Keyword Search and Oblivious Pseudorandom Functions

Michael J. Freedman[1], Yuval Ishai[2], Benny Pinkas[3], and Omer Reingold[4]

[1] New York University (mfreed@cs.nyu.edu)
[2] Technion (yuvali@cs.technion.ac.il)
[3] HP Labs, Israel (benny.pinkas@hp.com)
[4] Weizmann Institute of Science (omer.reingold@weizmann.ac.il)

Abstract. We study the problem of privacy-preserving access to a database. Particularly, we consider the problem of privacy-preserving keyword search (KS), where records in the database are accessed according to their associated keywords and where we care for the privacy of both the client and the server. We provide efficient solutions for various settings of KS, based either on specific assumptions or on general primitives (mainly oblivious transfer). Our general solutions rely on a new connection between KS and the oblivious evaluation of pseudorandom functions (OPRFs). We therefore study both the definition and construction of OPRFs and, as a corollary, give improved constructions of OPRFs that may be of independent interest.

Keywords: Secure keyword search, oblivious pseudorandom functions, private information retrieval, secure two-party protocols, privacy-preserving protocols

1 Introduction

Keyword search (KS) is a fundamental database operation. It involves two main parties: a server, holding a database comprised of a set of records and their associated keywords, and a client, who may send queries consisting of keywords and receive the records associated with these keywords. A natural question in the area of secure computation is the design of protocols for efficient, privacy-preserving keyword search. These protocols enable keyword queries while providing privacy for *both* parties: namely, (1) hiding the queries from the database (client privacy) and (2) preventing the clients from learning anything but the results of the queries (server privacy).

To be more specific, the private keyword-search problem may be defined by the following functionality. The database consists of n pairs $\{(x_i, p_i)\}_{i\in[n]}$; we denote x_i as the keyword and p_i as the payload (database record). A query from a client is a searchword w, and the client obtains the result p_i if there is a value i for which $x_i = w$ and obtains a special symbol $\perp$ otherwise. Given that KS allows clients to input an arbitrary searchword, as opposed to selecting p_i by an input i, keyword search is strictly stronger than the better-studied problems of oblivious transfer (OT) and symmetrically private information retrieval (SPIR).

1.1 Contributions

Our applied and conceptual contributions can be divided into the following:

- **Specific protocols for KS.** We construct direct instantiations of KS protocols, providing privacy for both parties, based on the use of oblivious polynomial evaluation and homomorphic encryption. The protocols have a communication complexity which is logarithmic in the size of the domain of the keywords and polylogarithmic in the number of records, and they require only one round of interaction, even in the case of malicious clients.[1] All previous fully-private KS protocols either require a linear amount of communication or multiple rounds of interaction, even in the semi-honest model.
- **KS using Oblivious Pseudorandom Functions (OPRFs).** We describe a generic, yet very efficient, *reduction* from KS to what we call *semi-private* KS, in which only the client's privacy is maintained. Specifically, we show that any KS protocol providing (only) client privacy can be upgraded to provide server privacy as well, by using an additional *oblivious evaluation of pseudorandom functions.* This reduction is motivated by the fact that efficient semi-private KS is quite easy to obtain by combining PIR with a suitable data structure supporting keyword searches [20, 7].[2] Thus, we derive a general construction of fully-private KS protocols based on PIR, a data structure supporting keyword searches, and an OPRF.
- **New notion of OPRF.** Motivated by the KS application and the above general reduction, we put forward a new relaxed notion of OPRF which facilitates more efficient constructions and is of independent theoretical interest.
- **Constructions of OPRF.** We show a construction of an OPRF protocol based on the DDH assumption. In addition, one of the our main contributions is a general construction of (relaxed) OPRF from OT. This construction is based on techniques from [23, 25], yet improves on these works as (1) it preserves privacy against (up to t) *adaptive* queries, (2) it is obliviously evaluated in *constant number of rounds*, and (3) it handles *exponential domain size*. These improvements are partially relevant also in the context of t-out-of-n OT, as originally studied in [23, 25]. We note that this is a *black-box* construction of t-time OPRF from OT. From a theoretical point-of-view, one of the most interesting open questions left by our work is to find an efficient black-box construction of *fully*-adaptive OPRF and KS (supporting an arbitrary number of queries) which only makes a black-box use of OT. In contrast, such a construction is easy to obtain by making a non-black-box use of OT. Thus, we have a rare example of a non-black-box construction in cryptography for which no black-box construction is known, even in the

[1] In the case of malicious parties, we use a slightly relaxed notion of security, following one suggested in the context of OT [1, 23] (see Section 2).

[2] In fact, if we allow a setup phase with linear communication complexity, we can obtain a semi-private KS supporting adaptive queries by simply sending the entire database to the client.

random-oracle model. In fact, we are not aware of any other such simple and natural example that fully resides in the semi-honest model.

1.2 Related Work

The work of Kushilevitz and Ostrovsky [20], which was the first to suggest a single-server PIR protocol, described how to use PIR together with a hash function for obtaining a semi-private KS protocol (we denote a KS protocol as "semi-private" if it does not ensure server privacy). Chor et al. [7] described how to implement semi-private KS using PIR and any data structure supporting keyword queries, and they added server privacy using a trie data structure and many rounds. Our reduction from KS to semi-secure KS provides a more efficient and general alternative, requiring only a small constant number of rounds.

Ogata and Kurosawa [27] show an ad-hoc solution for KS for adaptive queries, using a setup stage with linear communication. The security of their main construction is based on the random oracle assumption and on a non-standard assumption (related to the security of blind signatures). The system requires a public-key operation per item for every new query.

A problem somewhat related to KS is that of "search on encrypted data" [30, 3]. The scenario involves giving encrypted data to a third party. This party is later given a trapdoor key, enabling it to search the encrypted data for specific keywords, while hiding any other information about the data. This problem seems easier than ours since the search key is provided by the party which previously encrypted the data. Furthermore, there are protocols for "search on encrypted data" (*e.g.*, [30]) which use only symmetric-key crypto. therefore it is unlikely that they can be used for implementing KS, as KS implies OT.

Another related problem is that of "secure set intersection" [10], where two parties whose inputs consist of sets X, Y privately compute $X \cap Y$. KS is a special case of this problem with $|X| = 1$. On the other hand, set intersection can be reduced to KS by running a KS invocation for every item in X. Thus, our results can be applied to obtain efficient solutions to the set-intersection problem.

Cryptographic primitives. We make use of several standard cryptographic primitives that can be defined as instances of private two-party computation between a server and a client, including oblivious transfer (OT) [29, 9], single-server private information retrieval (PIR) [8, 20], symmetrically-private information retrieval (SPIR) [11, 23], and oblivious polynomial evaluation (OPE) [23]. Some specific constructions for non-adaptive KS require a semantically-secure homomorphic encryption system.

1.3 Organization

The remainder of this paper is structured as follows. We provide definitions and variants of keyword search in Section 2. Section 3 describes some direct constructions of (non-adaptive) KS protocols based on OPE and homomorphic encryption. In Section 4 we introduce our new relaxed notion of OPRF and use it to obtain a reduction from fully-private KS to semi-private KS. We conclude by providing efficient implementations of OPRFs in Section 5.

2 Preliminaries

This section defines the private keyword search problem and some of its variants. We assume the reader's familiarity with standard simulation-based definitions of secure computation (cf. [5, 13]).

2.1 Private keyword search

The system is comprised of a server S and a client C. The server's input is a database X of n pairs (x_i, p_i), each consisting of a keyword and a payload. Keywords can be strings of an arbitrary length and payloads are padded to some fixed length. We may also assume, without loss of generality, that all x_i are distinct. The client's input is a *searchword* w. If there is a pair in the database in which the keyword is equal to the searchword, then the output is the corresponding payload. Otherwise the output is a special symbol $\perp$.

Private keyword search (KS for short) requires privacy for both the client and the server, *i.e.*, neither party learns anything more than is defined by the above transaction. The strongest way of formalizing this requirement is by appealing to general definitions of secure computation from the literature. That is, a KS protocol can be defined as a secure two-party protocol realizing the above KS functionality. However, when constructing *specific* KS protocols—rather than general reductions from KS to other primitives—efficiency considerations dictate a slight relaxation of this definition which still suffices to capture the core correctness and privacy requirements. Specifically, when simulating a malicious server, the relaxed definition only requires one to simulate the server's view alone, without considering its joint distribution with the honest client's output. (In the setting of semi-honest parties, this relaxed definition is equivalent to the original one.)

With respect to a malicious server, this relaxed definition only requires that the client's query w remains private: It does not require the server to commit to or even "know" a database to which a client's search is effectively applied. Such a relaxation is standard for related primitives such as OT (cf. [1, 23]) or PIR (cf. [20, 4]). Moreover, it seems necessary for obtaining protocols that require only a single round of interaction yet still achieve security against malicious parties. (We note, however, that our protocols can be amended to satisfy the stronger definition by adding proofs of knowledge.)

It is interesting to contrast the goals of KS and those of zero-knowledge sets [22]. While KS provides privacy for both parties but does not require the server to commit to its input, zero-knowledge sets require the server to commit to its input but provides privacy for the server yet not the client.

The requirements of a private KS protocol can be divided into *correctness*, *client privacy*, and *server privacy* components. We first define these properties independently, and then define a private KS protocol as a protocol that satisfies these definitions. (To avoid cumbersome notation, we omit the auxiliary inputs required for sequential composition.)

Definition 1. (Correctness.) *If both parties are honest, then, after running the protocol on inputs (X, w), the client outputs p_i such that $w = x_i$, or $\perp$ if no such i exists.*

Definition 2. (Client's privacy: indistinguishability.) *For any* PPT $\mathcal{S}'$ *executing the server's part and for any inputs X, w, w', the views that $\mathcal{S}'$ sees on input X, in the case that the client uses the searchword w and the case that it uses w', are computationally indistinguishable.*

For both client and server privacy, indistinguishability is parameterized by a privacy parameter k, given to both parties as a common input. Note that this definition, protecting only the privacy of the client's query w, captures the aforementioned relaxation.

In order to show that the client does not learn more or different information from the protocol than from merely obtaining its output, we compare the protocol to the *ideal implementation*. In the ideal implementation, a trusted third party gets the server's database X and the client's query w as input, and outputs the corresponding payload to the client. Privacy requires that the protocol does not leak to the client more information than in the ideal implementation.

Definition 3. (Server's privacy: comparison with the ideal model.) *For every* PPT *machine $\mathcal{C}'$ substituting the client in the real protocol, there exists a* PPT *machine $\mathcal{C}''$ that plays the client's role in the ideal implementation, such that on any inputs (X, w), the view of $\mathcal{C}'$ is computationally indistinguishable from the output of $\mathcal{C}''$. (In the semi-honest model $\mathcal{C}' = \mathcal{C}$.)*

Remark 1. The protocols from Section 3, as originally described, will actually satisfy the following incomparable notion of server privacy: any computationally *unbounded* client $\mathcal{C}'$ can be simulated by an *unbounded* simulator $\mathcal{C}''$. This can be viewed as a pure form of information-theoretic privacy. Inefficient simulation seems necessary in order to obtain 1-round KS protocols (see a discussion in [1] for the similar case of OT). However, it is easy to convert these protocols to ones that support efficient simulation, using standard zero-knowledge proofs of knowledge: Clients should prove that they know the secret key corresponding to the public key they generate. Such proofs need to be performed only once, during the system's initialization.

Definition 4. (Private KS protocol.) *A two-party protocol satisfying Definitions 1 (correctness), 2 (client privacy) and 3 (server privacy).*

The above definition can be immediately applied to protocols computing any deterministic client-server functionality f. We refer to such a protocol as *private* protocol for f.

Finally, we will later use KS protocols in which the server privacy is not preserved (*i.e.*, satisfy only Definitions 1 and 2). We refer to such protocols as *semi-private* KS protocols.

2.2 Problem Variants

The default KS primitive can be extended and generalized in several ways. We first outline three orthogonal variations on the basic model, and then define the two main settings on which we focus.

- **Multiple queries.** The default notion of KS allows the client to search for a single keyword. While this procedure can be repeated several times, one may seek more efficient solutions allowing the client to retrieve t keywords at a reduced cost. This generalized notion of t-time KS is straightforward to define and makes sense even when $t \gg n$, since the client does not necessarily have an a-priori knowledge of the keywords. (This is in contrast to the case of 1-out-of-n OT or SPIR, where there is no point in letting $t > n$, since the entire database can be learned using t queries.)
- **Allowing setup.** By default, KS does not assume any previous interaction between the client and server. To facilitate prompt responses to future queries, the client and server may engage in a setup phase involving a polynomial amount of work. During the online phase, each keyword search may then only require a sub-linear amount of work.
- **Adaptive queries.** In the default *non-adaptive* setting, the client may ask multiple queries, but the queries must be defined before it receives the server's first answer. In the *adaptive* setting, the client can decide on the value of each query after receiving the answers to previous queries. An adaptive t-time KS protocol allows the client to make at most t adaptive queries. The privacy definition in this case extends the above in a natural way, similarly to that of adaptive OT in [24].

The results of this work have applications to all of the above variations. However, to make the presentation more focused, we restrict our discussion to two "typical" settings for KS:

Non-adaptive t-time KS without setup. In our default notion of KS, when t is unspecified, it is taken to be 1. This setting's main goal in this setting is to obtain solutions whose total *communication complexity* is sub-linear in n. Thus, the problem can be viewed as an extension of PIR and SPIR.

Adaptive t-time KS with setup. In this setting, allowing t adaptive queries, the setup phase typically consists of a single message in which the server sends the database in encrypted form to the client. (This is the default setting also considered in [23, 25, 27].) In general, however, the setup may be polynomial in the database size. Ideally, each adaptive query should involve a small amount of work—sub-linear in the database size—including both communication and computation. When t is unspecified, it is taken to be an arbitrary polynomial in the database size, where this polynomial may be larger than the cost of the setup. Thus, one cannot apply solutions that separately handle each future query.

For brevity, we subsequently refer to these settings as *non-adaptive KS* and *adaptive KS*, respectively.

3 Non-Adaptive KS from OPE

In this section, we construct a non-adaptive keyword search protocol using oblivious polynomial evaluation (OPE) [23]. The basic idea of the construction is to encode the database entries in $X = \{(x_1, p_1), \ldots, (x_n, p_n)\}$ as values of a polynomial, *i.e.*, to define a polynomial Q such that $Q(x_i) = (p_i)$. Note that this design is different than previous applications of OPE, where a polynomial (of degree k) was used only as a source for $(k+1)$-wise independent values. Compared to our other constructions and to previous solutions from the literature, this construction is unique in achieving sub-linear communication overhead in a single round of communication.[3]

The following scheme uses any generic OPE to build a KS protocol. We then show a specific implementation of the OPE based on homomorphic encryption.

Protocol 1 (Generic polynomial-based KS) .
Input: Client: an evaluation point w; Server: $\{x_i, p_i\}_{i\in[n]}$, all x_i's are distinct
Output: Client: p_i if $w = x_i$, nothing otherwise; Server: nothing

1. *The server defines L bins and maps the n items into the L bins using a random, publicly-known hash function H with a range of size L. H is applied to the database's keywords,* i.e., *(x_i, p_i) is mapped to bin $H(x_i)$. Let m be a bound such that, with high probability, at most m items are mapped to any single bin. (At this point, we keep L and m as parameters.)*
2. *For every bin j, the server defines two polynomials P_j and Q_j of degree $(m-1)$. The polynomials are defined such that for every pair (x_i, p_i) mapped to bin j, it holds that $P_j(x_i) = 0$ and $Q_j(x_i) = (p_i|0^\ell)$, where ℓ is a statistical security parameter.*
3. *For each bin j, the server picks a new random value r_j and defines the polynomial $Z_j(w) = r_j \cdot P_j(w) + Q_j(w)$.*
4. *The two parties run an OPE protocol in which the client evaluates all L polynomials at the searchword w.*
5. *The client learns the result of $Z_{H(w)}(w)$,* i.e., *of the polynomial associated with the bin $H(w)$. If this value if of the form $p|0^\ell$ the client outputs p, otherwise it outputs $\perp$.*

To instantiate this generic scheme, we need to detail the following three open issues: (1) the OPE method used by the parties, (2) the number of bins L, and (3) the method by which the client receives the OPE output for the relevant bin. Additionally, one could consider using carefully-chosen hashing methods to obtain a balanced allocation of items into bins, although this approach would not yield substantial improvements.

An OPE method. Our construction uses an OPE method based on homomorphic encryption[4] (such as Paillier's system [28]) in the following way. We first introduce this construction in terms of a single database bin.

[3] Protocol 1 uses a public hash function H. To run it in the "plain" model, the client can pick the hash function and send it to the server in its first message.

[4] Other OPE constructions could be based on the hardness of noisy polynomial interpolation or on using $\log |\mathcal{F}|$ 1-out-of-2 OTs, where $\mathcal{F}$ is the underlying field [23].

– The server's input is a polynomial of degree m, where $P(w) = \sum_{i=0}^{m} a_i w^i$. The client's input is a value w.
– The client sends to the server homomorphic encryptions of the powers of w up to the mth power, *i.e.*, $Enc(w), Enc(w^2), \ldots, Enc(w^m)$.
– The server uses the homomorphic properties to compute the following:

$$\prod_{i=0}^{m} Enc(a_i w^i) = Enc(\sum_{i=0}^{m} a_i w^i) = Enc(P(w))$$

The server sends this result back to the client.

In the case of semi-honest parties, it is clear that the OPE protocol is correct and private. Furthermore, the protocol can be applied in parallel to multiple polynomials, and the structure of the protocol enforces that the client evaluates all polynomials at the same point.

Now, consider that the server's input is L polynomials, one per bin. The protocol's overhead for computing all polynomials is the following. The client computes and sends m encryptions. Every polynomial P_j used by the server is of degree $d_j \leq m$ (where $d_j + 1$ items are mapped to bin j), and the server can evaluate it using $d_j + 1$ homomorphic multiplications of plaintexts. Thus, the total work of the server is $\sum_{j=0}^{L-1}(d_j + 1) = n$ exponentiations. The server returns just a single value for each of the L polynomials.

A simple protocol. Let the server assign the n items to L bins arbitrarily and evenly, ensuring that L items are assigned to every bin; thus, $L = \sqrt{n}$. The client need not know which items are mapped to which bin. The client's message during the OPE consists of $L = O(\sqrt{n})$ homomorphic encryptions; the server evaluates L polynomials by performing n homomorphic multiplications (exponentiations), and replies with the $L = \sqrt{n}$ results. This protocol has a communication overhead of $O(\sqrt{n})$, $O(n)$ computation overhead at the server's side, and $O(\sqrt{n})$ computation overhead at the client's side.

Reducing communication: Receiving the OPE output using PIR. Note that the client does not need to learn the outputs of all polynomials but rather only the value of the polynomial associated with the bin to which w could be mapped. To further lower the communication complexity, the protocol uses a public hash-function H and invokes PIR to retrieve the result of the relevant polynomial evaluation. Namely, the function H is chosen independently of the content of the database, and it is used to map items to bins. After the server evaluates the L polynomials on the client's input w, the client runs a 1-out-of-L PIR scheme to learn the result of the polynomial of bin $H(w)$.

The total communication overhead is $O(m) \approx n/L$ (client to server) plus the overhead of the PIR scheme. A good choice is to use a PIR scheme with a poly-logarithmic communication overhead, such as the scheme of Cachin et al. [4] (based on the Φ-hiding assumption) or the schemes of Cheng [6] or Lipmaa [21] (based on the Paillier and Damgård-Jurik cryptosystems, respectively). In these cases, setting $L = n/\log n$ gives a total communication of $O(\text{polylog } n)$. We note that the client can combine the first message from its KS scheme with that of its PIR scheme. Thus, the round overhead of the combined protocol is the same as that of the PIR protocol alone. The computation overhead of the server is $O(n)$ plus that of a PIR scheme with L inputs; the client's overhead is $O(m)$ plus that of a PIR scheme with L inputs.

Theorem 1. *There exists a KS system for semi-honest parties with a communication overhead of* $O(\text{polylog n})$ *and a computation overhead of* $O(\log n)$ *"public-key" operations for the client and* $O(n)$ *for the server. The security of the KS system is based on the assumptions used for proving the security of the KS protocol's homomorphic encryption system and of the PIR system.*

Proof (sketch for semi-honest parties): Given a pair (x_i, p_i) in the server's input such that $w = x_i$, it is clear that the client outputs p_i. If $w \neq x_i$ for all i, the client outputs $\perp$ with probability at least $1 - 2^{-\ell}$. The protocol is therefore correct. Since the server receives semantically-secure homomorphic encryptions and the PIR protocol protects the privacy of the client, the protocol ensures the client's privacy: The server cannot distinguish between any two client inputs x, x'. Finally, the protocol protects the server's privacy: If a polynomial Z with fresh randomness is prepared for every query on every bin, then the result of the client's query w is random if w is not a root of P, *i.e.,* if w is not in the server's input X. A party running the client's role in the ideal model can therefore simulate the client's view in the real execution.

Handling malicious servers. Assume that the PIR protocol provides client privacy in the face of a malicious server (as is the case with virtually all PIR protocols from the literature). Then the protocol is secure against malicious servers (per Definition 2), as the only information that the server receives, in addition to messages of the PIR protocol, is composed of semantically-secure encryptions of powers of the client's input searchword.

Handling malicious clients. If the client is malicious then server privacy is not guaranteed by Protocol 1 as given. For example, a malicious client could send encryptions that do not correspond to powers of a value w. However, if the OPE protocol used in Protocol 1 is secure against malicious clients, then the overall protocol provides security against malicious clients, regardless of the security of the PIR protocol. (Note that there are no server privacy requirements on PIR; it is used merely to reduce communication complexity.)

One conceptually-simple solution therefore requires the client to prove that the encryptions it sends in the OPE protocol are well-formed, *i.e.*, correspond to encryptions of a sequence of values $w, w^2, \ldots, w^m$. Unfortunately, such a proof in the standard model requires more than a single round of messages.

A more efficient solution can be based on a known reduction of the OPE of a polynomial of degree m, to m OPEs of linear polynomials [12]. The overhead of the resulting protocol is similar to that of a direct OPE of the polynomial, and the protocol consists of only a single round (the m OPEs of the linear polynomials are done in parallel). We describe the reduction of [12] in the full version of this paper.

When the OPE protocol (based on homomorphic encryption) is applied to a linear polynomial, any encrypted value (w) sent by the client corresponds to a valid input to the polynomial, and thus the OPE of the linear polynomial computes a legitimate value of the polynomial. Therefore, if we ensure that the client sends a legitimate encryption we obtain a linear OPE (and thus a general OPE) secure against malicious clients.

When considering concrete instantiations of the OPE protocol, we note that the El Gamal cryptosystem has the required property, namely that any ciphertext can be decrypted.[5] The El Gamal cryptosystem can therefore be used for implementing a single-round OPE secure against a malicious client. Yet, the El Gamal system has a different drawback: given that it is multiplicatively homomorphic, it can only be used for an OPE in which the receiver obtains $g^{P(x)}$, rather than $P(x)$ itself. Thus, a direct use of El Gamal in KS is only useful for short payloads, as it requires encoding the payload in the exponent and asking the receiver to compute its discrete log.

We can slightly modify the KS protocol to use El Gamal yet still support payloads of arbitrary length. A detailed description appears in the full version of the paper. The main idea, however, is to have the server map the items to $n/\log n$ bins as usual, but define, for every bin j, a random polynomial Z_j of degree $m = O(\log n)$. For an item (x_i, p_i), the server encrypts $p_i|0^\ell$ using the key $g^{Z_{H(x_i)}(x_i)}$. The client sends a first message for an El Gamal-based OPE, namely encryptions of $g^w, g^{w^2}, \ldots, g^{w^m}$. The server then prepares, for every bin j, a message $\langle\ g^{Z_j(w)}, \{Enc_{Z_j(x_{j,i})}(p_{j,i}|0^\ell)\}_{i\in[m]}\ \rangle$, where the $x_{j,i}$'s are the messages mapped to bin j. The client uses PIR to learn the message of its bin of interest, and then can decrypt the payload corresponding to w if $\exists\ x_{j,i} = w$.

The only difference with this modified protocol is that the message learned during the PIR is of size $O(|p_i| \log n)$ rather than of size $O(|p_i|)$. The overall communication complexity does not change, however, since the PIR has polylogarithmic overhead. We obtain essentially the same overhead, including round complexity, as Protocol 1. (Note also that the security of the new protocol is proved in the model of Remark 1.)

Multiple invocations. The privacy of the server in Protocol 1 and its variants is based on the fact that the client can evaluate each polynomial Z at most once. Therefore, fresh randomness r_i must be used in order to generate new polynomials $Z_1, \ldots, Z_L$ for every invocation of the protocol. This means that using the protocol for multiple queries must essentially be done by independent invocations of the protocol.

[5] Unfortunately, as was observed for example in [1], the Paillier cryptosystem is not verifiable. That is, given a public key and a ciphertext, it is not known how to verify that the ciphertext is valid and can be correctly decrypted.

4 Keyword Search from OPRFs

In this section, we describe a general *reduction* of KS to semi-private KS using oblivious pseudorandom functions (OPRFs). Unlike the protocol from the previous section, this reduction can yield fully-adaptive KS protocols. We first recall the original notion of OPRFs from the literature [26] and then introduce a new natural relaxation, which arguably suffices for most applications. Finally, we describe our reduction from KS to semi-private KS using the relaxed notion of OPRF. New constructions of such OPRFs will be presented in the next section.

4.1 Oblivious Pseudorandom Functions

The strongest definition of OPRF is as a secure two-party protocol realizing the functionality $g(r, w) = (\lambda, f_r(w))$ for some pseudorandom function family f_r. (Here and in the following, the first input or output corresponds to the server $\mathcal{S}$ and the second to the client $\mathcal{C}$; by λ we denote an empty output.) As usual, the term "secure" can be interpreted in several ways. For consistency with the security definitions of Section 2 and the constructions of the next section, we interpret "secure" here as "private". We note, however, that the definitions and results of this section naturally extend the case of full security.

Definition 5. Strongly-private OPRF (s-OPRF). *A two-party protocol π is said to be a strongly-private OPRF (or* strong OPRF *for short) if there exists some PRF family f_r, such that π privately realizes the following functionality.*

- *Inputs: Client holds an evaluation point w; Server holds a key r.*
- *Outputs: Client outputs $f_r(w)$; Server outputs nothing.*

One can similarly define adaptive and non-adaptive t-time variants of strong OPRFs. Note that server privacy guarantees that a malicious client $\mathcal{C}'$ cannot learn anything about r except what follows from $f_r(w')$ for some w'. Composability of secure computation [5] implies that a 1-time s-OPRF can be invoked multiple times (with the same r and different w_i) to realize an adaptive t-time s-OPRF, where t can be an arbitrary polynomial in the security parameter.[6]

It follows from known reductions between cryptographic primitives that strong OPRF exists if OT exists [14, 19, 16]. We note, however, that the construction of s-OPRF from OT makes a non-black-box use of the OT primitive, even in the semi-honest setting: The OT-based protocol for evaluating the PRF depends on the function's circuit representation [19], which in turn depends on the representation of the OT primitive from which the PRF is derived.

[6] Note that our definitions of KS and OPRF do not require protecting the client against a malicious server who may choose different keys r in different invocations. On the other hand, our definition coincides with that of [5] for the case of simulating a (potentially malicious) client.

A new relaxed type of OPRF. As noted above, a strong OPRF guarantees that the client learn no additional information about the PRF key r. As we shall see, some natural and efficient OPRF protocols do not satisfy the strong definition, yet are sufficient for the KS application. We thus turn our consideration to relaxing the definition of server privacy to the following.

Roughly speaking, we require that following the execution of the OPRF protocol, the client obtains no additional information about the *outputs* of a *random* function f_r, other than what follows from a legitimate set of queries, whose size is bounded by t in the t-time case. (Recall that the strong definition requires that no information be learned about the *key* of an *arbitrary* function f_r.) In other words, the outputs of f_r on unqueried inputs cannot be distinguished from the outputs of a *random* function, even given the client's view. Note that this does not prevent the client from learning substantial partial information about r (which does not provide information about other values of f_r).[7]

This intuitive property is relatively straightforward to formalize in the case of a semi-honest client. Specifically, one may require that following the protocol's execution, the client cannot efficiently distinguish between f_r and a random function if it only queries them on points not included in its queries $w_1, \ldots, w_t$. Obtaining a suitable definition for the case of malicious clients, however, requires more care. In particular, the inputs on which the client queries f_r in a particular execution of the protocol may not even be well-defined.

We formalize our relaxed notion of OPRF by a careful modification of the underlying functionality. The client's privacy is defined as before. However, for the purpose of defining the server's privacy, we view f_r as a *randomized* functionality (with randomness r picked by the TTP in the ideal implementation), and we allow *both* the client and the server to provide inputs to and receive outputs from this functionality.

Definition 6. Relaxed OPRF (r-OPRF). *A two-party protocol π is said to be a (non-adaptive, 1-time)* relaxed OPRF *if there exists some PRF family f_r, such that the following hold.*

Correctness and client's privacy. *These properties remain the same as in Definition 5,* i.e.*, using the functionality $g(r, w) = (\perp, f_r(w))$.*

Server's privacy. *To define server's privacy in π, we make the following mental experiment. Consider an augmented protocol $\tilde{\pi}$ in which the input of $\mathcal{S}$ consists of n evaluation points $x_1, \ldots, x_n$ (instead of a key r) and the input of $\mathcal{C}$ is an evaluation point w (as in π). Protocol $\tilde{\pi}$ proceeds as follows: (1) $\mathcal{S}$ picks a key r at random; (2) $\mathcal{S}, \mathcal{C}$ invoke π on inputs (r, w); (3) $\mathcal{S}$ outputs $(f_r(x_1), \ldots, f_r(x_n))$*

[7] As a concrete simple example, consider the following pseudo-random function based on the Naor-Reingold construction. The key r consists of two sets $x_1, \ldots, x_m$ and $y_1, \ldots, y_m$; the function is defined for inputs (i, j) such that $1 \leq i, j \leq m$, and its value is $f_r(i, j) = g^{x_i y_j}$ in a group where the DDH assumption holds and g is a generator. Consider a 1-time OPRF protocol where a client whose input is (i, j) learns x_i and y_j and uses them to compute $f_r(i, j)$. Although these values reveal part of the key r to the client, the other outputs of the function remain pseudo-random.

and $\mathcal{C}$ outputs its output in π. We require that the augmented protocol $\tilde{\pi}$ provide server security with respect to the following randomized *functionality $\tilde{g}$:*

- *Inputs: Client holds an evaluation point w; Server holds an arbitrary set of evaluation points $(x_1, \ldots, x_n)$.*
- *Outputs: Client outputs $f_r(w)$ and Server outputs $(f_r(x_1), \ldots, f_r(x_n))$, where the key r is uniformly chosen by the functionality.*[8]

Specifically, for any (efficient, malicious) client $\mathcal{C}'$ attacking $\tilde{\pi}$, there is a simulator $\mathcal{C}''$ playing the client's role in the ideal implementation of $\tilde{g}$, such that on all inputs $((x_1, \ldots, x_n), w)$, the view of $\mathcal{C}'$ concatenated with the output of $\mathcal{S}$ in $\tilde{\pi}$ is computationally indistinguishable from the output of $\mathcal{C}''$ concatenated with that of $\mathcal{S}$ in the ideal implementation of $\tilde{g}$.

This definition applies to the non-adaptive 1-time case. In the t-time case, we replace w with $w_1, \ldots, w_t$, and $f_r(w)$ with $(f_r(w_1), \ldots, f_r(w_t))$. In the adaptive case, the protocols $\pi, \tilde{\pi}$ and the functionalities $g, \tilde{g}$ have multiple phases, where the client's input w in each phase may depend on the outputs of previous phases.

The above server's privacy requirement implies that the client's view gives no information about the server's inputs and outputs $(x_1, f_r(x_1)), \ldots, (x_n, f_r(x_n))$, other than what follows from some valid set $(w'_1, f_r(w'_1)), \ldots, (w'_t, f_r(w'_t))$. Moreover, this holds for an arbitrary choice of points x_i made by the server (including those possibly intersecting w'_i). In fact, this is precisely the requirement needed for the keyword-search application.

Finally, we note that Definition 6 is indeed a relaxation of Definition 5.

Claim. If π is an s-OPRF, then it is also an r-OPRF.

Proof: The server's privacy requirement of Definition 5 implies, in particular, that on a *uniformly-chosen* r and an arbitrary w, the view V' of a malicious client $\mathcal{C}'$ concatenated with r is indistinguishable from the output V'' of its simulator $\mathcal{C}''$ concatenated with r. This in turn implies that $(V', \{(f_r(x_i)\}_{i\in[n]})$ is indistinguishable from $(V'', \{(f_r(x_i)\}_{i\in[n]})$, as required by Definition 6.

4.2 Reducing KS to Semi-Private KS

We now present a general method of using (either variant of) OPRF to upgrade any semi-private KS protocol into fully-private KS.

Recall that a semi-private KS protocol is a KS protocol which guarantees privacy for the client but not for the server, similar to the privacy offered by PIR protocols. (The notion of semi-private KS was first considered in [7], where it was referred to as *private information retrieval by keywords*.) Semi-private KS can be simply implemented by letting the server send its input X, or (better yet) a data structure Y representing X, to the client. When the communication

[8] Equivalently, f_r can be replaced here with a totally random function. We prefer the current formulation because of its closer correspondence with the notion of s-OPRF, as well as the convention that ideal functionalities are efficiently computable.

is required to be sublinear, semi-private KS can be implemented using PIR to probe the data structure Y, as suggested in [7].

Using the following high-level idea, we can now construct a fully-private KS protocol from a semi-private KS protocol: The server uses a PRF to assign random pseudo-identities to the original keywords x_i (as well as mask the payloads p_i), and the client uses an OPRF protocol to learn the values of the PRF on the selected searchword(s). Since the PRF values on unselected searchwords remain random from the client's point-of-view, knowledge of the original and pseudo-identity pairs of the selected searchwords does not provide any more information than does knowledge of just the set of searchwords that are in the database along with their payloads.

More formally, given a semi-private KS protocol and a (possibly relaxed) OPRF realizing f_r, the KS protocol proceeds as follows. For simplicity, we address below the case non-adaptive KS with $t = 1$.

Protocol 2 (A KS protocol based on semi-private KS and r-OPRF) .

1. *The server picks a random key r for the PRF. For $1 \leq i \leq n$, it parses $f_r(x_i)$ as $(\hat{x}_i, \hat{p}_i)$ and constructs a* pseudo-database *$X' = \{(x'_i, p'_i)\}_{i \in [n]}$ with $x'_i = \hat{x}_i$ and $p'_i = p_i \oplus \hat{p}_i$. (Both X and X' must be treated as* unordered *sets, whose representation does not reveal the index i of each element; alternatively, one may think of X and X' as lexicographically-sorted sequences.)*
2. *The parties invoke the r-OPRF protocol, with server input r and client input w. As a result, the client learns $f_r(w)$ and parses it as $(\hat{w}, \hat{p})$.*
3. *The parties invoke the semi-private KS protocol with server input X' and client input $\hat{w}$. As a result, the client learns whether $\hat{w} \in X'$, and if so, also learns the corresponding payload p'_i. If $\hat{w} \in X'$, the client outputs $p'_i \oplus \hat{p}$; otherwise, it outputs $\perp$.*

We stress that, due to the lack of server's privacy in *semi*-private KS, we should make the worst-case assumption that the client learns the *entire* pseudo-database X' in Step 3. Still, the use of an OPRF in Step 2 guarantees that the client does not learn more than it is entitled.

Remark 2. If a setup phase with linear communication is allowed, the semi-private KS in Step 3 can be replaced by having X' (or a corresponding data structure Y') sent to the client in the clear following Step 1.

Theorem 2. *Protocol 2 is a private KS protocol.*

Proof (sketch): The protocol's correctness is easy to verify. The client's privacy follows immediately from its privacy in the OPRF and the semi-private KS.

Server's privacy. Letting π denote the r-OPRF protocol, it is convenient to reformulate the above protocol in the following equivalent way:

- The parties invoke the augmented protocol $\tilde{\pi}$ (from Definition 6) on server input $(x_1, \ldots, x_n)$ and client input w. At the end of this protocol, $\mathcal{S}$ outputs $(f_r(x_1), \ldots, f_r(x_n))$ and $\mathcal{C}$ outputs $f_r(w)$.

– The server parses each $f_r(x_i)$ as $(\hat{x}_i, \hat{p}_i)$ and creates a pseudo-database $X' = \{(x'_i, p'_i)\}_{i\in[n]}$ with $x'_i = \hat{x}_i$ and $p'_i = p_i \oplus \hat{p}_i$, as before. Again, the client parses $f_r(w)$ as $(\hat{w}, \hat{p})$. The parties invoke the semi-private KS protocol with server input X' and client input $\hat{w}$. As a result, the client learns whether $\hat{w} \in X'$, in which case the client outputs $p'_i \oplus \hat{p}$; otherwise, it outputs $\perp$.

By Definition 6, when considering only the client's simulation, $\tilde{\pi}$ must be secure with respect to the randomized functionality $\tilde{g}$ mapping $(x_1, \ldots, x_n)$ and w to $(f_r(x_1), \ldots, f_r(x_n))$ and $f_r(w)$, respectively. Hence, using protocol composition [5], it suffices to prove the server's privacy in a simpler "hybrid" protocol, where the invocation of $\tilde{\pi}$ is replaced by a call to an oracle (or TTP) computing $\tilde{g}$. Moreover, by the pseudorandomness of f_r, we can replace the oracle $\tilde{g}$ by a similar oracle $\widetilde{G}$ in which f_r is replaced by a truly random function.

The resultant hybrid protocol is in fact *perfectly* private. Given a malicious client $\mathcal{C}'$ attacking the hybrid protocol, a corresponding simulator $\mathcal{C}''$ can proceed as follows. $\mathcal{C}''$ invokes $\mathcal{C}'$ on input w. In the first step, after learning the query w' which $\mathcal{C}'$ sends to the oracle computing $\widetilde{G}$, the simulator $\mathcal{C}''$ sends the query w' to the TTP computing KS. As a response, it gets p_i if $w' = x_i$ or $\perp$ if no such i exists. Now the second step can be simulated jointly with the response $(\hat{w}, \hat{p})$ of the $\widetilde{G}$ oracle. First, $\mathcal{C}''$ chooses X' to be a uniformly-random pseudo-database of size n. Next, it simulates $(\hat{w}, \hat{p})$ so that they are consistent with X' and the response of KS: if a payload p was obtained from KS, then $\hat{w}$ is taken to be a random keyword from X' and $\hat{p}$ is set to the exclusive-or of the keyword's corresponding payload and p; otherwise, $\hat{w}$ and $\hat{p}$ are chosen at random from their respective domains. Finally, $\mathcal{C}''$ simulates the view of $\mathcal{C}'$ in the semi-private KS protocol by simply running the protocol on inputs (X', w').

Efficiency. The cost of the protocol is dominated by that of the semi-private KS and the OPRF. In the t-time non-adaptive model, this cost is typically dominated by that of the semi-private KS, which in turn is dominated by the cost of the underlying PIR protocol. We note that the latter cost can be amortized over t non-adaptive queries [2, 18]. In the adaptive model—more generally, in any setting allowing setup—the offline cost is dominated by linear communication in the size of the database, and the online cost by the efficiency of the underlying OPRF. We now consider efficient implementations of the OPRF primitive.

5 Constructing OPRFs

A generic implementation of an s-OPRF can be based on general secure two-party evaluation. Namely, the server has as input a key r of a PRF f_r and, whenever the client wants to evaluate f_r on x, the parties perform a secure function evaluation (SFE), during which the client learns $f_r(x)$. As noted above, this gives rise to a *non-black-box* reduction from strong OPRF to OT. In this section, we discuss two other types of constructions:

– Constructions of fully-adaptive s-OPRFs based on specific assumptions (mainly on DDH). These constructions are either given or implicit in [26, 24] and are more efficient than the generic SFE-based construction sketched above.

– General constructions of t-time adaptive r-OPRFs making a *black-box* use of OT. From a theoretical point of view, one of the most interesting open questions left by our work is to come up with any efficient black-box construction of fully-adaptive r-OPRFs. This is indeed a rare example of a non-black-box construction in cryptography for which no black-box construction is known.

For simplicity, we discuss these constructions mainly from the viewpoint of the semi-honest model.

5.1 Strong OPRFs Based on DDH or Factoring

Naor and Reingold gave two constructions of PRFs based on number-theoretic assumptions in [26]: one based on the Decisional Diffie-Hellman assumption (DDH), and the other based on the hardness of factoring. The constructions have a simple algebraic structure, and they were used to give oblivious, fully-adaptive evaluations for these functions. While more efficient than general secure function evaluation, these s-OPRFs have the disadvantage of requiring a linear number of rounds and a linear number of exponentiations. Implicit in the work of Naor and Pinkas on OT [23, 24],[9] one can find a significantly more efficient evaluation of the DDH-based PRFs of [26]. We now sketch this construction.

Initialization: Let g be a generator of a group G_g of prime order p for which the DDH assumption holds. The key $\bar{r}$ of the pseudo-random function $f_{\bar{r}} : \{0,1\}^m \mapsto G_g$ contains m values $\{r_1, \ldots, r_m\}$, sampled uniformly at random in Z_p^*. The function $f_{\bar{r}}(x)$ is defined to be $g^{\Pi_{x_i=1} r_i}$, for any m-bit $x = x_1 x_2 \ldots x_m$. (This function was shown in [26] to be pseudorandom.)

Secure evaluation: The client has inputs $x = x_1 x_2 \ldots x_m$. The server selects m values $\{a_1, \ldots, a_m\}$ sampled uniformly at random in Z_p^*. For each i, the parties perform a 1-out-of-2 OT (denoted by $\binom{2}{1}$-OT), with the server using as inputs the two values a_i and $a_i \cdot r_i$. Thus, the client learns a_i if $x_i = 0$ and $a_i \cdot r_i$ otherwise. In addition, the server sends $\hat{g} = g^{1/\Pi_{i=1}^m a_i}$ in the clear. Let A be the product of the values learned by the client, then $A = (\Pi_{i=1}^m a_i) \cdot (\Pi_{x_i=1} r_i)$. Thus, the client can compute $\hat{g}^A$ and learn the desired value $f_{\bar{r}}(x)$.

Security: This protocol's security follows from the security of the OT protocol: The distribution of the m values learned by the m OTs, combined with $g^{1/\Pi_{i=1}^m a_i}$, can be easily sampled given access to $f_{\bar{r}}(x)$ alone.

Efficiency: The computational cost of the protocol (for both client and server) is m $\binom{2}{1}$-OTs and one exponentiation. The main cost in communication is that incurred by the m OTs. Given the work on batch OT of [17], the OTs performed by the oblivious evaluation protocol above can be considered, for practical purposes, to be almost as efficient as private-key operations. In particular, using these s-OPRFs in the transformation of Section 4.2 gives quite an efficient solution to KS. Unlike [27], this solution is in the standard model—rather than in the random oracle model—and only relies on standard assumptions.

[9] The construction was used to generate values that mask the server's input in an adaptive OT protocol.

5.2 Relaxed OPRFs Based on Black-Box OT

We now present a new construction of adaptive t-time r-OPRFs based on general assumptions, using the OT and PRF primitives in a black-box manner. (In fact, as discussed earlier, PRF is itself black-box implied by OT [14, 16, 15].) Our starting point is a construction of Naor and Pinkas [23] that gives PRFs—originally designed for sub-exponential domains—with some weak form of oblivious evaluation.

Consider a set of known PRFs $\{g_s\}$ over the domain $[N] = [M]^2$. Naor and Pinkas [23] construct related PRFs $\{f_{\bar{r}}\}$ over the same domain. First, let each key $\bar{r}$ be composed of two sets of M random g keys (*i.e.*, $\bar{r}_1 = \{r_{1,1}, \ldots, r_{1,M}\}$ and $\bar{r}_2 = \{r_{2,1}, \ldots, r_{2,M}\}$). Then, define $f_{\bar{r}}(x)$ as $g_{r_{1,x_1}}(x) \oplus g_{r_{2,x_2}}(x)$ for any $x = (x_1, x_2) \in M^2$.[10]

We can now use $f_{\bar{r}}$ in place of g_s to our advantage, as there exists a somewhat oblivious way of evaluating $f_{\bar{r}}(x)$. Namely, perform two independent $\binom{M}{1}$-OTs to retrieve $r_{1,x_1} \in \bar{r}_1$ and $r_{2,x_2} \in \bar{r}_2$, and then evaluate $f_{\bar{r}}(x)$ as desired using these random keys. Of course, the client now learns r_{1,x_1} and r_{2,x_2} in addition to just $f_{\bar{r}}(x)$. Still, it is easy to argue that $f_{\bar{r}}$, when restricted to all inputs other than x, remains pseudorandom. With a small additional effort, $f_{\bar{r}}$ can be turned into a 1-time r-OPRF.

What happens if we perform an oblivious evaluation of $f_{\bar{r}}$ on t different inputs? In this case, the client learns up to t keys in both $\bar{r}_1$ and $\bar{r}_2$, allowing it to evaluate $f_{\bar{r}}$ in up to t^2 places, which is certainly undesirable. Still, $f_{\bar{r}}$ maintains a considerable amount of pseudorandomness, as its output looks random other than at these t^2 locations. In light of this property, [23] gives a technique that can be translated into a construction of a *non-adaptive* t-time r-OPRF.

The PRF $F(\cdot)$ used in this construction is the exclusive-or of some ℓ functions $f_{\bar{r}^i}(\sigma^i(\cdot))$, where $f_{\bar{r}^i}$ is defined as before and each σ^i is a random permutation over $[N]$. All random inputs (for the sub-keys, $\bar{r}^i_1$ and $\bar{r}^i_2$, and for the permutations σ^i) are chosen independently by the server for all $1 \leq i \leq \ell$. The evaluation of $F(\cdot)$ on t inputs $x_1 \ldots x_t$ proceeds in ℓ rounds. In the i^{th} round, σ^i is sent to the client and the parties perform t oblivious evaluations of $f_{\bar{r}^i}$, as above.

This construction's main idea is the following: In each round, the client may learn at most t^2 values of the current $f_{\bar{r}^i}(\sigma^i(\cdot))$—a $t \times t$ sub-matrix—from the total of M^2 possible values over which the PRF is defined. However, to learn the value of F for $t+1$ distinct inputs, the client must learn all intermediate values for *each one of the ℓ functions* $f_{\bar{r}^i}(\sigma^i(\cdot))$ on these $t+1$ inputs. The random permutations σ^i—each learned only during the execution of subsequent rounds—ensure that this will only happen with negligible probability. See [23] for more details. Note that for this probabilistic argument to hold, the number of rounds ℓ must depend on the security parameter.

Challenges. The above construction raises the following challenges left open by [23] and the subsequent [24]: (1) Can the construction be made secure against adaptive queries? We note that the adaptive solutions given in [24] rely either

[10] This is a simple version of the construction; some useful optimizations are possible.

on specific assumptions or on random oracles. (2) Can one obtain oblivious evaluation in a *constant number of rounds*? Note that the number of rounds of the above protocols depends on the security parameter. (3) Can the construction handle an exponential domain size N? Various difficulties arise when naively extending the above solution to larger values of N. First, the random permutations σ^i are too large to sample and transmit. Second, one has to extend the construction to higher dimensions than two and view $[N]$ as $[M]^\ell$ for non-constant ℓ: We certainly want M to be sub-exponential, given that we are performing $\binom{M}{1}$-OTs. We can indeed perform this extension, but the natural method as used below reveals many more values of the PRFs: In t queries, the client learns t sub-keys in every dimension. Thus, it can evaluate the function at t^ℓ locations, where t^ℓ may be exponentially large (specifically, polynomially related to N). This expansion seems to complicate the analysis and, in particular, implies a larger number of rounds that also depends on t.

Our construction. In this section, we simultaneously answer all of the above challenges: We obtain *adaptive* t-time r-OPRFs that can handle an *exponential* domain size and can be securely evaluated in a *constant number of rounds*.

The technique of [23] for turning their 1-time r-OPRF into a t-time r-OPRF is based on providing only indirect access to the functions $f_{\bar{r}^i}$. Namely, the value of the PRF F on x depends on the values $f_{\bar{r}^i}(\sigma^i(x))$, rather than on $f_{\bar{r}^i}(x)$. However, since the permutation σ_i is transmitted in its entirety to the client, this type of indirection is not very useful for obliviousness by itself. Instead, the protocol must be designed using several functions, revealing additional information (each σ_i) in synchronous stages.

Instead, we will use only one function $f_{\bar{r}}$ and therefore will need only a single permutation σ for the indirect access to $f_{\bar{r}}$. Rather than transmitting the entire permutation σ to the client, we allow the client access only to t locations of σ in some oblivious way. Since σ is now not completely known to the client, we overcome both the need for a super-constant number of rounds and the large cost of sending σ for large domain sizes. Of course, if σ is random or pseudorandom, then the oblivious evaluation of σ is exactly the problem we wanted to solve in the first place! Therefore, we relax this randomness requirement by replacing σ with $(t+2)$-wise independent functions (although, in fact, even weaker requirements suffice).[11] We proceed to the detailed construction of adaptive t-time r-OPRFs, focusing on the setting where N is exponential.

Notion of privacy. In the above description and below, we argue that a function is an oblivious PRF if it remains pseudorandom on all inputs other than the ones retrieved by the client. This makes our discussion simpler and more intuitive. However, this type of definition seems only to make sense in the semi-honest

[11] A different variant of the construction uses the 1-time r-OPRFs based on [23] instead of the random permutations. This construction may be more efficient in some settings of the parameters. On the other hand, it seems theoretically inferior and somewhat more complicated (*e.g.*, it requires two levels of indirection). We therefore omit it from this version for clarity.

model (as otherwise, the inputs retrieved by the client may not be well-defined). Even in the semi-honest model, this notion—though sufficiently strong for the KS application—falls short of obtaining the requirements of a r-OPRF, which are defined in terms of simulation. Nevertheless, the protocol below gives a t-time r-OPRF: All that is needed is that the basic PRFs $\{g_s\}$ used by this protocol will have the additional property that, given t inputs and t corresponding outputs, a random seed s can be sampled under the restriction that g_s is consistent with these inputs and outputs. This is easy to obtain if each g_s is an exclusive-or of a PRF and a t-wise independent function (as t-wise independent functions usually have such "interpolation" property).

Extending the 1-time r-OPRF to higher dimensions. Let $\{g_s\}$ be PRFs over a domain $[N] = [M]^\ell$. Define the related PRFs $\{f_{\bar{r}}\}$ over the same domain, where each key $\bar{r}$ is composed of ℓ sets $\{\bar{r}_1, \ldots \bar{r}_\ell\}$ of M random g keys, where $\bar{r}_i = \{r_{i,1}, \ldots, r_{i,M}\}$. Thus, $\bar{r}$ defines an $\ell \times M$ matrix. For any $x = \{x_1, \ldots x_\ell\} \in M^\ell$, the value $f_{\bar{r}}(x)$ is defined to be $\bigoplus_{i=1}^{\ell} g_{i,x_i}(x)$.

The 1-time oblivious evaluation of $f_{\bar{r}}(x)$ goes as follows. First, perform ℓ independent $\binom{M}{1}$-OTs to retrieve $r_{i,x_i} \in \bar{r}_i$, for $i = 1, \ldots, \ell$. Then, the client can evaluate $f_{\bar{r}}(x)$ as desired. As mentioned above, t evaluations of $f_{\bar{r}}$ may now give information on t^ℓ values. However, $f_{\bar{r}}$ remains pseudorandom when restricted to all inputs other than x.

Oblivious evaluation of $(t+2)$-wise independent functions. The second ingredient in our construction is a family $H = \{h : [N] \mapsto [N]\}$ of $(t+2)$-wise independent functions. This definition means that, restricted to any $(t+2)$ inputs, a function h sampled from H is completely random.[12] We also rely on H to have an oblivious evaluation (or a t-time oblivious evaluation). This problem is an easier task than that of r-OPRFs. In particular, as $(t+2)$-wise independent functions exist unconditionally, they have oblivious evaluation based on OTs in a black-box manner. Note that while this observation is based on general secure evaluation, more efficient oblivious evaluations can be designed for specific families of hash functions: for example, an OPE-based evaluation can be used for a polynomial-based $(t+2)$-wise independent hash function.

The new adaptive t-time r-OPRFs. We set $M = 2t$ and assume without loss of generality that ℓ is at least the security parameter.[13] The key of these adaptive t-time r-OPRFs is composed of a $(t+2)$-wise independent hash function $h \in H$ and a key $\bar{r}$ of the ℓ-dimension 1-time r-OPRF $f_{\bar{r}}(\cdot)$ defined above. The value of this function $F_{h,\bar{r}}$ on any input $x \in [N]$ is given by $F_{h,\bar{r}}(x) \stackrel{\text{def}}{=} f_{\bar{r}}(h(x))$. The oblivious evaluation of $F_{h,\bar{r}}(x)$ proceeds by first evaluating $y = h(x)$ and then evaluating $f_{\bar{r}}(y)$, using the corresponding oblivious evaluation protocols.

---

[12] In fact, h can be only statistically close to random or even just pseudorandom.

[13] This implies r-OPRFs also for smaller values of N, although further optimizations may be possible for these cases.

Security of the construction (sketch). We want to claim that after t evaluations of $F_{h,\bar{r}}(\cdot)$, its restriction on all other inputs is indistinguishable from a random function. Intuitively, this is true since each dimension has $2t$ keys of which the client learns at most t, and the probability that another value of the function is evaluated using only these learned keys is at most $2^{-\ell}$. Consider the hybrid function $R(h(\cdot))$, where R is a random function. It is easy to argue that $R(h(\cdot))$ is indistinguishable from random: It only can be distinguished from random by querying inputs that cause collisions of h. Since conditioned on the values of h already learned by the client, h is still pair-wise independent, collisions are encountered with negligible probability. It remains to argue that $R(h(\cdot))$ is indistinguishable from $f_{\bar{r}}(h(\cdot))$. Note that at most t^{ℓ} values of $f_{\bar{r}}$ are compromised by the client, and $f_{\bar{r}}$ is still pseudorandom on the rest. To distinguish $R(h(\cdot))$ from $f_{\bar{r}}(h(\cdot))$, the distinguisher needs to query with an input that causes the output of h to fall into the compromised set. As the fraction of compromised $f_{\bar{r}}$-inputs is negligible (at most $2^{-\ell}$), this happens with negligible probability.

Acknowledgements. Michael Freedman is supported by a National Defense Science and Engineering Graduate Fellowship. Yuval Ishai is supported by Israel Science Foundation grant 36/03. Omer Reingold is the incumbent of the Walter and Elise Haas Career Development Chair at the Weizmann Institute of Science and is supported by US-Israel Binational Science Foundation Grant 2002246.

References

1. Bill Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, Innsbruck, Austria, May 2001.
2. Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers' computation in private information retrieval: Pir with preprocessing. In *CRYPTO*, Santa Barbara, CA, August 2000.
3. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, Interlaken, Switzerland, May 2004.
4. Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, Prague, Czech Republic, May 1999.
5. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
6. Yan-Cheng Chang. Single database private information retrieval with logarithmic communication. In *Proc. 9th ACISP*, Sydney, Australia, July 2004.
7. Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. Technical Report TR-CS0917, Dept. of Computer Science, Technion, 1997.
8. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proc. 36th FOCS*, Milwaukee, WI, 23–25 October 1995.
9. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.
10. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, Interlaken, Switzerland, May 2004.

11. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. 30th ACM STOC*, Dallas, TX, May 1998.
12. Niv Gilboa. *Topics in Private Information Retrieval*. PhD thesis, Technion - Israel Institute of Technology, 2000.
13. Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.
14. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.
15. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Construction of pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
16. Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *Proc. 30th FOCS*, Research Triangle Park, NC, October–November 1989.
17. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, Santa Barbara, CA, August 2003.
18. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch codes and their applications. In *Proc. 36th ACM STOC*, Chicago, IL, June 2004.
19. Joe Kilian. Founding cryptography on oblivious transfer. In *Proc. 20th ACM STOC*, Chicago, IL, May 1988.
20. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. 38th FOCS*, Miami Beach, FL, October 1997.
21. Helger Lipmaa. An oblivious transfer protocol with log-squared communication. Crypto ePrint Archive, Report 2004/063, 2004.
22. Silvio Micali, Michael Rabin, and Joe Kilian. Zero-knowledge sets. In *Proc. 44th FOCS*, Cambridge, MA, October 2003.
23. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. 31st ACM STOC*, Atlanta, GA, May 1999.
24. Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *CRYPTO*, Santa Barbara, CA, August 1999.
25. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proc. 12th SIAM SODA*, Washington, DC, January 2001.
26. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *Proc. 38th FOCS*, Miami Beach, FL, October 1997.
27. Wakaha Ogata and Kaoru Kurosawa. Oblivious keyword search. Crypto ePrint Archive, Report 2002/182, 2002.
28. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, Prague, Czech Republic, May 1999.
29. Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.
30. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proc. IEEE Symposium on Security and Privacy*, Berkeley, CA, May 2000.