REVIEW PAPER ON PRIVACY PROTECTION OF DATABASE ACCESS WITH PARTIAL SHUFFLE

Ms. P. V. Bhagat¹, Mr. A. V. Saurkar², Mr. V. V. Bhujade³

¹,²,³Department of Computer Science & Engineering, DMIETR, Sawangi(M), Wardha, MH, India

ABSTRACT

Privacy protection is one of the fundamental security requirements for database outsourcing. A major threat is information leakage from database access patterns generated by query executions. Recent works propose to protect access patterns by introducing a trusted component with constant storage size. The resulting privacy assurance is as strong as PIR, though with O(1) online computation cost, they still have O(n) amortized cost per query due to periodically full database shuffles. In this work, we design a novel scheme in the same model with provable security, which only shuffles a portion of the database.

Keywords: Database, Data Privacy, Information Security, Private Information Retrieval (PIR), REA Encryption.

1. INTRODUCTION

Private information retrieval (PIR) is the well-known cryptographic mechanism inhibiting information leakage from access patterns. In database applications, a malicious database server can derive sensitive information about user queries, simply by observing the database access patterns, e.g., the records being retrieved or frequent accesses to “hot” records. Such a threat is aggravated in the Database-as-a-Service (DaaS) model whereby a data owner outsources their database to an untrusted service provider. PIR disallows a server to infer any additional information about queries. Private information retrieval (PIR) enables a user to retrieve a data item from a database, replicated among one or more servers, while hiding the identity of the retrieved item. A PIR protocol allows a user to access a database such that the server storing the database does not gain any information on the records the user read. This scheme allows a user to retrieve a data item from a database without revealing information about the data item. The Private Information Retrieval problem is only concerned with user’s privacy, without requiring any protection of server’s privacy. The database use in the different session process in different areas. The entire database as a reply to the user, because
the heavy computation incurred at the server outweighs the saved communication expense. While comparing with the standard PIR schemes, these PIR schemes works on encrypted data records rather than bits in plaintext.

However, how query is efficient on the encrypted database becomes a challenge. This usually found that the system has to sacrifice the performance to obtain the security. When data is stored in encrypted form, we have to decrypt all the data before querying them. It is impractical because the cost of decryption over all the encrypted data is very expensive.

For this purpose, we put forward the innovative encryption algorithm, known as “Reverse Encryption Algorithm (REA)”. Reverse Encryption Algorithm is efficient and reliable. To protect access pattern of the database generated by query, we follow this line of research and design a novel scheme which only shuffles a portion of the database.

2. EXISTING SYSTEM

Software protection is one of the most important issues concerning computer practice. The problem is to sell programs that can be executed by the buyer, yet cannot be redistributed by the buyer to other users. Much engineering effort is put into trying to provide the “software protection”, but this effort seems to lack theoretical foundations. In particular, there is no crisp definition of what the problems are and what should be considered as a satisfactory solution. There exist many heuristics and ad-hoc methods for protection, but the problem as a whole has not received the theoretical treatment it deserves. The treatment of software protection is to use repeated execution of the “same” ram on several inputs. The ram starts its next execution with the work tapes of both CPU and memory having contents identical to their contents at termination of the previous execution. In this system provides the theoretical treatment of software protection. It reduces the problem of software protection to the problem of efficient simulation on oblivious RAM. The machine is oblivious if the sequence in which it accesses memory locations is equivalent for any two inputs with the same running time. RAM was proposed to protect software’s memory access pattern. A trusted hardware plays the role of CPU in ORAM and caches a constant number of data. The simulation of a RAM on an oblivious RAM is tamper-proof if the simulation remains oblivious even in case when an infinitely-powerful adversary examines and alters memory contents. A tamper-proof simulation means that either the tampered execution will equal the untampered execution for all the possible inputs of equal length or the tampered execution will be detected as faulty and suspended.

2.1 Disadvantages of existing system

- A standard problem with building an efficient protocol of this type is that need all group elements used to have only large prime factors in their orders.
- Increase communication cost, server storage cost, protected storage cost.
- To increase the turnaround time.
- Increase computational, computation complexity.

3. PROPOSED SYSTEM

In this project proposes a new model called PIR with preprocessing to prevent database access patterns from being exposed to a malicious server using hardware-based scheme. This model uses K servers each storing a copy of the database. Before a PIR execution, each server computes and stores polynomials many bits regarding the database. Private information retrieval is a communication protocol between a user and a server. In this protocol the user wishes to retrieve an item from a database stored in the server without revealing to the server which item is being retrieved. Private Information Retrieval (PIR) schemes allow a user to retrieve information from a
database while maintaining the privacy of the queries from the database. Private information retrieval (PIR) is the well-known cryptographic mechanism inhibiting information leakage from access patterns. Modeling the database service as bit retrieval from a bit array in plaintext, PIR disallows a server to infer any additional information about queries. Many PIR schemes have been proposed with the emphasis on lowering the communication complexity between the server and the user. The computation cost can be greatly reduced by embedding a trusted component at the server’s end.

This proposed two algorithms: a shuffle-based algorithm and a hierarchy-based algorithm. The shuffle-based algorithm is to design a PIR scheme with communication cost and computation cost for periodical shuffles, where a trusted hardware plays the role of CPU in ORAM and caches a constant number of data. The main algorithmic improvement was due to a shuffle algorithm. Therefore, the amortized computation complexity is where the hardware store k records. The hierarchical algorithm reduced the computation complexity by introducing storage at the client side. Therefore, if the database is not large, these hierarchy based algorithms are not necessarily more efficient than the shuffle-based algorithms. Initially, all database entries are labeled white. Once a record is fetched, it is labeled black. For a query executes a novel twin retrieval algorithm: if variable is in the cache, trusted component randomly fetches a pair of records, black and white, respectively; otherwise, it retrieves the needed record and another random record in a different color. When the cache is full, trusted component only shuffles and re-encrypts all black records, which is called a partial shuffle. Intuitively, host always spots a black and white pair being retrieved for queries in a session. Moreover, the information collected in one session is rendered obsolete for the succeeding sessions because partial shuffles remove the correlations across sessions. The security goal of Private Information Retrieval (PIR) is an encrypted record is not touched does not help the adversary to derive any information about user queries. The proposed scheme is rooted at an insightful observation: the full database shuffle is not indispensable, as long as user queries produce access patterns with the same distribution. Note that it is unnecessary to shuffle full records. A record does not leak any query information for the following two reasons. First, all records are encrypted and therefore a partial record itself does not compromise privacy. Second, there exists no access pattern involving it. It stress that the “square root” complexity of the shuffle based ORAM and our results are completely in different context. The square root solution of ORAM requires a sheltered storage storing items, which is equivalent to using a cache storing items at the client end in our setting. The standard private information retrieval (PIR) schemes are widely regarded as theoretical solutions; entail the computational overhead per query for a database with items. To protect access patterns by introducing a trusted component with constant storage size. In the loose-coupling architecture as suggested in a client/agent plays the role of trusted party. Note that the choice of architecture does not affect the complexity of the algorithms or the number of server operations. the database is encrypted and permuted using fresh secrets are generated. During execution, retrieves the requested item from the database if is not in the cache; otherwise, a random item is fetched to the cache. When the cache is full, the entire database is reshuffled and re-encrypted for the next session. The objective of database shuffles is to remix the touched database entries with the untouched ones, so that future executions appear independent with preceding ones. Due to the full database shuffle, these protocols in occur computation cost.

The resulting privacy assurance is as strong as PIR, though with online computation cost, they still have amortized cost per query due to periodically full database shuffles. It designs a novel scheme in the same model with provable security, which only shuffles a portion of the database. With a secure storage storing thousands of items, our scheme can protect the access pattern privacy of databases of billions of entries, at a lower cost than those using ORAM-based poly-logarithm algorithms. In fact, our scheme only uses a constant size cache and when our scheme has poly-logarithm complexity. It shows that the adversary gets negligible advantage by obtaining the
transaction scripts which is computationally indistinguishable from a random string of the same length. It begins with a definition of ideal implementation which dismisses those attacks on the permutations and encryption. PR-ORAM needs a protected storage whose size is independent of the database size. In this scheme, the hardware needs a cache to store a constant amount of data items. PR-ORAM also needs client end storage to store secret information. Since it does not store data, it requires less storage than our scheme. This scheme, the server storage grows with query executions. At maximum, it stores the database of items, two arrays and of size of tree. Although the proposed introduce a trusted hardware in the server side, the algorithms proposed in this paper can also be applied to client–server settings as ORAM-based PIR. Our scheme also has database read/write, though it need an additional cost for a binary search. For those PIR schemes without using caches, the computation cost per query. Our scheme substantially outperforms all other PIR schemes in terms of average query cost by paying a slightly higher price of online query processes. We remark that to solve the PIR problem, both our scheme and ORAM require a trusted entity. In the tight coupling architecture considered in our scheme, a secure hardware is the one, which supports multiple clients and has faster database accesses.

4. ARCHITECTURAL DIAGRAM

![Architectural Diagram](image-url)

Fig 1. Architectural Diagram
5. MODULE DESCRIPTION

5.1 Query Evaluation

This module is used to check whether the trusted server is valid trusted server or not. If the trusted server is valid trusted server then it will display the main page, otherwise a warning message “invalid trusted username or password” will be displayed. Then this module reads the input from the trusted server. Then the server is evaluated the query. The system consists of a group of trusted servers, a database modeled as an array of data items with equal length and a database host. A trusted component has an internal cache which stores up to data items. No adversary can tamper’s executions or access its private space including the cache. It is capable of performing symmetric key encryption/decryption and pseudorandom number generation. All messages exchanged between trusted servers and are through a confidential and authentic channel.

5.2 Access Pattern

This module is used to accessing the query. A server can derive sensitive information about trusted server queries, simply by observing the database access patterns. The access patterns also include the visited addresses and the frequency of accesses. The transcript of protocol execution within a period is referred to as access pattern. The adversary in our model is the database host which attempts to derive information about trusted server queries from access patterns. The adversary as a probabilistic polynomial time algorithm takes any access pattern as the input and outputs the value of a target query. There is no polynomial time adversary gets non negligible advantage in determining by observing access patterns including Query’s execution. The access patterns have the identical distribution for all query executions. The address can be derived from access pattern. Any access pattern observed by the adversary can be caused by all possible permutations with the same probability. Formally, this model shows that the adversary as a probabilistic polynomial time algorithm, which takes any access pattern as the input and outputs the value of a target query. It allows accessing a query oracle, through which issues queries arbitrarily as a regular trusted server and observes their executions. The adversary can issue queries.

5.3 PIR Scheme Module

This module is used to inhibiting information leakage from access patterns. In this module, PIR disallows a server to infer any additional information about queries. A PIR scheme prevents an adversary from inferring information about queries from observation of query executions. With a secure storage storing thousands of items, our scheme can protect the access pattern privacy of database of billions of entries, at a lower cost. A PIR protocol allows a trusted server to access a database such that the server storing the database does not gain any information on the records the trusted server read. This scheme allows a trusted server to retrieve a query item from a database without revealing information about the query item. PIR schemes incur even more turn around time than transferring the entire database as a reply to the trusted server, because the heavy computation incurred at the server outweighs the saved communication expense. It protects access patterns by introducing a trusted component with constant storage size.

5.4 Shuffle Module

This module is used to reduce the amortized server computation complexity. When the session ends, the shuffle algorithm empties the cache and produces. The information collected in one session is rendered obsolete for the succeeding sessions because partial shuffles remove the correlations across sessions with chooses a pseudorandom permutation and an encryption algorithm with a random secret key. This step can also be performed by a trusted authority. Trusted component executes queries using the retrieval algorithm. After executions, query is populated with addresses.
and generates a secret permutation and a new secret key. It shuffles the accessed query according to while leaving the database records intact. Since all records to be shuffled are in the cache, simply re-encrypts them using and then writes the cipher texts out in a batch to generate. It deletes all data items in the cache.

6. CONCLUSION

This project presented a novel hardware-based scheme to prevent database access patterns from being exposed to a malicious server. By virtue of twin-retrieval and partial-shuffle, the scheme avoids full-database shuffle and reduces the amortized server computation complexity. Although the hierarchy-based ORAM algorithm family can protect access patterns with at most cost, they are plagued with large constants hidden in the big-notations. In this scheme has much less server storage overhead. It formally proved the scheme’s security following the notion of PIR and showed the experiment results which confirm the performance analysis.

REFERENCES


