The Feasibility of Outsourced Database Search in the Plain Model

Carmit Hazay* Hila Zarosim†

Abstract

The problem of securely outsourcing computation to an *untrusted* server gained momentum with the recent penetration of *cloud computing* services. The ultimate goal in this setting is to design efficient protocols that minimize the computational overhead of the clients and instead rely on the extended resources of the server. In this paper, we focus on the *outsourced database search* problem which is highly motivated in the context of delegatable computing since it offers storage alternatives for massive databases, that may contain confidential data. This functionality is described in two phases: (1) setup phase and (2) query phase. The main goal is to minimize the parties workload in the query phase so that it is proportional to the query size and its corresponding response.

Our starting point is the semi-honest protocol from [FHV13] (ICALP 2013) that offers a simulation based secure protocol for outsourced pattern matching in the random oracle setting with optimal workload. In this work we study whether the random oracle is *necessary* for protocols with *minimal interaction* that meet the optimal communication/computation bounds in the query phase. We answer this question positively and demonstrate a lower bound on the communication or the computational overhead in this phase. We further abstract the security properties of the underlying cryptographic primitive that enables to obtain private outsourced database search with minimal interaction. For a large class of search functionalities the communication complexity of our protocol meets the above lower bound.

Keywords: Outsourced Computation, Database Search Functionalities, Lower Bound, Communication and Computational Complexities, Minimal Interaction

*Faculty of Engineering, Bar-Ilan University, Israel. Email: carmit.hazay@biu.ac.il

†Department of Computer Science, Bar-Ilan University, Israel. Email: zarosih@cs.biu.ac.il.

1 Introduction

Background on outsourced secure computation. The problem of securely outsourcing computation to an *untrusted* server gained momentum with the recent penetration of *cloud computing* services, where clients can lease computing services on demand rather than maintaining their own infrastructure. The ultimate goal in this setting is to design efficient protocols that minimize the computational overhead of the clients and instead rely on the extended resources of the server. Of course, the amount of work invested by the client in order to verify the correctness of the computation needs to be *substantially* smaller than running the computation by itself. Another ambitious challenge of delegatable computation is to design protocols that minimize the *communication* between the cloud and the client. This becomes of particular importance with the proliferation of smartphone technology and mobile broadband internet connections, as for mobile devices communication and data connectivity is often the more severe bottleneck.

The study of delagatable computation was initiated with the study of a restricted scenario where a *single client* outsources its computation to an external server. Two main approaches are examined in this context. In the first setting there is only one phase of interaction between the client and the server such that the overall amount of work performed by the client is smaller than performing the computation on its own. Correctness in this setting is achieved by succinct zero-knowledge proofs [GLR11, BCCT12, DFH12] with the aim of minimizing the number of rounds between the client and the server. In the *amortized setting* [GGP10, AIK10] the computational complexity of the client is analyzed in an amortized sense. Namely, the client can perform some expensive preprocessing (also known as the offline phase). After this phase is completed, it is required to run very efficient computations in the online phase.

Recent results also study an extended setting with *multiple r clients* that mutually distrust each other and wish to securely outsource a joint computation on their inputs with reduced costs [KMR11, KMR12, LATV12, AJLA$^+$12, CKKC13]. In particular, it is required that the communication between the clients and the server, as well as the communication between the clients, will be sufficiently smaller than running a secure computation in the standard setting. This more complex setting is strictly harder than the single client setting since one must handle potential corruptions of any (proper) subset of the clients, that might collude with the server. It is worth noting that in case only correctness is required then security in the multi clients setting is reduced to security in the single client setting. This is due to the fact that we can consider a protocol where $r-1$ clients send their inputs to the rth client, that communicates with the server using all inputs. It then forwards the server's proof to the other clients who can verify its correctness. Generally speaking, outsourced secure computation in the presence of collusion between any t clients and the server implies secure computation in the standard setting with $r-t+1$ parties. Thus, the problem of delegatable computation with multiple clients focuses on achieving privacy (with or without imposing correctness).

Modeling outsourced database search. To move towards more practical schemes, recent results give up on outsourcing arbitrary computation to the cloud, and instead focus on particularly efficient constructions for specific important functionalities. This approach has the potential to achieve improved complexities by exploiting the structure of the particular problem that is intended to solve. Some recent works have considered this question and proposed schemes for polynomial evaluation and keywords search [BGV11], set operations [PTT11], linear algebra [Moh11] and pattern matching [FHV13]. In this paper, we focus on the database search problem in the cloud. This problem, denoted by *outsourced database search*, is highly motivated in the context of delegatable computing since it offers storage alternatives for massive databases that may contain confidential data (e.g., health related data about patient history).

We consider a reactive database search functionality where one client has a database, and another set of clients search the database using a sequence of queries. To simplify the presentation we denote the former client by the sender and the other set of clients by the receiver (for simplicity, we focus on a single receiver

asking for multiple queries). The input of the sender is a database of size n.[1] The input queries of the receiver $\{q_i\}_{i\in[t]}$ are picked from a predefined set Q_n where Q_n is a set of queries that correspond to a database of size n. This functionality can be described in two phases. In the *setup phase* the sender uploads a function of its database to the server. This phase is run only once, where the sender's state after this phase is independent of n. Next, in the *query phase* the receiver picks a search query and obtains from the server the answer to this query (denoted by *record*). To restrict the number of queries, the sender must approve each query by providing a trapdoor that depends on the content of the query (however, recall that this cannot be solved in the trivial way since the sender maintains a small amount of state now and in particular does not hold the database any more). This formalization captures a large class of search functionalities. Two known examples are oblivious transfer (OT) with adaptive queries [NP99, GH11] and secure pattern matching [HT10]. In the former functionality the record size is bounded to a single element from the database, whereas in the latter functionality it is unbounded and might be $O(n)$.

Security is formalized using the ideal/real paradigm, considering the server as a separate entity that does not contribute any input to the computation. As in the standard static modeling, a corrupted party is either passively or actively controlled by an adversarial entity. In the passive case (a.k.a. the semi-honest case) a corrupted party follows the protocol's instructions and tries to gain additional information about the honest parties' inputs from its view; in the active case (a.k.a. the malicious case) a corrupted party is allowed to follow an arbitrary polynomial-time strategy. This modeling also captures a collusion between the server and one of the clients.[2] In order to take some advantage from this modeling, we would like the setup phase to require $O(n)$ workload, yet the overall cost of issuing a query should only *grow linearly with the size of the query's response* (which is as optimal as one can obtain). For some search functionalities, without a fixed bound on the database records, this optimization comes at the price of revealing some leakage about the database.

We are further interested in studying protocols with *minimal interaction*. That is, in the setup phase we require a single message sent from the sender to the server, whereas in the query phase we require that the sender and receiver exchange only two messages (one in each direction), and finally, one message in each direction between the receiver and the server (see Figure 1).[3] In this paper we focus on semi-honest security and study the *feasibility* of the outsourced database search functionality with *minimal interaction and using minimal resources* of communication and computation.

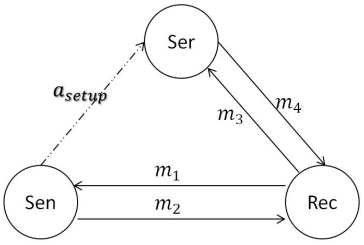

Figure 1: A graphical description of our modeling.

[1] We remark that the internal structure of the database is not important for our lower bounds proofs, whereas we require a concrete structured database in our feasibility result; see below for further information.

[2] As in the two-party setting, a collusion between the two clients is not interesting.

[3] We prove that if the order of communication between the receiver and the sender/server is swapped then our lower bounds follow more easily. We further note that our lower bounds are not restricted to a minimal interaction between the server and the receiver; we only impose this restriction in our construction.

The [FHV13] construction. This paper presents the first concrete protocols for the pattern matching problem in the cloud, where a text T that is outsourced to the cloud by a sender. In the query phase, the receiver learns the positions at which a pattern of length m matches the text (and nothing beyond that). This is called the outsourced pattern matching problem. These constructions offer simulation-based security in the presence of semi-honest and malicious adversaries and limit the communication in the query phase to $O(m)$ bits plus the number of occurrences – which is optimal (where the semi-honest protocol is with minimal interaction). Faust et al. use novel ideas based on a reduction to the subset sum problem, which do not rely on the hardness of the problem, but rather require instances that are solvable in polynomial-time.

Specifically, assume that in the setup phase the text is encoded by a random vector where each substring of length m is replaced by a random element. Moreover, in the query phase the sender hands the receiver the sum of elements that are in positions for which the corresponding substrings match the query, such that a solution to the subset sum instance reveals the positions in which the pattern appears. In order to avoid a blowup of the sender's state, the random vector is generated so that the sum of each set of positions that match a substring p equals $\mathrm{f}_k(p)$ (where f is a PRF). In addition, in order to keep the communication in the setup phase small, Faust et al. rely heavily on the programmability property of the random oracle, and use it to equivocate a fake text. In this work we study whether the random oracle is essential for achieving optimal overhead in the query phase. Namely,

Does there exist a private protocol with minimal interaction for the outsourced search functionality in the plain model, that meets the optimal communication/computation bounds in the query phase?

We prove that the answer for this question is negative. Namely, there exists a large class of search functionalities that *cannot* be realized privately with optimal resources in the query phase.

1.1 Our Results

Infeasibility of outsourced database search in the plain model. We prove that using the power of the random oracle is essential in order to reduce the resources of the receiver within protocols with minimal interaction, *even* if the sender's state in $o(n)$ and the number of rounds between the server and the receiver is arbitrary. This result has the consequence that for certain search functionalities (e.g, pattern matching and all its variants, and the indexing problem), the communication complexity or the number of steps made by the receiver must be as large as the size of the database. We examine both non-private and private channels scenarios (where in the latter setting corrupted parties do not see the communication between the honest parties), and prove that our lower bound holds in both settings. More formally, let $\mathcal{ANS}_{n,q}$ denote the set of *all potential responses* for the query q when ranging over all databases T of size n, and let $H_{n,Q} = \max_{q \in Q_n} \log |\mathcal{ANS}_{n,q}|$ (intuitively, $H_{n,Q}$ is the logarithm of the number of potential query responses when ranging over all databases of size n and all queries in Q_n; see Definition 3.3). We prove,

Theorem 1.1 *(informal) For any protocol with minimal interaction that securely implements the outsourced database search functionality in the presence of semi-honest adversaries, one of the following holds:*

1. *The communication complexity in the query phase is $O(H_{n,Q})$.*
2. *The number of random bits used by the receiver is $O(H_{n,Q})$.*

Our proof follows a similar intuition of the proof from [Nie02] when showing the impossibility of constructing non-interactive non-committing encryption schemes without a random oracle. Nevertheless, formalization is more challenging since our protocols are interactive and moreover, the number of involved parties is three. One consequence that we need to take into account is the order of rounds of which the receiver interacts with the other parties. This is because when we consider security in the private channels

setting we need to focus on a scenario where the server and the receiver collude. In this case, the view of the adversary contains both the randomness of the receiver and the server, as well as the message from the sender to the receiver. Thus, we must distinguish between the randomness of the receiver and that of the server and rely on the fact that the random tape of the server is uniformly independent of the receiver's view. Specifically, we need to show that when we fix a partial view of the receiver, then for almost all random tapes of the server, the receiver outputs the correct value. Note that if the receiver communicates with the server first then this independence no longer holds since the communication between the receiver and the sender may depend on the random tape of the server. On the other hand, if the receiver communicates with the sender first then independence follows, as semi-honest adversaries cannot pick their randomness arbitrarily. This subtlety is in contrast to the proof for non-committing encryption that allows to rely on the correctness of the non-interactive decryption algorithm of the underlying encryption scheme.

Intuitively, we consider the two possible orders of rounds in the query phase. If the receiver *first communicates with the server* and then with the sender, we show that the communication complexity of the protocol must be large because at the time the receiver communicates with the server, the server does not know anything about the database. It therefore does not know which information to send back, and essentially must send as much information as the maximal amount of information sent within any response to query q (when ranging over all databases of size n). If the receiver *first communicates with the sender* and then with the server, we consider the case where the receiver and the server are corrupted. Then in the simulation of the setup phase, the simulator must commit to a setup message independently of the sender's database. This message is fixed and cannot be later changed. Next in the query phase, the receiver makes a query q and learns the answer for this query. The simulator then has to simulate a view for the corrupted parties, so that it yields the correct output of the receiver. We show that this means that for every possible answer there must exist a view $(r_{\mathtt{Rec}}, \mathsf{m}_2)$ for the receiver (where $r_{\mathtt{Rec}}$ is the random tape of the receiver and m_2 the message form the sender to the receiver) such that with a high probability (over the random coins of the server), the receiver outputs the correct query response. This implies that the number of views $(r_{\mathtt{Rec}}, \mathsf{m}_2)$ must be proportional to the number of potential query responses when ranging over all databases (and hence the length of $(r_{\mathtt{Rec}}, \mathsf{m}_2)$ is linear in $H_{n,Q}$), which can be as large as the size of the database for certain search functionalities *even when the receiver's output size is small*. Interestingly, as shown in [FHV13], the random oracle allows for communication complexity and randomness that are proportional to the size of the query's response, in the non-private channel.

It is important to note that our lower bounds hold for *any* protocol with minimal interaction. Therefore, we can always focus on a protocol that makes use of a minimal number of random coins. Saying differently, our lower bounds consider the *effective* number of bits used by the receiver and even cover scenarios where the receiver's random tape is very large, for which the receiver ignores some portion of it. The reason for this is that for every such protocol, we can consider an equivalent protocol where the receiver's random tape does not contain any unused bits and apply our lower bounds to the new protocol. This further implies a lower bound on the number of steps of the receiver since these random bits are incorporated in the computation of the receiver. Moreover, our lower bounds hold *even if the receiver maintains no privacy* since they follow from the non-committing property that we require in the simulation.[4] Also, any attempt to replace the uniform randomness of the receiver by an output of a pseudorandom generator (PRG) in order to strengthen our lower bounds fails since it requires finding a preimage relative to the PRG; see more details in Section 4.

Finally, we note that our lower bounds also apply in the two-party setting for reactive search functionalities (with a preprocessing phase), which implies the infeasibility of private reactive pattern matching with optimal query response and minimal interaction in the plain model. This is in contrast to the non-private setting, where suffix trees [Wei73] are useful to store the text in a way that allows fast string operations. In

[4]We note that when privacy is not considered we prove that there exists a query for which our lower bounds hold. For private protocols this implies that these lower bounds hold for all queries or else some information about the query leaks.

particular, it illustrates that private pattern matching cannot be optimized in the preprocessing setting.

Efficient outsourced database search in the plain model. We further construct a private and efficient outsourced database search with minimal interaction and abstract the security properties of the underlying cryptographic primitive that enables to obtain this. Our framework implies that the sender's state in the query phase is independent of the database size. In addition, the communication complexity and the receiver's workload in the query phase depend only on its input and (worst case) output sizes. For a large class of search functionalities this communication complexity meets our lower bounds.

Our formulation uses an encryption scheme with a master secret key that produces multiple secret subkeys, where the goal is to associate a query with such a subkey that encrypts its record. This formalization is in the spirit of identity based encryption schemes (IBE) and functional encryption, where secret subkeys are associated with some functionality. Nevertheless, in our setting, queries (that can be interpreted as IBE identities) must be kept privately, which implies that the sender produces the secret keys obliviously. In addition, the sender generates both the database (plaintext) and the trapdoors (secret keys), which is different than the asymmetric scenario captured by IBE where one party holds the identity secret key whereas the other party holds the plaintext. This implies that our primitive can be designed based on weaker symmetric building blocks, such as PRFs, in contrast to more "expensive" primitives such as IBE.

A prime difficulty in proving security in this setting is due to the fact that the simulator must equivocate the fake database when simulating the view of a corrupted receiver (as it is committed to the database before observing any query). In order to capture this property we extend our notion of encryption and require secret keys equivocation, for secret keys that were generated in a fake (simulated) mode. This additional property has a flavour of adaptive security (as in the case of non-committing encryptions), but is strictly weaker since it does not enable security in the presence of adaptive corruptions of neither the receiver nor the sender. Nevertheless, this notion is still meaningful in our context. Combining these two properties of multiple secret keys and secret keys equivocation implies an encryption scheme for outsourced database search functionalities with communication complexity that depends on the largest query response size.

Note that the master secret key encodes all the information about the query secret keys succinctly, and thus the sender can avoid storing a key per query. It is also helpful in minimizing the communication between the sender and the receiver in the query phase, since it implies a mechanism that enables secret key transfer with communication that depends on the query size, but independent of the entire size of $|Q_n|$. Finally, we note that the design of OT with adaptive queries from [FIPR05], when executed in the outsourced setting, is captured by our abstraction. To conclude, we prove

Theorem 1.2 *(informal) Assume the existence of a semi-honest OT. Then, there exists a protocol that securely implements the outsourced database search functionality in the presence of semi-honest adversaries with minimal interaction and communication complexity that is proportional to the largest query response.*

Our construction is also secure in the presence of collusion between the receiver and the server and leaks the search pattern to the server. As a sanity check, we note that fully homomorphic encryption is not useful for our setting since the receiver must interact with the sender twice: once to obtain the public key and once to engage in a secure two-party protocol in order to decrypt the ciphertext returned from the server.

Symmetric searchable encryption (SSE). A related line of work regarding symmetric searchable encryption [CGKO11, KPR12, KP13, JJK$^+$13] allows a party to privately outsource its data to another party while maintaining the ability to search it, where the main focus of this primitive is typically on the search time. We note that SSE that supports adaptive queries, simulation-based security and query privacy can be applied in a setting with a single sender and multiple receivers, by letting the receivers learn their trapdoors using a secure two-party computation protocol that is engaged with the sender.

Importantly, our infeasibility result also applies to SSE with adaptive queries, where the user adaptively asks its queries *after* the preprocess phase is completed. Namely, we prove that there exists a class of search problems (e.g., the indexing problem), where there exists no SSE with adaptive queries, minimal interaction and communication that is proportional to the query's response. Finally, our framework regarding the feasibility of database search with no random oracles is also useful for SSE and provides an abstraction of the security properties of this primitive with minimal interaction, as well as a simplified instantiation based on PRFs.

2 Preliminaries

Basic notations. We denote the security parameter by κ and by $\mathcal{U}_n$ the uniform distribution over strings of length n. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$. We use the abbreviation PPT to denote probabilistic polynomial-time and the notation $[n]$ to denote the set of integers $\{1, \ldots, n\}$.

We specify the definition of computational indistinguishability.

Definition 2.1 (Computational indistinguishability by circuits) *Let $X = \{X(a,\kappa)\}_{a\in\{0,1\}^*,\kappa\in\mathbb{N}}$ and $Y = \{Y(a,\kappa)\}_{a\in\{0,1\}^*,\kappa\in\mathbb{N}}$ be two distribution ensembles. We say that X and Y are* computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, if for every PPT machine D, every $a \in \{0,1\}^*$, every positive polynomial $p(\cdot)$ and all sufficiently large κ:*

$$\left|\Pr\left[D(X(a,\kappa),1^\kappa)=1\right] - \Pr\left[D(Y(a,\kappa),1^\kappa)=1\right]\right| < \frac{1}{p(\kappa)}.$$

2.1 Secret Key Encryption (SKE)

We specify the definitions of secret key encryption and indistinguishability under chosen plaintext attacks.

Definition 2.2 (SKE) *We say that $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a* secret key encryption scheme *if $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ are polynomial-time algorithms specified as follows:*

- Gen*, given a security parameter κ (in unary), outputs key* SK*, where* SK *is a secret key. We denote this by* $\mathrm{SK} \leftarrow \mathsf{Gen}(1^\kappa)$.
- Enc*, given the secret key* SK *and a plaintext message m, outputs a ciphertext c encrypting m. We denote this by $c \leftarrow \mathsf{Enc}_{\mathrm{SK}}(m)$; and when emphasizing the randomness r used for encryption, we denote this by $c \leftarrow \mathsf{Enc}_{\mathrm{SK}}(m;r)$.*
- Dec*, given the secret key* SK *and a ciphertext c, outputs a plaintext message m s.t. there exists randomness r for which $c = \mathsf{Enc}_{\mathrm{SK}}(m;r)$ (or $\perp$ if no such message exists). We denote this by $m \leftarrow \mathsf{Dec}_{\mathrm{SK}}(c)$.*

For a secret key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a non-uniform adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we consider the following *indistinguishability game*:

$\mathrm{SK} \leftarrow \mathsf{Gen}(1^\kappa)$.
$(m_0, m_1, history) \leftarrow \mathcal{A}_1^{\mathsf{Enc}_{\mathrm{SK}}(\cdot)}(1^\kappa)$, s.t. $|m_0| = |m_1|$.
$c \leftarrow \mathsf{Enc}_{\mathrm{SK}}(m_b)$, where $b \in_R \{0,1\}$.
$b' \leftarrow \mathcal{A}_2^{\mathsf{Enc}_{\mathrm{SK}}(\cdot)}(c, history)$.
$\mathcal{A}$ wins if $b' = b$.

Denote by $\mathsf{AdvIND}_{\Pi,\mathcal{A}}(\kappa)$ the probability that $\mathcal{A}$ wins in the indistinguishability game.

Definition 2.3 (SKE IND-CPA) *A secret key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under chosen plaintext attacks (IND-CPA)*, if for every non-uniform adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* negl *such that* $\mathsf{AdvIND}_{\Pi,\mathcal{A}}(\kappa) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$.

2.2 Oblivious Pseudorandom Function Evaluation

Informally speaking, a pseudorandom function (PRF) is an efficiently computable function that looks like a truly random function to any PPT observer. Namely,

Definition 2.4 (Pseudorandom function ensemble) *Let* $F = \{\mathrm{f}_\kappa\}_{\kappa\in\mathbb{N}}$ *where for every* κ, $\mathrm{f}_\kappa : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^l$ *be an efficiently computable ensemble of keyed functions. We say that* $F = \{\mathrm{f}_\kappa\}_{\kappa\in\mathbb{N}}$ *is* a pseudorandom function ensemble *if for all PPT distinguishers* D*, there exists a negligible function* negl *such that for every* κ*:*

$$|\Pr D^{\mathrm{f}_\kappa(k,\cdot)}(1^\kappa) = 1 - \Pr D^{f_\kappa}(1^\kappa) = 1| \leq \mathsf{negl}(\kappa),$$

where k *is picked uniformly from* $\{0,1\}^\kappa$ *and* f_κ *is chosen uniformly at random from the set of functions mapping* m*-bit strings into* l*-bit strings. We sometimes omit* κ *from our notation when it is clear from the context.*

In our protocols, we consider a protocol π_{PRF} that *obliviously* evaluates a pseudorandom function in the presence of semi-honest adversaries. Specifically, let $k \in \{0,1\}^\kappa$ be a key sampled as above. Then the oblivious PRF evaluation functionality $\mathcal{F}_{\mathrm{PRF}}$ is defined by $(k, x) \mapsto (-, \mathrm{f}_k(x))$. An oblivious PRF protocol can be designed based on the Naor-Reingold pseudorandom function [NR97] and implemented by the protocol presented in [FIPR05], which involves running an oblivious transfer protocol [EGL85] for every bit of the input x. This function is defined by

$$\mathrm{f}((a_0, \ldots, a_m), x) = g^{a_0 \prod_{i=1}^m a_i^{x[i]}},$$

where g is a generator for a group $\mathbb{G}$ of prime order p, $a_1, \ldots, a_m \in \mathbb{Z}_p$ and $x = (x[1], \ldots, x[m]) \in \{0,1\}^m$. We remark that both the key and the range are not bit strings, as required by Definition 2.4, but they can be interpreted as such in a natural way.

More generally, a two rounds oblivious PRF protocol can be implemented in the semi-honest setting for any PRF, based on the garbling technique of Yao [Yao86] assuming a semi-honest oblivious transfer.

3 Our Modeling

In this section we model the reactive database search functionality where one client has a database, and another set of clients search the database using a sequence of queries. To simplify the presentation we denote the former client by the sender and the other set of clients by the receiver. (For simplicity, we focus on a single receiver asking for multiple queries).

Inputs and outputs. The input of the sender is a database T of size n. The input queries of the receiver $\{q_i\}_{i\in[t]}$ are picked from a predefined set Q_n where Q_n is a set of queries that correspond to a database of size n. Specifically, we let the set of queries $\{Q_n\}_{n\in\mathbb{N}}$ depend on the database size. This formalization captures search functionalities where Q_n changes with the database size, such as in oblivious transfer with adaptive queries. It further covers search functionalities where the same set of queries is used for databases

Functionality $\mathcal{F}_{\text{ODBS}}$

Let $m, t \in \mathbb{N}$ and $Q = \{Q_n\}_n$. Functionality $\mathcal{F}_{\text{ODBS}}$ sets a table $\mathcal{B}$ initially to be empty and proceeds as follows, running with sender Sen, receiver Rec, server Ser and ideal adversary $\mathcal{SIM}$.

1. Upon receiving a message (DB, T, m) from Sen, send $(\mathsf{preprocess}, |T|, m)$ to Ser and $\mathcal{SIM}$, and record (DB, T) and $n = |T|$.
2. Upon receiving a message (query, q_i) from Rec (for $i \in [t]$), where message $(\mathsf{DB}, \cdot)$ has been recorded, $|q_i| \leq m$ and $q_i \in Q_n$, check if the table $\mathcal{B}$ already contains an entry of the form $(q_i, \cdot)$. If not, then pick the next available identifier id from $\{0,1\}^*$ and add (q_i, id) to $\mathcal{B}$. Send $(\mathsf{query}, \mathtt{Rec})$ to Sen and $\mathcal{SIM}$.
   (a) Upon receiving $(\mathsf{approve}, \mathtt{Rec})$ from Sen send $(\mathsf{response}, \mathtt{Rec}, |T_i|, \mathsf{id})$ to server Ser. Otherwise, if no $(\mathsf{approve}, \mathtt{Rec})$ message has been received from Sen, send $\perp$ to Rec and abort.
   (b) If there exists a matched record T_i send $(\mathsf{response}, q_i, T_i, \mathsf{id})$ to Rec. Otherwise, send *"no match"*.

Figure 2: The outsourced database search functionality

of different sizes by fixing the same set of queries for all n, such as in pattern matching, (see Section 3.3 for the formal definitions of these functionalities).

The queries made by the receiver are determined adaptively by a PPT algorithm M that takes the receiver's initial input and the outputs of prior search results. Whenever we say that the honest receiver picks a search query $q_i \in Q_n$, we assume that the receiver applies its input selection algorithm M as specified above. Queries that do not have a suitable answer in the database will be responded with a *"no match"* message whenever queried by the receiver. Finally, we assume that $|q| \leq m$ for all $q \in Q_n$ and some fixed parameter $m = m(\kappa)$. We further assume that n is polynomial in the security parameter κ. We let T_q denote the response of the functionality on database T and query $q \in Q_n$.

The reactive search functionality. The reactive search functionality can be described in two phases.

1. In the *setup phase* the sender sends a message $a(T)$ to the server, where $a(\cdot)$ is some polynomial-time algorithm. This phase is run only once, such that the size of the sender's state s upon completion is bounded by $\mathsf{poly}(\kappa)$.[5]
2. In the *query phase* The receiver picks a search query and obtains from the server the answer to this query. Note that this definition is meaningful only if we restrict the number of queries made by the receiver. Otherwise, no notion of privacy is guaranteed for the sender, since the receiver (or even the server) can potentially search the database for as many queries as they wish. This requirement is formalized by asking the sender's "permission" whenever a query is made, and is an important feature of payment-based search applications where the receiver pays per search. Looking ahead, we implement this restriction using a secure protocol between the sender and the receiver that allows the receiver to learn the answer to its search query.

The formal definition of outsourced database search functionality appears in Figure 2.

Complexities. In order to take some advantage from this modeling, we would like the setup phase to require $O(n)$ workload, yet the overall cost of issuing a query should only *grow linearly with the size*

[5]For this to be meaningful, we requite that the size of the sender's state is strictly less than n. This is formalized by assuming the existence of two polynomials $p_1(\cdot)$ and $p_2(\cdot)$ such that $n \leq p_1(\kappa)$, $s \leq p_2(\kappa)$ and $s \in o(n)$.

of the query's response (which is as optimal as one can obtain). As mentioned before, for some search functionalities, where there is no fixed bound on the database records, this optimization comes with the price of revealing some leakage about the database. We further allow leaking the search pattern, where the server recognizes whether the same query already asked before. Finally, we require that the round complexity of any protocol implemented in this setting is minimal. I.e., in the setup phase we require a single message sent from the sender to the server, whereas in the query phase we require the receiver exchange only two messages (one in each direction) with each of the other clients.

Security definition. Security is formalized using the ideal/real paradigm, considering the server as a separate entity that does not contribute any input to the computation. In the ideal setting, such an entity is also communicating with the functionality and upon corruption, decides whether the functionality sends the outcome of the computation to the receiver. As in the standard static modeling a corrupted party is either semi-honest or malicious, where in the semi-honest setting the attacker follows the protocol's instructions but tries to gain additional information about the honest parties' inputs, whereas in the malicious setting the attacker follows an arbitrary efficient strategy. This modeling also captures collusion between some of the parties, when the adversary corrupts more than one party and the corrupted parties share a joint state. In this work we only consider collusion between the server and the receiver. In particular, collusion between the sender and the receiver is not interesting as the server has no input and no output, whereas the question of collusion between the server and the sender remains open. Intuitively, it is not clear how to protect the receiver's privacy when the adversary sees the sender's private information and the message from the receiver to the server.[6] We say that a protocol is secure in the presence of (P_1/P_2)-*collusion* if security holds against collusion between parties P_1 and P_2 (in addition to individual corruptions). In this work we consider both the private channels setting (where the communication between honest parties is not seen by the adversary) and the non-private channels setting. Note that any protocol in the private channels setting can be executed in a non-private channels environment by encrypting the communication between any pair of parties. This transformation increases the number of rounds as the parties need to exchange their public keys, where this additional message is sent only once at the beginning of the query phase.

Formally, denote by $\mathbf{IDEAL}_{\mathcal{F}_{\text{ODBS}},\mathcal{SIM}(z)}(\kappa,(-,T,(q_1,\ldots,q_t)))$ the output of an ideal adversary $\mathcal{SIM}$, server $\texttt{Ser}$, receiver $\texttt{Rec}$ and sender $\texttt{Sen}$ in the above ideal execution of $\mathcal{F}_{\text{ODBS}}$ upon given inputs $(-,(T,(q_1,\ldots,q_t)))$ and auxiliary input z to $\mathcal{SIM}$. Functionality $\mathcal{F}_{\text{ODBS}}$ is implemented via a protocol $\pi=(\pi_{\mathsf{Pre}},\pi_{\mathsf{Query}})$ consisting of a pair of protocols specified as follows. A two-party protocol π_{Pre} that is run in the setup phase by $\texttt{Sen}$, that preprocesses database T and forwards the outcome $a(T)$ to $\texttt{Ser}$. During the query phase protocol π_{Query} is run between $\texttt{Rec}$ (holding a query q) and $\texttt{Sen}, \texttt{Ser}$, where $\texttt{Rec}$ communicates with each party separately. We denote by $\mathbf{REAL}_{\pi,\mathcal{A}(z)}(\kappa,(-,T,(q_1,\ldots,q_t)))$ the output of a non-uniform PPT adversary $\mathcal{A}$, server $\texttt{Ser}$, sender $\texttt{Sen}$ and receiver $\texttt{Rec}$ in a real execution of $\pi=(\pi_{\mathsf{Pre}},\pi_{\mathsf{Query}})$ on inputs $(-,(T,(q_1,\ldots,q_t)))$ and auxiliary input z given to $\mathcal{A}$.

Definition 3.1 (Security of outsourced database search) *We say that* π securely implements $\mathcal{F}_{\text{ODBS}}$ *with respect to queries* $Q=\{Q_n\}_{n\in\mathbb{N}}$ *in the presence of* $(\texttt{Ser}/\texttt{Rec})$*-collusion and semi-honest (respectively, malicious) adversaries, if for any PPT semi-honest (respectively, malicious) adversary* $\mathcal{A}$ *there exists a PPT semi-honest (respectively, malicious) simulator* $\mathcal{SIM}$ *such that for any tuple of inputs* $(T,(q_1,\ldots,q_t))$ *such that* $q_1,\ldots,q_t\in Q_{|T|}$*, and auxiliary input* z*,*

$$\{\mathbf{IDEAL}_{\mathcal{F}_{\text{ODBS}},\mathcal{SIM}(z)}(\kappa,(-,T,(q_1,\ldots,q_t)))\}_{\kappa\in\mathbb{N}}\overset{\text{c}}{\approx}\{\mathbf{REAL}_{\pi,\mathcal{A}(z)}(\kappa,(-,T,(q_1,\ldots,q_t)))\}_{\kappa\in\mathbb{N}}.$$

[6] Notably, our lower bounds also apply to settings where all type of collusion are allowed since this only strengthens the model.

Functionality $\mathcal{F}_{\text{DBS}}$

Let $m, t \in \mathbb{N}$ and $Q = \{Q_n\}_n$. Functionality $\mathcal{F}_{\text{DBS}}$ sets a table $\mathcal{B}$ initially to be empty and proceeds as follows, running with sender Sen, receiver Rec, and ideal adversary $\mathcal{SIM}$.

1. Upon receiving a message (DB, T, m) from Sen, send $(\mathsf{preprocess}, |T|, m)$ to $\mathcal{SIM}$, and record (DB, T) and $n = |T|$.
2. Upon receiving a message (query, q_i) from Rec (for $i \in [t]$), where message $(\mathsf{DB}, \cdot)$ has been recorded, $|q_i| \leq m$ and $q_i \in Q_n$, check if the table $\mathcal{B}$ already contains an entry of the form $(q_i, \cdot)$. If not, then pick the next available identifier id from $\{0,1\}^*$ and add (q_i, id) to $\mathcal{B}$. Send $(\mathsf{query}, \mathtt{Rec})$ to Sen and $\mathcal{SIM}$.
   - Upon receiving $(\mathsf{approve}, \mathtt{Rec})$ from Sen check if there exists a matched record T_i and send $(\mathsf{response}, q_i, T_i, \mathsf{id})$ to Rec if so. Otherwise, send *"no match"* to Rec.
   - Otherwise, if no $(\mathsf{approve}, \mathtt{Rec})$ message has been received from Sen, send $\perp$ to Rec and abort.

Figure 3: The database search functionality

The $\mathcal{F}$-hybrid model. In the constructive part of this paper we will use secure two-party protocols as subprotocols. The standard way of doing this is to work in a "*hybrid model*" where parties both interact with each other (as in the real model) and use trusted help (as in the ideal model). Specifically, when constructing a protocol π that uses a subprotocol for securely computing some functionality $\mathcal{F}$, we consider the case that the parties run π and use "ideal calls" to a trusted party for computing $\mathcal{F}$. Upon receiving the inputs from the parties, the trusted party computes $\mathcal{F}$ and sends all parties their output. Then, after receiving these outputs back from the trusted party the protocol π continues.

Let $\mathcal{F}$ be a functionality and let π be a two-party protocol that uses ideal calls to a trusted party computing $\mathcal{F}$. Furthermore, let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine. Then, the *$\mathcal{F}$-hybrid execution of* π on inputs $(T, (q_1, \ldots, q_t))$, auxiliary input z to $\mathcal{A}$ and security parameter κ, denoted $\mathbf{HYBRID}_{\pi^{\mathcal{F}}, \mathcal{A}(z)}(\kappa, (-, T, (q_1, \ldots, q_t)))$, is defined as the output vector of the honest parties and the adversary $\mathcal{A}$ from the hybrid execution of π with a trusted party computing $\mathcal{F}$. By the composition theorem of [Can00] any protocol that securely implements $\mathcal{F}$ can replace the ideal calls to $\mathcal{F}$.

3.1 Outsourced Database Search and the Two-Party Setting

We prove that security of database search in our outsourced model implies secure two-party computation under certain corruption cases. We specify the definition of database search functionality in the standard two-party setting in Figure 3 (where no server is involved in the computation). The proof follows by the observation that a collusion between Ser and Rec can be reduced to a single player playing the role of the receiver in a two-party protocol implementing $\mathcal{F}_{\text{DBS}}$. The role of Sen remains unchanged. Note that the following statement is independent of the corruption model (i.e., semi-honest/malicious) and the communication/round complexities.

Theorem 3.2 *Let π be a protocol securely implementing $\mathcal{F}_{\text{ODBS}}$ with respect to queries $Q = \{Q_n\}_n$ in the presence of* (Ser/Rec)*-collusion. Then π securely implements $\mathcal{F}_{\text{DBS}}$.*

Proof Sketch: Let π be a protocol computing $\mathcal{F}_{\text{ODBS}}$ with sender Sen, receiver Rec and server Ser. We first adjust π into the two-party setting by defining a new pair of polynomial-time algorithms $(\mathtt{Sen}', \mathtt{Rec}')$ as follows. Define $\mathtt{Sen}'$ first; given input T and 1^κ, $\mathtt{Sen}'$ internally initializes $\mathtt{Sen}(T, 1^\kappa)$. Then, whenever $\mathtt{Sen}'$ receives a message from $\mathtt{Rec}'$, it invokes Sen on this message (and its internal state) and forwards

$\mathtt{Sen}$'s response to $\mathtt{Rec}'$. ($\mathtt{Sen}'$ is essentially identical to $\mathtt{Sen}$ only that it is communicating with a single party instead of two). Next, we define $\mathtt{Rec}'$ as follows. Given the description of an input selection algorithm M and 1^κ, $\mathtt{Rec}'$ internally initializes $\mathtt{Rec}(M, 1^\kappa)$ and $\mathtt{Ser}(1^\kappa)$. Whenever $\mathtt{Rec}'$ needs to generate a message to $\mathtt{Sen}'$, $\mathtt{Rec}'$ invokes $\mathtt{Rec}$ and $\mathtt{Sen}$ according to the specification of π and simulates the interaction between them if needed. Specifically, $\mathtt{Rec}'$ mimics the behaviour of a dummy adversary that controls $\mathtt{Ser}$ and $\mathtt{Rec}$ but acts honestly and follows the instructions of π.

Security follows from the security of π. Namely, let $\mathcal{A}$ denote a polynomial-time adversary corrupting $\mathtt{Sen}$, and let $\mathcal{SIM}$ denote a polynomial-time simulator guaranteed by the simulation-based security of π. Then for every adversary corrupting $\mathtt{Sen}'$ define the same simulator $\mathcal{SIM}$ for which security straightforwardly follows, as $\mathtt{Sen}'$ and $\mathtt{Sen}$ follow the same instructions. In addition, let $\mathcal{A}$ denote a polynomial-time dummy adversary corrupting both $\mathtt{Ser}$ and $\mathtt{Rec}$, and let $\mathcal{SIM}$ denote a polynomial-time simulator guaranteed by the simulation-based security of π with respect to this collusion. Then for every adversary corrupting $\mathtt{Rec}'$ define a new simulator $\mathcal{SIM}'$ that is hardwired with the code of $\mathcal{SIM}$, which internally simulates both $\mathtt{Ser}$ and $\mathtt{Rec}$. Upon completing the simulation, $\mathcal{SIM}'$ outputs the view returned by $\mathcal{SIM}$. ■

Note that Theorem 3.2 implies that our construction from Section 5 is also secure in the two-party setting.

3.2 Useful Notations

Let n be a natural number denoting the size of the database and let $Q = \{Q_n\}_{n\in\mathbb{N}}$ be such that Q_n is a set of appropriate queries for databases of size n.[7] We introduce important notations next.

Definition 3.3 *For every $q \in Q_n$, we let $\mathcal{ANS}_{n,q}$ denote the set of all potential responses T_q for the query q when ranging over all databases T of size n. Formally, $\mathcal{ANS}_{n,q} = \{T_q \mid T \in \{0,1\}^n\}$. Furthermore, let $H_{n,Q} = \max_{q\in Q_n} \log|\mathcal{ANS}_{n,q}|$, which intuitively captures the maximal amount of information that a response for* any *query $q \in Q_n$ provides.*

For instance, consider the oblivious transfer with adaptive queries functionality where every entry in the database is of size ℓ. In this case, $\mathcal{ANS}_{n,q}$ is the set of all ℓ-length binary strings.

Definition 3.4 *We specify the following definitions:*

1. *Denote by $\mathsf{cc}^{n,q}_{\mathtt{Ser}}(\kappa) = \mathsf{cc}_{\mathtt{Ser}}(\kappa, n, q)$ the communication complexity of the interaction between $\mathtt{Rec}$ and $\mathtt{Ser}$ within π_{Query} such that the receiver's input is the query q and the database is of size n. Namely, the number of bits being transferred between the receiver and the server in the query phase with parameters κ and q.*

2. *Analogically, denote by $\mathsf{cc}^{n,q}_{\mathtt{Sen}}(\kappa) = \mathsf{cc}_{\mathtt{Sen}}(\kappa, n, q)$ the communication complexity of the interaction between the receiver and the sender within π_{Query} such that the receiver's input is the query q and the database is of size n.*

3. *Denote by $\mathsf{cc}^{n,q}(\kappa) = \mathsf{cc}(\kappa, n, q)$ the overall communication complexity within π_{Query}. Namely, the overall number of bits being transferred during the execution of π_{Query} such that the receiver's input is the query q and the database is of size n.*

4. *Finally, denote by $\mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) = \mathsf{rand}_{\mathtt{Rec}}(\kappa, n, q)$ the size of the receiver's random tape within π_{Query} such that the receiver's input is the query q and the database is of size n.*

[7] We emphasize that the infeasibility proof holds for any database of length n (regardless of its internal structure).

3.3 Concrete Functionalities

We specify the description of two important functionalities in the context of database search.

3.3.1 Outsourced Oblivious Transfer with Adaptive Queries

The basic t-out-of-n oblivious transfer functionality between a sender and a receiver is denoted by $\mathcal{F}_{\mathrm{OT}}$: $((x_1, \ldots, x_n), (q_1, \ldots, q_t)) \mapsto (-, (x_{q_1}, \ldots, x_{q_t}))$, where $x_i \in \{0,1\}^{\ell}$ for all $i \in [n]$, and $\ell = \ell(\kappa), n = n(\kappa)$ are polynomials in κ. Namely, the receiver learns t elements from the input vector of the sender while the sender learns nothing. Note that by definition the receiver decides on the elements it wishes to obtain in advance. Alternatively, we can modify the description of $\mathcal{F}_{\mathrm{OT}}$ so that the receiver picks its input adaptively. This functionality is denoted by oblivious transfer with *adaptive* queries and is a special case of pattern matching defined next, where the queries are indices from $[n]$ and the outcome is the record in the ith database entry. To this end, we denote by $\mathcal{F}_{\mathrm{OT}}$ the oblivious transfer functionality with adaptive queries. We further denote the outsourced variant of this problem by $\mathcal{F}_{\mathrm{OOT}}$ where the server uploads its database to an external server.

3.3.2 Outsourced Pattern Matching

The inputs for the basic pattern matching problem are a text T of length n and a pattern p (i.e., keyword) of length m; the goal is to find all the text locations in which the pattern matches the text. A private distributed variant of this problem is defined in the two-party setting, where party P_1 holds a text T and party P_2 holds a pattern p. The goal of P_2 is to learn the positions in which p matches the text without revealing anything about the pattern to P_1; at the same time, P_2 should not learn anything else about the text. The outsourced variant of the problem which is specified in two phases, following the notation of Faust et al. [FHV13]. In the setup phase the sender uploads a (preprocessed) text $a(T)$ to an external server $\mathtt{Ser}$. In the query phase the receiver queries the text by searching patterns and learns the matched text locations. The reader can think of each record in the pattern matching database as a sequence of indices from $[n]$. In comparison with oblivious transfer, implementing this functionality is much more involved, since the database records are strongly related. This makes simulation (for the case the receiver is corrupted) much more challenging. To this end, we denote by $\mathcal{F}_{\mathrm{PM}}$ the pattern matching functionality. We further denote the outsourced variant of this problem by $\mathcal{F}_{\mathrm{OPM}}$ where the server uploads its pattern to an external server.

4 Infeasibility of Outsourced Database Search in the Plain Model

In this section we introduce our infeasibility result of outsourced database search in the plain model. We introduce our lower bound in two settings: (1) In Section 4.1 we prove the private channels case where corrupted parties do not see the communication between the honest parties. (2) In Section 4.2 we prove a similar theorem in the non-private case. In the later proof the adversary can observe the messages between the honest parties, which implies that a corrupted receiver observes the setup message. This simplifies our proof since the simulator does not need to generate the internal state of the server. The proof in the former setting holds only for protocols secure against $(\mathtt{Ser}/\mathtt{Rec})$-collusion and is slightly more involved.

4.1 The Private Channels Case

Our proof is shown in the presence of collusion between the receiver and the server and crucially relies on the assumption that the receiver communicates with the sender first. This ordering enables to split the randomness of an adversary controlling these parties into two distinct and independent sets. In Theorem 4.1

we show that this ordering in necessary, proving that if this order of rounds is modified then the communication complexity between the server and the receiver must be proportional to $H_{n,Q}$, that might be as large as the database size for some functionalities (see Lemma 4.6). Informally, this statement follows since at the time the receiver communicates with the server, the server does not know anything about the database. It therefore does not know the correct response to the receiver's query, and essentially must send as much information as the maximal amount of information sent within any response to query q (with respect to all possible databases of size n). Recall that we assume that the receiver communicates with each party only once. Formally,

Theorem 4.1 *Fix n and m, and let $\pi = (\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$ be a protocol with minimal interaction that securely implements $\mathcal{F}_{\text{ODBS}}$ with respect to queries $Q = \{Q_n\}_n$ in the presence of $(\mathtt{Ser}/\mathtt{Rec})$-collusion and semi-honest adversaries. Then, if π_{Query} is defined such that $\mathtt{Rec}$ communicates with $\mathtt{Ser}$ first, for every n there exists $q \in Q_n$ such that it holds that $\mathsf{cc}^{n,q}_{\mathtt{Ser}}(\kappa) \geq H_{n,Q} - s$.*[8]

Proof: Fix n and assume by contradiction that $\mathsf{cc}^{n,q'}_{\mathtt{Ser}}(\kappa) < H_{n,Q} - s$ for every $q' \in Q_n$ and consider the case that only the server is corrupted. Note that the sender can transmit at most s bits of information about the database since this is the state size it keeps after the setup phase is completed. Intuitively, this implies that if the receiver needs to learn more than s bits about the database it must receive them from the server. Yet, since the server does not know which bits to transfer, it will just transmit everything.

Formally, recall that $H_{n,Q} = \max_{q \in Q_n} \log |\mathcal{ANS}_{n,q}|$ and let $q^* \in Q_n$ be such that $\log |\mathcal{ANS}_{n,q^*}| = H_{n,Q}$. By the above assumption, we have that the overall length of the incoming communication to $\mathtt{Rec}$ on input q^* is *strictly less* than $H_{n,Q} = \log |\mathcal{ANS}_{n,q^*}|$. However, to obtain the correct output, the receiver must learn which of the elements in $\mathcal{ANS}_{n,q^*}$ is the correct response with respect to the sender's input T and hence for the very least, it must learn an overall of $\log |\mathcal{ANS}_{n,q^*}|$ bits of information. This implies that the receiver does not learn T_{q^*} correctly. ■

We stress that for every q, $|\mathcal{ANS}_{n,q}|$ is *independent* of the actual size of T_q for a concrete T, since it counts the number of potential responses when ranging over *all databases of length* n. Thus, the above lower bound is meaningful in the sense that it shows that the communication complexity might be large even if $|T_q|$ is small for some concrete T.

We are now ready to prove the following theorem.

Theorem 4.2 *Fix n and m, and let $\pi = (\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$ be a protocol with minimal interaction that securely implements $\mathcal{F}_{\text{ODBS}}$ with respect to queries $Q = \{Q_n\}_n$ in the presence of $(\mathtt{Ser}/\mathtt{Rec})$-collusion and semi-honest adversaries in the private channels setting, such that $\mathtt{Rec}$ communicates with $\mathtt{Sen}$ first. Then one of the following holds:*

1. For every query $q \in Q_n$ *the communication complexity* $\mathsf{cc}^{n,q}_{\mathtt{Sen}}(\kappa) \geq \frac{H_{n,Q}-3}{2}$ *or*
2. There exists a query $q \in Q_n$ *such that* $\mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) \geq \frac{H_{n,Q}-3}{2}$.

Proof: Let $\pi = (\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$ be as in Theorem 4.2, let $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ be a real-world semi-honest adversary controlling the server and the receiver, and let $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ be an ideal-world adversary guaranteed to exist by the security of $\pi = (\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$. By definition, upon given a message $(\mathsf{preprocess}, |T|, m)$ in the setup phase $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ outputs a string a_{Sim}. Moreover, upon given a message $(\mathsf{response}, q, T_q, \mathsf{id})$ in the query phase it outputs a valid view for $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ (recall that T_q represents the correct output for query q with respect to database T). This view is a triple $(r_{\mathtt{Rec}}, \mathsf{m}_2, r_{\mathtt{Ser}})$, where $r_{\mathtt{Rec}}$ and $r_{\mathtt{Ser}}$ are the respective random tapes of $\mathtt{Rec}$ and $\mathtt{Ser}$ and m_2 is a simulated message from $\mathtt{Sen}$ to $\mathtt{Rec}$.

[8] Recall that s denotes the size of the sender's state in the query phase and that $s \in o(n)$.

For a security parameter κ and a pair of query/record (q, T_q), let $\Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}},\kappa}[a_{\mathsf{Sim}}]$ denote the probability distribution over the simulated message of π_{Pre} and let $\Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}},\kappa,q,T_q}[a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}]$ denote the probability distribution on the values $(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}})$ where a_{Sim} is generated by $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ in the simulation of π_{Pre}, $(r_{\mathtt{Rec}}, \mathsf{m}_2)$ are generated by $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ in the simulation of π_{Query} and $r^*_{\mathtt{Ser}}$ is a uniformly random string. Moreover, let $\Pr_{\pi,\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}},\kappa,T,q}[a(T), r_{\mathtt{Rec}}, \mathsf{m}_2, r_{\mathtt{Ser}}]$ denote the probability distribution on the values $(a(T), r_{\mathtt{Rec}}, \mathsf{m}_2, r_{\mathtt{Ser}})$ that are generated in a real execution of π with $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$, on inputs T for the sender and q of the receiver. We further denote by $\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r_{\mathtt{Ser}})$ the output of the receiver in an execution of π with a message a_{Sim} from $\mathtt{Sen}$ to $\mathtt{Ser}$ in π_{Pre}, and a message m_2 from $\mathtt{Sen}$ to $\mathtt{Rec}$ in π_{Query}, where $r_{\mathtt{Rec}}$ and $r_{\mathtt{Ser}}$ denote the respective random tapes of the receiver and the server.

We begin with a claim that states that whenever $(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}})$ are sampled according to the distribution $\Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}},\kappa,q,T_q}[a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}]$, then the receiver outputs the correct output with probability at least $3/4$. Intuitively, this claim follows by the correctness of the real protocol and the indistinguishability of the ideal and real executions. That is, by the correctness of the protocol it holds that most of the real views $(r_{\mathtt{Ser}}, \mathsf{m}_2)$ yield the correct output, when $r_{\mathtt{Ser}}$ is randomly chosen (recall that by the order of the rounds, $r_{\mathtt{Ser}}$ is independent of $(r_{\mathtt{Rec}}, \mathsf{m}_2)$ in the real protocol). By the security of the protocol this must also hold in the simulation. Therefore, the simulated views must have the property that with a high probability the receiver returns the correct output when $r^*_{\mathtt{Ser}}$ is picked at random.

Claim 4.3 *There exists a κ_0 such that for all $\kappa > \kappa_0$, $T \in \{0,1\}^n$ and $q \in Q_n$,*

$$\Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}},\kappa,q,T_q}[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T_q] \geq \frac{3}{4}. \tag{1}$$

Proof Sketch: Assume that for infinitely many κ's there exists $T \in \{0,1\}^n$ and $q \in Q_n$ such that

$$\Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}},\kappa,q,T^q}[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T^q] < \frac{3}{4}. \tag{2}$$

By the correctness of π, we are guaranteed that for all sufficiently large κ, every $T \in \{0,1\}^n$ and every $q \in Q_n$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr_{\pi,\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}},\kappa,T,q}[\pi_{\mathsf{Output}}(a(T), r_{\mathtt{Rec}}, \mathsf{m}_2, r_{\mathtt{Ser}}) = T^q] > 1 - \mathsf{negl}(\kappa). \tag{3}$$

Therefore, we can construct a PPT distinguisher D that distinguishes between a real execution of π with $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ and an ideal execution of $\mathcal{F}_{\mathrm{ODBS}}$ with $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ as follows. Given input T, q and a view $(a, r_{\mathtt{Rec}}, \mathsf{m}_2, r_{\mathtt{Ser}})$ that is either generated by $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ or by the honest parties in a real execution of π, D chooses a uniform random string $r^*_{\mathtt{Ser}}$ and outputs 1 if and only if $\pi_{\mathsf{Output}}(a, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T^q$.

It is easy to see that if $(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r_{\mathtt{Ser}})$ were generated by $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$, then D outputs 1 with probability that equals to $\Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}},\kappa,q,T^q}[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T^q]$, whereas if $(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r_{\mathtt{Ser}})$ were generated in a real execution of π with $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$, then D outputs 1 with probability that equals to $\Pr_{\pi,\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}},\kappa,T,q}[\pi_{\mathsf{Output}}(a(T), r_{\mathtt{Rec}}, \mathsf{m}_2, r_{\mathtt{Ser}}) = T^q]$. Hence, by Equations. (3) and (2), D distinguishes the views with overwhelming probability. ■

To this end, we fix κ and q. Then, for every a_{Sim} and T_q let

$$\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q) = \left\{ (r_{\mathtt{Rec}}, \mathsf{m}_2) \Big|_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}},\kappa,q,T_q} \Pr[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T_q \mid a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2] > \frac{1}{2} \right\}.$$

Note that the above probability is only taken over the choice of $r^*_{\mathtt{Ser}}$ which is a uniformly random string. Next, for a fixed T_q we let $\mathcal{E}(T_q)$ denote the expected value of $|\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)|$ when a_{Sim} is generated by $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ in the simulation of π_{Pre}. That is,

$$\mathcal{E}(T_q) = \mathbb{E}_{a_{\mathsf{Sim}}}\left[|\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)|\right] = \sum_{a_{\mathsf{Sim}}} \Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec},\kappa}}[a_{\mathsf{Sim}}] \cdot |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)|\,.$$

Then, we prove the following claim,

Claim 4.4 *For every T_q, it holds that $\mathcal{E}(T_q) \geq \frac{1}{4}$.*

Proof: Let T_q be such that $\mathcal{E}(T_q) < 1/4$, we show that this contradicts Claim 4.3. First, recall that $\mathcal{E}(T_q) = \mathbb{E}_{a_{\mathsf{Sim}}}\left[|\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)|\right]$. By the Markov inequality it holds that

$$\Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec},\kappa}}\left[|\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| \geq 1\right] < \frac{1}{4}. \tag{4}$$

Then, by the total probability theorem it holds that

$$\begin{aligned}
&\Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec},n,q,T_q}}[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T_q] \\
&= \Pr\left[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T_q \,\middle|\, |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| \geq 1\right] \cdot \Pr\left[|\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| \geq 1\right] \\
&\quad + \Pr\left[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T_q \,\middle|\, |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| = 0\right] \cdot \Pr\left[|\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| = 0\right] \\
&\leq \Pr\left[|\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| \geq 1\right] + \Pr\left[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T_q \,\middle|\, |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| = 0\right] \\
&< \frac{1}{4} + \frac{1}{2} = \frac{3}{4}.
\end{aligned}$$

The last inequality is due to Eq. (4) and the definition of $\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)$. This contradicts Eq. (1). ■

Let $X_{n,q}$ denote the sum of the expected value $\mathcal{E}(T_q)$ when ranging over all possible T_q's. Then, by Claim 4.4 it holds that $X_{n,q} \geq \frac{1}{4} \cdot |\mathcal{ANS}_{n,q}|$. Moreover, it holds that

$$\begin{aligned}
X_{n,q} = \sum_{T_q \in \mathcal{ANS}_{n,q}} \mathcal{E}(T_q) &= \sum_{T_q \in \mathcal{ANS}_{n,q}} \sum_{a_{\mathsf{Sim}}} \Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}}[a_{\mathsf{Sim}}] \cdot |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| \\
&= \sum_{a_{\mathsf{Sim}}} \Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}}[a_{\mathsf{Sim}}] \cdot \sum_{T_q \in \mathcal{ANS}_{n,q}} |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)|\,.
\end{aligned}$$

Note that for a fixed a_{Sim}, every pair $(r_{\mathtt{Rec}}, \mathsf{m}_2)$ belongs to only one set $\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)$. This is due to the fact that if $(r_{\mathtt{Rec}}, \mathsf{m}_2) \in \mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)$ for some T_q then by definition the following probability $\Pr[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) = T_q \mid a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2] > \frac{1}{2}$. This implies that if a pair $(r_{\mathtt{Rec}}, \mathsf{m}_2)$ belongs to two distinct sets $T_q^0 \neq T_q^1$, then $\Pr[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, r^*_{\mathtt{Ser}}) \in \left\{T_q^0, T_q^1\right\} \mid a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2] > 1$. Therefore, for every a_{Sim} the sum $\sum_{T_q} |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)|$ is over disjoint sets. We conclude that

$$\sum_{T_q \in \mathcal{ANS}_{n,q}} |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| \leq |\{(r_{\mathtt{Rec}}, \mathsf{m}_2)\}| = \sum_{i \leq \mathsf{cc}^{n,q}_{\mathsf{Sen}}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa)} 2^i = 2^{\mathsf{cc}^{n,q}_{\mathsf{Sen}}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) + 1} - 1$$

where the second to the last equality is implied by the fact that $\mathsf{cc}^{n,q}_{\mathsf{Sen}}(\kappa)$ is a bound on the length of m_2 and $\mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa)$ is a bound on the length of $r_{\mathtt{Rec}}$. We therefore conclude that

$$X_{n,q} \leq \sum_{a_{\mathsf{Sim}}} \Pr_{\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}}[a_{\mathsf{Sim}}] \cdot \left(2^{\mathsf{cc}^{n,q}_{\mathsf{Sen}}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) + 1} - 1\right) \leq 2^{\mathsf{cc}^{n,q}_{\mathsf{Sen}}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) + 1} - 1.$$

Combining this with the observation that $X_{n,q} \geq \frac{1}{4} \cdot |\mathcal{ANS}_{n,q}|$, we obtain $2^{\mathsf{cc}^{n,q}_{\mathsf{Sen}}(\kappa)+\mathsf{rand}^{n,q}_{\mathsf{Rec}}(\kappa)+1} - 1 \geq \frac{1}{4} \cdot |\mathcal{ANS}_{n,q}|$ and hence for every query q,

$$\mathsf{cc}^{n,q}_{\mathsf{Sen}}(\kappa) + \mathsf{rand}^{n,q}_{\mathsf{Rec}}(\kappa) \geq \log\left(\frac{1}{4}\,|\mathcal{ANS}_{n,q}|\right) - 1 = \log|\mathcal{ANS}_{n,q}| - 3.$$

Therefore for every query q, it either holds that $\mathsf{cc}^{n,q}_{\mathsf{Sen}}(\kappa) \geq \frac{\log|\mathcal{ANS}_{n,q}|-3}{2}$ or $\mathsf{rand}^{n,q}_{\mathsf{Rec}}(\kappa) \geq \frac{\log|\mathcal{ANS}_{n,q}|-3}{2}$. Recall that $H_{n,Q} = \max_{q\in Q_n} \log|\mathcal{ANS}_{n,q}|$. We conclude that there exists a query $q \in Q_n$ for which either $\mathsf{cc}^{n,q}_{\mathsf{Sen}}(\kappa) \geq \frac{H_{n,Q}-3}{2}$ or $\mathsf{rand}^{n,q}_{\mathsf{Rec}}(\kappa) \geq \frac{H_{n,Q}-3}{2}$. Note that if the former inequality holds, then by the security of π the communication complexity is at least $\frac{H_{n,Q}-3}{2}$ *for all queries* $q \in Q_n$ (otherwise, the sender can learn the receiver's input by just looking at the length of the messages sent in π_{Query}, thus breaking privacy). This concludes the proof of Theorem 4.2. ■

Lemma 4.6 below demonstrates that for the pattern matching functionality there exists a family of queries Q such that $H_{n,Q} = n$ for every n. Combining this with Theorems 4.1-4.2, the following holds,

Corollary 4.5 *There exists a family of queries $Q = \{Q_n\}_n$ such that for* any *protocol with minimal interaction that implements the outsourced pattern matching functionality securely with respect to Q in the private channels setting, for every n one of the following holds:*

1. *There exists $q \in Q_n$ such that the communication complexity in the query phase is at least $\frac{n-3}{2} - s$;*
2. *There exists $q \in Q_n$ such that the length of the receiver's random tape is at least $\frac{n-3}{2} - s$.*

A bound on $H_{n,Q}$ for pattern matching. We prove the following simple observation relative to the pattern matching functionality; see Section 3.3 for the definition of this functionality.

Lemma 4.6 *For the pattern matching functionality there exists a family of queries Q such that $H_{n,Q} = n$ for every n.*

Proof: We prove the existence of a family of queries $Q = \{Q_n\}_n$ such that $H_{n,Q} = n$ for every n. Fix n and let $Q_n = \{0\}$ denote the single-bit pattern $q = 0$. In addition, recall that $H_{n,Q} = \max_{q\in Q_n} \log|\mathcal{ANS}_{n,q}|$ where $\mathcal{ANS}_{n,q} = \{T^q \mid T \in \{0,1\}^n\}$. Note that $\mathcal{ANS}_{n,q=0}$ includes all subsets of $[n]$ and thus, $|\mathcal{ANS}_{n,q=0}| = 2^n$ and $\log|\mathcal{ANS}_{n,q=0}| = n$, implying that $H_{n,Q} \geq \log|\mathcal{ANS}_{n,q=0}| = n$. ■

4.2 The Non-Private Channels Case

In this setting a corrupted party observes the communication between the honest parties. In our context this implies that a corrupted receiver sees the setup message sent from the sender to the server. Consequently, we only need to consider the corruption of the receiver, and the order of communication in the query phase does not matter as in the private channels case. We continue with our main theorem for this section.

Theorem 4.7 *Fix n and m, and let $\pi = (\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$ be a protocol with minimal interaction that securely implements $\mathcal{F}_{\mathrm{ODBS}}$ with respect to queries $Q = \{Q_n\}_n$ in the presence of semi-honest adversaries in the non-private channels setting. Then one of the following holds:*

1. For every query $q \in Q_n$ *the communication complexity* $\mathsf{cc}^{n,q} \geq \frac{H_{n,Q}-2}{2}$ *or*
2. There exists a query $q \in Q_n$ *such that* $\mathsf{rand}^{n,q}_{\mathsf{Rec}}(\kappa) \geq \frac{H_{n,Q}-2}{2}$.

Proof: The proof of Theorem 4.7 is very similar to the proof of Theorem 4.2. We present the outline of the proof. Let $\pi = (\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$ be as in Theorem 4.7, let $\mathcal{A}_{\mathtt{Rec}}$ be a real-world semi-honest adversary controlling the receiver (note that since we do not assume private channels, $\mathcal{A}_{\mathtt{Rec}}$ sees all communication between the honest parties and in particular the message within π_{Pre}), and let $\mathcal{SIM}_{\mathtt{Rec}}$ be an ideal-world adversary guaranteed to exist by the security of $\pi = (\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$. By definition, upon given a message $(\mathsf{preprocess}, |T|, m)$ in the setup phase $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ outputs a string a_{Sim}. Moreover, upon given a message $(\mathsf{response}, q, T_q, \mathsf{id})$ in the query phase it outputs a valid view for $\mathcal{A}_{\mathtt{Ser}}$which consists of a triple $(r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4)$, where $r_{\mathtt{Rec}}$ is the random tape of $\mathtt{Rec}$, m_2 is a simulated message from $\mathtt{Sen}$ to $\mathtt{Rec}$ and m_4 is a simulated message from $\mathtt{Ser}$ to $\mathtt{Rec}$.

For a security parameter κ and a pair of query/record (q, T_q), let $\Pr_{\mathcal{SIM}_{\mathtt{Rec}},\kappa}[a_{\mathsf{Sim}}]$ denote the probability distribution over the simulated message of π_{Pre} and let $\Pr_{\mathcal{SIM}_{\mathtt{Rec}},\kappa,q,T_q}[a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4]$ denote the probability distribution on the values $(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4)$ where a_{Sim} is generated in the simulation of π_{Pre}, and $r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4$ are generated in the simulation of π_{Query}. Moreover, let $\Pr_{\pi,\mathcal{A}_{\mathtt{Rec}},\kappa,T,q}[a(T), r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4]$ denote the probability distribution over the values $(a(T), r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4)$ that are generated in a real execution of π with $\mathcal{A}_{\mathtt{Rec}}$, on inputs T for the sender and q for the receiver. We further denote by $\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4)$ the output of the receiver in an execution of π with a message a_{Sim} from $\mathtt{Sen}$ to $\mathtt{Ser}$ in π_{Pre}, a message m_2 from $\mathtt{Sen}$ to $\mathtt{Rec}$ in π_{Query} and a message m_4 from $\mathtt{Ser}$ to $\mathtt{Rec}$, where $r_{\mathtt{Rec}}$ denotes the random tape of the receiver.

We continue with the following claim,

Claim 4.8 *There exists a κ_0 such that for all $\kappa > \kappa_0$ and $T \in \{0,1\}^n$, $q \in Q_n$,*

$$\Pr_{\mathcal{SIM}_{\mathtt{Rec}},\kappa,q,T_q}[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4) = T_q] \geq \frac{1}{2}. \tag{5}$$

Proof Sketch: Assume that for infinitely many κ's there exists $T \in \{0,1\}^n$ and $q \in Q_n$ such that

$$\Pr_{\mathcal{SIM}_{\mathtt{Rec}},\kappa,q,T_q}[\pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4) = T_q] < \frac{1}{2}.$$

By the correctness of protocol π, it is guaranteed that for all sufficiently large κ, every $T \in \{0,1\}^n$ and every $q \in Q_n$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr_{\pi,\mathcal{A}_{\mathtt{Rec}},\kappa,T,q}[\pi_{\mathsf{Output}}(a(T), r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4) = T_q] > 1 - \mathsf{negl}(\kappa)$$

Therefore we can construct a PPT distinguisher D that distinguishes a real execution of π with $\mathcal{A}_{\mathtt{Rec}}$ and an ideal execution of $\mathcal{F}_{\mathrm{ODBS}}$ with $\mathcal{SIM}_{\mathtt{Rec}}$ as follows. Given input T, q and view $a, r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4$, output 1 if and only if the receiver's output is T_q. It is easy to verify that there is a non-negligible gap relative to the real and the simulated views, and thus D distinguishes the executions with this gap. ■

To this end, we fix κ and q. Then, for every a_{Sim} and T_q let

$$\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q) = \{(r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4) \mid \pi_{\mathsf{Output}}(a_{\mathsf{Sim}}, r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4) = T_q\}.$$

For a fixed T_q, we let $\mathcal{E}(T_q)$ denote the expected value of $|\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)|$ when a_{Sim} is generated by $\mathcal{SIM}_{\mathtt{Rec}}$ in the simulation of π_{Pre}. The following claim is proved similarly to the proof of Claim 4.4:

Claim 4.9 *For every T_q, it holds that*

$$\mathcal{E}(T_q) \geq \frac{1}{2}.$$

Let $X_{n,q}$ denote the sum of the expected value $\mathcal{E}(T_q)$ when ranging over all possible T_q's. We have that

$$X_{n,q} = \sum_{a_{\mathsf{Sim}}} \Pr_{\mathcal{SIM}_{\mathtt{Rec}}}[a_{\mathsf{Sim}}] \cdot \sum_{T_q \in \mathcal{ANS}_{n,q}} |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| \,.$$

Then by Claim 4.9, we have that $X_{n,q} \geq \frac{1}{2} \cdot |\mathcal{ANS}_{n,q}|$.

Note that for a fixed a_{Sim}, every triple $(r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4)$ belongs to only one set $\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)$. This is due to the fact that a triple $(r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4)$ fixes the output of $\mathtt{Rec}$. Therefore, for every a_{Sim} the sum $\sum_{T_q} |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)|$ is of disjoint sets. We conclude that

$$\sum_{T_q \in \mathcal{ANS}_{n,q}} |\mathsf{GoodView}(a_{\mathsf{Sim}}, T_q)| \leq |\{(r_{\mathtt{Rec}}, \mathsf{m}_2, \mathsf{m}_4)\}| \leq \sum_{i \leq \mathsf{cc}^{n,q}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa)} 2^i = 2^{\mathsf{cc}^{n,q}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) + 1} - 1$$

where the second to the last inequality is implied by the fact that $\mathsf{cc}^{n,q}(\kappa)$ is a bound on the overall communication complexity in π_{Query} and $\mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa)$ is a bound on the length of $r_{\mathtt{Rec}}$. We therefore conclude that

$$X_{n,q} \leq \sum_{a_{\mathsf{Sim}}} \Pr_{\mathcal{SIM}_{\mathtt{Rec}}}[a_{\mathsf{Sim}}] \cdot \left(2^{\mathsf{cc}^{n,q}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) + 1} - 1\right) \leq 2^{\mathsf{cc}^{n,q}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) + 1} - 1.$$

Combining this with the observation that $X_{n,q} \geq \frac{1}{2} \cdot |\mathcal{ANS}_{n,q}|$, we obtain

$$2^{\mathsf{cc}^{n,q}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) + 1} - 1 \geq \frac{1}{2} \cdot |\mathcal{ANS}_{n,q}|$$

and hence for every query q,

$$\mathsf{cc}^{n,q}(\kappa) + \mathsf{rand}^{n,q}_{\mathtt{Rec}}(\kappa) \geq \log\left(\frac{1}{2}\,|\mathcal{ANS}_{n,q}|\right) - 1 = \log|\mathcal{ANS}_{n,q}| - 2.$$

We conclude the proof of Theorem 4.7 similarly to the proof of Theorem 4.2. ■

Applying Lemma 4.6 we obtain he following corollary,

Corollary 4.10 *There exists a family of queries $Q = \{Q_n\}_n$ such that for* any *protocol with minimal interaction that securely implements the outsourced pattern matching functionality with respect to Q in the non-private channels setting and for every n one of the following holds:*

1. *The communication complexity between the sender and the receiver in π_{Query} for any $q \in Q_n$ is at least $(n-2)/2$;*
2. *There exists $q \in Q_n$ such that the length of the receiver's random tape is at least $(n-2)/2$.*

Difficulties with proving a communication complexity lower bound. Recall that our infeasibility result provides a lower bound on either the communication complexity of an outsourced protocol or the size of the receiver's random tape. Clearly, it would be preferable if we could give a strict lower bound on each of these complexities separately. Towards achieving this goal, it seems very appealing to use a pseudorandom generator G that shortens the length of the receiver's random tape. Namely, replace the uniform randomness of the receiver in an outsourced protocol π by an output of a pseudorandom generator, computed on a shorter seed of length κ; thus obtaining a new protocol π' where the length of the random tape of the receiver is bounded by κ. It is simple to observe that the communication complexity of π' is exactly the same as the communication complexity of π. We can then apply our lower bound on π' in order to claim that either the

random tape of $\texttt{Rec}'$ in π' is large or the communication complexity of π' is large. Now, since we already know that the random tape of $\texttt{Rec}'$ is of length κ, we conclude that the communication complexity of π' must be large; hence obtaining that the communication complexity of π is large as well.

Unfortunately, this intuition fails when trying to formalize it. We demonstrate why it fails as follows. Let $\pi = (\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$ be a protocol for securely computing $\mathcal{F}_{\text{ODBS}}$ in the presence of $(\texttt{Ser}/\texttt{Rec})$-collusion and semi-honest adversaries, and let π' be a protocol obtained from π by having the receiver $\texttt{Rec}'$ pick a random seed $s \in \{0,1\}^\kappa$ and invoke $\texttt{Rec}$ with randomness $G(s)$. Our goal is to show that π' is also secure in the presence of $(\texttt{Ser}'/\texttt{Rec}')$-collusion and semi-honest adversaries by reducing its security to the security of π. Namely, we need to simulate the view of the corrupted parties in π' using the simulators constructed in the security proof of π. Consider the corruption case of the receiver $\texttt{Rec}$ in π. Then, in order to construct a simulator $\mathcal{SIM}'$ for the corrupted receiver $\texttt{Rec}'$ in π' we need to invoke simulator $\mathcal{SIM}$ and use its output in order to produce a simulated view for $\texttt{Rec}'$.

Recall that a valid view of $\texttt{Rec}$ consists of a pair $(r_{\texttt{Rec}}, \mathsf{trans})$, where $r_{\texttt{Rec}}$ is a random string of length $\mathsf{rand}^{n,q}_{\texttt{Rec}}(\kappa)$ and trans are the incoming messages that $\texttt{Rec}$ observes during the execution of π_{Query} with randomness $r_{\texttt{Rec}}$, whereas a valid view for $\texttt{Rec}'$ consists of a pair (s, trans) where s is a random seed of length κ and trans are the incoming message that $\texttt{Rec}'$ observes during the execution of π_{Query} with randomness $G(s)$. Then, it is not clear how to use the output $(r_{\texttt{Rec}}, \mathsf{trans})$ of $\mathcal{SIM}$ in order to construct a simulated view (s, trans) for $\texttt{Rec}'$ within π'. Specifically, the difficulty is mainly because it might be that $\mathcal{SIM}$ outputs only views for which $r_{\texttt{Rec}}$ is not in the range of G, and hence obtaining a corresponding s (that is part of $\mathcal{SIM}'$'s output) is not even possible.

Finally, we remark that any attempt to relax the security definition in a way that forces $\mathcal{SIM}$ to only output strings $r_{\texttt{Rec}}$ that have preimages relative to G, fails as well. This is because in this case the real and the ideal ensembles that correspond to $\texttt{Rec}'$'s view must consist of the seed s to the pseudorandom generator. This implies that the security argument cannot be based on the indistinguishability of $G(s)$ from a random string of the appropriate length.

5 Efficient Outsourced Database Search

In this section we demonstrate the feasibility of securely realizing functionality $\mathcal{F}_{\text{ODBS}}$ in the outsourced setting in the presence of $(\texttt{Ser}/\texttt{Rec})$-collusion and semi-honest adversaries in the private channels setting (we refer to Section 3 regarding a discussion of transforming a protocol in this setting into the non-private setting). Our construction assumes that database T is ordered by a sequence of pairs (q_i, T_i), where a record T_i is the corresponding answer to search query q_i. In some cases it may be useful to think of the pairs based database as a restructured database. Namely, we begin with some initial n bits database and then extract from it information according to some fixed set of queries. We further assume that the number of records n' can be extracted from the size of the database. The main underlying idea is to let the receiver learn a trapdoor for each query that is split into two parts; one is forwarded to the server that uses it to identify the corresponding (encrypted) record, and a second part that is used by the receiver to extract the content of the returned encrypted record. We formalize this intuition in the following section.

Ideally, we would like to work with databases with overall $O(\mathsf{polylog}(n) \cdot n)$ number of bits, since larger blowups do not make sense in the outsourced setting (even though our solution is not restricted in that sense). Nevertheless, it is important to note that such reconstructed databases are not necessarily the most concise way to represent a database, and may cause a significant blowup in the preprocessing phase. For instance, the structure of the pattern matching database implies $O(n^2)$ overhead and size since the upper bound of each record is $O(n)$. Alternative database reconstructions do exist. Specifically, for pattern matching one can encode every substring of length m separately, given that m is the length of the searched

pattern.[9] This implies $O(nm)$ preprocessing overhead. On the other hand, our goal in this section is to present a general framework that captures all database search functionalities, rather than presenting the most efficient construction for any particular functionality. In particular, we view this as a conceptual contribution that aims to abstract the cryptographic properties needed in order to achieve security in the outsourced setting. Moreover, our construction meets the lower bound from the prior section for functionalities with a fixed upper bound on the record size, and achieves optimal communication complexity bounds both in the preprocessing and query phases.

5.1 Encryption Scheme with Multiple Secret Keys

We define first a slightly modified notion of encryption scheme that captures the security properties needed for implementing reactive database functionalities securely and efficiently. Here we have a master secret key that produces multiple secret subkeys, where the goal is to have each subkey associated with a query $q \in Q$. We denote these subkeys by *query secret keys*. For such an encryption scheme $\Pi = (\mathsf{MGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, correctness is satisfied as follows: for all $q_i \in Q$ and $m \in \mathcal{M}$, $\mathsf{Dec}_{\mathrm{SK}_{q_j}}(\mathsf{Enc}_{\mathrm{SK}_{q_i}}(m)) = m$ if and only if $i = j$, where query secret key SK_q is generated using a key generation algorithm (Gen) that takes a master secret key (generated by MGen) and a query $q \in Q$, and $\mathcal{M}$ is the message space.

Security is also defined in the spirit of IBE security. Namely, the adversary asks its oracle for query secret keys. It then outputs two equal length messages m_0, m_1 and a query q^*, and is given back an encryption of one of these messages under SK_{q^*}. The only requirement is that the adversary does not ask for SK_{q^*} in the query phase. The adversary then needs to guess which message was encrypted. It is important to note the differences between our primitive and IBE (and also functional encryption). First, although queries can be interpreted as IBE identities, secret keys must be generated obliviously. This requirement is crucial for the privacy of the receiver in our protocol. In addition, the fact that the sender holds both the data and the secret trapdoors implies that our primitive can be designed based on weaker symmetric building blocks in contrast to more "expensive" primitives such as IBE; see below for a concrete example based on PRFs. This is in contrast to IBE that is, by definition, a public-key primitive.

Definition 5.1 (Encryption scheme with multiple secret keys) *We say that $\Pi = (\mathsf{MGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, parameterized by a family of queries $Q = \{Q_\kappa\}_\kappa$,[10] is an encryption scheme with multiple secret keys if $\mathsf{MGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ are polynomial-time algorithms specified as follows:*

1. MGen*, given the security parameter κ, outputs a master secret key* MSK.
2. Gen*, given the master secret key* MSK *and a query $q_i \in Q$, outputs a secret key* SK_{q_i}.
3. Enc*, given a query secret key* SK_{q_i} *for some $q_i \in Q$ and a message $m \in \mathcal{M}$, outputs a ciphertext c encrypting m. We denote this by $c \leftarrow \mathsf{Enc}_{\mathrm{SK}_{q_i}}(m)$; and when emphasizing the randomness r used for encryption, we denote this by $c \leftarrow \mathsf{Enc}_{\mathrm{SK}_{q_i}}(m; r)$.*
4. Dec*, given a query secret key* SK_{q_i} *for some $q_i \in Q$ and a ciphertext $c \leftarrow \mathsf{Enc}_{\mathrm{SK}_{q_i}}(m)$, outputs a plaintext message m. We denote this by $m \leftarrow \mathsf{Dec}_{\mathrm{SK}_{q_i}}(c)$.*

For a query based encryption scheme $\Pi = (\mathsf{MGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a non-uniform adversary $\mathcal{A}$, we consider the following *indistinguishability game*:

[9] Note that it is an open problem to design an outsourced protocol for pattern matching for all potential pattern lengths, with preprocessing overhead $o(n^2)$.

[10] Note that while in Protocol 1 we index the queries by the database size n, in this definition we index members in Q by the security parameter κ. This makes no difference since n is a fixed polynomial of κ, which implies that the two notations are interchangeable.

- Stage 1: The master secret key $\mathrm{MSK} \leftarrow \mathsf{MGen}(1^\kappa)$ is generated.
- Stage 2: The adversary obtains query secret keys. Whenever the adversary queries its challenger on query $q_i \in Q$ the challenger responds with SK_{q_i}. This stage repeats as long as the adversary desires.
- Stage 3: The adversary outputs to the challenger two equal length messages m_0, m_1 and a query q^*, and receives as a challenge an encryption of m_b computed under query secret key SK_{q^*} and denoted by $c \leftarrow \mathsf{Enc}_{\mathrm{SK}_{q^*}}(m_b)$, for a uniformly random $b \leftarrow \{0,1\}$. The only requirement is that the adversary does not ask for SK_{q^*} in the query phase.
- Stage 4: The adversary continues to issue secret key queries. This stage repeats as long as the adversary desires.
- Stage 5: The adversary makes a guess $b' \leftarrow \{0,1\}$, and wins iff $b' = b$.

Denote by $\mathsf{AdvQuery}_{\Pi,Q,\mathcal{A}}(\kappa)$ the probability that $\mathcal{A}$ wins the indistinguishability game.

Definition 5.2 (Security for encryption scheme with multiple secret keys) *An encryption scheme with multiple secret keys* $\Pi = (\mathsf{MGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* secure *if for every non-uniform adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that* $\mathsf{AdvQuery}_{\Pi,Q,\mathcal{A}}(\kappa) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$.

Next, we recall that in the security proof of database search functionalities, the simulator must be able to equivocate the database when simulating the view of a corrupted receiver (as it is committed to the database before observing any query). In order to capture this property we extend our notion of encryption, allowing secret keys equivocation for secret keys that were generated in a fake mode. More formally,

- *Secret key equivocation:* We require the existence of a PPT algorithm $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ that takes the master secret key MSK and outputs a fake ciphertext c and a trapdoor, such that for every $m \in \mathcal{M}$ it can generate a secret key that decrypts c into m. Formally, the following ensembles are computationally indistinguishable for every $m \in \mathcal{M}$ and $q \in Q$:

$$\{(q, \mathrm{SK}_q, c)\}_{\mathrm{SK}_q \leftarrow \mathsf{Gen}(\mathrm{MSK},q), c \leftarrow \mathsf{Enc}_{\mathrm{SK}_q}(m)}$$
$$\{(q, \mathrm{SK}, c)\}_{(c,\mathsf{td}) \leftarrow \mathsf{Sim}_1(1^\kappa), \mathrm{SK} \leftarrow \mathsf{Sim}_2(c,\mathsf{td},m)}.$$

This implies that the number of potential secret keys must be (at least) of the size of $\mathcal{M}$, since for each $m \in \mathcal{M}$ there should be a different secret key that equivocates c into m, and since fake keys are indistinguishable from real keys, they must be of the same size. In addition, we require that the query be part of the ensemble since this captures the corrupted receiver's view more accurately. Specifically, we need to enure that a real query secret key is indistinguishable from a fake key, even in the presence of the input query held by the receiver. Consequently, this implies that a query secret key does not reveal any information about the query it is associated with, in the sense that it is infeasible to associate between a secret key SK_q and a query q. Saying differently, the triples (q, q', SK_q) and $(q, q', \mathrm{SK}_{q'})$ are computationally indistinguishable for any two distinct queries $q, q' \in Q$. This holds because for any q, q' and m,

$$\{(q, q', \mathrm{SK}_q, c)\}_{\mathrm{SK}_q \leftarrow \mathsf{Gen}(\mathrm{MSK},q), c \leftarrow \mathsf{Enc}_{\mathrm{SK}_q}(m)} \overset{\mathrm{c}}{\approx} \{(q, q', \mathrm{SK}, c)\}_{(c,\mathsf{td}) \leftarrow \mathsf{Sim}_1(1^\kappa), \mathrm{SK} \leftarrow \mathsf{Sim}_2(c,\mathsf{td},m)}$$
$$\overset{\mathrm{c}}{\approx} \{(q, q', \mathrm{SK}_{q'}, c)\}_{\mathrm{SK}_{q'} \leftarrow \mathsf{Gen}(\mathrm{MSK},q'), c \leftarrow \mathsf{Enc}_{\mathrm{SK}_{q'}}(m)}$$

where the indistinguishability arguments follow due to our definition above. This property of query hiding is essential in order to enable the simulator to generate fake secret keys independently of the query (as the simulator is committed to a fake database that is independent of the real database). We denote our notion

of encryption by *equivocal encryption scheme with multiple secret keys*. Similarly, we claim that the tuples (q, q', m, m', c_q) and $(q, q', m, m', c_{q'})$ are indistinguishable, for $c_q \leftarrow \mathsf{Enc}_{\mathrm{SK}_q}(m)$ and $c_{q'} \leftarrow \mathsf{Enc}_{\mathrm{SK}_{q'}}(m')$. This holds because for any q, q' and m, m',

$$\{(q, q', m, m', c)\}_{\mathrm{SK}_q \leftarrow \mathsf{Gen}(\mathrm{MSK}, q), c \leftarrow \mathsf{Enc}_{\mathrm{SK}_q}(m)} \overset{\mathrm{c}}{\approx} \{(q, q', m, m', c^*)\}_{\mathrm{SK}_q \leftarrow \mathsf{Gen}(\mathrm{MSK}, q), c^* \leftarrow \mathsf{Enc}_{\mathrm{SK}_q}(m')}$$
$$\overset{\mathrm{c}}{\approx} \quad \{(q, q', m, m', c')\}_{\mathrm{SK}_{q'} \leftarrow \mathsf{Gen}(\mathrm{MSK}, q'), c' \leftarrow \mathsf{Enc}_{\mathrm{SK}_{q'}}(m')}$$

where the first indistinguishability follows due to IND-CPA and the second due to query hiding.

A concrete instantiation based on PRFs. One potential efficient instantiation is fixing the master secret key to be a PRF key k, such that $\mathrm{SK}_i = \mathrm{f}_k(i)$ and encryption is performed in a one-time pad style by masking the message with the secret key, yielding $c = m \oplus \mathrm{SK}_q$. Moreover, $\mathsf{Sim}_1(1^\kappa)$ outputs a random string c of the same length and an empty td. Equivocation is achieved by returning a uniform string r, such that $m = c \oplus r$ (note that query hiding follows from the security of the PRF). Looking ahead, to maintain the sender's privacy the receiver must only learn the individual random strings for the queries it asks. To maintain the receiver's privacy, the sender should not learn the values of the queries asked by the receiver. We therefore implement this secret key transfer using a secure two-party protocol, which boils down to a two rounds oblivious PRF evaluation protocol (see more about this functionality in Section 2.2).

A note about the message space. Note that for some functionalities, such as in pattern matching related functionalities, there is no constant upper bound on the record size, and in fact, the worst case bound grows linearly with the size of the database. To this end, whenever we assume the existence of a secure encryption scheme with query secret keys, we implicitly assume that this scheme is associated with message space $\{0,1\}^\ell$. Note that any such encryption scheme with a fixed message space $\{0,1\}^\ell$ can be used to encrypt messages from larger spaces $\{0,1\}^{\ell \cdot \mathsf{poly}(\kappa)}$ by first splitting the message into $\mathsf{poly}(\kappa)$ blocks, and then generating $\mathsf{poly}(\kappa)$ independent query secret keys. Moreover, the distinct secret keys for a receiver's query q_i are generated for queries $(q_i \| 1, \ldots, q_i \| \mathsf{poly}(\kappa))$.[11] In the outsourced setting the receiver must also retrieve a tag that is handed to the server in order to identify the appropriate record. Thus, the receiver learns an additional trapdoor defined by a query secret key for query $q_i \| 0$. Informally, security follows by a standard hybrid argument moving from a single challenge to multiple challenges. In the concrete PRF instantiation above we can invoke the PRF multiple times, generating secret keys by $F_k(i, j)$ where j is the block's index.

5.2 Our Protocol

Formally, let $(\mathsf{MGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Sim})$ be an equivocal encryption scheme with multiple secret keys for message space $\{0,1\}^\ell$ (see Section 5.1), and denote by $\mathcal{F}_{\mathrm{OQUERY}}(\mathrm{MSK}, q) \mapsto (-, \mathrm{SK}_q)$ the oblivious query functionality that obliviously implements algorithm Gen such that $\mathrm{MSK} \leftarrow \mathsf{MGen}(1^\kappa)$. Namely, the receiver privately learns the secret key that corresponds to its query $q \in Q_n$. Nevertheless, it might be the case that the database record space is much larger than $\{0,1\}^\ell$, requiring that the receiver learns multiple trapdoors to extract the entire record. We assume that the parties receive as an auxiliary information a parameter ℓ'_n which indicates the largest respone T_q when ranging over all databases of size n and all $q \in Q_n$, and fix $\tau = \lceil \ell'_n / \ell \rceil$. We thus consider an extended functionality for multiple query secret keys that is defined by $\mathcal{F}_{\mathrm{OMQUERY}}((\mathrm{MSK}, \tau), (q, \tau)) \mapsto (-, (\mathrm{SK}_{q^1}, \ldots, \mathrm{SK}_{q^\tau}))$ for obliviously learning multiple secret key queries. Our protocol is presented in the $\mathcal{F}_{\mathrm{OMQUERY}}$-hybrid model, where a trusted party realizes this functionality; efficient two rounds implementations can be designed based on particular instantiations (see the prior section for one example). We denote by $q^j = q || j$; the details of our protocol follow.

[11] This follows our assumption that a family of queries is represented by $\{0,1\}^m$; alternative formulations can be also considered.

Protocol 1 (Protocols $\pi = (\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$ for computing $\mathcal{F}_{\text{ODBS}}$.)

Protocol π_{Pre}: *Let 1^κ and $T = \{q_i, T_i\}_{i \in n'}$ be the input of* Sen*. Let ℓ'_n be an auxiliary information.*

1. Sen *first picks a random key* $\text{MSK} \leftarrow \mathsf{MGen}(1^\kappa)$ *for the encryption scheme with multiple secret keys.*
2. *For all $i \in n'$,* Sen *generates a pair (t_i, c_i) as follows:*
   (a) *Let $\tau = \lceil \ell'_n / \ell \rceil$ and let $\rho_i = \mathsf{Tag} \| \text{SK}_{q_i^1} \| \ldots \| \text{SK}_{q_i^\tau}$, where $\mathsf{Tag} \leftarrow \mathsf{Gen}(\text{MSK}, q_i \| 0)$ and $\text{SK}_{q_i^j} \leftarrow \mathsf{Gen}(\text{MSK}, q_i \| j)$ for independent invocations of* Gen*, all $j \in [\tau]$, and $\|$ for denoting concatenation.*
   (b) *Fix $t_i = \mathsf{Tag}$. Next, for every $j \in [\tau]$, encrypt the jth block $T_i[j]$ of T_i under secret key $\text{SK}_{q_i^j}$*[12]*. Let $c_i = c_i^1 \ldots c_i^\tau$ where $c_i^j = \mathsf{Enc}_{\text{SK}_{q_i^j}}(T_i[j])$.*
3. *Finally,* Sen *picks a permutation ω over n' and sends* Ser *the vector $a(T) = \left\{ \left(t_{\omega(i)}, c_{\omega(i)} \right) \right\}_{i=1}^{n'}$.*

Protocol π_{Query}: *Let $q \in Q_n$ be a query of the receiver* Rec*. Let ℓ'_n be an auxiliary information.*

1. Sen *and* Rec *make a call to functionality $\mathcal{F}_{\text{OMQUERY}}$ with inputs (MSK, τ) for* Sen *and (q, τ) for* Rec*. Let $\rho_q =$ denote the output obtained by* Rec *and denote by $\rho_q = \mathsf{Tag}_q \| \text{SK}_{q^1} \| \ldots \| \text{SK}_{q^\tau}$.*
2. Rec *sends Tag_q to* Ser*.*
3. *For every entry (t_i, c_i) in $a(T)$,* Ser *compares t_i with Tag_q. If there is a match for some index k,* Ser *sends c_k to* Rec*. Otherwise,* Ser *sends* "no match".
4. *Upon receiving a database entry $c_k = c^1 \ldots c^\tau$ from* Ser*,* Rec *does the following:*
   (a) Rec *uses SK_{q^j} to decrypt c^j for all $j \in [\tau]$. Let $T_k[j] \leftarrow \mathsf{Dec}_{\text{SK}_{q^j}}(c^j)$.*
   (b) Rec *outputs $T_k[1], \ldots, T_k[\tau]$. The sender and the server output an empty string.*

We prove that Protocol 1 is secure in the presence of semi-honest (Ser/Rec)-collusion.

Theorem 5.3 *Protocol 1 securely implements $\mathcal{F}_{\text{ODBS}}$ with respect to queries $Q = \{Q_n\}_n$ in the presence of (Ser/Rec)-collusion and semi-honest adversaries, in the private channels setting. Moreover, let $cc^n_{\mathsf{Pre}}(\kappa)$ denote the overall communication complexity within π_{Pre} and $cc^{n,q}_{\mathsf{Query}}(\kappa)$ denote the overall communication complexity within π_{Query} on a query q, then it holds that for every $q \in Q_n$, $cc^n_{\mathsf{Pre}}(\kappa) \in O(n' \ell'_n)$ and $cc^{n,q}_{\mathsf{Query}}(\kappa) \in O(\ell'_n)$.*

Proof: Note first that the communication complexity of our protocol in the preprocessing phase is upper bounded by n' entries, each of size ℓ'_n. Furthermore, the communication complexity in the query phase relative to both interactions with the sender and the server is upper bounded by the maximum length of a query response, which is ℓ'_n. We now prove the security of Protocol 1 by separately considering each corruption case. We denote by notation T_q the corresponding response of the server to query q. That is, T_q is either the record T_i or the message "no-match". In addition, Tag_q denotes the tag that is associated with query q and ρ_q denotes the receiver's output with respect to functionality $\mathcal{F}_{\text{OMQUERY}}$ when entering query q.

The sender is corrupted. This corruption case is simple to prove since the sender receives no messages within π_{Pre} and π_{Query}. It is therefore immediate that a semi-honest adversary $\mathcal{A}_{\mathtt{Sen}}$ controlling Sen learns no information about the receiver's queries within the executions of $(\pi_{\mathsf{Pre}}, \pi_{\mathsf{Query}})$.

[12] If T_i contains less than τ blocks, we use any standard padding technique to obtain τ blocks of length ℓ.

The server is corrupted. Let $\mathcal{A}_{\mathtt{Ser}}$ be a PPT real adversary that controls the server. We show the existence of a PPT ideal adversary $\mathcal{SIM}_{\mathtt{Ser}}$ such that the real execution of π with $\mathcal{A}_{\mathtt{Ser}}$ is computationally indistinguishable from an ideal execution with $\mathcal{F}_{\text{ODBS}}$ and $\mathcal{SIM}_{\mathtt{Ser}}$. Construction 1 describes the ideal adversary $\mathcal{SIM}_{\mathtt{Ser}}$. For the sake of generality, assume that $\mathcal{A}_{\mathtt{Ser}}$ outputs its entire view at the end of the protocol execution.

Construction 1 (Ideal adversary $\mathcal{SIM}_{\mathtt{Ser}}$)

Input: $1^\kappa, z$.

Initialization:

- *$\mathcal{SIM}_{\mathtt{Ser}}$ invokes $\mathcal{A}_{\mathtt{Ser}}$ on input $1^\kappa, z$.*

Simulating π_{Pre}:

1. *Upon receiving a message* $(\mathsf{preprocess}, |T|, m)$ *from $\mathcal{F}_{\text{ODBS}}$, fix n' as the number of records in the restructured database and $\tau = \lceil \ell'_n/\ell \rceil$.*[13] *Then define a database T^0 with n' records $(T^0_1, \ldots, T^0_{n'})$ of length ℓ'_n each, such that $T^0_i = 0^{\ell'_n}$ for all i.*
2. *$\mathcal{SIM}_{\mathtt{Ser}}$ picks a master secret key* $\mathsf{MSK} \leftarrow \mathsf{MGen}(1^\kappa)$.
3. *For every index $i \in [n']$, $\mathcal{SIM}_{\mathtt{Ser}}$ generates a pair (t_i, c_i) as follows:*
   (a) *$\mathcal{SIM}_{\mathtt{Ser}}$ picks a new arbitrary query $q_i \in Q_n$ (that was not picked thus far).*
   (b) *Let* $\rho_i = \mathsf{Tag} \| \mathrm{SK}_{q_i^1} \| \ldots \| \mathrm{SK}_{q_i^\tau}$, *where* $\mathsf{Tag} \leftarrow \mathsf{Gen}(\mathrm{MSK}, q_i \| 0)$ *and* $\mathrm{SK}_{q_i^j} \leftarrow \mathsf{Gen}(\mathrm{MSK}, q_i \| j)$ *for independent invocations of* Gen, *such that $j \in [\tau]$ and $\|$ denote concatenation.*
   (c) *Fix* $t_i = \mathsf{Tag}$.
   (d) *For every $j \in [\tau]$, encrypt the jth block $T^0_i[j]$ of T^0_i under secret key $\mathrm{SK}_{q_i^j}$. Let $c_i = c_i^1 \ldots c_i^\tau$ where* $c_i^j = \mathsf{Enc}_{\mathrm{SK}_{q_i^j}}(T^0_i[j])$.
4. *Finally, $\mathcal{SIM}_{\mathtt{Ser}}$ sends $\mathcal{A}_{\mathtt{Ser}}$ the vector $a(T^0) = \{(t_i, c_i)\}_{i=1}^{n'}$ on behalf of* Sen.

Simulating π_{Query}:

- *Upon receiving a message* $(\mathsf{response}, \mathtt{Rec}, |T_i|, \mathsf{id})$ *from $\mathcal{F}_{\text{ODBS}}$, and $|T_i|$ denotes a* no match *message, $\mathcal{SIM}_{\mathtt{Ser}}$ picks a new query $q \in Q_n$ that was not used in the simulation of protocol π_{Pre} and hands $\mathcal{A}_{\mathtt{Ser}}$ the tag* $\mathsf{Tag} \leftarrow \mathsf{Gen}(\mathrm{MSK}, q \| 0)$.[14]
  Otherwise, $\mathcal{SIM}_{\mathtt{Ser}}$ chooses a new random query q_i from the set of queries used in the simulation of protocol π_{Pre} and sends t_i to $\mathcal{A}_{\mathtt{Ser}}$ on behalf of Rec.

Output: *$\mathcal{SIM}_{\mathtt{Ser}}$ outputs whatever $\mathcal{A}_{\mathtt{Ser}}$ does.*

We now prove the following claim.

Claim 5.4 *For any tuple of inputs $(T, (q_1, \ldots, q_t))$ and auxiliary input z,*

$$\{\mathbf{IDEAL}_{\mathcal{F}_{\text{ODBS}}, \mathcal{SIM}_{\mathtt{Ser}}(z)}(\kappa, (-, T, (q_1, \ldots, q_t)))\}_{\kappa \in \mathbb{N}} \overset{\text{c}}{\approx} \{\mathbf{HYBRID}_{\pi^{\mathcal{F}_{\text{OMQUERY}}}, \mathcal{A}_{\mathtt{Ser}}(z)}(\kappa, (-, T, (q_1, \ldots, q_t)))\}_{\kappa \in \mathbb{N}}.$$

[13] Recall that ℓ'_n is a worst case upper bound on the record size that depends on the particular functionality and the database size, and that we assume that the parties receive it as an auxiliary information.

[14] We assume that given $|T_i|$ it is possible to tell whether query q_i is answered by the *no match* message or not.

Proof: We assume that the distribution $\mathbf{REAL}_{\pi,\mathcal{A}_{\mathtt{Ser}}(z)}(\kappa,(-,T,(q_1,\ldots,q_t)))$ contains the internal view of $\mathcal{A}_{\mathtt{Ser}}$ together with the input/output of $\mathtt{Rec}$ in an execution of $\pi = (\pi_{\mathsf{Pre}},\pi_{\mathtt{Rec}})$, whereas distribution $\mathbf{IDEAL}_{\mathcal{F}_{\mathrm{ODBS}},\mathcal{SIM}_{\mathtt{Ser}}(z)}(\kappa,(-,T,(q_1,\ldots,q_t)))$ contains the output of $\mathcal{SIM}_{\mathtt{Ser}}$ and the input/output of $\mathtt{Rec}$ in the ideal execution with $\mathcal{F}_{\mathrm{ODBS}}$.

More concretely, we prove that the following ensembles are computationally indistinguishable,

$$\left(a(T)=\left\{\left(t_{\omega(i)},c_{\omega(i)}\right)\right\}_{i=1}^{n'},\left(\mathsf{Tag}_{q_1},\ldots,\mathsf{Tag}_{q_t}\right),\left((q_1,\widetilde{T}_{q_1}),\ldots,(q_t,\widetilde{T}_{q_t})\right)\right)$$

and

$$\left(a(T^0)=\left\{\left(t_i^0,c_i^0\right)\right\}_{i=1}^{n'},\left(\mathsf{Tag}_{\tilde{q}_1^0},\ldots,\mathsf{Tag}_{\tilde{q}_t^0}\right),\left((q_1,T_{q_1})\ldots,(q_t,T_{q_t})\right)\right)$$

where $\tilde{q}_1^0,\ldots,\tilde{q}_t^0$ are the queries picked by the simulator during protocol π_{Query} and $\widetilde{T}_{q_1},\ldots\widetilde{T}_{q_t}$ are the database records the receiver learns in the hybrid execution.

We first claim that the honest real receiver learns the exact same records from database T as in the ideal execution, with overwhelming probability. This follows from the correctness of the encryption scheme used to encrypt the text and the fact that the real receiver learns the correct trapdoor for each input query q. We further note that the positions within the preprocessed text in which the server finds a match, are identically distributed in both executions. This is because in the real execution the sender randomly permutes its records using a random permutation ω. Moreover, in the simulation, the simulator picks the "matched" queries at random which implies the same distribution on the positions.

Next, note that the difference between the two ensembles is with respect to the preprocessed simulated database and the tags. Namely, $a(T^0)$ is a fake database that only contains zero entries, as opposed to the real database T that is input by the sender in the real execution. We further recall that $a(T^0)$ is comprised of tags/encrypted records pairs, associated with arbitrary queries. We claim that a corrupted server cannot detect this change and cannot link between a ciphertext to the query associated with it. Informally, this is due to the fact that by the security definition of our encryption scheme, the query secret keys do not leak any information about the query. Also, two ciphertexts encrypting different records under two different secret key queries are indistinguishable as well. These arguments follow from the text in Section 5.1.

More formally, the proof boils down to proving that the following preprocessed database given to the real server and real tags

$$\{\mathrm{SK}_{q_{\omega(i)}\|0},\mathsf{Enc}_{\mathrm{SK}_{q_{\omega(i)}\|1}}(T_{q_{\omega(i)}}),\ldots,\mathsf{Enc}_{\mathrm{SK}_{q_{\omega(i)}\|\tau}}(T_{q_{\omega(i)}})\}_{i\in n'},\left(\mathrm{SK}_{q_1\|0},\ldots,\mathrm{SK}_{q_t\|0}\right)$$

are indistinguishable from the sets

$$\{\mathrm{SK}_{q_i^0\|0},\mathsf{Enc}_{\mathrm{SK}_{q_i^0\|1}}(T_i^0),\ldots,\mathsf{Enc}_{\mathrm{SK}_{q_i^0\|\tau}}(T_i^0)\}_{i\in n'},\left(\mathrm{SK}_{\tilde{q}_1^0\|0},\ldots,\mathrm{SK}_{\tilde{q}_t^0\|0}\right)$$

given to a corrupted server in the ideal execution with respect to a fake database T^0 and an arbitrary set of queries $\{q_i^0\}_{i\in n'}$. In fact, indistinguishability holds even if the adversary is also given both the real and the simulated set of queries associated with each set. Specifically, $\mathrm{SK}_{q_{\omega(i)}\|0}$ and $\mathrm{SK}_{q_i^0\|0}$ are indistinguishable due to the query hiding property, whereas, the ciphertexts are indistinguishable due to IND-CPA and query hiding. Security follows by a sequence of standard hybrid arguments by replacing the secret keys associated with q_i^0 with the secret keys associated with q_i and then replacing T^0 with T. ■

The receiver is corrupted. Let $\mathcal{A}_{\mathtt{Rec}}$ be a PPT semi-honest adversary controlling $\mathtt{Rec}$. We show the existence of a PPT ideal adversary $\mathcal{SIM}_{\mathtt{Rec}}$ such that for any tuple of inputs $(T,(q_1,\ldots,q_t))$ and auxiliary input z, the hybrid execution of π with $\mathcal{A}_{\mathtt{Rec}}$ is indistinguishable from an ideal execution with $\mathcal{F}_{\mathrm{ODBS}}$ and $\mathcal{SIM}_{\mathtt{Rec}}$. Construction 2 describes the ideal adversary $\mathcal{SIM}_{\mathtt{Rec}}$. For the sake of generality, assume that $\mathcal{A}_{\mathtt{Rec}}$ outputs its entire internal view at the end of the protocol execution.

Construction 2 (Ideal adversary $\mathcal{SIM}_{\mathtt{Rec}}$)

Input: $1^\kappa, z$.

Initialize: *$\mathcal{SIM}_{\mathtt{Rec}}$ invokes $\mathcal{A}_{\mathtt{Rec}}$ on input $1^\kappa, z$.*

Simulating π_{Pre}:

- *Upon receiving a message* $(\mathsf{preprocess}, |T|, m)$ *from* $\mathcal{F}_{\text{ODBS}}$*, fix* n' *as the number of records in the database and let* $\tau = \lceil \ell'_n/\ell \rceil$*.. $\mathcal{SIM}_{\mathtt{Rec}}$ further picks a master secret key* $\text{MSK} \leftarrow \mathsf{MGen}(1^\kappa)$.

Simulating π_{Query} on input q (obtained by employing the input selection algorithm M):

1. *$\mathcal{SIM}_{\mathtt{Rec}}$ sends a message* (query, q) *to* $\mathcal{F}_{\text{ODBS}}$.
2. *Upon receiving a message* $(\mathsf{response}, q, T_q, \mathsf{id})$ *from the ideal functionality* $\mathcal{F}_{\text{ODBS}}$*, $\mathcal{SIM}_{\mathtt{Rec}}$ does the following:*
   (a) *It invokes $\mathcal{A}_{\mathtt{Rec}}$ and obtains q as the input to* $\mathcal{F}_{\text{OMQUERY}}$.
   (b) *It emulates* $\mathcal{F}_{\text{OMQUERY}}$ *by sending* $\rho_q = \mathsf{Tag} \| \text{SK}_{q^1} \| \ldots \| \text{SK}_{q^\tau}$*, where* $\mathsf{Tag} \leftarrow \mathsf{Gen}(\text{MSK}, q \| 0)$ *and* $\text{SK}_{q^j} \leftarrow \mathsf{Gen}(\text{MSK}, q \| j)$ *for all* $j \in [\tau]$.
   (c) *If T_q denotes a* no match *message, then upon receiving* Tag *from $\mathcal{A}_{\mathtt{Rec}}$, $\mathcal{SIM}_{\mathtt{Rec}}$ forwards the adversary a* "no match" *message on behalf of the server.*
   Otherwise, $\mathcal{SIM}_{\mathtt{Rec}}$ uses $\text{SK}_{q^1}, \ldots, \text{SK}_{q^\tau}$ *to generate ciphertexts* $c_1, \ldots, c_\tau$ *for encrypting T_q as would have done by the honest sender. $\mathcal{SIM}_{\mathtt{Rec}}$ then forwards the adversary ciphertexts* $c_1, \ldots, c_\tau$ *on behalf of the server.*

Output: *$\mathcal{SIM}_{\mathtt{Rec}}$ outputs whatever $\mathcal{A}_{\mathtt{Rec}}$ does.*

We now prove the following claim:

Claim 5.5 *For any tuple of inputs* $(T, (q_1, \ldots, q_t))$ *and auxiliary input* z*,*

$$\begin{aligned} &\{\mathbf{IDEAL}_{\mathcal{F}_{\text{ODBS}}, \mathcal{SIM}_{\mathtt{Rec}}(z)}(\kappa, (-, T, (q_1, \ldots, q_t)))\}_{\kappa \in \mathbb{N}} \\ &\equiv \quad \{\mathbf{HYBRID}_{\pi^{\mathcal{F}_{\text{OMQUERY}}}, \mathcal{A}_{\mathtt{Rec}}(z)}(\kappa, (-, T, (q_1, \ldots, q_t)))\}_{\kappa \in \mathbb{N}}. \end{aligned}$$

Proof: The proof is straightforward. Namely, it is easy to observe that $\mathcal{SIM}_{\mathtt{Rec}}$ perfectly simulates the honest players in π. It first picks a random master secret key MSK and perfectly simulates $\mathcal{F}_{\text{OMQUERY}}$ by sending ρ_q for every real query q. It then generates the receiver's response exactly as done by the honest sender. Hence, the simulated view of $\mathcal{A}_{\mathtt{Rec}}$ in the ideal execution of is identically distributed to the adversary's view in a hybrid execution of π. ■

The server and the receiver are corrupted. Let $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ be a PPT semi-honest adversary controlling both Ser and Rec. We show the existence of a PPT ideal adversary $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ such that for any tuple of inputs $(T, (q_1, \ldots, q_t))$ and auxiliary input z, the hybrid execution of π with $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ is indistinguishable from an ideal execution with $\mathcal{F}_{\text{ODBS}}$ and $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$. Construction 3 describes the ideal adversary $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$. For the sake of generality, assume that $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ outputs its entire internal view at the end of the protocol execution. Note that the difficulty in this case is that $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ has to simulate a set-up message for $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ without knowing the actual database T. Then, once given a pair of query/record (q, T_q), it should be able to simulate sender messages that are consistent both with the simulated message and with (q, T_q). To do this, we use the equivocation property of our encryption scheme. That is, $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ uses Sim_1 to generate fake ciphertexts in the simulation of π_{Pre} and then uses Sim_2 in the simulation of π_{Query} to obtain secret keys that are consistent with the actual query/record (q, T_q). This idea is formalized in Construction 3.

Construction 3 (Ideal adversary $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$)

Input: $1^\kappa, z$.

Initialize: $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *invokes* $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ *on input* $1^\kappa, z$.

Simulating π_{Pre}:

1. *Upon receiving a message* $(\mathsf{preprocess}, |T|, m)$ *from* $\mathcal{F}_{\text{ODBS}}$*, fix* n' *as the number of records in the restructured database and let* $\tau = \lceil \ell'_n/\ell \rceil$.
2. $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *picks a master secret key* $\mathsf{MSK} \leftarrow \mathsf{MGen}(1^\kappa)$.
3. *For all* $i \in [n']$, $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *generates a pair* (t_i, c_i) *as follows:*
   (a) $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *picks a new random query* $q_i \in Q_n$ *(that was not picked thus far).*
   (b) *Let* $\mathsf{Tag} \leftarrow \mathsf{Gen}(\mathsf{MSK}, q\|0)$ *and* $(c_i^j, \mathsf{td}_i^j) \leftarrow \mathsf{Sim}_1(1^\kappa)$ *for independent invocations of* Sim_1 *and all* $j \in [\tau]$.
   (c) $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *fixes* $t_i = \mathsf{Tag}$ *and* $c_i = c_i^1 \ldots c_i^\tau$.
4. *Finally,* $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *sends* $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ *the vector* $a(T^0) = \{(t_i, c_i)\}_{i=1}^{n'}$ *on behalf of* $\mathtt{Sen}$.

Simulating π_{Query} on input q:

1. $\mathcal{SIM}_{\mathtt{Rec},ser}$ *sends a message* (query, q) *to* $\mathcal{F}_{\text{ODBS}}$.
2. *Upon receiving a message* $(\mathsf{response}, q, T_q, \mathsf{id})$ *from* $\mathcal{F}_{\text{ODBS}}$*,* $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *does the following:*
   (a) *It invokes* $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ *and obtains* q *as the input to* $\mathcal{F}_{\text{OMQUERY}}$.
   (b) *If* T_q *is a* no match *message* $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *picks a new query* q' *that was not used in the simulation of protocol* π_{Pre}*, and emulates* $\mathcal{F}_{\text{OMQUERY}}$ *by sending* $\rho_{q'} = \mathsf{Tag}\|\mathrm{SK}_{q'^1}\| \ldots \|\mathrm{SK}_{q'^\tau}$ *to* $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$*, where* $\rho_{q'}$ *is computed as if* $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$*'s input to* $\mathcal{F}_{\text{OMQUERY}}$ *is* q'.
   (c) *Otherwise,* $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *picks a new query* q' *from the set of queries used in the simulation of protocol* π_{Pre} *(say* q' *is associated with the* i*th record* $(t_i, c_i = c_i^1, \ldots, c_i^\tau)$ *in the fake database), and uses* Sim_2 *to equivocate* c_i *as an encryption of* T_q*. Namely, for every* $j \in [\tau]$ *computes* $\mathrm{SK}_j \leftarrow \mathsf{Sim}_2(c_i^j, \mathsf{td}_i^j, T_q[j])$*. Let* $\rho_{q'} = t_i\|\mathrm{SK}_1\| \ldots \|\mathrm{SK}_\tau$.
   $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *emulates* $\mathcal{F}_{\text{OMQUERY}}$ *by sending* $\rho_{q'}$ *to* $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$.

Output: $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ *outputs whatever* $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ *does.*

We now prove the following claim:

Claim 5.6 *For any tuple of inputs* $(T, (q_1, \ldots, q_t))$ *and auxiliary input* z*,*

$$\begin{aligned}&\{\mathbf{IDEAL}_{\mathcal{F}_{\text{ODBS}},\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}(z)}(\kappa, (-, T, (q_1, \ldots, q_t)))\}_{\kappa\in\mathbb{N}}\\ &\overset{\text{c}}{\approx}\ \{\mathbf{HYBRID}_{\pi^{\mathcal{F}_{\text{OMQUERY}}},\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}(z)}(\kappa, (-, T, (q_1, \ldots, q_t)))\}_{\kappa\in\mathbb{N}}.\end{aligned}$$

Proof: We assume that the distribution $\mathbf{HYBRID}_{\pi^{\mathcal{F}_{\text{OMQUERY}}},\mathcal{A}_{\mathtt{Sen},\mathtt{Rec}}(z)}(\kappa, (-, T, (q_1, \ldots, q_t)))$ contains the internal view of real adversary $\mathcal{A}_{\mathtt{Ser},\mathtt{Rec}}$ that controls both the server and the receiver, whereas the distribution $\mathbf{IDEAL}_{\mathcal{F}_{\text{ODBS}},\mathcal{SIM}_{\mathtt{Sen},\mathtt{Rec}}(z)}(\kappa, (-, T, (q_1, \ldots, q_t)))$ contains the output of $\mathcal{SIM}_{\mathtt{Ser},\mathtt{Rec}}$ which simulates the view of the server and the receiver in the ideal execution with $\mathcal{F}_{\text{ODBS}}$.

More concretely, we prove that the following ensembles are computationally indistinguishable,

$$\left(a(T) = \left\{\left(t_{\omega(i)}, c_{\omega(i)}\right)\right\}_{i=1}^{n'}, (\rho_{q_1}, \ldots, \rho_{q_t}), \left((q_1, \widetilde{T}_{q_1}), \ldots, (q_t, \widetilde{T}_{q_t})\right)\right)$$

and

$$\left(a(T^0) = \left\{(t_i^0, c_i^0)\right\}_{i=1}^{n'}, \left(\rho_{\tilde{q}_1^0}, \ldots, \rho_{\tilde{q}_t^0}\right), ((q_1, T_{q_1}) \ldots, (q_t, T_{q_t}))\right)$$

where $\tilde{q}_1^0, \ldots, \tilde{q}_t^0$ are the queries picked by the simulator during the simulation of protocol π_{Query} and $\widetilde{T}_{q_1}, \ldots \widetilde{T}_{q_t}$ are the database records the receiver learns in the hybrid execution.

Note first that the positions within the preprocessed text in which the server finds a match, are identically distributed in both executions. This is because in the hybrid execution the sender randomly permutes its records using a random permutation ω. Moreover, in the simulation, the simulator picks the queries for the fake database at random.

Next, note that the difference between the two ensembles is with respect to the preprocessed simulated database, and the receiver's outputs with respect to functionality $\mathcal{F}_{\text{OMQUERY}}$. Namely, $a(T^0)$ is a fake database that only contains simulated ciphertexts, as opposed to the real database T that is input by the sender in the hybrid execution. We further recall that $a(T^0)$ is comprised of tags/encrypted records pairs, associated with arbitrary queries. Finally, note that the receiver's output with respect to functionality $\mathcal{F}_{\text{OMQUERY}}$ distributes differently in the simulation since the simulator picks arbitrary queries to compute the trapdoors of the receiver and the server.

Similarly to the case that the server is corrupted, we claim that a corrupted server and receiver cannot detect this change, and cannot link between a ciphertext (even a fake one) to the query associated with it. Informally, this is due to the fact that by the security definition of our encryption scheme, the query secret keys do not leak any information about the query. Also, a real ciphertext is indistinguishable from a fake one, even in the presence of the corresponding secret key. These arguments follow directly from Section 5.1.

More formally, the proof boils down to proving that the following preprocessed database and trapdoors given to the real server and receiver

$$\{\mathrm{SK}_{q_{\omega(i)}\|0}, \mathsf{Enc}_{\mathrm{SK}_{q_{\omega(i)}\|1}}(T_{q_{\omega(i)}}[1]), \ldots, \mathsf{Enc}_{\mathrm{SK}_{q_{\omega(i)}\|\tau}}(T_{q_{\omega(i)}}[\tau])\}_{i\in n'},$$
$$\left(\mathrm{SK}_{q_1\|0}, \mathrm{SK}_{q_1\|1}, \ldots, \mathrm{SK}_{q_1\|\tau}\right), \ldots, \left(\mathrm{SK}_{q_t\|0}, \mathrm{SK}_{q_t\|1}, \ldots, \mathrm{SK}_{q_t\|\tau}\right)$$

are indistinguishable from the set

$$\{\mathrm{SK}_{q_i^0\|0}, c_i^1, \ldots, c_i^\tau\}_{i\in n'}, \left(\mathrm{SK}_{\tilde{q}_1\|0}, \mathrm{SK}_{\tilde{q}_1\|1}, \ldots, \mathrm{SK}_{\tilde{q}_1\|\tau}\right), \ldots, \left(\mathrm{SK}_{\tilde{q}_t\|0}, \mathrm{SK}_{\tilde{q}_t\|1}, \ldots, \mathrm{SK}_{\tilde{q}_t\|\tau}\right)$$

given to a corrupted server in the ideal execution with respect to a fake database T^0 and a set of queries $\{q_i^0\}_{i\in n'}$ that the simulator uses in the simulation of π_{Query} to produce the trapdoors for the receiver.

We define a sequence of hybrid games $\{\mathbf{GAME}_j\}_{j\in n'}$ with simulator $\mathsf{Sim}^j_{\mathtt{Ser},\mathtt{Rec}}$ for each such game as follows. In game $\mathbf{GAME}^j_{\mathtt{Ser},\mathtt{Rec}}$, given database T, simulator Sim_j preprocesses the first j records as would have done by the real sender, whereas the remaining records are preprocessed as would have done by $\mathsf{Sim}_{\mathtt{Ser},\mathtt{Rec}}$. Namely, for the first j records $\{(q_1, T_1), \ldots, (q_j, T_j)\}_j$, $\mathsf{Sim}^j_{\mathtt{Ser},\mathtt{Rec}}$ uses their real values to generate the query secret keys and ciphertexts, whereas for the rest of the records it invokes algorithm Sim_1 to generate the fake ciphertexts. $\mathsf{Sim}^j_{\mathtt{Ser},\mathtt{Rec}}$ then permutes the preprocessed database and hands it to the corrupted server. Next, in the query phase, the simulator produces the appropriate response to the corrupted receiver. That is, if the receiver inputs $q \in \{q_1, \ldots, q_j\}$ the simulator hands the receiver the response as was generated in a hybrid execution. Otherwise, it invokes algorithm Sim_2 and continues as $\mathsf{Sim}_{\mathtt{Ser},\mathtt{Rec}}$ would have done. Clearly, the difference between every two subsequent games is computationally indistinguishable. This is due to the fact that tags $\mathrm{SK}_{q_{j+1}\|0}$ and $\mathrm{SK}_{q^0_{j+1}\|0}$ are indistinguishable due to query hiding. Furthermore, the ciphertexts associated with these queries are indistinguishable due to the equivocation property (which holds even in the presence of the secret keys).

Note that the view produced in game $\mathbf{GAME}_{n'}$ is almost as in the hybrid execution except that we still handle not matched queries as in the simulation with $\mathsf{Sim}_{\mathtt{Ser},\mathtt{Rec}}$. We claim that the views in the game and in the hybrid execution are computationally indistinguishable due to the query hiding of our encryption scheme. Namely, the adversary cannot conclude any information about the (not found) query by observing its secret key. This concludes the proof. ■

This concludes the proof of Theorem 5.3. ■

Summary. To summarize, we recall that our lower bound is related to the parameter $H_{n,Q}$ while our upper bound is related to ℓ'_n. We emphasize that while for some functionalities, these two parameters coincide, they are not equal in general. Specifically, $H_{n,Q}$ denotes the maximal amount of information within a response of some query when ranging over all databases of size n, whereas ℓ'_n implies the largest record length when ranging over all databases of the same size. For functionalities where $H_{n,Q} = O(\ell'_n)$, the communication complexity of our construction meets the lower bound from Section 4. Intuitively, this follows when there exists a query with exponentially many potential answers of size $O(\ell'_n)$, as for OT with adaptive queries and keyword search. Note that for functionalities with a fixed upper bound on the record size as above, the communication bounds introduced by Protocol 1 are optimal since for ℓ'_n the upper bound on the size of each record and n' the number of records, $n'\ell'_n = n$.

References

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP (1)*, pages 152–163, 2010.

[AJLA+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *EUROCRYPT*, pages 483–501, 2012.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In *CRYPTO*, pages 111–131, 2011.

[Can00] Ran Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13:143–202, 2000.

[CGKO11] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[CKKC13] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Carlos Cid. Multi-client non-interactive verifiable computation. In *TCC*, pages 499–518, 2013.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *TCC*, pages 54–74, 2012.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[FHV13] Sebastian Faust, Carmit Hazay, and Daniele Venturi. Outsourced pattern matching. In *ICALP*, 2013.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, pages 303–324, 2005.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.

[GH11] Matthew Green and Susan Hohenberger. Practical adaptive oblivious transfer from simple assumptions. In *TCC*, pages 347–363, 2011.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier cs-proofs. *IACR Cryptology ePrint Archive*, 2011:456, 2011.

[HT10] Carmit Hazay and Tomas Toft. Computationally secure pattern matching in the presence of malicious adversaries. In *ASIACRYPT*, pages 195–212, 2010.

[JJK+13] Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Outsourced symmetric private information retrieval. In *ACM Conference on Computer and Communications Security*, pages 875–888, 2013.

[KMR11] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. *IACR Cryptology ePrint Archive*, 2011:272, 2011.

[KMR12] Seny Kamara, Payman Mohassel, and Ben Riva. Salus: a system for server-aided secure function evaluation. In *ACM Conference on Computer and Communications Security*, pages 797–808, 2012.

[KP13] Seny Kamara and Charalampos Papamanthou. Parallel and dynamic searchable symmetric encryption. In *Financial Cryptography*, pages 258–274, 2013.

[KPR12] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *ACM Conference on Computer and Communications Security*, pages 965–976, 2012.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.

[Moh11] Payman Mohassel. Efficient and secure delegation of linear algebra. *IACR Cryptology ePrint Archive*, 2011:605, 2011.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, pages 111–126, 2002.

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *CRYPTO*, pages 573–590, 1999.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, pages 458–467, 1997.

[PTT11] Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Optimal verification of operations on dynamic sets. In *CRYPTO*, pages 91–110, 2011.

[Wei73] Peter Weiner. Linear pattern matching algorithms. In *SWAT (FOCS)*, pages 1–11, 1973.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.