Efficient Privacy and Data Confidentiality using A Trusted Outsourced Database

M. Kannan
Assistant Professor
Department of Computer Science and Engineering
Kongunadu College of Engineering Technology

D. Vikneshkumar
Assistant Professor
Department of Computer Science and Engineering
Sri Guru Institute of Technology, Coimbatore

Abstract:- **Any software-based cryptographic constructs then deployed, for server-side query process on the encrypted information, inherently limit query quality. Traditionally, as shortly as confidentiality became a priority, need arises to encrypt data before outsourcing to a service supplier Here, Trusted data base has been introduced, an outsourced database prototype that allows client to execute SQL queries with privacy and beneath restrictive compliance constraints by leverage server-hosted, tamper-proof trusted hardware in important question process stages, thereby removing any limitations on the sort of supported queries. Despite the value overhead and performance limitations of trusted hardware, it has been shown that the prices per question are orders of magnitude less than any potential future software-only mechanisms.**

Keywords - Database architectures, security, privacy, special-purpose hardware

1. INTRODUCTION

The overview of outsourcing and clouds are well known, significant challenges yet lie in the path of large-scale adoption since such services often require their customers to inherently trust the provider with full access to the outsourced data sets. Numerous instances of illicit insider behavior or data leaks have left clients reluctant to place sensitive data under the control of a remote, third-party provider, without practical assurances of privacy and confidentiality, especially in business, healthcare, and government frameworks. Moreover, today's privacy guarantees for such services are at best declarative and subject customers to unreasonable fine-print clauses. It allows the server operator to use customer behavior and content for commercial profiling or governmental surveillance purposes. Tamper resistant designs, however, are significantly constrained in both computational ability and memory capacity which makes implementing fully featured database solutions using Secure Coprocessors (SCPUs) very
challenging. Despite the cost overhead and performance limitations of trusted hardware, we show that the costs per query are orders of magnitude
lower than any (existing or) potential future software-only mechanisms. In most of these efforts, data is encrypted before outsourcing. Once encrypted however, inherent limitations in the types of primitive operations that can be performed on encrypted data lead to fundamental expressiveness and practicality constraints.

The introduction of new cost models and insights that explain and quantify the advantages of deploying trusted hardware for data processing; the design, development, and evaluation of Trusted, a trusted hardware based relational database with full data confidentiality; and detailed query optimization techniques in a trusted hardware-based query execution model. The cost-performance trade off seem to suggest that things stand somewhat differently. Specifically, at scale, in outsourced contexts, computation inside secure processors is orders of magnitude cheaper than any equivalent cryptographic operation performed on the provider's unsecured server hardware, despite the overall greater acquisition cost of secure hardware. This is so because the overheads for cryptography that allows some processing by the server on encrypted data are extremely high even for simple operations. This fact is rooted not in cipher implementation inefficiencies but rather in fundamental cryptographic hardness assumptions and constructs, such as trapdoor functions.

New mathematical hardness problems will need to be discovered to allow hope of more efficient cryptography. As a result, it has been posit that a full-fledged privacy enabling secure database leveraging server-side trusted hardware can be built and run at a fraction of the cost of any (existing or future) cryptography-enabled private data processing on common server hardware.

Query processing engines run on both the server and in the SCPU. Attributes in the database are classified as being either public or private. Private attributes are encrypted and can only be decrypted by the client or by the SCPU. Since the entire database resides outside the SCPU, its size is not bound by SCPU memory limitations. Pages that need to be accessed by the SCPU-side query processing are pulled in on demand by the Paging Module. A client defines a database schema and partially populates it. Sensitive attributes are marked using the SENSITIVE keyword which the client layer transparently processes by encrypting the corresponding attributes. A client sends a query request to the host server through a standard SQL interface. The query is transparently encrypted at the client site using the public key of the SCPU. The host server thus cannot decrypt the query. The host server forwards the encrypted query to the Request Handler inside the SCPU. The Request Handler decrypts the query and forwards it to the Query Parser. The query is parsed generating a set of plans. Each plan is constructed by rewriting the original client query into a set of sub queries, and, according to their target data set classification, each sub query in the plan is identified as being either public or private. The

Query Optimizer then estimates the execution costs of each of the plans and selects the best plan (one with least cost) for execution forwarding it to the dispatcher. The Query Dispatcher forwards the public queries to the host server and the private queries to the SCPU database engine while handling dependencies. The net result is that the maximum possible work is run on the host server's cheap cycles.

The final query result is assembled, encrypted, digitally signed by the SCPU Query Dispatcher, and sent to the client query parsing and execution. Sensitive attributes can occur anywhere within a query, e.g., in SELECT, WHERE, or GROUP-BY clauses, in aggregation operators, or within sub queries. The Query Parser's job is then: To ensure that any processing involving private attributes is done within the SCPU. All private attributes are encrypted using a shared data encryption keys between the client and the SCPU hence, the host server cannot decipher these attributes to optimize the rewrite of the client query such that most of the work is performed on the host server. To exemplify how public and private queries are generated from the original client query, use an example from the TPC-H benchmark. TPC-H does not specify any classification of attributes based on security. Therefore, it has been defined an attribute set classification into private (encrypted) and public (no encrypted). In brief, all attributes that convey identifying information about customers, suppliers, and parts are considered private. The resulting query plans, including rewrites into main CPU and SCPU components for TPC-H queries Q3 and Q6 are illustrated for queries that have WHERE clause conditions on public attributes, the server can first SELECT all the tuples that meet the criteria. The private attributes' queries are then performed inside the SCPU on these intermediate results, to yield the final result. The host server first executes a public query that filters all tuples which fall within the desired ship date and quantity range, both of these being public attributes.

The result from this public query is then used by the SCPU to perform the aggregation on the private attributes extended price and discount. While performing the aggregation the private attributes are decrypted inside the SCPU. Since the aggregation operation results in a new attribute composing of private attributes it is re encrypted within the SCPU before sending to the client. That the execution of private queries depends on the results from the execution of public queries and vice a-versa even though they execute in separate database engines. This is made possible by the Trusted DB Query Dispatcher in conjunction with the Paging Module.

2. RELATED WORK

Researchers have recently discovered several interesting, self-organized regularities from the World Wide Web, ranging from the structure and growth of the Web to the access patterns in Web surfing. What remains to be a great challenge in Web log mining is how to explain user behavior underlying observed Web usage regularities. In this paper, we will address the issue of how to characterize the strong regularities in Web surfing in terms of user navigation strategies, and present an information for aging agent based approach for describing user behavior.

By experimenting with the agent-based decision models of Web surfing, we aim to explain how some Web design factors as well as user cognitive factors may affect the overall behavioral patterns in Web usage. In order to further characterize user navigation regularities as well as to understand the effects of user interests, motivation, and content organization on the user behavior. An information for aging agent based model that takes into account the interest profiles, motivation aggregation, and content selection strategies of users and, thereafter, predicts the emerged regularities in user navigation behavior. In summary, our work offers a means for explaining strong Web regularities with respect to user Interest Profiles, Web Content Distribution and coupling, and user navigation strategies.

It enables us to predict the effects on emergent usage regularities if certain aspects of Web servers or user foraging behaviors are changed. While presenting an interesting and promising research direction, it has been point out that one of the useful extensions for future work would be to show how the quantitative representations or constructs as used in modeling Web contents and user interest profiles. The planning Techniques help to bridge the gap between the searching necessities and the Content Adaptation. To Monitor and Adapt the learning object of each learning route against unexpected contingencies.

Existing research addresses several such security aspects, including access privacy and searches on encrypted data. In most of these efforts data is encrypted before outsourcing. Once encrypted however, inherent limitations in the types of primitive operations that can be performed on encrypted data lead to fundamental expressiveness and practicality constraints. Recent theoretical cryptography results provide hope by proving the existence of universal homeomorphisms, i.e., encryption mechanisms that allow computation of arbitrary functions without decrypting the inputs. Unfortunately actual instances of such mechanisms seem to be decades away from being practical.

3. PROPOSED TRUSTED DATABASE ARCHITECTURE

A full-fledged, privacy enabling secure database leveraging server-side trusted hardware can be built and run at a fraction of the cost of any (existing or future) cryptography-enabled private data processing on common server hardware. The Proposed trusted database architecture is shown in Fig 1. It has been validated by designing and building Trusted DB, a SQL database processing engine that makes use of tamperproof cryptographic coprocessors such as the IBM 4764 in close proximity to the outsourced data. Tamper resistant designs however are significantly constrained in both computational ability and memory capacity which makes implementing fully featured database solutions using secure coprocessors (SCPUs) very challenging. Trusted achieves this by utilizing common unsecured server resources to the maximum extent possible. E.g., Trusted enables the SCPU to transparently access external storage

while preserving data confidentiality with on-the-fly encryption. This eliminates the limitations on the size of databases that can be supported. Moreover, client queries are pre-processed to identify sensitive components to be run inside the SCPU. Non-sensitive operations are off-loaded to the untrusted host server. This greatly improves performance and reduces the cost of transactions.

The outsourced data are stored at the host provider's site. Query processing engines run on both the server and in the SCPU. Attributes in the database are classified as being either public or private. Private attributes are encrypted and can only be decrypted by the client or by the SCPU. Since the entire database resides outside the SCPU, its size is not bound by SCPU memory limitations. Query execution entails a set of stages, in the first stage, a client defines a database schema and partially populates it. Sensitive attributes are marked using the SENSITIVE keyword which the client layer transparently processes by encrypting the corresponding attributes. Later, a client sends a query request to the host server through a standard SQL interface. The query is transparently encrypted at the client site using the public key of the SCPU. The host server thus cannot decrypt the query. The host server forwards the encrypted query to the Request Handler inside the SCPU. The Request Handler decrypts the query and forwards it to the Query Parser. The query is parsed generating a set of plans. Each plan is constructed by rewriting the original client query into a set of sub queries, and, according to data set classification, each sub query in the plan is identified as being either public or private. The Query Optimizer then estimates the execution costs of each of the plans and selects the best plan (one with least cost) for execution forwarding it to the dispatcher. The Query Dispatcher forwards the public queries to the host server and the private queries to the SCPU database engine while handling dependencies. The final query result is assembled, encrypted, digitally signed by the SCPU Query Dispatcher, and sent to the client.

3.1 QUERY PARSING AND EXECUTION

In the first stage a client defines a database schema and partially populates it. Sensitive attributes are marked using the SENSITIVE keyword which the client layer transparently processes by encrypting the corresponding attributes:

CREATE TABLE customer (ID integer primary key, Name char (72) SENSITIVE, Address char (120) SENSITIVE);

Later, a client sends a query request to the host server through a standard SQL interface. The query is transparently encrypted at the client site using the public key of the SCPU. The host server thus cannot decrypt the query. The host server forwards the encrypted query to the Request Handler inside the SCPU. The Request Handler decrypts the query and forwards it to the Query Parser. The query is parsed generating a set of plans. Each plan is constructed by rewriting the original client query into a set of sub-queries, and, according to their target data set classification, each sub-query in the plan is identified as being either public or private. The Query Optimizer then estimates the execution costs of each of the plans and selects the best plan (one with least cost) for execution forwarding it to the dispatcher. The Query Dispatcher forwards the public queries to the host server and the private queries to the SCPU database engine while handling dependencies. The net result is that the maximum possible work is run on the host server's cheap cycles. The final query result is assembled, encrypted, digitally signed by the SCPU Query Dispatcher, and sent to the client.

3.2 QUERY OPTIMIZATION PROCESS

The Query Plan constructs to multiple plans for the client query. The constructed plan the Query Cost Estimator computes an estimate of the execution cost of that plan. The selected and passed on to the Query Plan Interpreter for execution. The Query Cost Estimator due to the logical partitioning of data.

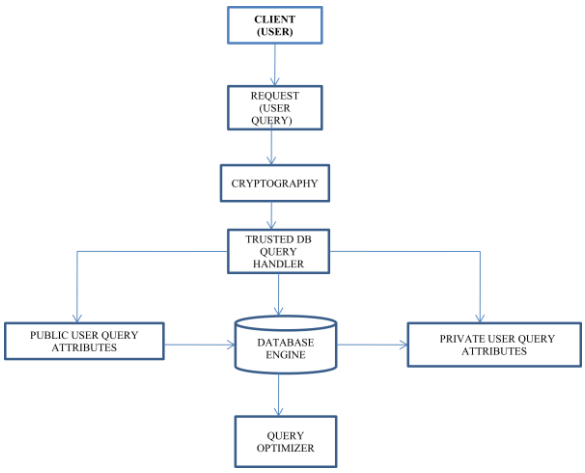

Fig 1.Trusted Database Architecture

At a high level query optimization in a database system works as follows. The Query Plan Generator constructs possibly multiple plans for the client query. For each constructed plan the Query Cost Estimator computes an estimate of the execution cost of that plan. The best plan i.e., one with the least cost, is then selected and passed on to the Query Plan Interpreter for execution. The query optimization process in Trusted Database works similarly with key differences in the Query Cost Estimator due to the logical partitioning of data mentioned above.

3.3 SYSTEM CATALOG

Any query plan is composed of multiple individual execution steps. To estimate the cost of the entire plan it is essential to estimate the cost of individual steps and aggregate them. In order to estimate these costs the Query Cost Estimator needs access to some key information. E.g., the availability of an index or the knowledge of possible distinct values of an attribute. These sets of information are collected and stored in the System Catalog. Most available DBMS today have some form of periodically updated System Catalog. The cost of a plan is the aggregate of the cost of the steps that comprise it. The execution times for a certain set of basic query plan steps are estimated.

4. CONCLUSION AND FUTURE ENHANCEMENTS

Queries on encrypted data, Propose division of data into secret partitions and rewriting of range queries over the original data in terms of the resulting partition identifiers. This balances a trade-off between client and server-side processing, as a function of the data segment size, the propose using tuples-level encryption and indexes on the encrypted tuples to support equality predicates. The main contribution here is the analysis of attribute exposure caused by query processing leading to two insights. The attribute exposure increases with the number of attributes used in an index, and the exposure decreases with the increase in database size. Range queries are processed by encrypting individual B+ Tree nodes and having the client, in each query processing step, retrieve a desired encrypted B+ Tree node from the server, decrypt and process it. However, this leads to minimal utilization of server resources thereby undermining the benefits of outsourcing. Moreover, transfer of entire B+ Tree nodes to the client results in significant network costs. In addition, a technique referred to as splitting and scaling issued to differ the frequency distribution of encrypted data from that of the plaintext data. Here, each plaintext value is encrypted using multiple distinct keys. Then, corresponding values are replicated to ensure that all encrypted values occur with the same frequency thereby thwarting any frequency-based attacks. Use a salted version of IDA scheme to split encrypted tuples data among multiple servers. In addition, a secure B+ Tree is built on the key attribute. The client utilizes the B+ Tree index to determine the IDA matrix columns that need to be accessed for data retrieval. To speed up client-side processing and reduce network overheads, it is suggested to cache parts of the B+ Tree index client-side.

The client query is split into multiple queries wherein each sub query fetches the relevant data from a server and the client combines results from multiple servers, also use vertical partitioning in a similar manner and for the same privacy goal, but differs in partitioning and optimization algorithms. Trusted is equivalent to both, when the size of the privacy subset is one and hence a single server suffices. In this case, each attribute column needs encryption to ensure privacy. Optimizer for querying encrypted columns since otherwise they rely on client-side decryption and processing, Introduce the concept of logical fragments to achieve the same partitioning effect as on a single server. A fragment here is simply areolation wherein attributes not desired to be visible in that fragment are encrypted. Trusted DB (and other solutions) are in effect concrete mechanisms to efficiently query any individual fragment from. The work on the other hand, can be used to determine the set of attributes that should be encrypted in propose an encryption scheme in a trusted-server model to ensure privacy of data residing on disk. The FCE scheme designed here is equivalently secure as a block cipher, however, with increased efficiency only ensures privacy of data residing on disk. In order to increase query functionality, a layered encryption scheme is used and then dynamically adjusted (by revealing key to the server) according to client queries. Trusted DB, on the other hand, operates in an un trusted server model, where sensitive data are protected, both on disk and during processing. Data that are encrypted on disk but processed in clear (in server memory), compromise privacy during the processing interval. The disclosures risks in such solutions are analyzed also propose a new query optimizer that takes into account both performance and disclosure risk for sensitive data. Individual data pages are encrypted by secret keys that are managed by a trusted hardware module. The decryption of the data pages and subsequent processing is done in server memory. Hence, the goal is to minimize the lifetime of sensitive data and keys in server memory after decryption.

REFERENCES

[1]. FIPS PUB 140-2, Security Requirements for Cryptographic Modules,http://csrc.nist.gov/groups/STM/cmvp/standards.html#02,2013.
[2]. TPC-H Benchmark, http://www.tpc.org/tpch/, 2013.
[3]. IBM 4764 PCI-X CryptographicCoprocessor,http://www03.ibm.com/security/cryptocards/pcixcc/overview.shtml, 2007.
[4]. G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K.Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu,"Two Can Keep a Secret: A Distributed Architecture for SecureDatabase Services," Proc. Conf. Innovative Data Systems Research(CIDR), pp. 186-199, 2005.
[5]. Iliev and S.W. Smith, "Protecting Client Privacy with TrustedComputing at the Server," IEEE Security and Privacy, vol. 3, no. 2,pp. 20-28, Mar./Apr. 2005.
[6]. M. Bellare, "New Proofs for NMAC and HMAC: Security WithoutCollision-Resistance," Proc. 26th Ann. Int'l Conf. Advances in
Cryptology, pp. 602-619, 2006.
[7]. B. Bhattacharjee, N. Abe, K. Goldman, B. Zadrozny, C. Apte, V.R.Chillakuru, and M. del Carpio, "Using Secure Coprocessors forPrivacy Preserving Collaborative Data Mining and Analysis,"Proc. Second Int'l Workshop Data Management on New Hardware(DaMoN '06), 2006.
[8]. M. Canim, M. Kantarcioglu, B. Hore, and S. Mehrotra, "BuildingDisclosure Risk Aware Query Optimizers for Relational Databases,"Proc. VLDB Endowment, vol. 3, nos. 1/2, pp. 13-24, Sept.2010.
[9]. S. Bajaj and R. Sion, "TrustedDB: A Trusted Hardware Based Database with Privacy and Data Confidentiality," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '11), pp. 205-216, 2011.
[10]. S. Bajaj and R. Sion, "TrustedDB: A Trusted Hardware Based Outsourced Database Engine," Proc. Int'l Conf. Very Large DataBases (VLDB), 2011.
[11]. S. Wu, F. Li, S. Mehrotra, and B.C. Ooi, "Query Optimization for Massively Parallel Data Processing," Proc. Second ACM Symp.Cloud Computing (CCS '11), Article 12, 2011.
[12]. V. Ciriani, S.D.C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Combining Fragmentation and Encryption to Protect Privacy in Data Storage," ACM Trans. Information and System Security, vol. 13, no. 3, pp. 22:1-22:33, July 2010.
[13]. E. Damiani, C. Vimercati, S. Jajodia, S. Paraboschi, and P.Samarati, "Balancing Confidentiality and Efficiency in Untrusted Relational DBMSs," Proc. 10th ACM Conf. Computer and Communications Security (CCS '12), 2003.