size M, and for any N-bit string x.

Proposition 6. *Let $N,M\in\mathsf{poly}(\lambda)$ and $D = O(1)$. Suppose that $m \ge (\log M + 3)^{D-1}t/2$. Then, there exists a passively t-secure 1-round m-server protocol Π for $\textsc{Circ}_{N,D,M}$ such that*

- $\mathrm{Comm}(\Pi) = O((\log M)^{D-1}N(\log N)\lambda m)$;

– c-Comp(Π) $= O((\log M)^{D-1} N (\log N) tm + (\log M)^2 \lambda m)$;
– s-Comp(Π) $= O(M (\log M) N \lambda)$.

The protocol is efficient since Comm(Π) and c-Comp(Π) are linear in N and polylogarithmic in the size M of circuits, omitting factors in λ and m.

4 Interactive Actively Secure Protocols

In this section, we show our compiler from one-round passively t-secure k-server protocols to $O(m^2)$-round actively t-secure m-server protocols such that $m \geq k + t$. To this end, we introduce a notion of *conflict-finding* protocols, which is an intermediate notion between passively secure and actively secure protocols. We show a generic compiler from conflict-finding to actively secure protocols in Section 4.3 and then show a generic compiler from passively secure to conflict-finding protocols.

4.1 Graph Theory

To begin with, we recall the standard terminology of graph theory (see [42, Chapter 2] for instance). A (simple and undirected) graph $\mathcal{G}$ is a pair (V, E), where V is a set of vertices and E is a set of edges $(i, j) \in V \times V$. Throughout the paper, we only consider the cases where V is either $[m]$ or a subset of $[m]$. Thus we may assume that V is a totally ordered set. The total order on V naturally induces a lexicographic order on E, which is also a total order on E. A graph $\mathcal{G}$ is called *connected* if there is a path between each pair of vertices. It is a standard result that there is a deterministic algorithm $\mathcal{D}$ which decomposes $\mathcal{G}$ into connected components in time $O(|V| + |E|)$ [42]. For $S \subseteq V$, we denote by $\mathcal{G}[S]$ the induced subgraph, i.e., the graph whose vertex set is S and whose edge set consists of the edges in E that have both endpoints in S.

We show a deterministic algorithm $\mathcal{C}'_k$ such that for any connected graph $\mathcal{G} = (V, E)$ with at least k vertices, $\mathcal{C}'_k(\mathcal{G})$ outputs a subset $S \subseteq V$ such that $|S| = k$ and $\mathcal{G}[S]$ is connected. First, $\mathcal{C}'_k$ chooses the minimum node s of V with respect to the total order on V. Secondly, $\mathcal{C}'_k$ runs the "textbook" depth-first search algorithm [42] starting at the vertex s, except that it stops searching if it visits k vertices. Finally, $\mathcal{C}'_k$ outputs the set S of all vertices it visited so far. By definition, S is of size k. Since any pair of vertices in S are connected via s, $\mathcal{G}[S]$ is connected. The running time of $\mathcal{C}'_k$ is $O(|V| + |E|)$.

Next, we show a deterministic algorithm $\mathcal{C}_k$ such that for any graph $\mathcal{G} = (V, E)$, if $\mathcal{G}$ contains a connected component of size at least k, $\mathcal{C}_k(\mathcal{G})$ outputs a subset $S \subseteq V$ of size k such that $\mathcal{G}[S]$ is connected, and otherwise, it outputs the empty set $\emptyset$. First, $\mathcal{C}_k$ lists all the connected components of $\mathcal{G}$, $(\mathcal{G}_1, \ldots, \mathcal{G}_q) \leftarrow \mathcal{D}(\mathcal{G})$. Secondly, $\mathcal{C}_k$ lets $q_{\min}$ be the minimum index q such that $\mathcal{G}_q$ has at least k vertices. If no component has k vertices, $\mathcal{C}_k$ outputs $\emptyset$. Otherwise, $\mathcal{C}_k$ outputs $S \leftarrow \mathcal{C}'_k(\mathcal{G}_{q_{\min}})$. The correctness of $\mathcal{C}_k$ immediately follows from those of $\mathcal{D}$ and $\mathcal{C}'_k$. The running time of $\mathcal{C}_k$ is $O(|V| + |E|)$.

Finally, we show a trivial but frequently-used algorithm $\mathcal{E}$, which takes as input a graph $\mathcal{G} = (V, E)$ and a pair of disjoint non-empty subsets $G_0, G_1 \subseteq V$, and outputs the minimum edge $e = (i, j) \in E$ (with respect to the total order on E) such that $i \in G_0$ and $j \in G_1$, or $j \in G_0$ and $i \in G_1$. The running time of $\mathcal{E}$ is $O(|E|)$.

4.2 Formalization of Conflict-finding Protocols

Roughly speaking, in a conflict-finding protocol, a client obtains (y, z), where y is the main output (supposed to be $F(p, x)$) and z is an auxiliary string. The string z is either output, $\mathsf{failure}$, or a non-trivial partition (G_0, G_1) of the set of servers[10]. The security requirements are:

Soundness. The probability that $z = \mathsf{output}$ and $y \neq F(p, x)$ is negligible, and the probability that the protocol outputs $z = \mathsf{failure}$ is also negligible;

Conflict-finding. If z is a non-trivial partition (G_0, G_1) of the set of servers, then one of G_0 or G_1 contains all honest servers (and hence the other group consists of malicious servers only);

Privacy. An adversary should not learn a client's input x even if she knows z.

Intuitively, the conflict-finding property ensures that a client learns a subset of malicious servers only, which allows him to find a pair of servers such that at least one of them is malicious. We require the privacy should hold even if z is leaked, in order for an adversary not to learn additional information from a set of servers the client removes. Below, we show formal definitions.

Definition 3. *We say that $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ is an* ℓ-round t-conflict-finding m-server protocol for $\mathcal{F} = (F_\lambda : P_\lambda \times X_\lambda \to Y_\lambda)_{\lambda \in \mathbb{N}}$ *if it satisfies the following properties:*

Syntax. *The syntax of* Query *and* Answer *is the same as that of Π as an ℓ-round m-server protocol for $\mathcal{F}$ (Definition 1). The algorithm* Output *takes a state $\mathsf{st}^{(\ell)}$ and answer $(\mathsf{ans}_i^{(\ell)})_{i \in [m]}$ in round ℓ as input, and outputs (y, z) such that (1) $y \in Y_\lambda$ and $z =$* output*, (2) $y = \perp$ and $z = (G_0, G_1)$, which is a non-trivial partition of $[m]$, or (3) $y = \perp$ and $z =$* $\mathsf{failure}$. *We call the first (resp. second) component of the output of* Output *the y-output (resp. z-output).*

Correctness. *There exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any $\lambda \in \mathbb{N}$ and any $(p, x) \in P_\lambda \times X_\lambda$, it holds that*

$$\Pr\left[(y, z) \leftarrow \mathsf{Output}(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i \in [m]}) : y = F_\lambda(p, x)\right] \geq 1 - \mathsf{negl}(\lambda),$$

where

$$((\mathsf{que}_i^{(j)})_{i \in [m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i \in [m]}),$$
$$\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(1^\lambda, p, \mathsf{que}_i^{(j)})$$

for all $j \in [\ell]$ and $i \in [m]$.

[10] We say that a partition (G_0, G_1) is *non-trivial* if $G_0 \neq \emptyset$ and $G_1 \neq \emptyset$.

Soundness. *There exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any stateful algorithm* $\mathcal{A}$ *and any* $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{Sound}_{\Pi,\mathcal{A}}(\lambda) = 1] < \mathsf{negl}(\lambda)\,, \tag{2}$$

where $\mathsf{Sound}_{\Pi,\mathcal{A}}(\lambda)$ *is the output of the following experiment:*

1. $(x, p, T) \leftarrow \mathcal{A}(1^\lambda)$, *where* $x \in X_\lambda$, $p \in P_\lambda$ *and* $T \subseteq [m]$ *is of size at most* t.
2. *For each* $j = 1, 2, \ldots, \ell$,
   (a) *Let* $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ *and give* $(\mathsf{que}_i^{(j)})_{i\in T}$ *to* $\mathcal{A}$.
   (b) $\mathcal{A}$ *outputs* $(\mathsf{ans}_i^{(j)})_{i\in T}$.
3. *Let* $(y, z) \leftarrow \mathsf{Output}(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]})$.
4. *Return* 1 *if* $y \in Y_\lambda \setminus \{F_\lambda(p, x)\}$ *and* $z = \mathsf{output}$, *or* $y = \perp$ *and* $z = \mathsf{failure}$. *Otherwise return* 0.

Conflict-finding. *For any stateful algorithm* $\mathcal{A}$ *and any* $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{CF}_{\Pi,\mathcal{A}}(\lambda) = 1] = 0,$$

where $\mathsf{CF}_{\Pi,\mathcal{A}}(\lambda)$ *is the output of the following experiment:*

1. $(x, p, T) \leftarrow \mathcal{A}(1^\lambda)$, *where* $x \in X_\lambda$, $p \in P_\lambda$ *and* $T \subseteq [m]$ *is of size at most* t.
2. *For each* $j = 1, 2, \ldots, \ell$,
   (a) *Let* $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ *and give* $(\mathsf{que}_i^{(j)})_{i\in T}$ *to* $\mathcal{A}$.
   (b) $\mathcal{A}$ *outputs* $(\mathsf{ans}_i^{(j)})_{i\in T}$.
3. *Let* $(y, z) \leftarrow \mathsf{Output}(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]})$.
4. *Return* 1 *if* $z = (G_0, G_1)$, $G_0 \not\subseteq T$ *and* $G_1 \not\subseteq T$. *Otherwise return* 0.

Privacy. *There exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any stateful algorithm* $\mathcal{A}$ *and any* $\lambda \in \mathbb{N}$,

$$\mathsf{Adv}^{\mathrm{CF}}_{\Pi,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Priv}^{\mathrm{CF},0}_{\Pi,\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{Priv}^{\mathrm{CF},1}_{\Pi,\mathcal{A}}(\lambda) = 0\right]\right| < \mathsf{negl}(\lambda)\,,$$

where for $b \in \{0, 1\}$, $\mathsf{Priv}^{\mathrm{CF},b}_{\Pi,\mathcal{A}}(\lambda)$ *is the output* b' *of* $\mathcal{A}$ *in the following experiment:*

1. $(x_0, x_1, p, T) \leftarrow \mathcal{A}(1^\lambda)$, *where* $x_0, x_1 \in X_\lambda$, $p \in P_\lambda$ *and* $T \subseteq [m]$ *is of size at most* t.
2. *For each* $j = 1, 2, \ldots, \ell$,
   (a) *Let* $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x_b, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ *and give* $(\mathsf{que}_i^{(j)})_{i\in T}$ *to* $\mathcal{A}$.
   (b) $\mathcal{A}$ *outputs* $(\mathsf{ans}_i^{(j)})_{i\in T}$.
3. *Let* $(y, z) \leftarrow \mathsf{Output}(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]})$ *and give* z *to* $\mathcal{A}$.
4. $\mathcal{A}$ *outputs a bit* $b' \in \{0, 1\}$.

For a (possibly non-negligible) function $\epsilon(\lambda)$, we define a weaker notion of a ϵ-*sound* t-conflict-finding protocol Π as the one satisfying the requirements in Definition 3 except that the condition (2) is replaced with

$$\Pr[\mathsf{Sound}_{\Pi,\mathcal{A}}(\lambda) = 1] < \epsilon.$$

We say that Π is *computationally t-conflict-finding* if it satisfies the above requirements for PPT adversaries $\mathcal{A}$.

4.3 Compiler from Conflict-finding to Actively Secure Protocols

We construct an actively t-secure m-server protocol from a t-conflict-finding $(m-t)$-server protocol. We give a sketch here and defer the formal proof to Appendix E.

Theorem 1. *Suppose that there exists an ℓ-round (resp. computationally) t-conflict-finding k-server protocol Π_{CF} for $\mathcal{F} = (F_\lambda : P_\lambda \times X_\lambda \to Y_\lambda)_{\lambda \in \mathbb{N}}$. If $m \geq t + k$, there exists an $O(\ell m^2)$-round (resp. computationally) actively t-secure m-server protocol Π for $\mathcal{F}$ such that*

- $\mathrm{Comm}(\Pi) = O(m^2 \cdot \mathrm{Comm}(\Pi_{\mathrm{CF}}))$;
- $\text{c-Comp}(\Pi) = O(m^2 \cdot \text{c-Comp}(\Pi_{\mathrm{CF}}) + m^4)$;
- $\text{s-Comp}(\Pi) = O(m^2 \cdot \text{s-Comp}(\Pi_{\mathrm{CF}}))$.

Proof (sketch). Define $N := \binom{m}{2} - \binom{m-t}{2} + 1 = O(m^2)$. Let V be the set of all m servers and $\mathcal{G}^{(1)}$ be the complete graph on V. Consider the following protocol Π: For each $j = 1, 2, \ldots, N$,

1. The client C finds a k-sized subset $S^{(j)}$ of V such that $\mathcal{G}^{(j)}[S^{(j)}]$ is connected, based on the algorithm $\mathcal{C}_k$ in Section 4.1.
2. C executes the conflict-finding protocol Π_{CF} with k servers in $S^{(j)}$, and obtain an output $(y^{(j)}, z^{(j)})$.
3. If $z^{(j)} = \mathsf{output}$, then C outputs the y-output $y^{(j)}$.
4. If $z^{(j)} = \mathsf{failure}$, then C outputs any default value y_0.
5. If $z^{(j)}$ is a non-trivial partition $(G_0^{(j)}, G_1^{(j)})$ of $S^{(j)}$, then C does the following:
   (a) Find an edge $e^{(j)}$ of $\mathcal{G}^{(j)}$ crossing the partition $(G_0^{(j)}, G_1^{(j)})$ based on the algorithm $\mathcal{E}$ in Section 4.1. Such an edge exists since $\mathcal{G}^{(j)}[S^{(j)}]$ is connected.
   (b) Let $\mathcal{G}^{(j+1)}$ be a graph obtained by removing $e^{(j)}$ from $\mathcal{G}^{(j)}$.
   (c) Go back to Step 1.

Privacy. An adversary corrupting a set T of at most t servers cannot learn a client's input from interaction at Step 2 due to the fact that $|T \cap S^{(j)}| \leq |T| \leq t$ and the privacy of Π_{CF}. The adversary can also see a sequence of graphs $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(N)}$ but as shown at Step 5, the sequence is determined only by a sequence of z-outputs $z^{(1)}, z^{(2)}, \ldots, z^{(N)}$. Since Π_{CF} guarantees privacy even if z-outputs are leaked, she learns no additional information.

Byzantine-robustness. The client C outputs an incorrect result only if one of the following events occurs: (1) $z^{(j)} = \mathsf{output}$ and $y^{(j)}$ is an incorrect result for some $j \in [N]$, (2) $z^{(j)} = \mathsf{failure}$ for some $j \in [N]$, or (3) $z^{(j)}$ is a non-trivial partition for all $j \in [N]$. It follows from the soundness of Π_{CF} that the first and second cases occur only with negligible probability.

We argue that the third case never occurs. Assume otherwise, then for all j, the z-output $z^{(j)}$ of the j-th iteration is a non-trivial partition $(G_0^{(j)}, G_1^{(j)})$ of $S^{(j)}$. Since the conflict-finding property of Π_{CF} ensures that either $G_0^{(j)}$ or $G_1^{(j)}$ includes the set of honest servers $H := [m] \setminus T$, the removed edge $e^{(j)} = (i_1, i_2)$ satisfies $i_1 \in T$ or $i_2 \in T$ and hence the subgraph $\mathcal{G}^{(j)}[H]$ is a complete graph for all j. Since N is larger than the total number $N' = \binom{m}{2} - \binom{m-|T|}{2}$ of unordered pairs (i_1, i_2) such that $i_1 \in T$ or $i_2 \in T$, $\mathcal{G}^{(N')}$ has no edge $e = (i_1, i_2)$ such that $i_1 \in T$ or $i_2 \in T$. Therefore, a set of servers $S^{(N')}$ involved in the N'-th iteration is a subset of H since $k \leq m - t \leq |H|$. We have assumed that $z^{(N')}$ is a non-trivial partition $(G_0^{(N')}, G_1^{(N')})$ of $S^{(N')}$ but the conflict-finding property ensures that $H \subseteq G_0^{(N')}$ or $H \subseteq G_1^{(N')}$, which is contradiction. $\square$

4.4 Compiler from Passively Secure to Conflict-finding Protocols

First, we show a basic construction of ϵ-sound conflict-finding protocols for non-negligible ϵ. We give a sketch here and defer the formal proof to Appendix F.

Proposition 7. *Let Π be a 1-round (resp. computationally) passively t-secure m-server protocol for $\mathcal{F} = (F_\lambda : P_\lambda \times X_\lambda \to Y_\lambda)_{\lambda \in \mathbb{N}}$. Let $M = \mathsf{poly}(\lambda)$. Then, there exists a 2-round (resp. computationally) ϵ-sound t-conflict-finding m-server protocol Π' for $\mathcal{F}$ such that*

- $\mathrm{Comm}(\Pi') = O(mM \cdot \mathrm{Comm}(\Pi))$;
- $\text{c-Comp}(\Pi') = O(m^2 M \cdot \text{c-Comp}(\Pi))$;
- $\text{s-Comp}(\Pi') = O(mM \cdot \text{s-Comp}(\Pi))$;

where $\epsilon = m/M + \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\lambda)$.

Proof (sketch). Consider the following protocol Π':

First round.

1. The client C chooses μ_* uniformly at random from $[M]$.
2. For all $\mu \in [M]$, C computes queries $(\mathsf{que}_1^{\langle \mu \rangle}, \ldots, \mathsf{que}_m^{\langle \mu \rangle})$ of Π on his true input x if $\mu = \mu_*$, and on a default input x_{def} otherwise.
3. C sends the queries $(\mathsf{que}_i^{\langle \mu \rangle})_{\mu \in [M]}$ to each server S_i as usual, who returns answers $(\mathsf{ans}_i^{\langle \mu \rangle})_{\mu \in [M]}$ to them.

Second round.

1. C sends all the queries $(\mathsf{que}_k^{\langle \mu \rangle})_{k \in [m], \mu \neq \mu_*}$ for the default input x_{def} to all servers.
2. For all $k \in [m]$ and $\mu \in [M] \setminus \{\mu_*\}$, each server S_i returns an answer $\mathsf{ans}_k^{\langle \mu \rangle}(i)$ as S_k would answer to $\mathsf{que}_k^{\langle \mu \rangle}$.

To obtain an output, C defines $v_i = (\mathsf{ans}_k^{\langle\mu\rangle}(i))_{k\in[m],\mu\neq\mu_*}$ for all $i \in [m]$. For simplicity, we here assume that $\mathsf{ans}_i^{\langle\mu\rangle}(i) = \mathsf{ans}_i^{\langle\mu\rangle}$ for all $i \in [m]$. This is because otherwise, it means that a server S_i returns different answers in the first and second rounds and hence S_i is immediately found malicious. The client C partitions the set of servers into equivalence classes $G'_0, \ldots, G'_\ell$ under the equivalence relation defined as: $i \sim j \overset{\text{def}}{\iff} v_i = v_j$. If $\ell = 0$ (i.e., all servers belong to the same equivalence class), then he runs the output algorithm of Π on the answers $(\mathsf{ans}_1^{\langle\mu_*\rangle}, \ldots, \mathsf{ans}_m^{\langle\mu_*\rangle})$ to the queries for his true input. He then outputs the result y along with $z = \mathsf{output}$. If $\ell \geq 1$, then he outputs $y = \perp$ and $z = (G_0, G_1)$, where $G_0 = G'_0$ and $G_1 = G'_1 \cup \cdots \cup G'_\ell$.

Conflict-finding. Let T be a set of corrupted servers. Since honest servers $i, j \notin T$ always return the same answer to the same query, we have that $\mathsf{ans}_k^{\langle\mu\rangle}(i) = \mathsf{ans}_k^{\langle\mu\rangle}(j)$ for all $k \in [m]$ and $\mu \in [M] \setminus \{\mu_*\}$, and hence $v_i = v_j$. Therefore, the set of honest servers is contained in an equivalence class and it holds that $\overline{T} \subseteq G_0 = \overline{G_1}$ or $\overline{T} \subseteq G_1 = \overline{G_0}$.

Soundness. In the first place, the protocol Π' never outputs $z = \mathsf{failure}$. Assume that Π' outputs $z = \mathsf{output}$. Then, all servers belong to the same equivalence class, which implies that $v_i = v_j$ for any $i, j \in [m]$. To let the client accept an incorrect result, an adversary needs to let at least one corrupted server S_i submit an incorrect answer exactly to the query $\mathsf{que}_i^{\langle\mu_*\rangle}$ for the client's true input. (This is because if a corrupted server submits incorrect $\mathsf{ans}_i^{\langle\mu\rangle}$ for some $\mu \neq \mu_*$, then it is detected when compared with an answer $\mathsf{ans}_i^{\langle\mu\rangle}(j)$ from an honest server $j \notin T$.) However, the adversary cannot learn which query encodes the client's true input due to the privacy of Π. Therefore, her best possible strategy is to guess μ_* uniformly at random, which succeeds only with probability $1/M$. The union bound implies that the error probability is at most m/M.

Privacy. Since M queries are generated independently, an adversary learns no information on the client's input x in the first round. The queries revealed in the second round are the ones for a default input x_{def}, which is independent of x, and hence the adversary learns no additional information. The privacy holds even if the z-output z is leaked, since z is determined only by $(v_i)_{i\in[m]}$, which can be simulated from information that the adversary learns up to the second round. □

Next, we show that the error probability of the basic construction can be made negligible by parallel execution. The proof is deferred to Appendix G.

Theorem 2. *Let Π be a 1-round (resp. computationally) passively t-secure m-server protocol for $\mathcal{F} = (F_\lambda : P_\lambda \times X_\lambda \to Y_\lambda)_{\lambda\in\mathbb{N}}$. Then there exists a 2-round (resp. computationally) t-conflict-finding m-server protocol Π_{CF} for $\mathcal{F}$ such that*

- $\text{Comm}(\Pi_{\text{CF}}) = O(m^2\lambda \cdot \text{Comm}(\Pi))$;
- $\text{c-Comp}(\Pi_{\text{CF}}) = O(m^3\lambda \cdot \text{c-Comp}(\Pi))$;
- $\text{s-Comp}(\Pi_{\text{CF}}) = O(m^2\lambda \cdot \text{s-Comp}(\Pi))$.

Finally, by combining Theorems 1 and 2, we obtain our generic construction of an $O(m^2)$-round actively t-secure m-server protocol from any 1-round passively t-secure k-server protocol for $k \leq m - t$.

Theorem 3. *Suppose that $m > 2t$. Let $k \leq m - t$ and Π be a 1-round (resp. computationally) passively t-secure k-server protocol for $\mathcal{F}$. Then there exists an $O(m^2)$-round (resp. computationally) actively t-secure m-server protocol Π' for $\mathcal{F}$ such that*

- $\mathrm{Comm}(\Pi') = O(m^4\lambda \cdot \mathrm{Comm}(\Pi))$;
- $\text{c-Comp}(\Pi') = O(m^5\lambda \cdot \text{c-Comp}(\Pi))$;
- $\text{s-Comp}(\Pi') = O(m^4\lambda \cdot \text{s-Comp}(\Pi))$.

4.5 Instantiations

By applying our compiler in Theorem 3 to the passively secure protocols in Propositions 1 and 2, we obtain actively secure protocols for $\textsc{Index}_N$.

Corollary 1. *Suppose that $m \geq 3^t + t$. Let $N \in \mathsf{poly}(\lambda)$. Then, there exists an actively t-secure $O(m^2)$-round m-server protocol Π for $\textsc{Index}_N$ such that*

- $\mathrm{Comm}(\Pi) = \exp(O(\sqrt{\log N \log\log N})) \cdot t3^t m^4\lambda$;
- $\text{c-Comp}(\Pi) = \exp(O(\sqrt{\log N \log\log N})) \cdot t3^t m^5\lambda$;
- $\text{s-Comp}(\Pi) = N^2 \cdot \exp(O(\sqrt{\log N \log\log N})) \cdot 2^t m^4\lambda$.

In particular, $\max\{\mathrm{Comm}(\Pi), \text{c-Comp}(\Pi)\} = N^{o(1)} \cdot 2^{O(t)}\lambda$.

Corollary 2. *Assume a pseudorandom generator $G : \{0,1\}^\lambda \to \{0,1\}^{2(\lambda+1)}$. Suppose that $m \geq 2^t + t$. Let $N \in \mathsf{poly}(\lambda)$. Then, there exists a computationally actively t-secure 1-round m-server protocol Π for $\textsc{Index}_N$ such that*

- $\mathrm{Comm}(\Pi) = O(\log N \cdot \lambda^2 \cdot t2^t m^4)$;
- $\text{c-Comp}(\Pi)$ *is* $O(\log N \cdot t2^t m^5)$ *invocations of G;*
- $\text{s-Comp}(\Pi)$ *is* $O(N^2 \log N \cdot tm^4)$ *invocations of G.*

In particular, $\max\{\mathrm{Comm}(\Pi), \text{c-Comp}(\Pi)\} = \log N \cdot 2^{O(t)} \cdot \mathsf{poly}(\lambda)$.

By applying Theorem 3 to Proposition 3, we obtain an actively secure protocol for multivariate polynomials.

Corollary 3. *Let $N, D, M \in \mathsf{poly}(\lambda)$. Let R be a ring such that for any $a \in \{1, 2, \ldots, m-1\}$, an element $a \cdot 1_R$ has an inverse in R. Suppose that*

$$m > \left(\frac{D}{2} + 1\right) t.$$

Then, there exists an actively t-secure 1-round m-server protocol Π for $\textsc{Poly}_{N,D,M}(R)$ such that

- $\mathrm{Comm}(\Pi) = O(Nm^4\lambda)$ *ring elements;*

– c-Comp(Π) $= O(Ntm^6\lambda)$ *ring operations;*
– s-Comp(Π) $= O(NMDm^4\lambda)$ *ring operations.*

In particular, $\max\{\mathrm{Comm}(\Pi), \text{c-Comp}(\Pi)\} = N \cdot \mathsf{poly}(m, \lambda)$.

By applying Theorem 3 to Proposition 4, we can reduce the required number of servers by a factor of d assuming homomorphic encryption for degree-d polynomials.

Corollary 4. *Let $d = O(1)$ and R be a ring such that for any $a \in \{1, 2, \ldots, \max\{d, m-1\}\}$, an element $a \cdot 1_R$ has an inverse in R, where 1_R is the multiplicative identity of R. Assume a homomorphic encryption scheme* HE *for degree-d polynomials over R. Suppose that*

$$m > \left(\frac{D}{d+1} + 1\right) t$$

Let $M, N \in \mathsf{poly}(\lambda)$ and $D = O(1)$. Then there exists a computationally actively t-secure $O(m^2)$-round m-server protocol Π for $\mathrm{POLY}_{N,D,M}(R)$ such that

– Comm(Π) $= O(Nm^5\lambda \cdot \ell_{\mathsf{ct}})$*, where ℓ_{ct} is the description length of ciphertexts of* HE*;*
– c-Comp(Π) $= O((Nt \cdot \tau_{\mathsf{Enc}} + \tau_{\mathsf{Dec}})m^6\lambda)$*, where τ_{Dec} and τ_{Enc} are the running time of the decryption and encryption algorithms of* HE*, respectively;*
– s-Comp(Π) $= O(M \cdot m^4\lambda\tau_{\mathsf{Eval}})$*, where τ_{Eval} is the running time per operation of the evaluation algorithm of* HE.

In particular, $\max\{\mathrm{Comm}(\Pi), \text{c-Comp}(\Pi)\} = N \cdot \mathsf{poly}(m, \lambda)$.

By applying Theorem 3 to Proposition 5, we obtain an actively t-secure protocol for polynomials achieving the minimum number of servers $2t + 1$.

Corollary 5. *Suppose that $m = 2t+1$. Assume that the (δ, q)-sLPN assumption holds for a constant $0 \leq \delta \leq 1$ and a sequence $q = (q(\lambda))_{\lambda \in \mathbb{N}}$ of prime powers that are computable in polynomial time in λ. Let $L, M, N \in \mathsf{poly}(\lambda)$ and $D = O(\log \lambda / \log \log \lambda)$. Then, there exists a computationally actively t-secure $O(m^2)$-round m-server protocol Π for $\mathrm{POLY}^L_{N,D,M}(\mathbb{F}_q)$ such that*

– Comm(Π) $= \widetilde{O}((M^{2/\delta}N + L)(\log q)m^5\lambda^2)$*;*
– c-Comp(Π) $= \widetilde{O}((M^{2/\delta}N + L)(\log q)m^6\lambda^2)$*;*
– s-Comp(Π) $= \widetilde{O}(M^{1/\delta+1}L(\log q)m^4\lambda^2)$.

In particular, $\max\{\mathrm{Comm}(\Pi), \text{c-Comp}(\Pi)\} = (M^{2/\delta}N + L) \log q \cdot \mathsf{poly}(m, \lambda)$.

Finally, by applying Theorem 3 to Proposition 6, we obtain an actively secure protocol for constant-depth circuits.

Corollary 6. *Let* $N, M \in \mathsf{poly}(\lambda)$ *and* $D = O(1)$. *Suppose that*

$$m \geq \left(\frac{(\log M + 3)^{D-1}}{2} + 1 \right) t.$$

Then, there exists an actively t*-secure* $O(m^2)$*-round* m*-server protocol* Π *for* $\textsc{Circ}_{N,D,M}$ *such that*

- $\mathrm{Comm}(\Pi) = O((\log M)^{D-1} N (\log N) \lambda^2 m^5)$*;*
- $\mathrm{c\text{-}Comp}(\Pi) = O((\log M)^{D-1} N (\log N) \lambda t m^6 + (\log M)^2 \lambda^2 m^6)$*;*
- $\mathrm{s\text{-}Comp}(\Pi) = O(M (\log M) N m^4 \lambda^2)$.

In particular, $\max\{\mathrm{Comm}(\Pi), \mathrm{c\text{-}Comp}(\Pi)\} = N \cdot \mathsf{poly}(m, \lambda)$.

5 Non-interactive Actively Secure Protocols

In this section, we show our compiler from one-round passively t-secure k-server protocols to one-round actively t-secure m-server protocols such that $m = O(k \log k) + 2t$. To this end, we introduce a novel combinatorial object of *locally surjective map families*, which is a variant of perfect hash families with a stronger property. We show a probabilistic construction of such families in Section 5.1 and then show a generic compiler from passively secure to actively secure protocols in Section 5.2.

5.1 Locally Surjective Map Family

We show the formal definition of locally surjective map families.

Definition 4. *Let* $m, h, k \in \mathbb{N}$ *and* $\mathcal{L}$ *be a family of maps from* $[m]$ *to* $[k]$. *We call* $\mathcal{L}$ *an* (m, h, k)-locally surjective map family *if* $|A_H| > |\mathcal{L}|/2$ *for any* $H \in \binom{[m]}{h}$, *where* $A_H = \{f \in \mathcal{L} : f(H) = [k]\}$.

A locally surjective map family satisfies a stronger property than a nearly perfect hash family $\mathcal{L}'$ introduced in [11], which assumes that for any $H \in \binom{[m]}{h}$, there exists *at least one* map $f \in \mathcal{L}'$ such that $f(H) = [k]$.

We show a probabilistic construction of an (m, h, k)-locally surjective map family of size $O(m)$ for $k = O(h/\log h)$. The formal proof is deferred to Appendix H.

Proposition 8. *Let* $m, h, k \in \mathbb{N}$ *be such that* $h \geq 15$, $m \geq 15$ *and* $k \leq h/(\gamma \ln h)$, *where* $\gamma := 1 + (\ln 3 - \ln \ln 15)/(\ln 15) < 1.04$. *Then, there exists an* (m, h, k)*-locally surjective map family* $\mathcal{L}$ *such that* $w := |\mathcal{L}| = 14m$.

5.2 Compiler from Passively Secure to Actively Secure Protocols

Based on locally surjective map families, we show our construction of one-round actively secure protocols from any one-round passively secure protocol. We give a sketch here and defer the formal proof to Appendix I.

Theorem 4. *Suppose that there exists a* 1*-round (resp. computationally) passively* t*-secure* k*-server protocol* $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ *for* $\mathcal{F} = (F_\lambda : P_\lambda \times X_\lambda \to Y_\lambda)_{\lambda \in \mathbb{N}}$*. If there exists an* $(m, m-2t, k)$*-locally surjective map family* $\mathcal{L}$ *of size* $w = \mathsf{poly}(\lambda)$*, there exists a* 1*-round (resp. computationally) actively* t*-secure* m*-server protocol* $\Pi' = (\mathsf{Query}', \mathsf{Answer}', \mathsf{Output}')$ *for* $\mathcal{F}$ *such that*

- $\mathrm{Comm}(\Pi') = O(twm \cdot \mathrm{Comm}(\Pi))$;
- $\text{c-Comp}(\Pi') = O(twm \cdot \text{c-Comp}(\Pi))$;
- $\text{s-Comp}(\Pi') = O(tw \cdot \text{s-Comp}(\Pi))$.

Proof (sketch). Let $\mathcal{L} = \{f_1, \ldots, f_w\}$ be an (m, h, k)-locally surjective map family, where $h = m - 2t$. For $u \in [w]$ and $j \in [k]$, define $G_{u,j} = f_u^{-1}(j) = \{i \in [m] : f_u(i) = j\}$. Consider the following protocol Π': For all $u \in [w]$ and $\ell \in [t+1]$ (in parallel),

1. The client C computes k queries $(\mathsf{que}_1^{(u,\ell)}, \ldots, \mathsf{que}_k^{(u,\ell)})$ of Π.
2. C sends $\mathsf{que}_{f_u(i)}^{(u,\ell)}$ to each server S_i.
3. Each S_i returns an answer $\mathsf{ans}_i^{(u,\ell)}$ as the $f_u(i)$-th server would answer to $\mathsf{que}_{f_u(i)}^{(u,\ell)}$ in Π.

To obtain an output, C sets $S \leftarrow [m]$ and $L \leftarrow 1$, and does the following:

1. Check whether for all $u \in [w]$ and $j \in [k]$, the answers $\mathsf{ans}_i^{(u,L)}$ returned by servers S_i in $G_{u,j}$ are identical with each other.
2. If so, let $\alpha_{u,j}$ be the unique answer by servers in $G_{u,j}$ and run the output algorithm of Π on $(\alpha_{u,1}, \ldots, \alpha_{u,k})$ to obtain y_u. Then, output the majority of $y_1, \ldots, y_w$.
3. Otherwise, find a pair (i_1, i_2) of servers who are mapped to the same group $G_{u,j}$ but returned different answers. That is, $f_u(i_1) = f_u(i_2)$ and $\mathsf{ans}_{i_1}^{(u,L)} \neq \mathsf{ans}_{i_2}^{(u,L)}$ for some $u \in [w]$. Note that at least one of them are malicious. Then, update $S \leftarrow S \setminus \{i_1, i_2\}$ and $L \leftarrow L + 1$, and go back to Step 1.

Privacy. An adversary corrupting a set T of at most t servers can only learn queries received by a set $f_u(T)$ of servers in Π. Since $|f_u(T)| \leq |T| \leq t$, the privacy of Π' follows from that of Π.

Byzantine-robustness. An adversary succeeds in letting the client accept an incorrect result only if at least $w/2$ out of $y_1, \ldots, y_w$ are incorrect in some iteration (say, L) in the output phase of C. This implies that for at least $w/2$ u's, there exists a remaining corrupted server $i \in T \cap S$ who submits an incorrect answer $\widetilde{\mathsf{ans}}_i^{(u,L)} \neq \mathsf{ans}_i^{(u,L)}$. On the other hand, since at most one honest server is eliminated from S in each iteration, it holds that $|H \cap S| \geq (m - t) - t =$

$m-2t$, where H is the set of all honest servers. Therefore, the property of locally surjective map families ensures that $f_u(H \cap S) = [k]$ holds for at least one of the above $w/2$ u's. In other words, there exists a remaining honest server $i' \in H \cap S$ such that $f_u(i') = f_u(i)$, and the answer $\widetilde{\mathsf{ans}}_i^{(u,L)}$ is compared with the correct answer $\mathsf{ans}_{i'}^{(u,L)}$ from the honest server i'. Thus, the client can detect the malicious behavior of the corrupted server i. Therefore, the client can successfully eliminate at least one malicious server in each iteration and obtain the correct result after at most t iterations. $\square$

To obtain a concrete compiler from Theorem 4, we plug in the (m, h, k)-locally surjective map family in Proposition 8 with $h = m - 2t$.

Theorem 5. *Suppose that there exists a* 1*-round (resp. computationally) passively* t*-secure* k*-server protocol* Π *for* $\mathcal{F}$*. If*

$$m \geq 2t + 15 \text{ and } \frac{m-2t}{\gamma \ln(m-2t)} \geq k,$$

where $1 < \gamma < 1.04$ *is the constant in Proposition 8, then there exists a* 1*-round (resp. computationally) actively* t*-secure* m*-server protocol* Π' *for* $\mathcal{F}$ *such that*

- $\mathrm{Comm}(\Pi') = O(tm^2 \cdot \mathrm{Comm}(\Pi))$;
- $\text{c-Comp}(\Pi') = O(tm^2 \cdot \text{c-Comp}(\Pi))$;
- $\text{s-Comp}(\Pi') = O(tm \cdot \text{s-Comp}(\Pi))$.

Remark 3. The computational complexity of the construction in Theorem 5 does not take into account that of finding a locally surjective map family $\mathcal{L}$. We note that the choice of $\mathcal{L}$ does not affect the security of a protocol. Hence we can construct it before the protocol starts and the family is reusable any number of times.

5.3 Instantiations

By applying our compiler in Theorem 5 to the protocols in Propositions 1 and 2, we obtain the following corollaries. The formal proof appears in Appendix J.

Corollary 7. *Suppose that* $m \geq \max\{2t3^t + 2t, 2t + 15\}$*. Let* $N \in \mathsf{poly}(\lambda)$*. Then, there exists a computationally actively* t*-secure* 1*-round* m*-server protocol* Π *for* $\textsc{Index}_N$ *such that*

- $\mathrm{Comm}(\Pi) = \exp(O(\sqrt{\log N \log\log N})) \cdot t^2 3^t m^2$;
- $\text{c-Comp}(\Pi) = \exp(O(\sqrt{\log N \log\log N})) \cdot t^2 3^t m^2$;
- $\text{s-Comp}(\Pi) = N^2 \cdot \exp(O(\sqrt{\log N \log\log N})) \cdot t 2^t m$.

In particular, $\max\{\mathrm{Comm}(\Pi), \text{c-Comp}(\Pi)\} = N^{o(1)} \cdot 2^{O(t)}$.

Corollary 8. *Assume a pseudorandom function* $G : \{0,1\}^\lambda \to \{0,1\}^{2(\lambda+1)}$*. Suppose that* $m \geq \max\{t2^{t+1} + 2t, 2t + 15\}$*. Let* $N \in \mathsf{poly}(\lambda)$*. Then, there exists an actively* t*-secure* 1*-round* m*-server protocol* Π *for* $\textsc{Index}_N$ *such that*

– $\mathrm{Comm}(\Pi) = O(\log N \cdot \lambda \cdot t^2 2^t m^2)$;
– $\text{c-Comp}(\Pi)$ *is* $O(\log N \cdot t^2 2^t m^2)$ *invocations of* G;
– $\text{s-Comp}(\Pi)$ *is* $O(N^2 \log N \cdot t 2^t m)$ *invocations of* G.

In particular, $\max\{\mathrm{Comm}(\Pi), \text{c-Comp}(\Pi)\} = \log N \cdot 2^{O(t)} \cdot \mathsf{poly}(\lambda)$.

Note that it is possible to apply the compiler in Theorem 5 to the passively secure k-server protocols in Propositions 3, 4, 5, and 6. Since $k > t$, the number of servers of the resulting protocols is $\Omega(k \log k) + 2t = \Omega(t \log t)$. On the other hand, these protocols can also be made actively secure by using the standard error correction algorithm [47] or the technique of [43], and one can then obtain actively secure protocols that has a smaller number of servers $O(t)$. We thus do not show instantiations based on these protocols.

Acknowledgements

This research was partially supported by JSPS KAKENHI Grant Numbers JP20J20797 and JP19H01109, Japan, JST CREST Grant Numbers JPMJCR2113 and JPMJCR22M1, Japan, and JST AIP Acceleration Research JPMJCR22U5, Japan.

References

1. Ananth, P., Chandran, N., Goyal, V., Kanukurthi, B., Ostrovsky, R.: Achieving privacy in verifiable computation with multiple servers – without fhe and without pre-processing. In: Public-Key Cryptography – PKC 2014. pp. 149–166 (2014)
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: Efficient verification via secure computation. In: Automata, Languages and Programming. pp. 152–163 (2010)
3. Augot, D., Levy-dit Vehel, F., Shikfa, A.: A storage-efficient and robust private information retrieval scheme allowing few servers. In: Cryptology and Network Security. pp. 222–239 (2014)
4. Banawan, K., Ulukus, S.: The capacity of private information retrieval from byzantine and colluding databases. IEEE Transactions on Information Theory **65**(2), 1206–1219 (2019)
5. Barkol, O., Ishai, Y.: Secure computation of constant-depth circuits with applications to database search problems. In: Advances in Cryptology — CRYPTO 2005. pp. 395–411 (2005)
6. Barkol, O., Ishai, Y., Weinreb, E.: On d-multiplicative secret sharing. Journal of Cryptology **23**(4), 580–593 (2010)
7. Barkol, O., Ishai, Y., Weinreb, E.: On locally decodable codes, self-correctable codes, and t-private PIR. Algorithmica **58**(4), 831–859 (2010)
8. Beimel, A., Ishai, Y., Kushilevitz, E., Raymond, J.F.: Breaking the o(n/sup 1/(2k-1)/) barrier for information-theoretic private information retrieval. In: The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings. pp. 261–270 (2002)
9. Beimel, A., Ishai, Y.: Information-theoretic private information retrieval: A unified construction. In: Automata, Languages and Programming. pp. 912–926 (2001)

10. Beimel, A., Ishai, Y., Kushilevitz, E., Orlov, I.: Share conversion and private information retrieval. In: 2012 IEEE 27th Conference on Computational Complexity. pp. 258–268 (2012)
11. Beimel, A., Stahl, Y.: Robust information-theoretic private information retrieval. Journal of Cryptology **20**(3), 295–321 (2007)
12. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Theory of Cryptography. pp. 325–341 (2005)
13. Boyer, R.S., Moore, J.S.: MJRTY—a fast majority vote algorithm. Automated Reasoning: Essays in Honor of Woody Bledsoe pp. 105–117 (1991)
14. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Advances in Cryptology – EUROCRYPT 2015. pp. 337–367 (2015)
15. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Advances in Cryptology – CRYPTO 2016, Part I. pp. 509–539 (2016)
16. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: Improvements and extensions. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1292–1303. CCS '16 (2016)
17. Boyle, E., Gilboa, N., Ishai, Y., Nof, A.: Sublinear GMW-style compiler for MPC with preprocessing. In: Advances in Cryptology – CRYPTO 2021. pp. 457–485 (2021)
18. Boyle, E., Kohl, L., Scholl, P.: Homomorphic secret sharing from lattices without FHE. In: Advances in Cryptology – EUROCRYPT 2019, Part II. pp. 3–33 (2019)
19. Bunn, P., Kushilevitz, E., Ostrovsky, R.: CNF-FSS and its applications. In: Public-Key Cryptography – PKC 2022. pp. 283–314 (2022)
20. Canetti, R., Riva, B., Rothblum, G.N.: Refereed delegation of computation. Information and Computation **226**, 16–36 (2013)
21. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: Topics in Cryptology – CT-RSA 2015. pp. 487–505 (2015)
22. de Castro, L., Lee, K.: VeriSimplePIR: Verifiability in simplepir at no online cost for honest servers. In: 33rd USENIX Security Symposium (USENIX Security 24) (to appear), https://www.usenix.org/conference/usenixsecurity24/presentation/de-castro
23. Choi, S.G., Katz, J., Kumaresan, R., Cid, C.: Multi-client non-interactive verifiable computation. In: Theory of Cryptography. pp. 499–518 (2013)
24. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. Journal of the ACM **45**(6), 965–982 (1998)
25. Colombo, S., Nikitin, K., Corrigan-Gibbs, H., Wu, D.J., Ford, B.: Authenticated private information retrieval. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 3835–3851 (2023)
26. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Public Key Cryptography. pp. 119–136 (2001)
27. Damgård, I., Orlandi, C., Simkin, M.: Yet another compiler for active security or: Efficient MPC over arbitrary rings. In: Advances in Cryptology – CRYPTO 2018. pp. 799–829 (2018)
28. Dao, Q., Ishai, Y., Jain, A., Lin, H.: Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In: Advances in Cryptology – CRYPTO 2023. pp. 315–348 (2023)
29. Devet, C., Goldberg, I., Heninger, N.: Optimally robust private information retrieval. In: 21st USENIX Security Symposium (USENIX Security 12). pp. 269–283 (2012)

30. Di Crescenzo, G., Malkin, T., Ostrovsky, R.: Single database private information retrieval implies oblivious transfer. In: Advances in Cryptology – EUROCRYPT 2000. pp. 122–138 (2000)
31. Dvir, Z., Gopi, S.: 2-server PIR with subpolynomial communication. Journal of the ACM **63**(4), 1–15 (2016)
32. Efremenko, K.: 3-query locally decodable codes of subexponential length. SIAM Journal on Computing **41**(6), 1694–1703 (2012)
33. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)
34. Eriguchi, R., Kurosawa, K., Nuida, K.: On the optimal communication complexity of error-correcting multi-server PIR. In: Theory of Cryptography. pp. 60–88 (2022)
35. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Advances in Cryptology – CRYPTO 2010. pp. 465–482 (2010)
36. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. pp. 169–0178. STOC '09 (2009)
37. Gilboa, N., Ishai, Y.: Distributed point functions and their applications. In: Advances in Cryptology – EUROCRYPT 2014. pp. 640–658 (2014)
38. Goldberg, I.: Improving the robustness of private information retrieval. In: 2007 IEEE Symposium on Security and Privacy (SP'07). pp. 131–148 (2007)
39. Hazay, C., Ishai, Y., Marcedone, A., Venkitasubramaniam, M.: LevioSA: Lightweight secure arithmetic computation. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 327–344. CCS '19 (2019)
40. Hazay, C., Venkitasubramaniam, M., Weiss, M.: The price of active security in cryptographic protocols. In: Advances in Cryptology – EUROCRYPT 2020. pp. 184–215 (2020)
41. Ishai, Y., Lai, R.W.F., Malavolta, G.: A geometric approach to homomorphic secret sharing. In: Public-Key Cryptography – PKC 2021. pp. 92–119 (2021)
42. Korte, B.H., Vygen, J.: Combinatorial optimization, vol. 1. Springer (2011)
43. Kurosawa, K.: How to correct errors in multi-server PIR. In: Advances in Cryptology – ASIACRYPT 2019. pp. 564–574 (2019)
44. Orlandi, C., Scholl, P., Yakoubov, S.: The rise of paillier: Homomorphic secret sharing and public-key silent ot. In: Advances in Cryptology – EUROCRYPT 2021. pp. 678–708 (2021)
45. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology — EUROCRYPT '99. pp. 223–238 (1999)
46. Roy, L., Singh, J.: Large message homomorphic secret sharing from DCR and applications. In: Advances in Cryptology – CRYPTO 2021. pp. 687–717 (2021)
47. Rudra, A.: Lecture 27: Berlekamp-Welch algorithm https://cse.buffalo.edu/faculty/atri/courses/coding-theory/lectures/lect27.pdf
48. Spitzbart, A.: A generalization of Hermite's interpolation formula. The American Mathematical Monthly **67**(1), 42–46 (1960)
49. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Advances in Cryptology – EUROCRYPT 2010. pp. 24–43 (2010)
50. Woodruff, D., Yekhanin, S.: A geometric approach to information-theoretic private information retrieval. SIAM Journal on Computing **37**(4), 1046–1056 (2007)

51. Yao, X., Liu, N., Kang, W.: The capacity of multi-round private information retrieval from byzantine databases. In: 2019 IEEE International Symposium on Information Theory (ISIT). pp. 2124–2128 (2019)
52. Yoshida, M., Obana, S.: Verifiably multiplicative secret sharing. IEEE Transactions on Information Theory **65**(5), 3233–3245 (2019)
53. Zhang, L.F., Wang, H.: Multi-server verifiable computation of low-degree polynomials. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 596–613 (2022)
54. Zhang, L.F., Wang, H., Wang, L.P.: Byzantine-robust private information retrieval with low communication and efficient decoding. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. pp. 1079–1085. ASIA CCS '22 (2022)

A Passively Secure Protocols from [10] and [16]

There is a 3-server protocol for $\textsc{Index}_N$ with sub-polynomial complexity in N.

Proposition 9 ([10]). *There exists a passively* 1*-secure* 1*-round* 3*-server protocol* Π *for* $\textsc{Index}_N$ *such that*

- *The query length is* $c_{\mathsf{que}} = \exp(O(\sqrt{\log N \log\log N})) = N^{o(1)}$*;*
- *The answer length is* $c_{\mathsf{ans}} = 2$*;*
- $\text{c-Comp}(\Pi) = \exp(O(\sqrt{\log N \log\log N})) = N^{o(1)}$*;*
- $\text{s-Comp}(\Pi) = N \cdot \exp(O(\sqrt{\log N \log\log N})) = N^{1+o(1)}$.

There is a technique to amplify the privacy threshold in [7]. Let Π be a passively t_0-secure m_0-server protocol for $\textsc{Index}_N$ whose query length is c_{que} and answer length is c_{ans}. Then, Π can be generically transformed into a passively t_0t-secure m_0^t-server protocol for $\textsc{Index}_N$ whose query length is $O(c_{\mathsf{que}}tm_0^t)$ and answer length is $O(c_{\mathsf{ans}}^t)$. In particular, by applying this transformation to the passively 1-secure 3-server protocol in Proposition 9, we obtain the passively t-secure 3^t-server protocol in Proposition 1.

Under the assumption of one-way functions, there is a passively 1-secure 2-server protocol for $\textsc{Index}_N$ with logarithmic communication complexity in N [16].

Proposition 10 ([16]). *Assume a pseudorandom generator* $G : \{0,1\}^\lambda \to \{0,1\}^{2(\lambda+1)}$*. There exists a passively* 1*-secure* 1*-round* 2*-server protocol* Π *for* $\textsc{Index}_N$ *such that*

- $\text{Comm}(\Pi) = O(\log N \cdot \lambda)$*;*
- $\text{c-Comp}(\Pi)$ *is* $O(\log N)$ *invocations of* G*;*
- $\text{s-Comp}(\Pi)$ *is* $O(N \log N)$ *invocations of* G*.*

Again, by applying the privacy amplification technique in [7], we obtain a passively t-secure 2^t-server protocol in Proposition 2.

B Homomorphic Secret Sharing

Let $\mathcal{P} = (P_\lambda)_{\lambda\in\mathbb{N}}$ be a sequence of families of functions. We recall the notion of a *homomorphic secret sharing (HSS)* scheme for $\mathcal{P}$.

Definition 5. *Let* $\mathcal{X} = (X_\lambda)_{\lambda\in\mathbb{N}}$ *and* $\mathcal{Y} = (Y_\lambda)_{\lambda\in\mathbb{N}}$ *be sequences of sets with polynomial-size descriptions and* $\mathcal{P} = (P_\lambda)_{\lambda\in\mathbb{N}}$ *be a sequence of sets of functions such that each* P_λ *consists of functions* $p : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$ *with polynomial-size descriptions. A* passively t-secure (resp. computationally passively t-secure) m-server homomorphic secret sharing scheme for $\mathcal{P}$ (with public-key setup) *is a tuple of four polynomial-time algorithms* $\Pi = (\mathsf{KeyGen}, \mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$*, where:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: An algorithm* KeyGen *takes a security parameter* λ *as input and outputs a public key* pk *and a secret key* sk*;*

- $\mathsf{Share}(\mathsf{pk}, x) \rightarrow (\mathsf{sh}_i)_{i \in [m]}$: *An algorithm* Share *takes a public key* pk *and* $x \in X_\lambda$ *as input, and outputs shares* $(\mathsf{sh}_i)_{i \in [m]}$*;*
- $\mathsf{Eval}(\mathsf{pk}, p, \mathsf{sh}_i) \rightarrow \mathsf{out}_i$: *An algorithm* Eval *takes a public key* pk*, a function* $p \in P_\lambda$ *and a share* sh_i *as input, and outputs an output share* out_i*;*
- $\mathsf{Dec}(\mathsf{sk}, (\mathsf{out}_i)_{i \in [m]}) \rightarrow y$: *An algorithm* Dec *takes a secret key* sk *and output shares* $(\mathsf{out}_i)_{i \in [m]}$ *as input, and outputs* $y \in Y_\lambda$*;*

satisfying the following properties:

Correctness. *There exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any* $\lambda \in \mathbb{N}$ *and any* $(p, x) \in P_\lambda \times X_\lambda$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, (\mathsf{out}_i)_{i \in [m]}) = p(x)\right] \geq 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $(\mathsf{sh}_i)_{i \in [m]} \leftarrow \mathsf{Share}(\mathsf{pk}, x)$, *and* $\mathsf{out}_i \leftarrow \mathsf{Eval}(\mathsf{pk}, p, \mathsf{sh}_i)$ *for all* $i \in [m]$.

Privacy. *There exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any stateful (resp. PPT stateful) algorithm* $\mathcal{A}$ *and any* $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\mathsf{Priv}^0_{\Pi,\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{Priv}^1_{\Pi,\mathcal{A}}(\lambda) = 0\right]\right| < \mathsf{negl}(\lambda),$$

where for $b \in \{0, 1\}$, $\mathsf{Priv}^b_{\Pi,\mathcal{A}}(\lambda)$ *is the output of* $\mathcal{A}$ *in the following experiment:*

1. *Let* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
2. *Let* $(x_0, x_1, p, T) \leftarrow \mathcal{A}(\mathsf{pk})$, *where* $x_0, x_1 \in X_\lambda$, $p \in P_\lambda$ *and* $T \subseteq [m]$ *is of size at most* t.
3. *Let* $(\mathsf{sh}_i)_{i \in [m]} \leftarrow \mathsf{Share}(\mathsf{pk}, x_b)$ *and give* $(\mathsf{sh}_i)_{i \in T}$ *to* $\mathcal{A}$.
4. $\mathcal{A}$ *outputs a bit* $b' \in \{0, 1\}$.

Let $\Pi = (\mathsf{KeyGen}, \mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ be a passively t-secure m-server homomorphic secret sharing scheme for $\mathcal{P}$. Define $\mathcal{F} = (F_\lambda : P_\lambda \times X_\lambda \rightarrow Y_\lambda)_{\lambda \in \mathbb{N}}$ as $F_\lambda(p, x) = p(x)$. We can view Π as a passively t-secure 1-round m-server protocol $\Pi' = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ for $\mathcal{F}$ in the following manner: Query is defined as the algorithm that takes 1^λ and x as input, runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $(\mathsf{sh}_i)_{i \in [m]} \leftarrow \mathsf{Share}(\mathsf{pk}, x)$, and outputs $\mathsf{que}_i := (\mathsf{pk}, \mathsf{sh}_i)$ for $i \in [m]$ and $\mathsf{st} := (\mathsf{pk}, \mathsf{sk})$; $\mathsf{Answer}(p, \mathsf{que}_i = (\mathsf{pk}, \mathsf{sh}_i))$ just runs $\mathsf{out}_i \leftarrow \mathsf{Eval}(\mathsf{pk}, p, \mathsf{sh}_i)$ and outputs $\mathsf{ans}_i := \mathsf{out}_i$; and finally $\mathsf{Output}((\mathsf{ans}_i)_{i \in [m]}, \mathsf{st} = (\mathsf{pk}, \mathsf{sk}))$ outputs $y \leftarrow \mathsf{Dec}(\mathsf{sk}, (\mathsf{sh}_i)_{i \in [m]})$.

The evaluation algorithm Eval of a homomorphic secret sharing scheme is not always defined to be deterministic in the literature. We notice that all instantiations considered in this paper actually have deterministic Eval and thus, the induced protocols have deterministic answer algorithms Answer, which complies with the syntax in Definition 1.

In the literature, there is a notion of homomorphic secret sharing *in the plain model*, where there is no key generation algorithm KeyGen and public and secret keys are the empty string. Note that this notion is also captured by Definition 1 in the same manner as above except that pk and sk are omitted.

To be used for specific applications, HSS schemes in the plain model are often required to have *linear reconstruction*, i.e., $\mathsf{Dec}((\mathsf{out}_i)_{i\in[m]})$ is a linear function on $(\mathsf{out}_i)_{i\in[m]}$. On the other hand, this paper targets applications where a single client can do arbitrary computation as long as the complexity is sufficiently lower than that of computing $F_\lambda(p, x)$ by himself. Therefore, we may allow HSS schemes to have non-linear reconstruction.

C Passively Secure Protocol from [28]

Let $n = n(\lambda)$ be the dimension, $N = N(\lambda)$ the number of samples, $k = k(\lambda) \leq n$ the sparsity parameter, $q = q(\lambda)$ the field size, $\epsilon = \epsilon(\lambda)$ the noise rate with $0 \leq \epsilon \leq 1$. Define the sparse LPN distribution $\mathcal{D}_{\mathsf{sLPN},n,N,k,\epsilon,q}$ as the distribution of an output of the following process:

1. Sample $\mathbf{s}$ uniformly at random from $\mathbb{F}_q^{1\times n}$.
2. Sample $\mathbf{A}$ uniformly at random from $\mathbb{F}_q^{n\times N}$ conditioned on every column of $\mathbf{A}$ has exactly k non-zero elements.
3. Sample $\mathbf{e}$ according to $(\mathsf{Ber}(\mathbb{F}_q, \epsilon)^{1\times N}$, where $\mathsf{Ber}(\mathbb{F}_q, \epsilon)$ returns 0 with probability $1 - \epsilon$ and a uniformly random non-zero element of $\mathbb{F}_q$ otherwise.
4. Compute $\mathbf{b} = \mathbf{s} \cdot \mathbf{A} + \mathbf{e}$ and output $(\mathbf{A}, \mathbf{b})$

Similarly, define $\mathcal{D}_{\mathsf{rand},n,N,k,\epsilon,q}$ as the distribution that is identical to $\mathcal{D}_{\mathsf{sLPN},n,N,k,\epsilon,q}$ except that $\mathbf{b}$ is chosen uniformly at random from $\mathbb{F}_q^{1\times m}$.

Let $0 \leq \delta \leq 1$ be a constant and $q = (q(\lambda))_{\lambda\in\mathbb{N}}$ be a sequence of prime powers that is computable in polynomial time in λ. We say that the (δ, q)-sLPN assumption holds if for all parameters n, N, k, ϵ that are computable in polynomial time in λ such that $k = \omega(1)$ and $\epsilon = O(n^{-\delta})$, the distributions $\mathcal{D}_{\mathsf{sLPN},n,N,k,\epsilon,q}$ and $\mathcal{D}_{\mathsf{rand},n,N,k,\epsilon,q}$ are computationally indistinguishable.

Let $D = O(\log\lambda/ \log\log\lambda)$ and $N, M \in \mathsf{poly}(\lambda)$. Similarly to [28], we choose the sparsity parameter $k = (\log\lambda)^c$ for a sufficiently small constant c and the dimension $n = M^{1/\delta}\lambda^{1/2}$. Assume the (δ, q)-sLPN assumption holds. Then there exists a computationally passively t-secure 1-round $(t+1)$-server protocol Π for $\textsc{Poly}_{N,D,M}(\mathbb{F}_q)$ such that

– $\mathrm{Comm}(\Pi) = O(n^2N(\log q)m) = \widetilde{O}(M^{2/\delta}N(\log q)m\lambda)$;
– $\text{c-Comp}(\Pi) = \widetilde{O}(n^2N(\log q)m) = \widetilde{O}(M^{2/\delta}N(\log q)m\lambda)$;
– $\text{s-Comp}(\Pi) = \widetilde{O}(nM(\log q)) = \widetilde{O}(M^{1/\delta+1}(\log q)\lambda)$.

Note that the original scheme for evaluating a single polynomial does not satisfy our efficiency requirements since $\mathrm{Comm}(\Pi)$ and $\text{c-Comp}(\Pi)$ are not less than the description length $\widetilde{O}(M)$ of N-variate polynomials of degree D with M monomials. Let $L = \mathsf{poly}(\lambda)$. We observe that a slightly modified version of the original scheme can support a function class $\textsc{Poly}^L_{N,D,M}(\mathbb{F}_q)$. Indeed, a client computes a query based on his input $\mathbf{x}$ as in the original scheme but now servers evaluate L polynomials on the query and return L answers to the client. We then obtain Proposition 5.

The passively secure protocol [28] can be made actively t-secure by using the Berlekamp-Welch algorithm, which is a standard error correction algorithm for the Reed-Solomon code (see, e.g., [47]). Technically, if ϕ is a polynomial of degree at most $m-2t$, the algorithm can recover ϕ from any vector whose Hamming distance from $(\phi(1),\ldots,\phi(m))$ is at most t. In the protocol [28], answers from servers have the form of $(\phi(1),\ldots,\phi(m))$ for a polynomial ϕ of degree $t+1$ such that $\phi(0)=p(\mathbf{x})$. Therefore, if $m \geq 3t+1$, then one can correct errors with the Berlekamp-Welch algorithm and achieve active security. A similar technique can be applied to the passively secure protocol in [5] or Proposition 6 at the cost of increasing the number of servers by $2t$, although these facts were not explicitly stated in [5, 28].

D Passively Secure Protocol from [41]

First, we recall the definition of homomorphic encryption.

Definition 6. *An* IND-CPA *secure homomorphic encryption scheme* $\mathsf{HE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *for degree-d polynomials over a ring R consists of the following PPT algorithms:*

- $\mathsf{KGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: Given the security parameter 1^λ, the key generation algorithm outputs a public key* pk *and a secret key* sk*;*
- $\mathsf{Enc}(\mathsf{pk}, \mathbf{x}) \to \mathbf{c}$*: Given the public key* pk *and a vector of n messages $\mathbf{x} \in R^n$, the encryption algorithm outputs a vector of n ciphertexts $\mathbf{c} \in \mathcal{C}^n$ in some ciphertext space $\mathcal{C}$;*
- $\mathsf{Eval}(\mathsf{pk}, p, \mathbf{c})$*: Given the public key* pk*, a degree-d polynomial $p \in R[X_1,\ldots,X_n]$, and a vector of n ciphertexts $\mathbf{c} \in \mathcal{C}^n$, the evaluation algorithm outputs a ciphertext $c' \in \mathcal{C}$;*
- $\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$*: Given the secret key* sk *and a vector of n ciphertexts $\mathbf{c} \in \mathcal{C}^n$, the decryption algorithm outputs a vector of n plaintexts $\mathbf{x} \in R^n$;*

satisfying the following properties:

Correctness. *There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for any $\lambda \in \mathbb{N}$, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$, any $n \in \mathsf{poly}(\lambda)$, any degree-$d$ polynomial $p \in R[X_1,\ldots,X_n]$, and any vector of n messages $\mathbf{x} = (x_1,\ldots,x_n) \in R^n$, it holds that*

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathbf{x})) = \mathbf{x}] \geq 1 - \mathsf{negl}(\lambda) \quad \text{and}$$
$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, c') = p(x_1,\ldots,x_n) : \begin{array}{c} \mathbf{c} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{x}) \\ c' \leftarrow \mathsf{Eval}(\mathsf{pk}, p, \mathbf{c}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins of Enc *and* Eval*.*

IND-CPA Security. *There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for any PPT stateful adversary $\mathcal{A}$, it holds that*

$$|\Pr[\mathsf{IND\text{-}CPA}^0_{\mathsf{HE},\mathcal{A}} = 1] - \Pr[\mathsf{IND\text{-}CPA}^1_{\mathsf{HE},\mathcal{A}} = 1]| < \mathsf{negl}(\lambda),$$

where $\mathsf{IND\text{-}CPA}^b_{\mathsf{HE},\mathcal{A}}$ is defined in Fig. 1 for $b \in \{0,1\}$.

$\underline{\mathsf{IND\text{-}CPA}^b_{\mathsf{HE},\mathcal{A}}(1^\lambda):}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
$(x_0, x_1) \leftarrow \mathcal{A}(\mathsf{pk})$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b)$
$b' \leftarrow \mathcal{A}(c)$
return b'

Fig. 1. IND-CPA experiment for public-key encryption

Let $d, D \in O(1)$ such that $d < D$, and $N, M \in \mathsf{poly}(\lambda)$. It is shown in [41] that assuming an IND-CPA secure homomorphic encryption scheme for degree-d polynomials over a ring R and

$$m > \frac{Dt}{d+1},$$

then there exists a computationally passively t-secure 1-round m-server protocol Π for $\textsc{Poly}_{N,D,M}(R)$ such that

- $\mathrm{Comm}(\Pi) = O(Nm \cdot \ell_{\mathsf{ct}})$, where ℓ_{ct} is the description length of ciphertexts of HE;
- $\text{c-Comp}(\Pi) = O((\tau_{\mathsf{Dec}} + N \cdot \tau_{\mathsf{Enc}})m^2)$, where τ_{Dec} and τ_{Enc} are the running time of Dec and Enc, respectively;
- $\text{s-Comp}(\Pi) = O(M \cdot \tau_{\mathsf{Eval}})$, where τ_{Eval} is the running time of Eval per operation.

Note that the original construction assumes that R is a field. We observe that their construction still works as long as for any $a \in \{1, 2, \ldots, \max\{d, m-1\}\}$, an element $a \cdot 1_R$ has an inverse in R, where 1_R is the multiplicative identity of R. Indeed, one needs to compute multiplicative inverses only when reconstructing the value of a polynomial from output shares in the output algorithm of the protocol (one performs only addition and multiplication in the query and answer algorithms, which can be done in a ring). Concretely, if a client wants to compute $p(\mathbf{x})$, he decodes ciphertexts of the homomorphic encryption scheme to obtain ring elements $(\phi(i), \phi^{(1)}(i), \ldots, \phi^{(d)}(i))_{i \in [m]}$ for a polynomial $\phi \in R[X]$ such that $\phi(0) = p(\mathbf{x})$, where $\phi^{(e)}$ is the e-th derivative of ϕ, and then the client recovers $\phi(0)$ by Hermite interpolation. According to the interpolation formula in [48], $\phi(0)$ can be recovered if $(i-j) \cdot 1_R$ and $a \cdot 1_R$ have inverses in R for any $i \neq j \in [m]$ and $a \in \{1, 2, \ldots, d\}$.

To ensure that the protocol in Proposition 4 has a deterministic answer algorithm Answer, the underlying homomorphic encryption scheme must have a deterministic evaluation algorithm Eval. This is the case for the instantiations [33, 12, 45, 26, 21] mentioned in [41]. We notice that some evaluation algorithms in these works inject randomness when evaluating functions, but this is only for the scheme to satisfy *circuit privacy*, i.e., to keep information on functions hidden from evaluated ciphertexts. In particular, our required security notion (i.e., IND-CPA security) is satisfied even if evaluation algorithms are made deterministic.

E Proof of Theorem 1

Consider a protocol Π described in Fig. 2. We denote the query, answer and output algorithms of Π_{CF} by $(\mathsf{Query}_0, \mathsf{Answer}_0, \mathsf{Output}_0)$. We call the execution of Π_{CF} in the j-th iteration at Step 3 of Π the *j-th instance of Π_{CF}*.

Correctness. Assume that all servers are honest. The correctness of Π_{CF} ensures that in the first instance of Π_{CF}, it holds with overwhelming probability that $(y^{(1)}, z^{(1)}) = (F_\lambda(p,x), \mathsf{output})$. Then $y^{(1)} = F_\lambda(p,x)$ is added to $\mathcal{Y}$ at Step 3(c) of Π. After that, $\mathcal{Y}$ is never updated since $\mathcal{Y} \neq \emptyset$. Therefore, C outputs $F_\lambda(p,x)$ at Step 4.

Privacy. Let $\mathcal{A}$ be a stateful adversary against the privacy of Π. For $b \in \{0,1\}$, let $\mathsf{Priv}^b_{\Pi,\mathcal{A}}(\lambda)$ be a random variable defined in Definition 2. We will show that

$$\mathsf{Adv}_{\Pi,\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Priv}^0_{\Pi,\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{Priv}^1_{\Pi,\mathcal{A}}(\lambda) = 0\right]\right|.$$

is negligible.

To simplify notations, we denote the following experiment by $\mathrm{EXPT}_{\Pi_{\mathrm{CF}},\mathcal{A},p,T}(S,x)$, where T is a subset of $[m]$ of size at most t, $S = \{i_1, i_2, \ldots, i_k\}$ $(i_1 < i_2 < \cdots < i_k)$ is a subset of $[m]$ of size k, $p \in P_\lambda$, and $x \in X_\lambda$:

$\mathrm{EXPT}_{\Pi_{\mathrm{CF}},\mathcal{A},p,T}(S,x)$

1. Execute Π_{CF} with k servers in S on a client input x and a common server input p, in which answers of servers in T are computed by $\mathcal{A}$. Formally, for each $j = 1, \ldots, \ell$,

$$((\mathsf{que}_v^{(j)})_{v\in[k]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}_0(1^\lambda, x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_v^{(j-1)})_{v\in[k]}),$$
$$\mathsf{ans}_v^{(j)} \leftarrow \mathsf{Answer}_0(1^\lambda, p, \mathsf{que}_v^{(h)})$$

   except that for any v with $i_v \in T$, $\mathsf{ans}_v^{(j)}$ is computed by $\mathcal{A}$.
2. Let $(y,z) \leftarrow \mathsf{Output}_0(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_v^{(\ell)})_{v\in[k]})$.
3. Return z.

We define a hybrid distribution $\mathsf{Hyb}^{(h)}_{\Pi,\mathcal{A}}(\lambda)$ for $1 \leq h \leq N$ as the output b' of $\mathcal{A}$ in the following experiment $\mathrm{HybEXPT}^{(h)}_{\Pi,\mathcal{A}}(\lambda)$:

Notations.

- Let $\Pi_{\mathrm{CF}} = (\mathsf{Query}_0, \mathsf{Answer}_0, \mathsf{Output}_0)$ be an ℓ-round t-conflict-finding k-server protocol.
- Let $m \geq t + k$.
- A client C has an input $x \in X_\lambda$ and every server S_i has a common input $p \in P_\lambda$.
- Let $V := [m]$ and $N := \binom{m}{2} - \binom{m-t}{2} + 1$.

Sub-algorithms.

$\mathcal{C}_k(\mathcal{G}) \to S$**.** On input a graph $\mathcal{G} = (V, E)$, $\mathcal{C}_k$ outputs a subset $S \subseteq V$ of size k such that $\mathcal{G}[S]$ is connected if $\mathcal{G}$ contains a connected component of size k. The construction is given in Section 4.1.

$\mathcal{E}(\mathcal{G}, (G_0, G_1)) \to e$**.** On input a graph $\mathcal{G} = (V, E)$ and a pair of disjoint non-empty subsets $G_0, G_1 \subseteq V$, $\mathcal{E}$ outputs the minimum edge $e = (i, j) \in E$ (with respect to the total order on E) such that $i \in G_0$ and $j \in G_1$, or $j \in G_0$ and $i \in G_1$. The construction is given in Section 4.1.

$\mathsf{Next}_k(\mathcal{G}, S, z) \to (\mathcal{G}', S')$**.** On input a graph $\mathcal{G} = (V, E)$, a size-k subset S of V, and a z-output z of Π_{CF},

- If $z = \mathsf{output}$ or $z = \mathsf{failure}$, output $(\mathcal{G}', S') \leftarrow (\mathcal{G}, S)$.
- If $z = (G_0, G_1)$, where G_0, G_1 are disjoint non-empty subsets of S,
  1. Compute an edge $e \leftarrow \mathcal{E}(\mathcal{G}, (G_0, G_1))$.
  2. Set $E' \leftarrow E \setminus \{e\}$, $\mathcal{G}' \leftarrow (V, E')$, $S' \leftarrow \mathcal{C}_k(\mathcal{G}')$ and output $(\mathcal{G}', S')$.

Protocol.

1. C sets $\mathcal{Y} = \emptyset$.
2. C sets $\mathcal{G}^{(1)} = (V, E^{(1)})$ to the complete graph over V and computes a size-k subset $S^{(1)} \leftarrow \mathcal{C}_k(\mathcal{G}^{(1)})$.
3. For each $j = 1, 2, \ldots, N$, the client and servers do the following:
   (a) C executes Π_{CF} with k servers in $S^{(j)}$ on a client input x and a common server input p. Formally, if $S^{(j)} = \{i_1, \ldots, i_k\}$, where $i_1 < \cdots < i_k$, then for each $h = 1, 2, \ldots, \ell$,
      i. C computes
      $$((\mathsf{que}_v^{(h)})_{v \in [k]}, \mathsf{st}^{(h)}) \leftarrow \mathsf{Query}_0(1^\lambda, x, \mathsf{st}^{(h-1)}, (\mathsf{ans}_v^{(h-1)})_{v \in [k]})$$
      and sends $\mathsf{que}_v^{(h)}$ to S_{i_v}.
      ii. In response, S_{i_v} computes
      $$\mathsf{ans}_v^{(h)} \leftarrow \mathsf{Answer}_0(1^\lambda, p, \mathsf{que}_v^{(h)})$$
      and sends it back to C.
   (b) C obtains $(y^{(j)}, z^{(j)}) \leftarrow \mathsf{Output}_0(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_v^{(\ell)})_{v \in [k]})$.
   (c) If $z^{(j)} = \mathsf{output}$ and $\mathcal{Y} = \emptyset$, C adds $y^{(j)}$ to $\mathcal{Y}$. If $z^{(j)} = \mathsf{failure}$ and $\mathcal{Y} = \emptyset$, C adds a default value $y_0 \in Y_\lambda$ to $\mathcal{Y}$.
   (d) C computes $(\mathcal{G}^{(j+1)}, S^{(j+1)}) \leftarrow \mathsf{Next}_k(\mathcal{G}^{(j)}, S^{(j)}, z^{(j)})$.
4. If $\mathcal{Y} = \{y\}$, C outputs y and otherwise, outputs the default value y_0.

Fig. 2. An actively secure protocol Π based on a conflict-finding protocol

HybEXPT$^{(h)}_{\Pi,\mathcal{A}}(\lambda)$

1. $(x_0, x_1, p, T) \leftarrow \mathcal{A}(1^\lambda)$, where $x_0, x_1 \in X_\lambda$, $p \in P_\lambda$ and $T \subseteq [m]$ is of size at most t.
2. Compute $(\mathcal{G}^{(1)}, S^{(1)})$ in the same manner as Step 2 of Π.
3. For each $j = 1, 2, \ldots, h-1$,
   (a) Execute $z^{(j)} \leftarrow \mathrm{EXPT}_{\Pi_{\mathrm{CF}},\mathcal{A},p,T}(S^{(j)}, x_1)$.
   (b) Compute $(\mathcal{G}^{(j+1)}, S^{(j+1)}) \leftarrow \mathsf{Next}_k(\mathcal{G}^{(j)}, S^{(j)}, z^{(j)})$.
4. For each $j = h, h+1, \ldots, N$,
   (a) Execute $z^{(j)} \leftarrow \mathrm{EXPT}_{\Pi_{\mathrm{CF}},\mathcal{A},p,T}(S^{(j)}, x_0)$.
   (b) Compute $(\mathcal{G}^{(j+1)}, S^{(j+1)}) \leftarrow \mathsf{Next}_k(\mathcal{G}^{(j)}, S^{(j)}, z^{(j)})$.
5. $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

Note that $\mathsf{Hyb}^{(1)}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{Priv}^0_{\Pi,\mathcal{A}}(\lambda)$. We define $\mathsf{Hyb}^{(N+1)}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{Priv}^1_{\Pi,\mathcal{A}}(\lambda)$ and

$$\mathsf{Adv}^{(h)}_{\Pi,\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Hyb}^{(h)}_{\Pi,\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{Hyb}^{(h+1)}_{\Pi,\mathcal{A}}(\lambda) = 0\right]\right|.$$

Then we have that $\mathsf{Adv}_{\Pi,\mathcal{A}}(\lambda) \leq \sum_{h=1}^{N} \mathsf{Adv}^{(h)}_{\Pi,\mathcal{A}}(\lambda)$.

We show that for any h, there exists an adversary $\mathcal{B}$ against the privacy of Π_{CF} such that $\mathsf{Adv}^{(h)}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathrm{CF}}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda)$. If this can be shown, since $N = O(m^2)$ is polynomial in λ, it follows that $\mathsf{Adv}_{\Pi,\mathcal{A}}(\lambda) \leq N \cdot \sup_{\mathcal{B}} \mathsf{Adv}^{\mathrm{CF}}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda) = \mathsf{negl}(\lambda)$.

We construct $\mathcal{B}$ as follows:

$\mathcal{B}$

Setup. On input 1^λ,
1. Run $(x_0, x_1, p, T) \leftarrow \mathcal{A}(1^\lambda)$.
2. For each $j = 1, 2, \ldots, h-1$,
   (a) Execute $z^{(j)} \leftarrow \mathrm{EXPT}_{\Pi_{\mathrm{CF}},\mathcal{A},p,T}(S^{(j)}, x_1)$.
   (b) Compute $(\mathcal{G}^{(j+1)}, S^{(j+1)}) \leftarrow \mathsf{Next}_k(\mathcal{G}^{(j)}, S^{(j)}, z^{(j)})$.
3. Output (x_0, x_1, p, T'), where $S^{(h)} = \{i_1, i_2, \ldots, i_k\}$ $(i_1 < i_2 < \cdots < i_k)$, and $T' = \{v \in [k] : i_v \in T\}$.

Protocol. During the execution of Π_{CF}, $\mathcal{B}$ emulates the behavior of $\mathcal{A}$ in the h-th instance of Π_{CF}. More precisely, to compute an answer of the v-th server such that $i_v \in T$ for $v \in [k]$, $\mathcal{B}$ lets $\mathcal{A}$ compute an answer of S_{i_v} in the h-th instance of Π_{CF}.

Output. To output a guess b', $\mathcal{B}$ receives a z-output z of Π_{CF} and does the following:
1. Compute $(\mathcal{G}^{(h+1)}, S^{(h+1)}) \leftarrow \mathsf{Next}_k(\mathcal{G}^{(h)}, S^{(h)}, z)$.
2. For each $j = h+1, h+2, \ldots, N$,
   (a) Execute $z^{(j)} \leftarrow \mathrm{EXPT}_{\Pi_{\mathrm{CF}},\mathcal{A},p,T}(S^{(j)}, x_0)$.
   (b) Compute $(\mathcal{G}^{(j+1)}, S^{(j+1)}) \leftarrow \mathsf{Next}_k(\mathcal{G}^{(j)}, S^{(j)}, z^{(j)})$.
3. Let $\mathcal{A}$ output a guess $b' \in \{0,1\}$.

Consider the experiment $\mathrm{EXPT}^{\mathrm{CF},0}_{\Pi_{\mathrm{CF}},\mathcal{B}}$ defining $\mathsf{Priv}^{\mathrm{CF},0}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda)$ (see Definition 3):

$\mathrm{EXPT}^{\mathrm{CF},0}_{\Pi_{\mathrm{CF}},\mathcal{B}}$

1. $(x_0, x_1, p, T') \leftarrow \mathcal{B}(1^\lambda)$.
2. Execute Π_{CF} with k servers on a client input x_0 and a common server input p, in which answers of servers in T' are computed by $\mathcal{B}$. Formally, for each $j = 1, \ldots, \ell$,

$$((\mathsf{que}_v^{(j)})_{v\in[k]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}_0(1^\lambda, x_0, \mathsf{st}^{(j-1)}, (\mathsf{ans}_v^{(j-1)})_{v\in[k]}),$$
$$\mathsf{ans}_v^{(j)} \leftarrow \mathsf{Answer}_0(1^\lambda, p, \mathsf{que}_v^{(h)})$$

except that for v with $i_v \in T$, $\mathsf{ans}_v^{(j)}$ is computed by $\mathcal{B}$.
3. Let $(y, z) \leftarrow \mathsf{Output}_0(1^\lambda, \mathsf{st}^{(\ell)}, (\mathsf{ans}_v^{(\ell)})_{v\in[k]})$.
4. $\mathcal{B}$ is given z and outputs a bit $b' \in \{0, 1\}$.

We have that $\mathsf{Hyb}^{(h)}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{Priv}^{\mathrm{CF},0}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda)$ from the following observations:

- From the definition of **Setup** of $\mathcal{B}$ and the fact that a client input is set to x_1, **Setup** corresponds to Steps 1–3 in $\mathrm{HybEXPT}^{(h)}_{\Pi,\mathcal{A}}(\lambda)$ and hence the distributions of $(z^{(1)}, \mathcal{G}^{(2)}, \mathcal{S}^{(2)}), \ldots, (z^{(h-1)}, \mathcal{G}^{(h)}, S^{(h)})$ are the same in both experiments $\mathrm{EXPT}^{\mathrm{CF},0}_{\Pi_{\mathrm{CF}},\mathcal{B}}$ and $\mathrm{HybEXPT}^{(h)}_{\Pi,\mathcal{A}}(\lambda)$.
- From the definition of **Protocol** of $\mathcal{B}$ and the fact that a client input is set to x_0, the distribution of z in the experiment $\mathrm{EXPT}^{\mathrm{CF},0}_{\Pi_{\mathrm{CF}},\mathcal{B}}$ is the same as that of $z^{(h)}$ in the experiment $\mathrm{HybEXPT}^{(h)}_{\Pi,\mathcal{A}}(\lambda)$.
- Since $(\mathcal{G}^{(h+1)}, S^{(h+1)})$ is a function of $(\mathcal{G}^{(h)}, S^{(h)}, z^{(h)})$, the distributions of $(\mathcal{G}^{(h+1)}, S^{(h+1)})$ are the same in both experiments.
- From the definition of **Output** of $\mathcal{B}$ and the fact that a client input is set to x_0, the distributions of $(z^{(h+1)}, \mathcal{G}^{(h+2)}, \mathcal{S}^{(h+2)}), \ldots, (z^{(N)}, \mathcal{G}^{(N+1)}, S^{(N+1)})$ are the same in both experiments.
- The distributions of b' outputted by $\mathcal{A}$ are the same in both experiments.

In a similar way, we have that $\mathsf{Hyb}^{(h+1)}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{Priv}^{\mathrm{CF},1}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda)$. Therefore, we obtain that $\mathsf{Adv}^{(h)}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathrm{CF}}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda)$.

Byzantine-robustness. Let $\mathcal{A}$ be a stateful adversary against the Byzantine-robustness of Π. Let $\mathsf{BR}_{\Pi,\mathcal{A}}(\lambda)$ be a random variable defined in Definition 2. That is, $\mathsf{BR}_{\Pi,\mathcal{A}}(\lambda)$ is the output of the following experiment:

$\mathrm{EXPT}^{\mathrm{BR}}_{\Pi,\mathcal{A}}$

1. $(x, p, T) \leftarrow \mathcal{A}(1^\lambda)$.
2. Execute Π on a client input x and a common server input p, in which answers of servers in T are computed by $\mathcal{A}$.
3. Return 1 if the output of C is different from $F_\lambda(p, x)$, and otherwise return 0.

We show the probability that $\mathsf{BR}_{\Pi,\mathcal{A}}(\lambda) = 1$ is negligible. The event that $\mathsf{BR}_{\Pi,\mathcal{A}}(\lambda) = 1$ happens only if after N iterations of Step 3 of Π, one of the following three conditions holds: (1) $\mathcal{Y} = \emptyset$, (2) $\mathcal{Y} = \{y_0\}$, where y_0 is the default value, or (3) $\mathcal{Y} = \{y\}$ and $y \neq F_\lambda(p, x)$.

We show that the first case never occurs. Assume otherwise, then for all j, the z-output $z^{(j)}$ of the j-th instance of Π_{CF} computed at Step 3(b) of Π is a non-trivial partition $(G_0^{(j)}, G_1^{(j)})$ of a set of servers $S^{(j)}$ involved in the j-th instance of Π_{CF}. From the construction of Next_k, for all j, $\mathcal{G}^{(j+1)}$ is a graph obtained by removing one edge $e^{(j)}$, which has one endpoint in each subset of $(G_0^{(j)}, G_1^{(j)})$, from $\mathcal{G}^{(j)}$. Since the conflict-finding property of Π_{CF} ensures that either $G_0^{(j)}$ or $G_1^{(j)}$ includes $H := [m] \backslash T$, the removed edge $e^{(j)} = (i_1, i_2)$ satisfies $i_1 \in T$ or $i_2 \in T$ and hence for all j, the subgraph $\mathcal{G}^{(j)}[H]$ is a complete graph. Since N is larger than the total number $N' = \binom{m}{2} - \binom{m-|T|}{2}$ of unordered pairs (i_1, i_2) such that $i_1 \in T$ or $i_2 \in T$, $\mathcal{G}^{(N')}$ has no edge $e = (i_1, i_2)$ such that $i_1 \in T$ or $i_2 \in T$. Therefore, a set of servers $S^{(N')} \leftarrow \mathcal{C}_k(\mathcal{G}^{(N')})$ involved in the N'-th instance of Π_{CF} is a subset of H since $k \leq m - t \leq |H|$. We have assumed that $z^{(N')}$ is a non-trivial partition $(G_0^{(N')}, G_1^{(N')})$ of $S^{(N')}$ but the conflict-finding property ensures that $H \subseteq G_0^{(N')}$ or $H \subseteq G_1^{(N')}$, which is contradiction.

If the second or third case occurs, then there exists $h \in [N]$ for which the following event $\mathsf{E}^{(h)}$ occurs: The outputs $(y^{(h)}, z^{(h)})$ of the h-th instance of Π_{CF} satisfy

$$y^{(h)} \neq F_\lambda(p, x) \text{ and } z^{(h)} = \mathsf{output}, \text{ or } y^{(h)} = \bot \text{ and } z^{(h)} = \mathsf{failure}. \tag{3}$$

We show that for all h, there exists an adversary $\mathcal{B}$ against the soundness of Π_{CF} such that

$$\Pr\left[\mathsf{E}^{(h)}\right] = \Pr\left[\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda) = 1\right], \tag{4}$$

where $\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda)$ is defined in Definition 3.

We construct $\mathcal{B}$ as follows:

$\mathcal{B}$

Setup. On input 1^λ,
1. Run $(x, p, T) \leftarrow \mathcal{A}(1^\lambda)$.
2. For each $j = 1, 2, \ldots, h-1$,
   (a) Execute $z^{(j)} \leftarrow \mathrm{EXPT}_{\Pi_{\mathrm{CF}},\mathcal{A},p,T}(S^{(j)}, x)$.
   (b) Compute $(\mathcal{G}^{(j+1)}, S^{(j+1)}) \leftarrow \mathsf{Next}_k(\mathcal{G}^{(j)}, S^{(j)}, z^{(j)})$.
3. Output (x, p, T'), where $S^{(h)} = \{i_1, i_2, \ldots, i_k\}$ $(i_1 < i_2 < \cdots < i_k)$, and $T' = \{v \in [k] : i_v \in T\}$.

Protocol. During the execution of Π_{CF}, $\mathcal{B}$ emulates the behavior of $\mathcal{A}$ in the h-th instance of Π_{CF}. More precisely, to compute an answer of the v-th server such that $i_v \in T$ for $v \in [k]$, $\mathcal{B}$ lets $\mathcal{A}$ compute an answer of S_{i_v} in the h-th instance of Π_{CF}.

Then the experiment defining $\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda)$ corresponds to the first-to-h-th iterations of Step 3 of Π in the experiment $\mathrm{EXPT}^{\mathrm{BR}}_{\Pi,\mathcal{A}}$. Hence the distribution of $(y^{(h)}, z^{(h)})$ is the same in both experiments. Since the condition (3) that $\mathsf{E}^{(h)}$ happens is the same as the condition that $\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda) = 1$ (see Definition 3), we obtain the equality (4).

Therefore, we conclude that

$$\Pr\left[\mathsf{BR}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \leq \sum_{h=1}^{N} \Pr\left[E^{(h)}\right] \leq N \cdot \sup_{\mathcal{B}} \Pr\left[\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{B}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,.$$

Complexity. The round complexity is $\ell' = \ell N = O(\ell m^2)$. Since the running time of $\mathcal{C}_k$ and $\mathcal{E}$ is $O(m^2)$ (see Section 4.1), the running time of Next_k is also $O(m^2)$. Since $N = O(m^2)$, the client-side computational complexity is $O(N(\text{c-Comp}(\Pi_{\mathrm{CF}}) + m^2)) = O(m^2 \cdot \text{c-Comp}(\Pi_{\mathrm{CF}}) + m^4)$ and communication complexity is $O(m^2 \cdot \mathrm{Comm}(\Pi_{\mathrm{CF}}))$. Also, the server-side computational complexity is $O(N \cdot \text{c-Comp}(\Pi_{\mathrm{CF}})) = O(m^2 \cdot \text{c-Comp}(\Pi_{\mathrm{CF}}))$.

F Proof of Proposition 7

Consider a protocol Π described in Fig. 3.

Correctness. Assume that all servers are honest. For any $i \in [m]$, it holds that

$$\mathsf{ans}_i^{\langle\mu\rangle} = \mathsf{ans}_i^{\langle\mu\rangle}(i) = \mathsf{Answer}(p, \mathsf{que}_i^{\langle i\rangle}) \ (\forall \mu \in [M] \setminus \{\mu_*\}).$$

Also, for any $i, j \in [m]$, it holds that

$$\mathsf{ans}_k^{\langle\mu\rangle}(i) = \mathsf{ans}_k^{\langle\mu\rangle}(j) = \mathsf{Answer}(p, \mathsf{que}_k^{\langle\mu\rangle}) \ (\forall k \in [m], \mu \in [M] \setminus \{\mu_*\}).$$

Hence, any pair (i, j) are equivalent under the equivalence relation defined at Step (c)ii of $\mathcal{Z}$. Therefore, $\mathcal{Z}$ outputs $z = \mathsf{output}$. Then C outputs

$$\mathsf{Output}(\mathsf{st}^{\langle\mu_*\rangle}, \mathsf{ans}_1^{\langle\mu_*\rangle}, \ldots, \mathsf{ans}_m^{\langle\mu_*\rangle}) = F_\lambda(p, x)$$

since $((\mathsf{que}_i^{\langle\mu_*\rangle})_{i\in[m]}, \mathsf{st}^{\langle\mu_*\rangle}) \leftarrow \mathsf{Query}(x)$ and $\mathsf{ans}_1^{\langle\mu_*\rangle} = \mathsf{Answer}(p, \mathsf{que}_i^{\langle\mu_*\rangle})$.

Privacy. Let $\mathcal{A}$ be a stateful adversary against the privacy of Π'. Let $\mathsf{Priv}^b_{\Pi',\mathcal{A}}(\lambda)$ be a random variable defined in Definition 3. That is, $\mathsf{Priv}^b_{\Pi',\mathcal{A}}(\lambda)$ is the output of $\mathcal{A}$ in the following experiment $\mathrm{EXPT}^{\mathrm{Priv},b}_{\Pi',\mathcal{A}}$:

$\mathrm{EXPT}^{\mathrm{Priv},b}_{\Pi',\mathcal{A}}$

1. $(x_0, x_1, p, T) \leftarrow \mathcal{A}(1^\lambda)$.
2. Execute Π' on a client input x_b and a common server input p, in which answers of servers in T are computed by $\mathcal{A}$.
3. Let (y, z) be the output of C.
4. Return $b' \leftarrow \mathcal{A}(z)$.

Notations.

- Let $\mathit{\Pi} = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be a 1-round m-server protocol.
- Let $M \in \mathbb{N}$.
- Let $x_{\mathrm{def}} \in X_\lambda$ be a default value.
- A client C has an input $x \in X_\lambda$ and every server S_i has a common input $p \in P_\lambda$.

First round.

1. C chooses $\mu_* \leftarrow_\$ [M]$ and computes

$$((\mathsf{que}_i^{\langle\mu\rangle})_{i\in[m]}, \mathsf{st}^{\langle\mu\rangle}) \leftarrow \begin{cases} \mathsf{Query}(1^\lambda, x), & \text{if } \mu = \mu_*, \\ \mathsf{Query}(1^\lambda, x_{\mathrm{def}}), & \text{otherwise,} \end{cases}$$

for all $\mu \in [M]$. C sends $(\mathsf{que}_i^{\langle\mu\rangle})_{\mu\in[M]}$ to each server S_i.

2. In response, S_i computes

$$\mathsf{ans}_i^{\langle\mu\rangle} = \mathsf{Answer}(1^\lambda, p, \mathsf{que}_i^{\langle\mu\rangle}),\ \forall\mu \in [M]$$

and sends $(\mathsf{ans}_i^{\langle\mu\rangle})_{\mu\in[M]}$ to C.

Second round.

1. C sends $(\mathsf{que}_k^{\langle\mu\rangle})_{k\in[m],\mu\neq\mu_*}$ to all servers.
2. In response, each server S_i computes

$$\mathsf{ans}_k^{\langle\mu\rangle}(i) = \mathsf{Answer}(1^\lambda, p, \mathsf{que}_k^{\langle\mu\rangle}),\ \forall k \in [m],\ \forall\mu \in [M] \setminus \{\mu_*\}$$

and sends $(\mathsf{ans}_k^{\langle\mu\rangle}(i))_{k\in[m],\mu\neq\mu_*}$ to C.

Output.

1. C runs the following algorithm $\mathcal{Z}$ to obtain z:
$\mathcal{Z}((\mathsf{ans}_i^{\langle\mu\rangle})_{i\in[m],\mu\neq\mu_*}, (\mathsf{ans}_k^{\langle\mu\rangle}(i))_{i\in[m],k\in[m],\mu\neq\mu_*}) \to z$**.**
   (a) Let $G_0 \leftarrow \emptyset$.
   (b) If there exist $i \in [m]$ and $\mu \in [M] \setminus \{\mu_*\}$ such that $\mathsf{ans}_i^{\langle\mu\rangle} \neq \mathsf{ans}_i^{\langle\mu\rangle}(i)$, then output $z = (G_0, G_1)$, where $G_0 = \{i\}$ and $G_1 = [m] \setminus \{i\}$.
   (c) Otherwise,
      i. Define $v_i = (\mathsf{ans}_k^{\langle\mu\rangle}(i))_{k\in[m],\mu\neq\mu_*}$ for all $i \in [m]$.
      ii. Partition $[m]$ into equivalence classes $G_0', G_1', \ldots, G_\ell'$ under the following equivalence relation: $i \sim j \overset{\text{def}}{\iff} v_i = v_j$.
      iii. If $\ell = 0$ (i.e., all servers belong to the same equivalence class), then output $z = \mathsf{output}$. Otherwise, output $z = (G_0, G_1)$, where $G_0 = G_0'$ and $G_1 = G_1' \cup \cdots \cup G_\ell'$.
2. If $z = \mathsf{output}$, C computes

$$y \leftarrow \mathsf{Output}(1^\lambda, \mathsf{st}^{\langle\mu_*\rangle}, \mathsf{ans}_1^{\langle\mu_*\rangle}, \ldots, \mathsf{ans}_m^{\langle\mu_*\rangle}).$$

and outputs (y, z). Otherwise, C outputs $(\perp, z)$.

Fig. 3. A conflict-finding protocol $\mathit{\Pi}'$ with non-negligible soundness error

We show that $\mathsf{Adv}_{\Pi',\mathcal{A}}(\lambda) := |\Pr\left[\mathsf{Priv}^0_{\Pi',\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{Priv}^1_{\Pi',\mathcal{A}}(\lambda) = 0\right]|$ is negligible. To this end, we show that there exists an adversary $\mathcal{B}$ against the privacy of Π such that $\mathsf{Priv}^0_{\Pi,\mathcal{B}}(\lambda) = \mathsf{Priv}^0_{\Pi',\mathcal{A}}(\lambda)$, $\mathsf{Priv}^1_{\Pi,\mathcal{B}}(\lambda) = \mathsf{Priv}^1_{\Pi',\mathcal{A}}(\lambda)$ and hence $\mathsf{Adv}_{\Pi',\mathcal{A}}(\lambda) = \mathsf{Adv}_{\Pi,\mathcal{B}}(\lambda)$. If this can be shown, we conclude that $\mathsf{Adv}_{\Pi',\mathcal{A}}(\lambda) \leq \sup_{\mathcal{B}} \mathsf{Adv}_{\Pi,\mathcal{B}}(\lambda) = \mathsf{negl}(\lambda)$.

We construct $\mathcal{B}$ as follows:

$\mathcal{B}$

Setup. On input 1^λ, output $(x_0, x_1, p, T) \leftarrow \mathcal{A}(1^\lambda)$.

Protocol. Assume that $\mathcal{B}$ receives challenge queries $(\mathsf{que}_i)_{i \in T}$ during the execution of Π. $\mathcal{B}$ does the following:

1. Choose $\mu_* \leftarrow_{\$} [M]$ and emulate the behavior of $\mathcal{A}$ in the first round, embedding the challenge queries into the μ_*-th query $(\mathsf{que}_j^{\langle \mu_* \rangle})_{j \in T}$. More precisely,
   (a) Choose $\mu_* \leftarrow_{\$} [M]$.
   (b) Compute
   $$((\mathsf{que}_i^{\langle \mu \rangle})_{i \in [m]}, \mathsf{st}^{\langle \mu \rangle}) \leftarrow \mathsf{Query}(1^\lambda, x_{\mathrm{def}}),\ \forall \mu \in [M] \setminus \{\mu_*\}.$$
   (c) Let $(\mathsf{que}_i^{\langle \mu_* \rangle})_{i \in T} \leftarrow (\mathsf{que}_i)_{i \in T}$.
   (d) Let $(\mathsf{ans}_i^{\langle \mu \rangle})_{i \in T, \mu \in [M]} \leftarrow \mathcal{A}((\mathsf{que}_i^{\langle \mu \rangle})_{i \in T, \mu \in [M]})$.
2. Let $(\mathsf{ans}_k^{\langle \mu \rangle}(i))_{i \in T, k \in [m], \mu \in [M]} \leftarrow \mathcal{A}((\mathsf{que}_k^{\langle \mu \rangle})_{k \in [m], \mu \neq \mu_*})$.
3. For each $i \notin T$, let
   $$\mathsf{ans}_k^{\langle \mu \rangle}(i) \leftarrow \mathsf{Answer}(1^\lambda, p, \mathsf{que}_k^{\langle \mu \rangle}),\ \forall k \in [m], \forall \mu \in [M] \setminus \{\mu_*\}.$$
4. Let $z \leftarrow \mathcal{Z}((\mathsf{ans}_i^{\langle \mu \rangle})_{i \in [m], \mu \neq \mu_*}, (\mathsf{ans}_k^{\langle \mu \rangle}(i))_{i \in [m], k \in [m], \mu \neq \mu_*})$, where $\mathcal{Z}$ is the algorithm defined in Fig. 3.
5. Output $b' \leftarrow \mathcal{A}(z)$.

Consider the experiment $\mathrm{EXPT}^{\mathrm{Priv},0}_{\Pi,\mathcal{B}}$ defining $\mathsf{Priv}^0_{\Pi,\mathcal{B}}(\lambda)$, where challenge queries are computed on input x_0. Observe that at Step 1, $\mathcal{B}$ perfectly simulates queries that $\mathcal{A}$ receives in the first round of Π'. Also, at Step 2, $\mathcal{B}$ perfectly simulates queries that $\mathcal{A}$ receives in the second round of Π'. This is because $\mathcal{B}$ itself has computed $(\mathsf{que}_i^{\langle \mu \rangle})_{i \in [m]}$ for $\mu \in [M] \setminus \{\mu_*\}$. $\mathcal{B}$ correctly simulates answers from honest servers in the second round since the computation of the answers do not involve a client input (see "Second round" in Fig. 3). Thus, the joint distributions of $(\mathsf{ans}_i^{\langle \mu \rangle})_{i \in [m], \mu \neq \mu_*}$ and $(\mathsf{ans}_k^{\langle \mu \rangle}(i))_{i \in [m], k \in [m], \mu \neq \mu_*}$ are the same in both experiments $\mathrm{EXPT}^{\mathrm{Priv},0}_{\Pi,\mathcal{B}}$ and $\mathrm{EXPT}^{\mathrm{Priv},0}_{\Pi',\mathcal{A}}$. In particular, the distributions of the output z of $\mathcal{Z}$ are also identical. We therefore conclude that $\mathsf{Priv}^0_{\Pi,\mathcal{B}}(\lambda) = \mathsf{Priv}^0_{\Pi',\mathcal{A}}(\lambda)$, and similarly that $\mathsf{Priv}^1_{\Pi,\mathcal{B}}(\lambda) = \mathsf{Priv}^1_{\Pi',\mathcal{A}}(\lambda)$.

Conflict-finding. Let $\mathcal{A}$ be a stateful adversary against the conflict-finding property of Π'. We show that it always hold that $\mathsf{CF}_{\Pi',\mathcal{A}}(\lambda) = 0$. That is, if Π' outputs $z = (G_0, G_1)$ as the z-output, it always hold that $\overline{T} \subseteq \overline{G_0}$ or $\overline{T} \subseteq \overline{G_1}$, where T is a set of servers corrupted by $\mathcal{A}$ in the experiment defining $\mathsf{CF}_{\Pi',\mathcal{A}}(\lambda)$.

To this end, observe that for any $i \notin T$, it holds that

$$\mathsf{ans}_i^{\langle\mu\rangle} = \mathsf{ans}_i^{\langle\mu\rangle}(i) = \mathsf{Answer}(p, \mathsf{que}_i^{\langle i\rangle}) \ (\forall \mu \in [M] \setminus \{\mu_*\}).$$

That is, if $\mathcal{Z}$ outputs $z = (\{i\}, [m] \setminus \{i\})$ for some i at Step (b), then it holds that $\overline{T} \subseteq [m] \setminus \{i\} = \overline{G_0}$. Furthermore, for any $i, j \notin T$, it holds that

$$\mathsf{ans}_k^{\langle\mu\rangle}(i) = \mathsf{ans}_k^{\langle\mu\rangle}(j) = \mathsf{Answer}(p, \mathsf{que}_k^{\langle\mu\rangle}) \ (\forall k \in [m], \mu \in [M] \setminus \{\mu_*\})$$

and hence $\overline{T}$ is contained in one of equivalent classes $G'_0, \ldots, G'_\ell$ under the equivalence relation defined at Step (c)ii of $\mathcal{Z}$. If $\overline{T} \subseteq G'_0$, then $\overline{T} \subseteq \overline{G_1}$, and if $\overline{T} \subseteq G'_j$ for some $j \neq 0$, then $\overline{T} \subseteq \overline{G_0}$.

Soundness. Let $\mathcal{A}$ be a stateful algorithm. Let $\mathsf{Sound}_{\Pi',\mathcal{A}}(\lambda)$ be a random variable defined in Definition 3. That is, $\mathsf{Sound}_{\Pi',\mathcal{A}}(\lambda)$ is the output of the following experiment $\mathrm{EXPT}^{\mathrm{Sound}}_{\Pi',\mathcal{A}}$:

$\mathrm{EXPT}^{\mathrm{Sound}}_{\Pi',\mathcal{A}}$

1. $(x, p, T) \leftarrow \mathcal{A}(1^\lambda)$.
2. Execute Π' on a client input x and a common server input p, in which answers of servers in T are computed by $\mathcal{A}$.
3. Let (y, z) be the output of C.
4. Return 1 if $y \neq F_\lambda(p, x)$ and $z = \mathsf{output}$, or $y = \perp$ and $z = \mathsf{failure}$. Otherwise return 0.

We will show that

$$\Pr[\mathsf{Sound}_{\Pi',\mathcal{A}}(\lambda) = 1] \leq \frac{m}{M} + \mathsf{negl}(\lambda) \tag{5}$$

for some negligible function $\mathsf{negl}(\lambda)$. Define Good be the event that

$$\mathsf{ans}_i^{\langle\mu_*\rangle} = \mathsf{Answer}(p, \mathsf{que}_i^{\langle\mu_*\rangle}),\ \forall i \in [m].$$

Also, define $\mathsf{Verified}$ be the event that

$$\mathsf{ans}_i^{\langle\mu\rangle} = \mathsf{Answer}(p, \mathsf{que}_i^{\langle\mu\rangle}),\ \forall \mu \in [M] \setminus \{\mu_*\},\ \forall i \in [m].$$

From the construction of Π', it is clear that C never outputs (y, z) such that $y = \perp$ and $z = \mathsf{failure}$. Hence $\mathsf{Sound}_{\Pi',\mathcal{A}}(\lambda) = 1$ if and only if $y \neq F_\lambda(p, x)$ and $z = \mathsf{output}$. Furthermore, if $z = \mathsf{output}$, then the algorithm $\mathcal{Z}$ must not have proceeded to Step (b). This means that $\mathsf{ans}_i^{\langle\mu\rangle} = \mathsf{ans}_i^{\langle\mu\rangle}(i)$ for any $i \in [m]$ and any $\mu \in [M] \setminus \{\mu_*\}$. Furthermore, the equivalence class computed at Step (c)ii is $G'_0 = [m]$ and hence any $i \in [m]$ is equivalent to any $j \notin T$, i.e., $v_i = v_j$. Since $v_j = (\mathsf{Answer}(p, \mathsf{que}_k^{\langle\mu\rangle}))_{k \in [m], \mu \neq \mu_*}$ if $j \notin T$, we have that $\mathsf{ans}_i^{\langle\mu\rangle} = \mathsf{ans}_i^{\langle\mu\rangle}(i) = \mathsf{ans}_i^{\langle\mu\rangle}(j) = \mathsf{Answer}(p, \mathsf{que}_i^{\langle\mu\rangle})$ for all $\mu \in [M] \setminus \{\mu_*\}$, which implies that the event

$\mathsf{Verified}$ occurs. Therefore, we have

$$
\begin{aligned}
&\Pr[\mathsf{Sound}_{\Pi',\mathcal{A}}(\lambda) = 1] \\
\le\ &\Pr[y \ne F_\lambda(p,x) \wedge \mathsf{Verified}] \\
=\ &\Pr[y \ne F_\lambda(p,x) \wedge \mathsf{Verified} \wedge \mathsf{Good}] + \Pr\left[y \ne F_\lambda(p,x) \wedge \mathsf{Verified} \wedge \overline{\mathsf{Good}}\right] \\
\le\ &\Pr[y \ne F_\lambda(p,x) \wedge \mathsf{Verified} \wedge \mathsf{Good}] + \Pr\left[\mathsf{Verified} \wedge \overline{\mathsf{Good}}\right]
\end{aligned}
$$

The first term is upper bounded by a negligible function $\mathsf{negl}'(\lambda)$ since the correctness of Π ensures that $\mathsf{Output}(\mathsf{st}^{\langle\mu_*\rangle}, (\mathsf{Answer}(p, \mathsf{que}_i^{\langle\mu_*\rangle}))_{i\in[m]}) = F_\lambda(p,x)$ except with negligible probability.

We denote the second term by p_0, i.e., p_0 is the probability that the events $\mathsf{Verified}$ and $\overline{\mathsf{Good}}$ occurs in the experiment $\mathrm{EXPT}^{\mathrm{Sound}}_{\Pi',\mathcal{A}}$. Let $\mathrm{HybEXPT}^{\mathrm{Sound}}_{\Pi',\mathcal{A}}$ be the same experiment as $\mathrm{EXPT}^{\mathrm{Sound}}_{\Pi',\mathcal{A}}$ except that Π' is executed on the default value x_{def} instead of x. We define events $\mathsf{HybVerified}$ and $\mathsf{HybGood}$ in the same manner as $\mathsf{Verified}$ and Good in $\mathrm{EXPT}^{\mathrm{Sound}}_{\Pi',\mathcal{A}}$, respectively. Also, define p_1 as the probability that $\mathsf{HybVerified}$ and $\overline{\mathsf{HybGood}}$ occurs in the experiment $\mathrm{HybEXPT}^{\mathrm{Sound}}_{\Pi',\mathcal{A}}$. From the privacy of Π, we can see that $|p_0 - p_1|$ is negligible. Indeed, we construct an adversary $\mathcal{B}$ against the privacy of Π as follows:

$\mathcal{B}$

Setup. On input 1^λ,
1. Run $(x,p,T) \leftarrow \mathcal{A}(1^\lambda)$.
2. Output $(x, x_{\mathrm{def}}, p, T)$.

Protocol. Assume that $\mathcal{B}$ receives challenge queries $(\mathsf{que}_j)_{j\in T}$ during the execution of Π. $\mathcal{B}$ chooses $\mu_* \leftarrow_\$ [M]$ and simulates the behavior of $\mathcal{A}$ in the first round, embedding the challenge queries into the μ_*-th query $(\mathsf{que}_j^{\langle\mu_*\rangle})_{j\in T}$. Return 1 if $\mathcal{A}$ modifies the μ-th answers exactly for $\mu = \mu_*$. Otherwise return 0. More precisely,
1. Choose $\mu_* \leftarrow_\$ [M]$.
2. Compute $((\mathsf{que}_j^{\langle\mu\rangle})_{j\in[m]}, \mathsf{st}^{\langle\mu\rangle}) \leftarrow \mathsf{Query}(1^\lambda, x_{\mathrm{def}})$ for all $\mu \in [M] \setminus \{\mu_*\}$.
3. Let $(\mathsf{que}_j^{\langle\mu_*\rangle})_{j\in T} \leftarrow (\mathsf{que}_j)_{j\in T}$.
4. Let $(\mathsf{ans}_j^{\langle\mu\rangle})_{j\in T,\mu\in[M]} \leftarrow \mathcal{A}((\mathsf{que}_j^{\langle\mu\rangle})_{j\in T,\mu\in[M]})$.
5. Return 1 if

$$
\begin{aligned}
&\mathsf{ans}_i^{\langle\mu_*\rangle} \ne \mathsf{Answer}(1^\lambda, p, \mathsf{que}_i^{\langle\mu_*\rangle})\ (\exists i \in [m]) \\
\text{and } &\mathsf{ans}_i^{\langle\mu\rangle} = \mathsf{Answer}(1^\lambda, p, \mathsf{que}_i^{\langle\mu\rangle})\ (\forall \mu \in [M]\setminus\{\mu_*\}, \forall i \in [m]).
\end{aligned}
$$

Otherwise, return 0.

Observe that the experiment defining $\mathsf{Priv}^0_{\Pi,\mathcal{B}}(\lambda)$ executes Π on a client input $x_0 = x$ while the experiment defining $\mathsf{Priv}^1_{\Pi,\mathcal{B}}(\lambda)$ executes Π on $x_1 = x_{\mathrm{def}}$. Furthermore, if x_0 is inputted, $\mathcal{B}$ returns 1 if and only if $\mathsf{Verified}$ and $\overline{\mathsf{Good}}$ occurs. Similarly, if x_1 is inputted, $\mathcal{B}$ returns 1 if and only if $\mathsf{HybVerified}$ and

$\overline{\mathsf{HybGood}}$ occurs. Therefore, we have that for $b \in \{0,1\}$,

$$\Pr\left[\mathsf{Priv}^{b}_{\Pi,\mathcal{B}}(\lambda)=1\right]=p_b$$

and hence there is a negligible function $\mathsf{negl}''(\lambda)$ such that

$$|p_0-p_1|=\mathsf{Adv}_{\Pi,\mathcal{B}}(\lambda)\leq \sup_{\mathcal{B}}\mathsf{Adv}_{\Pi,\mathcal{B}}(\lambda)\leq \mathsf{negl}''(\lambda).$$

Note that μ_* is perfectly hidden from the adversary $\mathcal{A}$ in the first round of Π' executed in $\mathrm{HybEXPT}^{\mathrm{Sound}}_{\Pi',\mathcal{A}}$. Furthermore, whether $\mathsf{HybVerified}$ and $\mathsf{HybGood}$ occur is determined by the behavior of $\mathcal{A}$ in the first round. Since μ_* is randomly chosen from $[M]$, we have that

$$\begin{aligned}
p_1 &= \Pr\left[\mathsf{HybVerified}\wedge\overline{\mathsf{HybGood}}\right]\\
&\leq \sum_{i\in[m]}\Pr\left[\begin{array}{c}\text{In }\mathrm{HybEXPT}^{\mathrm{Sound}}_{\Pi',\mathcal{A}}\text{, it holds that}\\ \mathsf{ans}_i^{\langle\mu\rangle}=\mathsf{Answer}(p,\mathsf{que}_i^{\langle\mu\rangle})\text{ for all }\mu\in[M]\setminus\{\mu_*\}\\ \text{and }\mathsf{ans}_i^{\langle\mu_*\rangle}\neq\mathsf{Answer}(p,\mathsf{que}_i^{\langle\mu_*\rangle})\end{array}\right]\\
&= \frac{m}{M}
\end{aligned}$$

Therefore, we can upper bound the second term as

$$p_0\leq p_1+\mathsf{negl}''(\lambda)\leq\frac{m}{M}+\mathsf{negl}''(\lambda).$$

Letting $\mathsf{negl}(\lambda)=\mathsf{negl}'(\lambda)+\mathsf{negl}''(\lambda)$, we obtain (5).

Complexity. The round complexity is 2. In the first round, the client needs to compute M queries of Π and then the computational complexity is at most $O(\lambda\cdot\text{c-Comp}(\Pi))$. The computational complexity of Step 1 in the output phase is at most $O(m^2M\cdot\mathrm{Comm}(\Pi))=O(m^2M\cdot\text{c-Comp}(\Pi))$ since the client can verify the equivalence between each pair of servers in $O(M\cdot\mathrm{Comm}(\Pi))$ time. The computational complexity of Step 2 is at most $O(\text{c-Comp}(\Pi))$. Thus, we have that $\text{c-Comp}(\Pi')=O(m^2M\cdot\text{c-Comp}(\Pi))$. Every server computes answers to at most $O(mM)$ queries for Π and hence $\text{s-Comp}(\Pi')=O(mM\cdot\text{s-Comp}(\Pi))$. The communication complexity is at most $O(mM\cdot\mathrm{Comm}(\Pi))+O(m^2M\cdot\mathrm{Comm}(\Pi))=O(m^2M\cdot\mathrm{Comm}(\Pi))$.

G Proof of Theorem 2

Let $M=16m=\mathsf{poly}(\lambda)$ and Π' be a ϵ-sound t-conflict-finding protocol for $\epsilon=m/M+\mathsf{negl}(\lambda)=1/16+\mathsf{negl}(\lambda)$ given by Proposition 7. Let $\kappa=\lambda$. Consider a protocol Π_{CF} described in Fig. 4.

Correctness. Assume that all servers are honest. Then the correctness of Π' ensures that $z^{\langle j\rangle}=\mathsf{output}$ and $y^{\langle j\rangle}=F_\lambda(p,x)$ for all $j\in[\kappa]$, except with negligible probability $\kappa\cdot\mathsf{negl}(\lambda)$. Then, $K=\emptyset$ and C outputs $(F_\lambda(p,x),\mathsf{output})$.

Notations.

- Let Π' be the ϵ-sound t-conflict-finding m-server protocol described in Fig. 3.
- Let $\kappa \in \mathbb{N}$.
- A client C has an input $x \in X_\lambda$ and every server S_i has a common input $p \in P_\lambda$.

Protocol.

1. C and servers execute κ independent instances of Π' in parallel.
2. C lets $(y^{\langle 1\rangle}, z^{\langle 1\rangle}), \ldots, (y^{\langle \kappa\rangle}, z^{\langle \kappa\rangle})$ be the outputs of the κ instances of Π'.
3. C defines
$$K = \{j \in [\kappa] : z^{\langle j\rangle} \text{ is a non-trivial partition of } [m]\}.$$
4. If $K \neq \emptyset$, C outputs $(y^{\langle j\rangle}, z^{\langle j\rangle})$ for the minimum number j of K.
5. If $K = \emptyset$ and there exists $y \in Y_\lambda$ such that
$$|\{j \in [\kappa] : y^{\langle j\rangle} = y\}| > \frac{\kappa}{2},$$
then C outputs (y, output).
6. Otherwise, C outputs $(\perp, \mathsf{failure})$.

Fig. 4. A conflict-finding protocol Π_{CF} with negligible soundness error

Privacy. Observe that Π_{CF} just independently runs Π' κ times in parallel. Furthermore, the z-output of Π_{CF} is determined by the z-outputs $z^{\langle 1\rangle}, \ldots, z^{\langle \kappa\rangle}$ of Π'. Thus, the privacy of Π_{CF} follows from that of Π'.

Conflict-finding. If Π_{CF} outputs a non-trivial partition $z = (G_0, G_1)$, it must be equal to the z-output $z^{\langle j\rangle}$ of the j-th instance Π' for some $j \in [\kappa]$. Thus, the conflict-finding property of Π' ensures that either G_0 or G_1 includes the set of all honest servers.

Soundness. Let $\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{A}}(\lambda)$ be the random variable defined in Definition 3, where $\mathcal{A}$ is a stateful adversary against the soundness of Π. We will show that

$$\Pr[\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{A}}(\lambda) = 1] < \mathsf{negl}''(\lambda) \tag{6}$$

for some negligible function $\mathsf{negl}''(\lambda)$. Let Good be the event that there exist more than $\kappa/2$ instances j such that $z^{\langle j\rangle} = \mathsf{output}$ and $y^{\langle j\rangle} = F_\lambda(p, x)$. Then

$$\begin{aligned}&\Pr[\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{A}}(\lambda) = 1]\\ &= \Pr[\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{A}}(\lambda) = 1 \wedge \mathsf{Good}] + \Pr\left[\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{A}}(\lambda) = 1 \wedge \overline{\mathsf{Good}}\right].\end{aligned}$$

- The first term: If Good happens, then C outputs $y = F_\lambda(p, x)$ and $z = \mathsf{output}$, or $y = \perp$ and a non-trivial partition $z = (G_0, G_1)$. Therefore,
$$\Pr[\mathsf{Sound}_{\Pi_{\mathrm{CF}},\mathcal{A}}(\lambda) = 1 \wedge \mathsf{Good}] = 0.$$

– The second term: In Π', the client never outputs $z = \mathsf{failure}$ as shown in the proof of the soundness of Π' (see Proposition 7). That is, for all $j \in [\kappa]$, $z^{\langle j \rangle} = \mathsf{output}$ or $z^{\langle j \rangle}$ is a non-trivial partition. Hence, if Good does not happen, then there exist at least $\kappa/2$ instances j such that $y^{\langle j \rangle} \neq F_\lambda(p, x)$ and $z^{\langle j \rangle} = \mathsf{output}$, or $y^{\langle j \rangle} = \perp$ and $z^{\langle j \rangle}$ is a non-trivial partition. Note that if $z^{\langle j \rangle}$ is a non-trivial partition for some j, then the client outputs $y = \perp$ and a non-trivial partition $z = (G_0, G_1)$, and hence $\mathsf{Sound}_{\Pi_{\mathrm{CF}}, \mathcal{A}}(\lambda) = 0$. Since Π' is an ϵ-sound for $\epsilon = 1/16 + \mathsf{negl}(\lambda)$ and we set $\kappa = \lambda$, we have

$$
\begin{aligned}
& \Pr\left[\mathsf{Sound}_{\Pi_{\mathrm{CF}}, \mathcal{A}}(\lambda) = 1 \wedge \overline{\mathsf{Good}}\right] \\
& \leq \Pr\left[\begin{array}{c}\mathsf{Sound}_{\Pi_{\mathrm{CF}}, \mathcal{A}}(\lambda) = 1 \text{ and there exist } \kappa/2 \text{ instances } j \\ \text{such that } y^{\langle j \rangle} \neq F_\lambda(p, x) \text{ and } z^{\langle j \rangle} = \mathsf{output}\end{array}\right] \\
& \leq \Pr\left[\begin{array}{c}\text{There exist } \kappa/2 \text{ instances } j \\ \text{such that } y^{\langle j \rangle} \neq F_\lambda(p, x) \text{ and } z^{\langle j \rangle} = \mathsf{output}\end{array}\right] \\
& \leq \epsilon^{\frac{\kappa}{2}} \cdot \binom{\kappa}{\kappa/2} \\
& \leq \left(\frac{1}{16} + \mathsf{negl}(\lambda)\right)^{\frac{\kappa}{2}} \cdot 2^\kappa \\
& \leq \left(\frac{1}{4} + \mathsf{negl}'(\lambda)\right)^{\frac{\lambda}{2}} \\
& \leq \mathsf{negl}''(\lambda).
\end{aligned}
$$

Therefore, we obtain (6).

Complexity. Since we set $M = 16m$ and $\kappa = \lambda$, the communication complexity of Π_{CF} is $O(\kappa \cdot \mathrm{Comm}(\Pi')) = O(m^2 \lambda \cdot \mathrm{Comm}(\Pi))$. It can also be seen that $\text{s-Comp}(\Pi_{\mathrm{CF}}) = O(m^2 \lambda \cdot \text{s-Comp}(\Pi))$. Observe that the client-side computational complexity of Step 1 is $\kappa \cdot \text{c-Comp}(\Pi') = O(m^3 \lambda \cdot \text{c-Comp}(\Pi))$. The other steps can be done in time $O(\kappa \log |\mathcal{Y}_\lambda|) = O(\lambda \cdot \text{c-Comp}(\Pi))$ since it is possible to find the majority of a sequence $y^{\langle 1 \rangle}, \ldots, y^{\langle \kappa \rangle} \in \mathcal{Y}_\lambda$ in time $O(\kappa \log |\mathcal{Y}_\lambda|)$, e.g., by [13]. Thus we have that $\text{c-Comp}(\Pi_{\mathrm{CF}}) = O(m^3 \lambda \cdot \text{c-Comp}(\Pi))$.

H Proof of Proposition 8

Fix $H \in \binom{[m]}{h}$. We first show that $\Pr[f \leftarrow_{\$} \mathrm{Map}(m, k) : f(H) \neq [k]] < 1/3$. Indeed, if $f(H) \neq [k]$, there is $j \in [k]$ such that $f(s) \neq j$ for any $s \in H$. Since the total number of maps f such that $f(s) \neq j$ for any $s \in H$ is at most $k^{m-h}(k-1)^h$, we obtain that

$$
\Pr[f \leftarrow_{\$} \mathrm{Map}(m, k) : f(H) \neq [k]] \leq \frac{k^{m-h}(k-1)^h k}{k^m} \leq k\left(1 - \frac{1}{k}\right)^h.
$$

Since $1 - x \leq \exp(-x)$ and $k \leq h/(\gamma \ln h)$, this probability is further upper bounded by

$$
k \cdot \exp\left(-\frac{h}{k}\right) \leq \frac{h}{\gamma \ln h} \exp(-\gamma \ln h) \leq \frac{1}{h^{\gamma-1} \ln h} \leq \frac{1}{15^{\gamma-1} \ln 15} = \frac{1}{3}.
$$

Let X be a random variable over $\{0,1\}$ defined as $X = 1$ if and only if $f(H) = [k]$, where $f \leftarrow_{\$} \mathrm{Map}(m,k)$. Let $p = \Pr[X = 1]$. We have that $p \geq 2/3$. Let $X_1, \ldots, X_w$ be i.i.d. random variables over $\{0,1\}$ such that $\Pr[X_u = 1] = p$ for all u. Note that $p = \mathbb{E}\left[(1/w)\sum_{u\in[w]} X_u\right]$. From the Chernoff bound, we obtain that

$$
\begin{aligned}
\Pr\left[\sum_{u\in[w]} X_u \leq \frac{w}{2}\right] &= \Pr\left[\frac{1}{w}\sum_{u\in[w]} X_u \leq p - \left(p - \frac{1}{2}\right)\right] \\
&\leq \left(\left(\frac{p}{1/2}\right)^{1/2}\left(\frac{1-p}{1/2}\right)^{1/2}\right)^{w} \\
&= (4p(1-p))^{w/2} \\
&\leq \left(\frac{8}{9}\right)^{w/2} \\
&\leq \exp\left(-\frac{w}{18}\right).
\end{aligned}
$$

From the definition of X_u, we have that

$$
\Pr\left[f_1, \ldots, f_w \leftarrow_{\$} \mathrm{Map}(m,k) : |\{u \in [w] : f_u(H) = [k]\}| \leq \frac{w}{2}\right] \leq \exp\left(-\frac{w}{18}\right).
$$

It follows from the union bound that

$$
\begin{aligned}
&\Pr\left[f_1, \ldots, f_w \leftarrow_{\$} \mathrm{Map}(m,k) : \exists H \in \binom{[m]}{h}, |\{u \in [w] : f_u(H) = [k]\}| \leq \frac{w}{2}\right] \\
&\leq \binom{m}{h}\exp\left(-\frac{w}{18}\right) \\
&\leq 2^m \exp\left(-\frac{w}{18}\right) \\
&= \exp\left(m \ln 2 - \frac{w}{18}\right).
\end{aligned}
$$

We can also see that if $f_1, \ldots, f_w \leftarrow_{\$} \mathrm{Map}(m,k)$, the probability that there are $i, j \in [w]$ such that $f_i = f_j$ is at most

$$
\binom{w}{2}\frac{1}{k^m} < \frac{w^2}{2}\cdot\frac{1}{2^m} = \frac{w^2}{2^{m+1}}.
$$

Therefore, if $f_1, \ldots, f_w \leftarrow_{\$} \mathrm{Map}(m,k)$, the probability that the set $\mathcal{L} = \{f_1, \ldots, f_w\}$ is of size w and $|\{f \in \mathcal{L} : f(H) = [k]\}| > w/2$ for all $H \in \binom{[m]}{h}$ is at least

$$
1 - \exp\left(m \ln 2 - \frac{w}{18}\right) - \frac{w^2}{2^{m+1}}.
$$

If we set $w = 14m$, then the above value is

$$\begin{aligned}
&1 - \exp\left(-\left(\frac{7}{9} - \ln 2\right) m\right) - \frac{14^2 m^2}{2^{m+1}} \\
&\geq 1 - \exp\left(-\left(\frac{7}{9} - \ln 2\right) \cdot 15\right) - \frac{14^2 \cdot 15^2}{2^{16}} \\
&= 1 - 0.2809\cdots - 0.6729\cdots \\
&> 0
\end{aligned}$$

since $m \geq 15$ and $7/9 - \ln 2 > 0$. Therefore, an (m, h, k)-locally surjective map family of size $w = 14m$ indeed exists.

I Proof of Theorem 4

The protocol Π' is described in Fig. 5.

Correctness. Assume that correct answers $(\mathsf{ans}_i)_{i\in[m]}$ are inputted into Output'. At Step 3 in the first iteration (i.e., $L = 1$), for all $u \in [w]$ and $j \in [k]$, it holds that

$$\mathsf{ans}_i^{(u,1)} = \mathsf{Answer}(1^\lambda, p, \mathsf{que}_{f_u(i)}^{(u,1)}) = \mathsf{Answer}(1^\lambda, p, \mathsf{que}_j^{(u,1)}),\ \forall i \in G_{u,j},$$

and hence that $\alpha_j^{(u)} = \mathsf{Answer}(1^\lambda, p, \mathsf{que}_j^{(u,1)})$. It follows from the correctness of Π that $y^{(u)} = F_\lambda(p, x)$ for all $u \in [w]$. Hence Output' goes to Step 3(a)-i and outputs $y = F_\lambda(p, x)$.

Privacy. Let $\mathcal{A}$ be a stateful adversary against the privacy of Π'. For $b \in \{0, 1\}$, let $\mathsf{Priv}^b_{\Pi',\mathcal{A}}(\lambda)$ be a random variable defined in Definition 2. We show that $\mathsf{Adv}_{\Pi',\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Priv}^0_{\Pi',\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{Priv}^1_{\Pi',\mathcal{A}}(\lambda) = 0\right]\right|$ is negligible. To this end, we suppose that $I := \{(u, \ell) : u \in [w], \ell \in [t + 1]\}$ has a total order induced by the lexicographic order. We define a hybrid distribution $\mathsf{Hyb}^{(u,\ell)}_{\Pi',\mathcal{A}}(\lambda)$ for $(u, \ell) \in I$ as the output b' of $\mathcal{A}$ in the following experiment $\mathrm{HybEXPT}^{(u,\ell)}_{\Pi',\mathcal{A}}(\lambda)$:

$\mathrm{HybEXPT}^{(u,\ell)}_{\Pi',\mathcal{A}}(\lambda)$

1. $(x_0, x_1, p, T) \leftarrow \mathcal{A}(1^\lambda)$.
2. For each $(u', \ell') < (u, \ell)$, compute
$$(\mathsf{que}_1^{(u',\ell')}, \ldots, \mathsf{que}_k^{(u',\ell')}; \mathsf{st}^{(u',\ell')}) \leftarrow \mathsf{Query}(1^\lambda, x_1).$$
3. For each $(u', \ell') \geq (u, \ell)$, compute
$$(\mathsf{que}_1^{(u',\ell')}, \ldots, \mathsf{que}_k^{(u',\ell')}; \mathsf{st}^{(u',\ell')}) \leftarrow \mathsf{Query}(1^\lambda, x_0).$$
4. Give $(\mathsf{que}_i)_{i\in T}$ to $\mathcal{A}$, where $\mathsf{que}_i = \{\mathsf{que}^{(u',\ell')}_{f_{u'}(i)} : (u', \ell') \in I\}$.
5. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

Notations.

– Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be a 1-round passively t-secure k-server protocol.
– Let $\mathcal{L} = \{f_1, \ldots, f_w\}$ be an (m, h, k)-locally surjective map family of size w, where $h = m - 2t$.
– For each $u \in [w]$ and $j \in [k]$, let $G_{u,j} = \{i \in [m] : f_u(i) = j\}$.

$\mathsf{Query}'(1^\lambda, x)$**.**

1. For each $u \in [w]$ and $\ell \in [t+1]$, compute

$$(\mathsf{que}_1^{(u,\ell)}, \ldots, \mathsf{que}_k^{(u,\ell)}; \mathsf{st}^{(u,\ell)}) \leftarrow \mathsf{Query}(1^\lambda, x).$$

2. Output $((\mathsf{que}_i)_{i \in [m]}, \mathsf{st})$, where

$$\mathsf{que}_i = \{\mathsf{que}_{f_u(i)}^{(u,\ell)} : u \in [w], \ell \in [t+1]\},$$
$$\mathsf{st} = \{\mathsf{st}^{(u,\ell)} : u \in [w], \ell \in [t+1]\}.$$

$\mathsf{Answer}'(1^\lambda, p, \mathsf{que}_i)$**.**

1. Parse $\mathsf{que}_i = \{\mathsf{que}_{f_u(i)}^{(u,\ell)} : u \in [w], \ell \in [t+1]\}$.
2. For each $u \in [w]$ and $\ell \in [t+1]$, compute

$$\mathsf{ans}_i^{(u,\ell)} = \mathsf{Answer}(1^\lambda, p, \mathsf{que}_{f_u(i)}^{(u,\ell)}).$$

3. Output $\mathsf{ans}_i = \{\mathsf{ans}_i^{(u,\ell)} : u \in [w], \ell \in [t+1]\}$.

$\mathsf{Output}'(1^\lambda, \mathsf{st}, (\mathsf{ans}_i)_{i \in [m]})$**.**

1. Parse

$$\mathsf{ans}_i = \{\mathsf{ans}_i^{(u,\ell)} : u \in [w], \ell \in [t+1]\},\ \mathsf{st} = \{\mathsf{st}^{(u,\ell)} : u \in [w], \ell \in [t+1]\}.$$

2. Set $L \leftarrow 1$ and $S \leftarrow [m]$.
3. If $S = \emptyset$ or $L > t+1$, output a default value $y_0 \in Y_\lambda$. Otherwise, do the following:
   (a) If for all $u \in [w]$ and $j \in [k]$, there exists $\alpha_j^{(u)}$ such that

$$\{\mathsf{ans}_i^{(u,L)} : i \in G_{u,j} \cap S\} = \{\alpha_j^{(u)}\}, \tag{7}$$

   then compute $y^{(u)} = \mathsf{Output}(1^\lambda, \mathsf{st}^{(u,L)}, \alpha_1^{(u)}, \ldots, \alpha_k^{(u)})$.
      i. If there exists $y \in Y_\lambda$ such that $|\{u \in [w] : y^{(u)} = y\}| > w/2$, then output y.
      ii. Otherwise, output a default value $y_0 \in Y_\lambda$.
   (b) Otherwise, let (u, j) be such that the condition (7) does not hold, and (i_1, i_2) be a pair such that $i_1, i_2 \in S \cap G_{u,j}$ and $\mathsf{ans}_{i_1}^{(u,L)} \neq \mathsf{ans}_{i_2}^{(u,L)}$. Set $L \leftarrow L + 1$ and $S \leftarrow S \setminus \{i_1, i_2\}$. Repeat Step 3.

Fig. 5. An actively secure protocol Π' based on a locally surjective map family

Note that $\mathsf{Hyb}^{(1,1)}_{\Pi',\mathcal{A}}(\lambda) = \mathsf{Priv}^0_{\Pi',\mathcal{A}}(\lambda)$. We define $\mathsf{Hyb}^{(w,t+2)}_{\Pi',\mathcal{A}}(\lambda) = \mathsf{Priv}^1_{\Pi',\mathcal{A}}(\lambda)$ and

$$\mathsf{Adv}^{(u,\ell)}_{\Pi',\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Hyb}^{(u,\ell)}_{\Pi',\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{Hyb}^{\mathsf{next}(u,\ell)}_{\Pi',\mathcal{A}}(\lambda) = 0\right]\right|,$$

where $\mathsf{next}(u,\ell)$ is the next element of (u,ℓ) with respect to the total order on I. Then we have that

$$\mathsf{Adv}_{\Pi',\mathcal{A}}(\lambda) \leq \sum_{(u,\ell)\in I} \mathsf{Adv}^{(u,\ell)}_{\Pi',\mathcal{A}}(\lambda).$$

We show that for any (u,ℓ), there exists an adversary $\mathcal{B}$ against Π such that $\mathsf{Adv}^{(u,\ell)}_{\Pi',\mathcal{A}}(\lambda) = \mathsf{Adv}_{\Pi,\mathcal{B}}(\lambda)$ for all λ. If this can be shown, since $|I| = w(t+1)$ is polynomial in λ, we conclude that

$$\mathsf{Adv}_{\Pi',\mathcal{A}}(\lambda) \leq w(t+1)\cdot \sup_{\mathcal{B}} \mathsf{Adv}_{\Pi,\mathcal{B}}(\lambda) = \mathsf{negl}(\lambda)\,.$$

We construct $\mathcal{B}$ as follows.

$\mathcal{B}$

Setup. On input 1^λ,
1. Run $(x_0, x_1, p, T) \leftarrow \mathcal{A}(1^\lambda)$.
2. Output (x_0, x_1, p, T'), where $T' = f_u(T)$. Note that $|T'| \leq |T| \leq t$.

Protocol. Receiving queries $(\mathsf{que}'_j)_{j\in T'}$ during the execution of Π, $\mathcal{B}$ does the following:
1. For each $(u',\ell') < (u,\ell)$, compute
$$(\mathsf{que}^{(u',\ell')}_1, \ldots, \mathsf{que}^{(u',\ell')}_k; \mathsf{st}^{(u',\ell')}) \leftarrow \mathsf{Query}(1^\lambda, x_1).$$
2. For each $(u',\ell') > (u,\ell)$, compute
$$(\mathsf{que}^{(u',\ell')}_1, \ldots, \mathsf{que}^{(u',\ell')}_k; \mathsf{st}^{(u',\ell')}) \leftarrow \mathsf{Query}(1^\lambda, x_0).$$
3. Set $\mathsf{que}^{(u,\ell)}_{f_u(i)} = \mathsf{que}'_{f_u(i)}$ for $i \in T$.
4. Give $(\mathsf{que}_i)_{i\in T}$ to $\mathcal{A}$, where $\mathsf{que}_i = \{\mathsf{que}^{(u',\ell')}_{f_{u'}(i)} : (u',\ell') \in I\}$.
5. Output a guess b' outputted by $\mathcal{A}$.

In the experiment defining $\mathsf{Priv}^0_{\Pi,\mathcal{B}}(\lambda)$, if $(u',\ell') < (u,\ell)$, the (u',ℓ')-th component of each que_i is a query of Π on input x_1 and otherwise, it is a query on input x_0. Therefore, we have that $\mathsf{Priv}^0_{\Pi,\mathcal{B}}(\lambda) = \mathsf{Hyb}^{(u,\ell)}_{\Pi',\mathcal{A}}(\lambda)$. Similarly, we have that $\mathsf{Priv}^1_{\Pi,\mathcal{B}}(\lambda) = \mathsf{Hyb}^{\mathsf{next}(u,\ell)}_{\Pi',\mathcal{A}}(\lambda)$. We conclude that $\mathsf{Adv}^{(u,\ell)}_{\Pi',\mathcal{A}}(\lambda) = \mathsf{Adv}_{\Pi,\mathcal{B}}(\lambda)$.

Byzantine-robustness. Let $\mathcal{A}$ be a stateful adversary against the Byzantine-robustness of Π'. Consider the experiment defining $\mathsf{BR}_{\Pi',\mathcal{A}}(\lambda)$ (see Definition 2). Assume that in the setup, $\mathcal{A}(1^\lambda)$ outputs $x \in X_\lambda$, $p \in P_\lambda$ and a subset $T \subseteq [m]$ of size at most t. Let $(\mathsf{que}_i)_{i\in[m]}$ and st be an output of $\mathsf{Query}'(x)$, and ans_i be the correct output of $\mathsf{Answer}'(p, \mathsf{que}_i)$ for $i \in [m]$. Suppose that during the protocol execution, $\mathcal{A}$ outputs possibly incorrect answers $(\widetilde{\mathsf{ans}}_i)_{i\in T}$, where $\widetilde{\mathsf{ans}}_i = \{\widetilde{\mathsf{ans}}_i^{(u,\ell)} : u \in [w], \ell \in [t+1]\}$.

First, assume that there exists $\ell \in [t+1]$ such that $\widetilde{\mathsf{ans}}_i^{(u,\ell)} = \mathsf{ans}_i^{(u,\ell)}$ for all $i \in T$ and all $u \in [w]$ (i.e., all malicious servers behave honestly in the (u,ℓ)-th instance of Π). It then follows from the correctness of Π that Output' proceeds to Step 3(a)-i in the ℓ-th iteration, and computes the correct value $y = F_\lambda(p, x)$.

Next, assume that for all $\ell \in [t+1]$, there exist $i \in T$ and $u \in [w]$ such that $\widetilde{\mathsf{ans}}_i^{(u,\ell)} \neq \mathsf{ans}_i^{(u,\ell)}$ (i.e., at least one malicious server submits an incorrect answer). Let $H = [m] \setminus T$. For $L = 0, 1, \ldots, t+1$, define the number t_L as $t_L = |S \cap T|$, where S is the set after iterating Step 3 of Output' L times. t_L denotes the number of malicious servers remaining in S after L iterations. Observe that $t_0 = t$.

Consider the L-th iteration of Step 3 of Output' for $1 \leq L \leq t+1$. We will show that either of the following two events occurs: (1) Output' goes to Step 3(a)-i and outputs the correct result $y = F_\lambda(p, x)$; (2) Output' goes to Step 3(b) and removes at least one element $i \in T$ from S. In other words, Output' never goes to Step 3(a)-ii.

To show the above claim by induction on L, we consider the first iteration, i.e., $L = 1$. We have that $T = [m]$ and $|H| \geq m - 2t = h$. Let $\mathcal{L}_{\mathrm{Bad}}$ be the set of maps $f_u \in \mathcal{L}$ such that at least one malicious server submits an incorrect answer in the instance corresponding to f_u, i.e.,

$$\mathcal{L}_{\mathrm{Bad}} = \{f_u \in \mathcal{L} : \exists i \in [m], \widetilde{\mathsf{ans}}_i^{(u,1)} \neq \mathsf{ans}_i^{(u,1)}\}.$$

If $|\mathcal{L}_{\mathrm{Bad}}| \geq w/2$, there exists $f_u \in \mathcal{L}_{\mathrm{Bad}}$ such that $f_u(H) = [k]$ due to the property of the locally surjective map family $\mathcal{L}$. Let $i \in [m]$ be such that $\widetilde{\mathsf{ans}}_i^{(u,1)} \neq \mathsf{ans}_i^{(u,1)}$. If $i \in G_{u,j}$ (i.e., $f_u(i) = j$), $\widetilde{\mathsf{ans}}_i^{(u,1)}$ conflicts with $\mathsf{ans}_{i'}^{(u,1)}$ for any $i' \in H \cap G_{u,j}$ since $\mathsf{ans}_i^{(u,1)} = \mathsf{ans}_{i'}^{(u,1)} = \mathsf{Answer}(p, \mathsf{que}_j^{(u,1)})$. Note that $H \cap G_{u,j} \neq \emptyset$ since $f_u(H) = [k] \ni j$. Then, Output' goes to Step 3(b). Since for any $i_1, i_2 \in H$ with $f_u(i_1) = f_u(i_2)$, the answers do not conflict, i.e., $\mathsf{ans}_{i_1}^{(u,1)} = \mathsf{ans}_{i_2}^{(u,1)}$, a pair removed from S necessarily includes i and hence the second case (2) occurs. In particular, we obtain that $t_1 \leq t - 1$.

On the other hand, suppose that $|\mathcal{L}_{\mathrm{Bad}}| < w/2$ and Output' goes to Step 3(a). Note that for any $u \in [w]$ such that $f_u \notin \mathcal{L}_{\mathrm{Bad}}$,

$$y^{(u)} = \mathsf{Output}((\mathsf{Answer}(p, \mathsf{que}_j^{(u,1)}))_{j\in[k]}; \mathsf{st}^{(u,1)}) = F_\lambda(p, x).$$

It thus holds that $|\{u \in [w] : y^{(u)} = F_\lambda(p, x)\}| > w/2$ and Output' outputs $y = F_\lambda(p, x)$, i.e., the first case (1) occurs.

Let L be such that $1 \leq L \leq t$ and consider the situation where the second case (2) continues to occur and Output' proceeds to the $(L+1)$-th iteration. We have that $|S| = m - 2L$ just before proceeding to the $(L+1)$-th iteration. We also have that $h' := |H \cap S| \geq (m - t) - L \geq m - 2t = h$. The property of the locally surjective map family $\mathcal{L}$ implies that

$$|\{f \in \mathcal{F} : f(H \cap S) = [k]\}| > \frac{w}{2}.$$

Similarly to $\mathcal{L}_{\mathrm{Bad}}$, let $\mathcal{L}'_{\mathrm{Bad}}$ denote the set of maps $f_u \in \mathcal{L}$ such that at least one remaining malicious server submits an incorrect answer in the instance corresponding to f_u, i.e.,

$$\mathcal{L}'_{\mathrm{Bad}} = \{f_u \in \mathcal{L} : \exists i \in S, \widetilde{\mathsf{ans}}_i^{(u,L)} \neq \mathsf{ans}_i^{(u,L)}\}.$$

If $|\mathcal{L}'_{\mathrm{Bad}}| \geq w/2$, there exists $f_u \in \mathcal{L}'_{\mathrm{Bad}}$ such that $f_u(H \cap S) = [k]$. Let $i \in S$ be such that $\widetilde{\mathsf{ans}}_i^{(u,L)} \neq \mathsf{ans}_i^{(u,L)}$ for that $u \in [w]$. As in the first iteration, if $i \in G_{u,j}$ (i.e., $f_u(i) = j$), $\widetilde{\mathsf{ans}}_i^{(u,L)}$ conflicts with $\mathsf{ans}_{i'}^{(u,L)}$ for any $i' \in H \cap G_{u,j}$ since $\mathsf{ans}_i^{(u,L)} = \mathsf{ans}_{i'}^{(u,L)} = \mathsf{Answer}(p, \mathsf{que}_j^{(u,L)})$. Then, Output' goes to Step 3(b). Since the answers from any $i_1, i_2 \in H$ with $f_u(i_1) = f_u(i_2)$ do not conflict, a pair removed from S necessarily includes i and hence the second case (2) occurs. In this case, it holds that $t_{L+1} \leq t_L - 1$. On the other hand, suppose that $|\mathcal{L}'_{\mathrm{Bad}}| < w/2$ and Output' goes to Step 3(a). Note that for any $u \in [w]$ such that $f_u \notin \mathcal{L}_{\mathrm{Bad}}$,

$$y^{(u)} = \mathsf{Output}((\mathsf{Answer}(p, \mathsf{que}_j^{(u,L)}))_{j \in [k]}; \mathsf{st}^{(u,L)}) = F_\lambda(p, x).$$

It thus holds that $|\{u \in [w] : y^{(u)} = F_\lambda(p, x)\}| > w/2$ and Output' outputs $y = F_\lambda(p, x)$, i.e., the first case (1) occurs.

Therefore, after the L-th iteration, Output' outputs $y = F_\lambda(p, x)$ and terminates; or it holds that $t_L \leq t - L$ and the algorithm proceeds to the next iteration. In particular, there exists L with $L \leq t$ such that $t_L = 0$, or Output' outputs $F_\lambda(p, x)$ in the L'-th iteration for some $L' \leq L$. In the former case, there is no remaining malicious server, i.e., $S \cap T = \emptyset$, in the $(L+1)$-th iteration. Then Output' goes to Step 3(a)-i and outputs the correct value $F_\lambda(p, x)$ since $\mathsf{ans}_i^{(u,L+1)}$ is a correct answer for every $i \in S$.

Complexity. Observe that Query' outputs $O(tw)$ queries $\{\mathsf{que}_j^{(u,\ell)} : u \in [w], \ell \in [t{+}1]\}$ for $\varPi$ to all $i \in G_{u,j}$. Since $|G_{u,j}| \leq m$, we have that $\mathrm{Comm}(\varPi') = O(twm \cdot \mathrm{Comm}(\varPi))$. As for the computational complexity, Query' generates $O(tw)$ queries of $\varPi$ and to make $O(m)$ copies of each of them. Output' checks consistency among $O(wm)$ answers in each of $t+1$ iterations and finally runs the algorithm Output. We note that it is possible to find the majority of a sequence $x^{(1)}, \ldots, x^{(w)} \in \mathcal{X}$ in time $O(w \log |\mathcal{X}|)$, e.g., by [13]. Thus, we have that c-Comp$(\varPi') = O(twm \cdot$ c-Comp$(\varPi))$. It can be easily seen that s-Comp$(\varPi') = O(tw \cdot$ s-Comp$(\varPi))$.

J Proof of Corollary 7

If $m \geq 2t3^t + 2t$, then $m - 2t \geq 2t3^t \geq 6$ and

$$\frac{m - 2t}{\gamma \ln(m - 2t)} \geq \frac{2t}{\gamma \ln(2t3^t)} \cdot 3^t.$$

Since

$$2t - \gamma \ln(2t) \geq 2t - 1.04 \ln t - 1.04 \ln 2 \geq 1.2t \geq (\gamma \ln 3)t,$$

we have that $2t \geq \gamma \ln(2t3^t)$ and that

$$\frac{m-2t}{\gamma \ln(m-2t)} \geq 3^t.$$

We can then apply Theorem 5 to the passively t-secure 3^t-server protocol in Proposition 1 and obtain Corollary 7.

Similarly, we can see that if $m \geq t2^{t+1} + 2t$, then it holds that

$$\frac{m-2t}{\gamma \ln(m-2t)} \geq 2^t.$$

We can then apply Theorem 5 to the protocol in Proposition 1 and obtain Corollary 8.