Private Inference Control *

[Extended Abstract]

David Woodruff[†]
Massachusetts Institute of Technology
dpwood@mit.edu

Jessica Staddon
Palo Alto Research Center
staddon@parc.com

ABSTRACT

Access control can be used to ensure that database queries pertaining to sensitive information are not answered. This is not enough to prevent users from learning sensitive information though, because users can combine non-sensitive information to discover something sensitive. Inference control prevents users from obtaining sensitive information via such "inference channels", however, existing inference control techniques are not private - that is, they require the server to learn what queries the user is making in order to deny inference-enabling queries.

We propose a new primitive - private inference control (PIC) - which is a means for the server to provide inference control without learning what information is being retrieved. PIC is a generalization of private and symmetrically-private information retrieval (PIR/SPIR). While it is straightforward to implement access control using PIR (simply omit sensitive information from the database), it is nontrivial to implement *inference* control efficiently. We measure the efficiency of a PIC protocol in terms of its communication complexity, its round complexity, and the work the server performs per query. Under existing cryptographic assumptions, we give a PIC scheme which is simultaneously optimal, up to logarithmic factors, in the work the server performs per query, the total communication complexity, and the number of rounds of interaction. We also present a scheme requiring more communication but sufficient storage of state by the server to facilitate private user revocation. Finally, we present a generic reduction which shows that one can focus on designing PIC schemes for which the inference channels take a particularly simple threshold form.

[*]A full version of this paper is available on eprint at http://eprint.iacr.org/2004/130/

[†]Most of this research was conducted while this author was an intern at PARC.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption; H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Security, Theory

Keywords

Private information retrieval, oblivious transfer, inference control

1. INTRODUCTION

Consider a user who wants to query a database containing sensitive information. The user should learn the response to a query if and only if it does not enable them to learn something sensitive. There are two ways to learn something sensitive: by accessing sensitive information directly or by making a sequence of non-sensitive queries whose answers, when taken together, allow the user to infer something sensitive. Preventing the former is known as *access control* and the latter as *inference control* (see, for example, [18]). A sequence of queries whose responses allow a sensitive inference to be made is known as an *inference channel*, and the goal of inference control is to prevent the user from *completing* any inference channel. As an example, consider the problem of protecting the identity of individuals in a database [15, 31, 30]. Social security numbers are clearly identifying and access control ensures that they cannot be queried without appropriate authorization. In addition, while attributes such as city of residence, year of birth, profession, marital status and political party, may not be identifying individually (and so not subject to access control), when these values are received for a single record in a database they may be identifying and thus they form an inference channel.[1] Inference control ensures that colluding users aren't able to together make enough queries to complete such a channel.

An equally important aspect of this setting is the potential privacy loss on the user's side due to the exposure of their queries to the database server. To solve this problem private information retrieval (PIR) schemes (see, for example, [12, 24]) have been developed, but such schemes are not concerned with inference control. The question then arises: *How can we ensure users are not able to infer sensitive information without knowing what information they*

[1]Note that inference control is the same as access control if and only if all inference channels have length one.

are retrieving? To solve this problem we introduce a new primitive that we call private inference control, or PIC. In a PIC scheme, the server learns no information in the computational sense about the user's queries, even if the user is unable to retrieve the answer to their query. Note that any PIC protocol implements oblivious transfer [29, 17], so it must be interactive. Hence, the server learns the number of queries made so far simply by counting the number of interactions with the user. We emphasize that this is in fact the *only* information a computationally-bounded server learns about the user's queries. In fact, it is precisely the knowledge of the number of queries made so far that will enable the server to do collusion-resistant inference control.

Oblivious transfer has been studied in the context of access control [27, 2]. However, both [27] and [2] can only be applied to protect against certain collections of inference channels, rather than the more general collections that we consider here (see Section 3 for a precise description). In addition, they don't apply to the relational databases [11] that are our main focus because they view the data as consisting of a single record. Running multiple instances of schemes in [27, 2] in parallel to accommodate multiple records causes the efficiency of their techniques, and much of the user privacy that we desire, to be lost. Indeed, we believe that new ideas are needed to solve the PIC problem.

Our enabling technique is the transfer of a small amount of extra information to the user with a query. In subsequent queries, the extra information is used to encrypt data that is communicated through the execution of a PIR or SPIR protocol. The encryption is done in such a way to ensure that users who have made a permissible sequence of queries thus far (as defined by the inference channels) are able to decrypt. The difficulty comes in determining how to do this encryption to accommodate an arbitrary collection of inference channels while keeping communication overhead and server work at reasonable levels.

As suggested by our earlier example, PIC is useful in relational databases. Once the inference channels have been identified (with a tool such as [30], for example) a PIC scheme can be designed to ensure that users are prevented from making undesired inferences while privately accessing the database, even when users collude. In addition, even when data perturbation is used to ensure privacy of the data (see, for example, [3, 35, 1]), PIC can be useful. For example, in [15] it's shown that in a statistical database in which sums are queried, a large amount of perturbation is needed in order to preserve privacy. So much perturbation, in fact, that the data is potentially useless. If a PIC scheme were employed in such a setting it might be possible to retain the usefulness of the data by reducing the amount of perturbation while ensuring that no user can query enough data to determine the individual, unperturbed data values.[2] Finally, PIC may be useful in Internet content distribution, as it can allow clients to hide their items of interest from merchants (thus avoiding marketing intrusions) while providing merchants with the guarantee that only prescribed content can be retrieved.

Overview. This paper is organized as follows. Section 1.1 summarizes our results, and related work appears in Section 1.2. Section 2 provides tools and notation and Section 3 describes our model of PIC. In Section 4 we show that it suffices to consider inference channels of a certain "threshold" form. Section 5 contains a stateful protocol with optimal server work and private user revocation based on query histories. Section 6 describes our PIC scheme that is simultaneously optimal in communication/round complexity and server work per query. We conclude in Section 7. Due to space constraints, background is kept at a minimum and some proofs are omitted. The details can be found in the full version of this paper [36].

[2]However, with such an approach the user can learn the structure of the inference channels (but not the data that completes the channels) which may not occur when data is protected solely by perturbation. See Section 3 for more on this point.

1.1 Results

Besides the contribution we make by rigorously defining the security model for private inference control, we also achieve extremely efficient implementations. Note that a possible implementation of PIC is just to run a generic secure function evaluation (also referred to as a secure cicuit evaluation) [22, 37]. However, the communication complexity of such schemes is at least linear in the input size. In contrast, we obtain solutions with substantially sublinear communication. Our key idea for achieving this is to carefully divide up the inputs/state of our schemes so as to run secure function evaluation on a small set of inputs while running an efficient private information retrieval scheme on a large set of inputs. Of course, the difficulty of doing this is that this division of inputs must respect user privacy and inference control. Our final scheme in section 6 accomplishes this division. Plugging in the best known PIR parameters [10, 25], we obtain (loosely speaking):

For databases of size n, there exists a 1-round PIC scheme with $O(polylog(n))$ communication and $O(npolylog(n))$ server work. (Section 6)

In addition, in Section 5 we present a scheme that incurs more overhead but has the benefit of supporting private user revocation. That is, the server can ensure that users who have accessed certain information are no longer able to query the database *without* knowing who these users are.

Finally, we present a reduction that shows that one can focus on designing PIC schemes for which the inference channels take a particularly simple "threshold" form and transform them into PIC schemes handling arbitrary inference channels, without much cost in efficiency for a certain range of parameters.

1.2 Related Work

Private information retrieval (PIR) was introduced in a seminal paper by Chor *et al* [12]. In such schemes a server holds an n-bit binary string $x \in \{0,1\}^n$, representing a database, and a user has some index $i \in [n]$. At the end of the protocol the user should learn x_i and the server should learn nothing about i. This notion of privacy was later relaxed [10, 24, 25] to prevent any computationally-bounded server from learning anything about i.

A *symmetrically private information retrieval* (SPIR) protocol [20] guarantees that the user learns nothing about any index in x other than the one requested, in addition to providing user privacy. Note that neither SPIR nor PIR accounts for changes in a user's permissions over time; these permissions must change in order to enforce inference control. However, both PIR and SPIR protocol are useful building blocks for PIC and are used in our protocols.

SPIR is essentially equivalent to the older notion of 1-out-of-n oblivious transfer (OT) [29, 7], where the emphasis in SPIR is on the communication complexity. The oblivious transfer works that bare the most similarity to PIC are [2, 27]. In [27], Naor and Pinkas present schemes enabling a user to query a database adaptively for at most a parameter τ number of times, while preserving the privacy of both the user and the database. Hence, [27] can be viewed as studying a simplified version of our problem in which the database consists of a single record and the inference channels are all sets of $\tau + 1$ attributes. It is not clear that the techniques of [27] can be efficiently extended to our setting. In particular, the natural extension in which one instance of a protocol from [27] is run for each record results in linear communication and doesn't offer the privacy of our protocols since the server knows when each record is queried.

In [2], Aiello, Ishai and Reingold also only consider the single record setting and the natural extension of their protocols to multiple records is less efficient, and less privacy-preserving, than our protocols. However, the schemes in [2] have the advantage of allowing for more general inference channels than those in [27]. Phrased in terms of our setting, in [2] a price is associated with each attribute of a record and a user can only query attributes until their budget is exceeded. Although this allows for a wider variety of inference channels, it is not as general as the setting we consider. We consider inference channels that are subsets of the attributes (this is made more precise in Section 3), and so an inference can be drawn when all of the attributes in a subset are received for any particular record. As an example of inference channels that cannot be realized in the "price" model of [2], consider the following two inference channels (A_i denotes the ith attribute in the database), $I_1 = \{A_1, A_2\}, I_2 = \{A_3, A_4\}$. In order to provide inference control with the protocols of [2], there must be at least one attribute in each with a price of more than 1/2 of the budget, but then no two of these attributes can be "bought" given the budget, however no such pair of attributes forms an inference channel.

An alternate approach to inference control, and access control in general, is data perturbation [3, 35, 1]. Data perturbation has the benefit that queries are never blocked and hence the user gains no information about what's sensitive in the database. This property is especially important when inference channels are data dependent. However, recent work [15] suggests that in some settings the amount of perturbation that must be applied to the data to guarantee privacy may be so much as to render the data useless.

A solution to the PIC problem is possible using techniques for search on encrypted data (see, [33] and the more recent work [8, 19]). The idea is that the records a user has queried are stored in encrypted form on the server and the user gives the server the capability to determine whether or not the current record of interest has been queried too much. With such a solution communication cost is essentially the cost of the SPIR protocol employed. However such an approach doesn't offer complete privacy to the user as the server is able to determine exactly how many times a record has been queried and exactly when it was queried, even though the correct index of the record may be hidden.

Tools and techniques for ensuring inference control without user privacy can be found in [30, 23, 34].

2. PRELIMINARIES

2.1 Tools

We use the following tools: computational symmetrically-private information retrieval [20, 26, 17] on records, homomorphic encryption [28], non-malleable encryption [16, 13], and zero-knowledge proofs of knowledge [6].

In the full version of this paper [36], we use [10, 26] to derive a construction of SPIR on records of size $O(\text{polylog}(n))$ using one round of interaction, $O(\text{polylog}(n))$ communication, and $O(n \text{ polylog}(n))$ server work per query. We will use the term SPIR to refer to our specific construction on records. The SPIR protocol we use has the additional property that it is secure in the real/ideal model. That is, to guarantee the user does not learn more than a single index i of the database x, there is an efficient simulator which can extract the index i learned by any (possibly malicious) user U^* interacting with the honest server.

We briefly describe the other tools, leaving the details to the full version. An encryption scheme, $E : (G_1, +) \to (G_2, \cdot)$ is *homomorphic* if for all $a, b \in G_1$, $E(a + b) = E(a) \cdot E(b)$. Informally, an encryption scheme is *non-malleable* if in addition to being semantically secure, it is impossible to generate a different ciphertext from a given ciphertext so that the two respective plaintexts are related. Intuitively, a *zero-knowledge proof of knowledge* allows a prover to convince a verifier of some fact in zero-knowledge if and only if the prover "knows something". This is formally captured by an efficient algorithm known as a knowledge extractor.

2.2 Notation

For an integer m, $[m]$ denotes the set $\{1, \dots, m\}$. Further, $2^{[m]}$ denotes the set of all subsets of $[m]$.

For a vector s, s_i refers to its ith coordinate, and if s_i is itself a vector, $s_{i,j}$ denotes the jth coordinate of s_i. For $i \leq j$, we let $s_{i,\dots,j}$ denote the $(j-i+1)$-tuple $(s_i, s_{i+1}, \dots, s_{j-1}, s_j)$.

For two strings or vectors s and t, let $s \circ t$ denote their concatenation. Let $|s|$ denote the length of s.

By $\eta(a, b)$ we denote an arbitrary negligible function, i.e., a function of a, b which is less than any inverse polynomial in a, b for a and b sufficiently large.

Two families of random variables U_n and V_n are computationally indistinguishable if for all probabilistic polynomial time (PPT) algorithms A,

$$|\Pr[A(U_n) = 1] - \Pr[A(V_n) = 1]| < \eta(n).$$

The notation $\tilde{O}$ suppresses terms that are polylogarithmic in the number of database records n.

3. THE MODEL

In this paper we restrict our attention to single-server computationally-private schemes. We first consider honest users, U, and the honest server, S. We consider malicious users U^* in our definition of inference control, and honest but curious[3] servers S^* in our definition of user privacy. We require all users and servers to be efficient probabilistic algorithms. For notational convenience, we assume all entries in

[3]Honest but curious means that, as a black box, the server behaves the same way as the honest server, but the server may keep all messages from a user and run arbitrary efficient algorithms on them.

the database are single bits, but our definitions easily extend to handle entries in $\{0,1\}^l$ for constant l.

A database is a string $x \in (\{0,1\}^m)^n$. x_i denotes the ith *record* of the database, and $x_{i,j}$ denotes the jth *attribute* value of the ith record. In our asymptotic analysis, we assume the number of attributes, m, is at most $O(\log\log n)$, whereas the number of records n is very large, as is the case for many relational databases. In the important case when the inference channels take a threshold form, or are otherwise easy to describe, we show how to dispense with this assumption on m. In fact in section 6 we give a best-possible PIC implementation for any m. However, even in section 6, we will assume m is at most polynomially larger[4] than n.

We assume that given the description of x, there is a mechanism for generating a collection $\mathcal{C}$ of sets $F \subseteq [m]$ denoting the inference channels in x. The meaning of $\mathcal{C}$ is that, for all $i \in [n]$ and $F \in \mathcal{C}$, the user should not learn $x_{i,j}$ for all $j \in F$. A tool such as the one in [30] can be used to determine $\mathcal{C}$. We take $\mathcal{C}$ to be an input to the server. We sometimes think of $\mathcal{C}$ and $F \in \mathcal{C}$ as sets, and other times as tuples.

As discussed in [15, 31, 30], a driving motivation is that we are concerned with information that is sensitive because it identifies individuals. We can think of an individual as a record. An individual is more likely to be identified the more of their attributes are revealed. For example, it's almost impossible to identify someone from zip code alone, but there might be a single person with a given zip code, car make and place of work, and if so, these attributes constitute an inference channel. Hence, our inference channels consist of subsets of attributes, and inference control ensures a user doesn't learn all the attributes in such a subset for any particular record. If one insists on an inference channel existing between some set S of distinct records in the database, one can merge the set S into a single record and apply our schemes with an efficiency cost.

Note that as is clear in the above example, we assume $\mathcal{C}$ is monotone, that is, if $A \in \mathcal{C}$ and $A \subseteq B$, then $B \in \mathcal{C}$. We also assume that $\mathcal{C}$ is an input to the *user*. Indeed, we cannot expect to hide $\mathcal{C}$ from the user since the user can see which queries are blocked, and thus learn information about $\mathcal{C}$. Note that $\mathcal{C}$ describes the *structure* of the database, and does not reveal any actual database entries.

There are two stages in a private inference control scheme: the offline stage and the online stage. In the offline stage the user and server are given their inputs, and the server is allowed to preprocess the database. In the online stage an honest user generates coordinates of a $|T|$-tuple, $T = ((i_1, j_1), \ldots, (i_{|T|}, j_{|T|}))$ one at a time, where (i_t, j_t) is allowed to depend on all previous interactions with the server. In the lth execution of the online stage an honest user should output x_{i_l,j_l}. The fact that the user generates coordinates of T one at a time reflects the fact that a user typically queries a database adaptively.

For simplicity we will only require correctness when T consists of distinct pairs. Hence, $|T| \leq (m-1)n$. However, all of our protocols can be easily modified to allow for repeat queries (see Section 7). Also, we give a direct construction of a PIC scheme in section 6 which handles both repeated queries and rejected queries. That is, even if a user has a query blocked because it is inference-enabling, they can continue to make future queries and retrieve database entries so long as these queries are not inference-enabling.

We call a query sequence T of distinct pairs *permissable* if it doesn't complete any inference channels, that is, if for all $F \in \mathcal{C}$ and all $i \in [n]$, there exists an $\ell \in F$ such that $(i, \ell) \notin T$. We can think of $T = T(U, x)$ (here U denotes the code of U) as a random variable induced by the uniform distribution on ρ and γ, where ρ and γ are random strings generated by the user and server, respectively. If U is honest, T assumes a particular permissable query sequence for fixed ρ and γ.

We proceed to define the algorithms and stages that a user and server run in any PIC protocol. In every algorithm, $n, m, \mathcal{C}$, and 1^k are inputs, where $k = k(n)$ is a security parameter. For notational convenience we do not include these inputs in the algorithms' descriptions.

Private Inference Control: A PIC protocol consists of the following stages and algorithms.

Offline Stage: The user gets a random string ρ and the server a random string γ. $|\rho|$ and $|\gamma|$ are polynomials in n and k. The same ρ (resp. γ) will be an input to every user (resp. server) algorithm for every interaction with the server (resp. user). We may assume all user and server algorithms are deterministic given ρ and γ. The server is given $x \in (\{0,1\}^m)^n$ and both parties are given $\mathcal{C} \in 2^{[m]}$. The server runs an efficient *preprocessing algorithm* $P(\cdot,\cdot)$, which outputs a string $y = P(x, \gamma)$.

Each execution of the online stage is allowed to be a multi-round protocol, and we let r denote the number of rounds per execution. Without loss of generality we may assume r is the same in every execution. In the wth round of the lth execution the user computes and sends a message $m_{w,l}$ to the server and the server computes and sends an answer $a_{w,l}$ back to the user. Define:

$$M_{w,l} = \{m_{i,j} \mid 1 \leq j < l,\ 1 \leq i \leq r, \text{ or } j = l,\ 1 \leq i \leq w\},$$

and

$$A_{w,l} = \{a_{i,j} \mid 1 \leq j < l,\ 1 \leq i \leq r, \text{ or } j = l,\ 1 \leq i \leq w\}.$$

Note that $M_{0,l} = M_{r,l-1}$ and $A_{0,l} = A_{r,l-1}$. During the protocol the user will have some state $\sigma_U(A_{w,l}, \rho)$ and the server some state $\sigma_S(M_{w,l}, x, \gamma)$ for some w and l. For instance, $\sigma_U(A_{w,l}, \rho)$ and $\sigma_S(M_{w,l}, x, \gamma)$ may contain the entire message history.

Online Stage : Let $Q(\cdot,\cdot,\cdot,\cdot)$, $D(\cdot,\cdot,\cdot,\cdot,\cdot)$ and $R(\cdot,\cdot,\cdot,\cdot,\cdot)$ be efficient algorithms. Q is the *query generator*, D the *database algorithm*, and R the *reconstruction algorithm*. The following process is run between a user U and server S:

- For $1 \leq l \leq |T|$,
  1. U generates $T_l = (i_l, j_l)$ for a permissable $T_{1,\ldots,l}$.
  2. For $1 \leq w \leq r$,
     (a) U computes $m_{w,l} = Q(i_l, j_l, \sigma_U(A_{w-1,l}, \rho), \rho)$ and sends $m_{w,l}$ to S.
     (b) S computes
     $$D(m_{w,l}, x, y, \sigma_S(M_{w-1,l}, x, \gamma), \gamma),$$
     which returns the pair
     $$(a_{w,l}, \sigma_S(M_{w,l}, x, \gamma)),$$

[4] Notationally this means that $\tilde{O}(\text{polylog } m) = \tilde{O}(1)$

and S sends $a_{w,l}$ to U.

(c) U computes $\sigma_U(A_{w,l},\rho)$ from

$$R(a_{w,l}, i_l, j_l, \sigma_U(A_{w-1,l},\rho),\rho),$$

and if $w = r$, U outputs $out_l \in \sigma_U(A_{r,l},\rho)$.

We define the *view*, $V_{U^*}(x,\rho,\gamma)$, of an arbitrary user, U^*, with respect to an arbitrary honest but curious server to consist of ρ, and $A_{r,|T|}$. We define the view, $V_{S^*}(U^*, x, \rho, \gamma)$, of an arbitrary server S^* with respect to an arbitrary user U^* to consist of x, γ, and $M_{r,|T|}$. As with the algorithm descriptions, $n, m, \mathcal{C}$, and 1^k are also part of U^* and S^* views.

Correctness and Security Definitions: The algorithms P, Q, D, R constitute a PIC protocol if the following hold for all n, m, k, and monotone $\mathcal{C} \in 2^{[m]}$:

1. Correctness: For all $x \in (\{0,1\}^m)^n$ and all honest users U,

$$\Pr_{\rho,\gamma}[\forall\alpha \in [|T(U,x)|],\ out_\alpha = x_{i_\alpha j_\alpha}] = 1.$$

2. User Privacy: For all $x \in (\{0,1\}^m)^n$, for all honest users U and any two sequences T_1, T_2 with $|T_1| = |T_2|$, for all γ and all honest but curious servers S^*, S^* can distinguish user query sequence T_1 from sequence T_2 with only negligible probability. That is, for all efficient (in n, 1^k) algorithms A,

$$|\Pr_\rho[A(V_{S^*}(U,x,\rho,\gamma)) = 1 \mid T(U,x) = T_1]$$

$$-\Pr_\rho[A(V_{S^*}(U,x,\rho,\gamma)) = 1 \mid T(U,x) = T_2]| < \eta(n,k),$$

where the probability is also taken over the coin tosses of the adversary's algorithm A, where A is additionally given $n, 1^k, m$, and $\mathcal{C}$.

3. Inference Control: We make a comparison with the ideal model in which a trusted third party, Charlie, when given $x \in (\{0,1\}^m)^n$ and a *permissable* T, gives the user $x_{i,j}$ for all pairs $(i,j) \in T$. More precisely, we require that for every U^* and every ρ, there exists an efficient U', given the code of U^* and ρ, that plays U^*'s role in the ideal model, such that for every $x \in (\{0,1\}^m)^n$, U' can find a permissable T such that the output of U^* interacting with the honest server and any the output of U' interacting with Charlie given x and T are computationally indistinguishable. Note that U' is also allowed to make queries adaptively, so T may depend on x. In general, the T found by U' is a random variable induced by the uniform distribution on γ. Observe that the inference channels, $\mathcal{C}$, are known to both U^* and U' and are part of their views.

We phrase our definition in terms of a single user for simplicity. Clearly, for collusion-resistance it does not suffice to protect against inferences by a single user. It is straightforward to modify our protocols to protect against collusion – we simply permit each user to access less of each channel. The same technique is used in [34] to ensure collusion-resistant inference control without privacy, however it is easily adopted in our setting as it only depends on server knowledge of the number of queries a user has made, and *not* the content of the queries. Adjusting the definition is similarly straightforward, one would instead consider malicious users, $U_1^*, \ldots, U_c^*$, where c is the desired level of collusion resistance. We revisit collusion resistance in Section 7.

Since our goal is to ensure that a malicious user doesn't learn any unintended information, the above definition is similar to what is termed "sender's security" in [27] and "vendor's security" in [2]. However, we differ from both notions in that inference channels are a more general concept than the query counts in [27] and the query prices in [2], that determine access controls in those works. We could also define inference control and user privacy to hold with respect to non-uniform adversaries (and in fact our schemes can be made secure against such adversaries), but for simplicity, we do not take that up here.

Efficiency: For fixed n, m, and k, we measure the efficiency of a PIC protocol as follows. All of our efficiency measures are *per query*, that is, for a single execution of the online stage. The *communication complexity* $C(n,m,k)$ is defined to be: $\max_{\mathcal{C},\ x,\ U,\ \gamma,\ \rho,\ l} \ \sum_{w=0}^{r} (|m_{w,l}| + |a_{w,l}|)$. The *server work* of a PIC protocol $W(n,m,k)$ is the max, over all $\mathcal{C}, x, U, \gamma, \rho$, and l, of the sum of the running times of $D(m_{w,l}, x, y, \sigma_S(M_{w-1,l}, x, \gamma), \gamma)$ over w. The *round complexity* of a PIC protocol $R(n,m,k)$ is r.

4. A GENERIC REDUCTION

We show that it suffices to focus on developing protocols that meet the definition of PIC when the inference channels, $\mathcal{C}$, take a particularly simple form. We note that for m larger than $O(\log\log n)$, this reduction is not particularly efficient. Thus, in Section 6 we directly construct a PIC scheme with an exponential savings over that achieved via this reduction.

Definition 1. *A protocol is a* **threshold private inference control scheme***, abbreviated TPIC, if it satisfies the requirements of a private inference control scheme under the assumption that* $\mathcal{C} = \{[m]\}$.

Note that a TPIC need not satisfy the requirements of a PIC for $\mathcal{C} \neq \{[m]\}$. However, the following theorem shows that given any TPIC, we can construct a PIC with related efficiency (for m not too large):

Theorem 1. *Let (P', Q', D', R') be a TPIC with communication $C'(n,m,k)$, server work $W'(n,m,k)$, and round complexity $R'(n,m,k)$. Then there exists a PIC scheme (P,Q,D,R) with communication $C(n,m,k) = m2^m C'((m+1)2^m n, m, k)$, $W(n,m,k) = m2^m W'((m+1)2^m n, m, k)$ work, and $R(n,m,k) = m2^m R'((m+1)2^m n, m, k)$ rounds.*

Proof Sketch: The idea behind the proof is to make a new record for each channel in an original record. The new records have variable size between 1 and m, so we pad them with random bits so that they all have the same length. Bits in the pad are XORed to values in the record in order to require the record be queried m times to learn the channel it represents. Because of the expansion of each record, an attribute value may now occur in multiple records. We force the user to read all records containing this element by using secret sharing to "split" the element across each of these records. The only way to recover the element is to query all records containing it. The user and server of the new PIC scheme essentially simulate the user and server of the TPIC by running multiple online executions of the TPIC per single

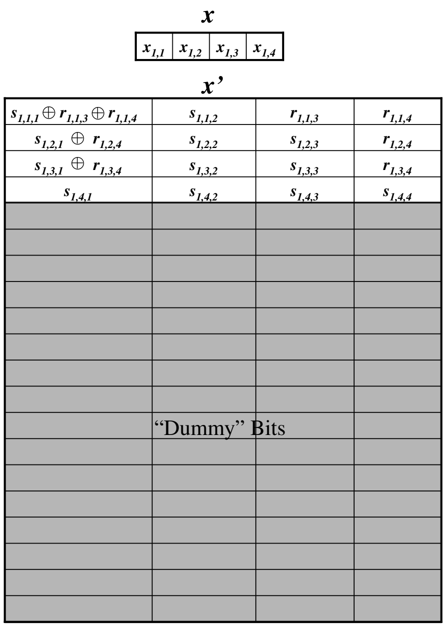

Figure 1: For a database, x, with $n = 1$, $m = 4$, and $\mathcal{C} = ((1,2),(1,2,3),(1,2,4),(1,2,3,4))$, the new database x' (depicted as a $|\mathcal{C}|(m+1)n = 20 \times 4 = m$, binary matrix), produced by the generic reduction. Note that $s_{1,4,1} \oplus s_{1,3,1} \oplus s_{1,2,1} \oplus s_{1,1,1} = x_{1,1}$, $s_{1,4,2} \oplus s_{1,3,2} \oplus s_{1,2,2} \oplus s_{1,1,2} = x_{1,2}$, $s_{1,4,3} \oplus s_{1,2,3} = x_{1,3}$ and $s_{1,4,4} \oplus s_{1,3,3} = x_{1,4}$.

execution of the PIC. The formal proof can be found in [36] and a small example of the reduction is in Figure 1.

Note that to obtain $x_{1,1}$ in the figure, 8 entries of x' must be probed, while to obtain $x_{1,4}$, only 2 entries need to be probed. The dummy bits in the figure help give user privacy by ensuring that the number of rounds (i.e., probes) per online execution is always the same. The calculation of the number of dummy bits needed is given in the full version.

Again, we stress that this is a general theorem which constructs a PIC out of any TPIC, with complexity exponential in m. Note that $|\mathcal{C}|$ itself may be exponential in m, so this reduction is efficient for general $\mathcal{C}$. In Section 6 we provide a protocol with better complexity though, which will be useful in the multi-user setting when m may be large.

5. PIC WITH PRIVATE REVOCATION AND LINEAR WORK

One of the main difficulties in constructing PIC protocols is that the large number of permissable query sequences puts upward pressure on the size of the SPIR tables employed, and thus, on the communication complexity. The protocol of this section manages communication cost by allowing the server to store encrypted information about each user's queries. This has several significant benefits. First, in this protocol the server stores *encrypted* query histories. This facilitates user revocation as an adjustment to changes in the inference channel. If, for example, it is determined that certain information in the database is sufficiently sensitive that a user who has accessed it should no longer be permitted to query the database this can easily be privately "checked" by the database and all future queries by such a user rebuffed. User privacy is preserved since the server doesn't know whether the user passed or failed the check. Further, the presence of the encrypted query histories makes it easy to adapt to changes in the inference channels (likely with a dynamic database). In addition, the protocol keeps the server's work at essentially an optimal level, $O(k^2 n)$, where k is a security parameter taken to be $\tilde{O}(1)$ in the following.[5] Note that PIC necessarily implements PIR, and the best known single-server PIR scheme, even with preprocessing, uses $\Omega(n)$ work per query.

The protocol makes use of a homomorphic encryption function, $E^{hom}(\cdot)$ (instantiated with Paillier's scheme [28], for example). We describe the protocol as a TPIC, and apply Theorem 1 to obtain a PIC scheme.

PROTOCOL 1. *Let U's t^{th} query be $T_t = (i_t, j_t)$, $1 \le i_t \le n$, $1 \le j_t \le m$ and let $\mathcal{C} = \{[m]\}$. We describe how this TPIC protocol proceeds on the t^{th} query.*

1. *If $t = 1$, U generates public and private keys for $E^{hom}(\cdot)$ and sends the public key to the server.*
2. *Sending query information: U sends the server $E^{hom}(i_t)$ and $E^{hom}(j_t)$ and gives a zero-knowledge proof of knowledge [6] that the ciphertexts are well-formed.*
3. *Generating authorization:*
   - (a) *If $t < m$ then the server sets $\mathcal{S}_t = 0$ and proceeds to step 4. Else, the server generates a secret $\mathcal{S}_t$ and randomly generates shares, $y_1, \ldots, y_{t-1}$, forming a $(t-m+1)$-out-of-$(t-1)$ sharing of $\mathcal{S}_t$ [32] and sends the user*
   $$E^{hom}((i_1 - i_t)y_1), \ldots, E^{hom}((i_{t-1} - i_t)y_{t-1})$$
   using the homomorphic property of $E^{hom}(\cdot)$.
   - (b) *U decrypts and recovers at least $t-m+1$ of the values in $\{y_1, \ldots, y_{t-1}\}$ if U has made a permissible sequence of queries, and thus is able to reconstruct the secret $\mathcal{S}_t$.*
4. *Query Processing:*
   - (a) *The server generates random values, $v_{i,j}^{(1)}, v_{i,j}^{(2)}$, for every $1 \le i \le n$, $1 \le j \le m$, and forms the table with nm entries $\mathcal{T} = (E^{hom}(v_{i,j}^{(1)}(j - j_t) + v_{i,j}^{(2)}(i - i_t) + \mathcal{S}_t + x_{ij}))_{i,j}$.*
   - (b) *U and the server engage in the SPIR protocol on $\mathcal{T}$. In order to be able to recover $x_{i,j}$, U must know $\mathcal{S}_t$ and the following equalities must hold: $i = i_t$ and $j = j_t$. Hence, U can at most recover x_{i_t,j_t}, and can only do so if U has made a permissible sequence of queries.*

We discuss the key points pertaining to the protocol's security here and defer formal proofs to the full version [36]. Inference control intuitively holds because step 2 ensures

[5]We are ignoring logarithmic factors and using $\tilde{O}(1)$ for $k = \log^{O(1)}(n)$, as is done for security against PPT algorithms for the best known PIR/SPIR schemes [10, 25, 26]. Since k will also be a security parameter for a homomorphic encryption scheme E^{hom}, we asume $k = \tilde{O}(1)$ is enough for semantic security. If one instantiates E^{hom} with say, Paillier's scheme, such is the case w.r.t. known attacks. More security is possible by replacing $k = \tilde{O}(1)$ with $k = n^{\epsilon}$ for any $\epsilon > 0$.

that a user who is in danger of completing an inference channel won't be able to recover $\mathcal{S}_t$ because in step $2(a)$, for at least $m-1$ values of j, $E^{hom}((i_j - i_t)y_j) = E^{hom}(0)$. The use of encryption in the SPIR phase[6] ensures that the user will only be able to decrypt entries in the record the user is authorized to query. Note that this scheme has the property that a user who is blocked from making an inference-enabling query *will* be able to continue to receive information from the database (provided subsequent queries are permissible given past ones). User privacy relies upon the SPIR protocol and the strength of the encryption schemes.

The per query communication cost is $O(kt)$ plus the cost of a SPIR invocation. As mentioned in Section 1.2, the $O(kt)$ part of the cost can be removed with a decrease in user privacy, by using techniques for search on encrypted data [33, 8, 19] to enable the server to determine whether or not a server is authorized by searching the user's encrypted query history, and thus removing step 2. Server work is essentially optimal in this protocol at $O(k^2 n)$ with the bulk of the work occurring in step 3, and the protocol can be implemented in one round.

Applying Theorem 1 turns this TPIC scheme into a PIC scheme with $\tilde{O}(n)$ server work per query and $\tilde{O}(t)$ communication per query, which may be $\tilde{O}(n)$, depending on t.

Finally we note that since the server stores the encrypted query histories it is simple to accommodate user revocation based on the user's history. As an extreme example, if it is determined that access to $x_{i,j}$ is sufficiently valuable that a user with that information should no longer be allowed to access the database, then the server can easily "check" for $E(i)$ and $E(j)$ using a similar secret sharing technique to that in step 3. This does not immediately compromise the user's privacy as the server does not know that the user was unable to recover the secret, although the user should maintain a consistent query pattern in order to ensure that the server does not learn they were revoked.

6. ONE-ROUND PIC WITH $\tilde{O}(1)$ COMMUNICATION AND LINEAR WORK

In this section we present, up to logarithmic factors, a PIC protocol that is simultaneously optimal in its communication complexity, round complexity, and server work per query. Again, as in Section 5, we assume the security parameter $k = \tilde{O}(1)$.[7] Our protocol achieves communication complexity $\tilde{O}(1)$, uses 1 round, and has $\tilde{O}(n)$ work per query. This is optimal in all three aspects with respect to known PIR schemes, even with preprocessing [5]. As with the previous protocol, this protocol allows users to continue to make queries even if some of their queries were blocked in the past. We also show how to directly handle repeat queries, but we note that the previous protocol can do this too via a general technique given in section 7. In addition, although it prohibits the revocation capabilities of the previous protocol, it may be preferable because of its essentially stateless nature (and also its improved complexity).

Without loss of generality, we assume n is a power of 2. We use a data structure B consisting of a balanced binary tree with n leaves, where in addition, we connect m children to each leaf of the binary tree. The idea is that the leaves of B denote entries $x_{i,j}$ of the database, and the parents of the leaves denote records x_i. U will obtain keys $K(w, z)$ associated with each leaf w indicating whether or not the value at the leaf has been accessed. Here $z \in \{0, 1\}$. Internal nodes w also have associated keys $K(w, z)$, where here z is an integer indicating the total number of times leaves in the subtree rooted at w have been accessed. These keys will be used to do inference control. When a user U retrieves a database entry they use their keys to go up the tree. If U tries to use "older" keys indicating that nodes have been accessed less than they actually have, their keys will be inconsistent with S's knowledge of the total number of queries made thus far, and U will not be able to recover the desired database entry.

Before going into detail, let us fix some notation. Let α denote the root of B. We say $w \in B$ is at height d if it is d levels above the leaves. The leaves are at height 0. Each node in B of height 1 is denoted by i for some $i \in [n]$, and each of the m children of i are denoted by (i, j) for some $j \in [m]$ (we sometimes use w to refer to an arbitrary node in B, including those at height 0 and 1). For a non-root node w in B, let $\mathrm{sib}(w)$ denote w's siblings (there is either 1 or $m-1$). For a non-leaf node w, let $\mathrm{children}(w)$ denote w's children. For a leaf node w, let $\mathrm{anc}(w)$ denote the set of $\log n + 1$ ancestors along the path from w to α, inclusive. We say a node w is *accessed* whenever $x_{i,j}$ is retrieved by the user for which $w \in \mathrm{anc}(i, j)$. Finally, for leaves w define the set of $2 \log n + m - 1$ nodes that is the set of ancestors (as defined above) together with the siblings of the ancestors:

$$\mathrm{sibanc}(w) = \mathrm{anc}(w) \cup \{u \mid \exists v \in \mathrm{anc}(w) \text{ s.t. } u = \mathrm{sib}(v)\}$$

6.1 Protocol Description

When the honest user queries $x_{i,j}$, he will use the set of keys $\pi = \{K(w, f_w) \mid w \in \mathrm{sibanc}(i, j)\}$, where f_w is the number of times w has been accessed. If the user is dishonest, for some $w \in \mathrm{sibanc}(i, j)$, he may substitute $K(w, z)$ in place of $K(w, f_w)$ for some integer $z \neq f_w$. However, the invariant we maintain is that with all but negligible probability, any user U^* cannot determine $K(w, z)$ for any $z > f_w$, and so if $K(w, z)$ is substituted for $K(w, f_w)$, it holds that $z < f_w$. If the user is given $x_{i,j}$, he will also obtain the *updated* set of keys $\{K(w, f_w + 1) \mid w \in \mathrm{sibanc}(i, j)\}$.

Our scheme enforces inference control by what we call *sum-consistency*. For any non-leaf node w and its children, denoted $\mathrm{children}(w)$, we say the keys $K(w, i), \{K(u, j_u) \mid u \in \mathrm{children}(w)\}$ are *sum-consistent* if $i = \sum_{u \in \mathrm{children}(w)} j_u$. Suppose the honest user U wants to retrieve $x_{i,j}$ on the $(t+1)$st query. We claim the set of keys π gives a *proof* that U is not in danger of completing an inference channel. Indeed, if U is honest, π has the following three properties:

1. For each non-leaf node w in $\mathrm{anc}(i, j)$, $K(w, f_w)$ and $\{K(u, f_u) \mid u \in \mathrm{children}(w)\}$ are sum-consistent.
2. $f_\alpha = t$.
3. If U is not in danger of completing an inference channel

[6] PIR may suffice instead of SPIR. We use the stonger primitive SPIR to facilitate the security proof (see [36])

[7] This is possible for security against PPT algorithms for the best known PIR/SPIR schemes (e.g., transforming the PIR protocol of [10] to a SPIR protocol using [26]) and for the symmetric non-malleable encryption E used in our protocol. For example, w.r.t. known attacks one can securely use Cramer-Shoup [13] based on either the Paillier assumption or Decisional-Diffie-Hellman assumption with $k = \tilde{O}(1)$. In our protocol we only need E to be symmetric-key with plaintext domain cardinality $\Theta(\log n)$. See [16] for more information.

by learning $x_{i,j}$, then for all inference channels $F \in \mathcal{C}$, there is some $j' \in F$ such that $j' \neq j$ for which $K((i,j'),0) \in \pi$.

A dishonest user U^* will not be able to furnish a proof π to obtain $x_{i,j}$ when learning $x_{i,j}$ completes an inference channel F. Indeed, if U^* does not substitute $K(w,z)$ for $K(w,f_w)$ for some $z \neq f_w$ and $w \in \mathrm{sibanc}(i,j)$, the third property above cannot hold. On the other hand, by the invariant mentioned above, if U^* substitutes $K(w,z)$ for $K(w,f_w)$ for some $z \neq f_w$ for some w, then z is necessarily less than f_w so properties (1) and (2) cannot hold simultaneously.

Of course, for user privacy, U cannot simply give π to S. Instead, U will prove knowledge of π via a secure function evaluation (SFE) [22, 37] with S. The idea is that U will input π to the SFE, which will give the user a certain secret if and only if π is a valid proof. Now the communication complexity of an SFE for a function is linear in the circuit size computing the function, and therefore, if the keys $K(w,i)$ are completely independent of each other, the communication complexity of a generic SFE for π will be $\Omega(mn)$, which is prohibitive. Indeed, since the server cannot know which keys the user will use because of user privacy, and since the keys are independent random values, the server must input all possible user keys [8] into the circuit so the circuit can perform comparisons on them.

To get by with low communication, we observe that the keys associated with different nodes need not be totally independent of one another, as long as they are not predictable from each other. Let E be a non-malleable [16] symmetric-key encryption scheme, and let K be a secret key of E known only to S. Then for each node w and integer i, we define the key $K(w,i)$ to be the encryption $E_K(w,i)$ under key K. Since E is non-malleable, $E_K(w,i)$ is unpredictable from $E_K(v,j)$ for $(w,i) \neq (v,j)$. The point here is that the server just needs to input the decryption key K of E to the secure circuit in order to perform the comparisons on the user's keys, rather than $\Omega(mn)$ truly random keys.

We would also like to handle repeated and rejected queries. To this end the user will have keys $E_K(``reject'', z_1)$ and $E_K(``repeat'', z_2)$ for integers z_1 and z_2. The idea is that z_1 denotes the total number of rejected queries made so far, and z_2 denotes the total number of queries that were repeats. Since E is non-malleable and K is unknown to U, U cannot construct/modify such encryptions. These encryptions will be updated by the server "obliviously" in the secure function evaluation we now describe.

The user's input to the circuit, C, is a set of keys π', an $i \in [n]$, a $j \in [m]$, and two k-bit numbers P, Q, which if U is honest, denote $E_K(``reject'', z_1)$ and $E_K(``repeat'', z_2)$ for some integers z_1, z_2. The server input is K, the number of queries made so far t, theinference channels $\mathcal{C}$, and a seed s to a pseudorandom function h.

The circuit functionality C first checks if $E_K((i,j),1) \in \pi'$, which it can do efficiently since it knows K. If this is the case, C checks if Q is of the form $E_K(``repeat'', z_2)$ for some integer $z_2$, and if so, outputs $E_K(``repeat'', z_2+1)$ to the user by decrypting and re-encrypting. If Q is not of this form, C outputs $E_K(``repeat'', 1)$. In either case C additionally gives $h(s,i,j)$ to U. Otherwise, if $E_K((i,j),1) \notin \pi'$, C checks that the keys in π' satisfy the properties of a valid proof π, that is, they form a sum-consistent path from (i,j) to α. Note that, we replace the property that $f_\alpha = t$ with the property that $f_\alpha = t - z_1 - z_2$, since the user could not have updated his keys in the $z_1 + z_2$ queries which were rejects or repeats. Note that C can do this efficiently by decrypting the keys in π' using K along with simple arithmetic operations.

Let $g(\pi')$ denote the set of keys $\{K(w, f_w+1) \mid K(w,f_w) \in \pi'\}$. If π' passes the above test, the user output of C is $h(s,i,j)$ and $g(\pi')$. If π' does not pass the test, C checks if P is of the form $E_K(``reject'', z_1)$ for an integer $z_1$, and if so, outputs $E_K(``reject'', z_1+1)$ to the user. Otherwise, C outputs $E_K(``reject'', 1)$ to the user. In either case, the server has no output. Since the input size to C is $\tilde{O}(|\mathcal{C}|+m)$ (for $|K|, |s| = \tilde{O}(1)$), the total communication and server work required for this step is $\mathrm{poly}(|\mathcal{C}|, m)$ [22, 37] since C is an efficient computation. Note that, by outputting $E_K(``reject'', 1)$ and $E_K(``repeat'', 1)$ we don't need the user to already know $E_K(``reject'', 0)$ and $E_K(``repeat'', 0)$ at the beginning of the protocol.

The following is the complete protocol:

Protocol 2. *In the offline stage the server S randomly chooses a seed s to a pseudorandom function h and a key K to a non-malleable encryption scheme E. In the t-th execution of the online stage, the user U first generates a query (i_t, j_t). Then U and S perform the following SFE and SPIR reading in parallel. These two steps encompass the query generation algorithm Q, database algorithm D, and reconstruction algorithm R discussed in section 3:*

1. *Let C be a secure circuit implementing the functionality described above. U constructs the set of keys $\pi = \{K(w,f_w) \mid w \in \mathrm{sibanc}(i_t,j_t)\}$, and feeds these with keys $P = E_K(``reject'', z_1)$ and $Q = E_K(``repeat'', z_2)$ into C, where z_1, z_2 denote the number of rejected and repeated queries made thus far. If no such queries have been made in one of these two cases (i.e., z_1 or z_2 is 0), then U substitutes a random value in the range of E in its place. S then feeds s, K, and the inference channels $\mathcal{C}$ into C. S gets no output from C, while U's output is divided into cases (recall the protocol description refers to honest U and S):*
   - (a) *If learning x_{i_t,j_t} is inference-enabling, U's output is $E(``reject'', z_1+1)$.*
   - (b) *If x_{i_t,j_t} was previously queried, U's output is*
   $$E(``repeat'', z_2+1), h(s,i_t,j_t).$$
   - (c) *Otherwise U's output is $h(s,i_t,j_t)$ and the updated keys $\{K(w,f_w+1) \mid K(w,f_w) \in \pi\}$.*
2. *S prepares a table T with i,jth entry set to $x_{i,j} \oplus h(s,i,j)$ and U and S engage in SPIR on records on T. U should obtain T_{i_t,j_t}.*

If learning x_{i_t,j_t} is not inference-enabling, U reconstructs x_{i_t,j_t} from $h(s,i_t,j_t)$ and $T_{i_t,j_t} = x_{i_t,j_t} \oplus h(s,i_t,j_t)$.

6.2 Efficiency Wrap-up

Since server work and communication are dominated by the SPIR invocation, we see that the overall server work of our scheme is $\tilde{O}(mn)$ and uses $\tilde{O}(\mathrm{poly}(m, |\mathcal{C}|))$ bits of communication. Note that m may be $O(\log\log n)$ without

[8] Of course, the server can use its knowledge of the number of queries made thus far to exclude some keys from the circuit input. This does not asymptotically help though, once the user makes enough queries.

asymptotically affecting the work or communication, and that this is best possible, since $|\mathcal{C}|$ may be as large as 2^m in any inference control scheme. Finally we note that both the secure circuit evaluation and the SPIR invocation each take one round (with say, using the circuit evaluation of [9]), and further, they can be parellelized. Hence the whole scheme can be done in a single round.

In the special case where $\mathcal{C}$ is of the form $\{S \subset [m] \mid |S| \geq \tau\}$, i.e., a threshold access structure, $|\mathcal{C}| = \log m$ as we just need to describe τ. In this case we can modify our scheme so that the communication is $O(\text{polylog}(mn))$ and the work remains at $\tilde{O}(mn)$. To do this, instead of having m keys for each attribute for a given record i, contributing a factor of m to the communication of the SFE, we instead have a single key for each record i encoding the total number of attributes in i which have been queried. Note that by definition of $\mathcal{C}$, the actual attributes queried are immaterial. This can be done as long as the key size is larger than $\log m$ bits, and hence the communication drops to $O(\text{polylog}(mn))$. We can still handle repeat queries by distributing a special key with each attribute of i, which can be an additional input to the secure circuit evaluation and checked to see if a repeat is being made. Again, this is just one additional key per secure circuit evaluation, so we get an exponential (in m) savings in communication. This allows m to be polynomial in n without asymptotically affecting the communication.

Finally observe that the server uses only $\tilde{O}(mn)$ space and $\tilde{O}(1)$ bits of randomness.

6.3 Security proofs

We give proof sketches that the above protocol meets the requirements of a PIC protocol:

Correctness: This is straightforward and is omitted.

User Privacy: Observe that an honest but curious server S^*'s view is that of a polynomial (in n) number of SFE and SPIR interactions with an honest U. Note that there are efficient simulators S_1 for the SFE [22, 37] and S_2 for the SPIR (to construct S_2, simply fix an arbitrary input for U, and simulate the SPIR interaction of U with S^* for that input). Since S^* is honest but curious, its inputs (importantly x) are fixed at the beginning of the protocol, and since S^* gets no output from either the SFE or the SPIR, its view is just the concatenation of the outputs of S_1 and S_2 evaluated on its inputs at the beginning of the protocol. This defines a simulator for the view of S^*, which implies user privacy. For more background see [21].

Inference Control: The proof will use two efficient algorithms S_1 and S_2 which we refer to as simulators. Given the code of a malicious user U^* and his inputs to the SFE in our protocol, it holds that S_1 can efficiently generate a new set of inputs π', i, j, P, Q for which the view of U^* in the SFE of our protocol is computationally indistinguishable from the view of a user who just hands π', i, j, P, Q to a trusted third party who, given the server's inputs, computes the functionality of the SFE and hands the user the response (see [21], comment after Definition 2.1.6). Secondly, using the PIR to SPIR transformation of [26], it holds that S_2, given the code U^* and his inputs to the SPIR subprotocol in our protocol, can extract the indices requested (explicitly or implicitly) by U^* with the server.

Now let U' play U^*s role in the ideal model. One issue that arises is that as U^* makes more queries, his "knowledge" of the database and his set of keys increases. These new values are inputs of U^* in future online executions. Thus to apply S_1 and S_2, it must be the case that U' can efficiently derive the inputs of U^* to these subprotocols from U^*s knowledge. This is not a problem, though, as it will inductively hold that U' can efficiently derive the entire view of U^* from the outputs U^* receives in prior online executions (known also to U') and U^*s randomness ρ, and thus we may think of U^* as an algorithm whose inputs are just the outputs received from prior executions, together with ρ.

U' first chooses a random key K for E and a seed s for the pseudorandom function h. Suppose after t queries (including repeated/rejected queries), the view generated by U' is computationally indistinguishable from that generated by U^* interacting with S. By induction, we may assume that U^* has only a negligible advantage of predicting $E_K(w, j)$ for any $w \in B$ and any $j > f_w$. This holds in the base case (information-theoretically) since U^* doesn't know K. Again by induction, we may assume that U^* can only generate $E_K(\text{"}reject'', j), E_K(\text{"}repeat'', j)$ for j less than or equal to the total number of rejected/repeated queries thus far. We describe U''s simulation on the $(t+1)$st query, and show the above inductive steps.

By the inductive hypothesis, U' can run S_1 to extract U^*s effective inputs π', i, j, P, Q. Given π', i, j, P, Q, we now show how U' can produce a random variable Out which is computationally indistinguishable from the output the trusted third party Tr of the SFE returns in the ideal model.

Given π', i, j and the previously requested indices I of U' to the trusted third party Charlie (of the overall PIC protocol), U' essentially simulates the SFE functionality, setting $Out = E_K(\text{"}reject'', z+1)$ if $x_{i,j}$ together with values indexed by I are inference-enabling, where $P = E_K(\text{"}reject, z)$. Similarly for a repeat query, U' sets $Out = E_K(\text{"}repeat'', z+1), x_{i,j} \oplus h(s, i, j)$ if $Q = E_K(\text{"}repeat'', z)$, and since it is a repeat query, U' already knows $x_{i,j}$ from a previous interaction with Charlie.

The most interesting case is if (i, j) is neither inference-enabling nor already in I, U' checks if π' has the three properties of π given in the protocol description. Here, again, we replace the second property "$f_\alpha = t$" with the property "$f_\alpha = t - z_1 - z_2$ ", where $P = E_K(\text{"}reject'', z_1)$ and $Q = E_K(\text{"}repeat'', z_2)$. If π' has these three properties, U' updates I to include (i, j), asks the trusted third party Charlie for $x_{i,j}$, and sets $Out = x_{i,j} \oplus h(s, i, j), g(\pi')$, where $g(\pi')$ is the updated list defined in the protocol description. Otherwise U' sets $Out = E_K(\text{"}reject'', z+1)$ or $Out = E_K(\text{"}reject'', 1)$ depending if P is a valid encryption.

Clearly Out is indistinguishable from the output of Tr, since $h(s, i, j)$ and $x_{i,j} \oplus h(s, i, j)$ are both pseudorandom (over s). We give the user $x_{i,j} \oplus h(s, i, j)$ as the output of the SFE instead of $h(s, i, j)$ so that U' doesn't have to talk to Charlie when simulating the SPIR interaction, and hence, I, as defined, is exactly the set of database values the server has already queried from Charlie.

U' also runs the simulator S_2 for SPIR to extract the requested indices (i', j') and simulates the output of U^* based on $h(s, i', j')$ and U^*s inputs (which are known to U' inductively). Since the SFE and SPIR are parallel, one-round protocols, the view of U^* is just the concatenation of the view generated from Out, together with the view generated from $h(s, i', j')$. This shows that the view generated by U' is indistinguishable from that in an actual execution. Note that U' keeps Out and (i', j') for future executions.

It remains to show the inductive steps for the claims in the second paragraph. Loosely speaking, these follow from the fact that $E_K(\text{"}reject\text{"}, z), E_K(\text{"}repeat\text{"}, z)$, and node keys are updated when they should be, and only the encryptions U^* knows depend on K, but these do not help form new encryptions since E is non-malleable.

7. EXTENSIONS AND OPEN PROBLEMS

We've described our protocols in the context of a single user, however they naturally support many users. In the multi-user setting the database server only needs the ability to link queries made by the same user and so it's possible for users to maintain some anonymity through the use of pseudonyms. In addition, collusion resistance can easily be added to any of our protocols. To ensure c-collusion resistance for an m-channel, meaning no c users can collude to complete the channel, we modify the protocols to only allow a user to query a record $\frac{(m-1)}{c}$ times. With this modification, c users can together query at most $m-1$ attributes in an inference channel. Although, this may restrict information access quite a bit if the database isn't queried much, it does ensure collusion resistance in a fair manner.

As mentioned in Section 3, a PIC protocol can be generically modified to allow for repeat queries. This is achieved by distributing a "master key", $K_{i,j}$, with each $x_{i,j}$ and requiring the server to maintain a table containing each $x_{i,j}$ encrypted under $K_{i,j}$. A user can engage with the server in SPIR protocols on this table at will. If the user wants to protect against the server learning that a repeat query is made, they can perform the repeat table lookup on every query, with at most a constant factor slowdown in efficiency, together with a query on the actual database.

Finally, as this is a new primitive there are a number of open questions. Perhaps there are applications of PIC, similar to those in [2], that lie outside of the realm of protecting sensitive subsets of information. Simpler implementations without a secure circuit would also be interesting.

Acknowledgements

The authors thank Benny Pinkas for helpful discussions and Diana Smetters for suggesting this problem. Much of this work was supported by DARPA contract F30602-03-C-0037. The first author was supported by an NDSEG fellowship towards the end of this work.

8. REFERENCES

[1] R. Agrawal and R. Srikant. *Privacy-preserving data mining.* In ACM SIGMOD Conference on Management of Data, pp. 439-450, 2000.

[2] W. Aiello, Y. Ishai and O. Reingold. *Priced oblivious transfer: how to sell digital goods.* In Advances in Cryptology - Eurocrypt '01.

[3] L. Beck. *A security mechanism for statistical databases.* In ACM TODS, pp. 316-338, 1980.

[4] A. Beimel, Y. Ishai, E. Kushilevitz and T. Malkin. *One-way functions are essential for single-server private information retrieval.* In proceedings of STOC '99.

[5] A. Beimel, Y. Ishai and T. Malkin. *Reducing the Servers Computation in Private Information Retrieval: PIR with Preprocessing.* In Advances in Cryptology – Crpypto 2000.

[6] M. Bellare, O. Goldreich. *On Defining Proofs of Knowledge.* In Advances in Cryptology – Crypto 1992.

[7] M. Blum. *Three applications of the oblivious transfer: Part i: Coin flippinf by telephone; part ii: How to exchange secrets;part iii: How to send electronic mail.* Department of EECS, University of California, Berkeley, CA, 1981.

[8] D. Boneh, G. Di Crescenzo, R. Ostrovsky and G. Persiano. *Searchable public key encryption.* To appear in Advances in Cryptology – Eurocrypt 2004. http://eprint.iacr.org/2003/195/

[9] C. Cachin, J. Camenisch, J. Kilian and J. Mller. *One-round secure computation and secure autonomous mobile agents.* In Ugo Montanari, Jos P. Rolim, and Emo Welzl, editors, Proc. 27th International Colloquium on Automata, Languages and Programming (ICALP), volume 1853 of Lecture Notes in Computer Science, pages 512-523. Springer, 2000.

[10] C. Cachin, S. Micali and M. Stadler. *Computationally private information retrieval with polylogarithmic communication.* In Advances in Cryptology – Eurocrypt '99.

[11] S. Castano, M. Fugini, G. Martella and P. Samarati. *Database Security.* ACM Press, 1995.

[12] B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan. *Private information retrieval.* In proceedings of FOCS '95.

[13] R. Cramer and V. Shoup. *Universal hash proofs and a paradigm for chosen ciphertext secure public-key encryption.* In Advances in Cryptoloy – Eurocrypt 2002.

[14] G. Di Crescenzo, T. Malkin and R. Ostrovsky. *Single database private information retrieval implies oblivious transfer.* In Advances in Cryptology – Eurocrypt '00.

[15] I. Dinur and K. Nissim. *Revealing information while preserving privacy.* In proceedings of PODS 2003.

[16] D. Dolev, C. Dwork, and M. Naor. *Non-malleable cryptography*, SIAM. J. Computing, Vol. 30, No. 2, 2000, pp. 391–437.

[17] S. Even, O. Goldreich and A. Lempel. *A randomized protocol for signing contracts.* In Communications of the ACM, 1985.

[18] C. Farkas and S. Jajodia. *The inference problem: a survey.* In SIGKDD Explorations, December 2002.

[19] E. Goh. *How to search efficiently on encrypted compressed data.* In the Cryptology ePrint Archive, Report 2003/216, October 7, 2003. http://eprint.iacr.org/2003/216/

[20] Y. Gertner, Y. Ishai, E. Kushilevitz and T. Malkin. *Protecting data privacy in private information retrieval schemes.* In proceedings of STOC '98.

[21] O. Goldreich. *Secure Multi-Party Computation*, 1998. Available at http://philby.ucsd.edu/

[22] O. Goldreich, S. Micali, and A. Wigderson. *How to Play Any Mental Game.* In proceedings of 19th STOC, pp. 218-229, 1987.

[23] T. Keefe, M. Thuraisingham, W. Tsai. *Secure query-processing strategies.* IEEE Computer, Vol. 22, No. 3, 1989.

[24] E. Kushilevitz and R. Ostrovsky. *Replication is not needed: single database, computationally-private information retrieval.* In Proceedings of FOCS '97.

[25] H. Lipmaa. *An Oblivious Transfer Protocol with Log-Squared Total Communication*, http://eprint.iacr.org/2004/063/

[26] M. Naor and B. Pinkas. *Oblivious transfer and polynomial evaluation.* In proceedings of STOC 1999.

[27] M. Naor and B. Pinkas. *Oblivious transfer with adaptive queries.* In Advances in Cryptology - Crypto '99, pp. 573-590.

[28] P. Paillier. *Public-key cryptosystems based on composite degree residuaosity classes.* In Advances in Cryptology – Eucrocrypt 1999, pp. 223-238.

[29] M. Rabin. *How to exchange secrets by oblivious transfer.* Tech report TR 81, Aiken Computation Lab, 1981.

[30] X. Qian, M. Stickel, P. Karp, T. Lunt, and T. Garvey. *Detection and elimination of inference channels in multilevel relational database systems.* In Proc. of the IEEE Symposium on Research in Security and Privacy, pp. 196–205, 1993.

[31] P. Samarati and L. Sweeney. *Generalizing Data to Provide Anonymity when Disclosing Information.* In PODS 1998.

[32] A. Shamir. *How to Share a Secret.* In Communications of the ACM, 22(11):612- 613, November 1979.

[33] D. Song, D. Wagner and A. Perrig. *Practical Techniques for Searches on Encrypted Data.* In Proc. of the 2000 IEEE Security and Privacy Symposium, May 2000.

[34] J. Staddon. *Dynamic Inference Control.* 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2003.

[35] J. Traub, Y. Yemini and H Wozniakowski. *The statistical security of a statistical database.* In TODS, pp.672-679, 1984.

[36] D. Woodruff and J Staddon. *Private Inference Control* IACR eprint, http://eprint.iacr.org/2004/130/

[37] A. C. Yao. *Protocols for secure computations.* In proc. of 23rd FOCS, 1982, pp. 160-164. 16.