DISTRIBUTED DATABASE SYSTEM FOR ORGANIZATION'S SECURITY MAINTANANCE

[1]Akash J. Jadhav, [2]Supriya H. Patil,[3]Dr.K.Vengatesan,

Software Engineer, Tudip Technologies,Pune

Associate Technology Scientist,Infintus Innovations Pvt Ltd, Pune,

Associate Professor, Department of Computer Engineering, Sanjivani College of Engineering.

ABSTRACT

Earlier patterns towards database outsourcing, and in addition concerns and laws overseeing information protection, have prompted extraordinary enthusiasm for empowering secure database administrations. Past ways to deal with empowering such an administration have been founded on information encryption, causing an extensive overhead in query preparing. We propose another, disseminated design that enables an association to outsource its information administration to multiple untrusted servers while protecting information security. We indicate how the nearness of two servers empowers proficient parceling of information with the goal that the substance at any one server are ensured not to rupture information protection. We demonstrate to advance and execute questions in this engineering, and talk about new difficulties that rise in outlining the database representation.

Keywords: Database, Security, Proficient Queries, Service Provider.

INTRODUCTION

The database group is seeing the rise of two late patterns set on an impact course. From one viewpoint, the outsourcing of information administration has turned out to be progressively appealing for some associations [1]; the utilization of an outer database benefit guarantees dependable information stockpiling requiring little to no effort, taking out the requirement for costly in-house information administration foundation, e.g., [2]. Then again, heightening worries about information protection, late administrative enactment [3], and also prominent occasions of database robbery [4], have started an unmistakable fascination in empowering secure information stockpiling. The two patterns portrayed above are in coordinate clash with each other. A client utilizing a database benefit Need's to believe the specialist co-op with conceivably delicate information, welcoming harming holes of private data. Thusly, there has been much late enthusiasm for an alleged Secure Database Service {a DBMS that gives dependable capacity and productive query execution, while not knowing the substance of the database [5]. Such an administration additionally helps the specialist co-op by constraining their risk in the event of break-ins into their framework {if the specialist co-ops don't have a clue about the substance of the database, neither will a programmer who breaks into the framework. Existing proposition for secure database administrations have normally been established on encryption [2], [4]. Information is encoded on the (trusted) client side before being put away in the (untrusted) outer database. Watch that there is dependably an inconsequential method to answer all database questions: the client can get the whole database from the server, unscramble it, and execute the query on this decoded database. Obviously, such an approach is dreadfully costly to be useful.

Rather, the expectation is that questions can be changed by the client to execute straightforwardly on the encoded information; the aftereffects of such changed inquiries could be post-prepared by the client to acquire the last outcomes. Lamentably, such expectations are regularly dashed by the security proficiency exchange of encryption. Frail encryption works that permit productive questions spill considerably an excess of data and in this way don't safeguard information protection [5]. Then again, more grounded encryption works frequently require turning to Plan A for questions {fetching the whole database from the server {which is essentially excessively costly. Additionally, in spite of expanding processor rates, encryption and decoding are not precisely shoddy, particularly when performed over information at a fine granularity. We propose another way to deal with empowering a secure database server. The key thought is to enable the client to segment its information crosswise over two, (and all the more by and large, any number of) consistently free database frameworks that can't speak with each other. The dividing of information is performed in such a form as to guarantee that the introduction of the substance of any one database does not bring about an infringement of protection. The client executes inquiries by transmitting proper sub-questions to every database and afterward sorting out the outcomes at the client side.

LITERATURE REVIEW

Secure Database Services As discussed in the introduction, the outsourcing of data management has motivated the model where a DBMS provides reliable storage and efficient query execution, while not knowing the contents of the database [6]. Schemes proposed so far for this model encrypt data on the client side and then store the encrypted database on the server side [3], [8], [9]. However, in order to achieve efficient query processing, all the above schemes only provide very weak notions of data privacy. In fact a server that is secure under formal cryptographic notions can be proved to be hopelesslyinefficient for data processing [5]. Our architecture of using multiple servers helps to achieve bothefficiency and provable privacy together. Trusted Computing With trusted computing [4], a tamper-proof secure co-processor could be installed on the server side, which allows executing a function while hiding the function from the server. Using trusted tamper-proof hardware for enabling secure database services has been proposed in [5]. How-ever, such a scheme could involve significant computational overhead due to repeated encryption and decryption at the tuple level.

Understanding the role of tamper-proof hardware in our architecture remains a subject of future work. Secure Multi-party Computation Secure multi- party computation [3], [9] discusses how to compute the output of a function whose inputs are stored at different parties, such that each party learns only the function output and nothing about the inputs of the other parties. In our context, there are two par-ties { the server and the client { with the server's input being encrypted data, the client's input being the encryption key, and the function being the desired query. In principle, the client and the server could then en- gage in a one-sided secure computation protocol to compute the function output that is revealed only to the client. However, \in principle" is the operative phrase, as the excessive communication overhead involved makes this approach even more inefficient than the trivial scheme in which the client fetches the entire database from the server. More efficient specialized secure multi-party computation techniques have been studied recently [1], [2]. However all of this work is to enable different organizations to securely analyze their combined data, rather than the client-server model we are interested in.

Privacy-preserving Data Mining Different approaches for privacy-preserving data mining studied recently include:

(1) Perturbationtechniques [4], [7],[8].

(2) Query restriction/auditing [5].

(3) k-anonymity [3].

However, research here is motivated by the need to ensure individual privacy while at the same time allowing the inference of higher- granularity patterns from the data. Our problem is rather different in nature, and the above techniques are not directly relevant in our context. Access Control Access control is used to control which parts of data can be accessed by different users. Several models have been proposed for specifying and enforcing access control in databases [5]. Access control does not solve the problem of maintaining an untrusted storage server as even the administrator or an insider having complete control over the data at the server is not trusted by the client in our model.

PROPOSED MODEL

Untrusted Service Providers: The client does not need to believe the chairmen of either database to ensure protection. Inasmuch as an enemy does not access the two databases, information security is completely ensured. On the off chance that the client were to acquire database administrations from two unique sellers, the odds of a foe breaking into the two frameworks is decreased enormously. Moreover, \insider" assaults at one of the sellers don't trade off the security of the framework all in all. Provable Privacy. The nearness of two databases empowers the productive encoding of touchy characteristics in a data hypothetically secure mold. To delineate, consider a delicate settled length numerical property, for example, a charge card number.

We may speak to a charge card number c, by putting away c XORed with an irregular number r in one database, and putting away r in the other database. The arrangement of bits used to speak to the charge card number in either database is totally irregular, in this manner giving flawless protection. Be that as it may, we may recoup the number only by XORing the qualities put away in the two databases, which is more productive than utilizing costly encryption and unscrambling capacities.

Proficient Queries: The nearness of different databases empowers the capacity of numerous trait esteems in decoded frame. Normally, the introduction of an arrangement of at-tribute esteems relating to a tuple may bring about a security infringement, while the presentation of just some subset of it might be safe. For instance, uncovering a person's name and her Visa number might be a genuine security infringement. Notwithstanding, uncovering the name alone, or the Visa number alone, may not be a major ordeal [2]. In such cases, we may put people's names in a single database, while putting away their Mastercard number in the other, abstaining from having to en-tomb either characteristic. A result is that inquiries including the two names and Mastercard numbers might be executed significantly more effectively than if the properties had been encoded.

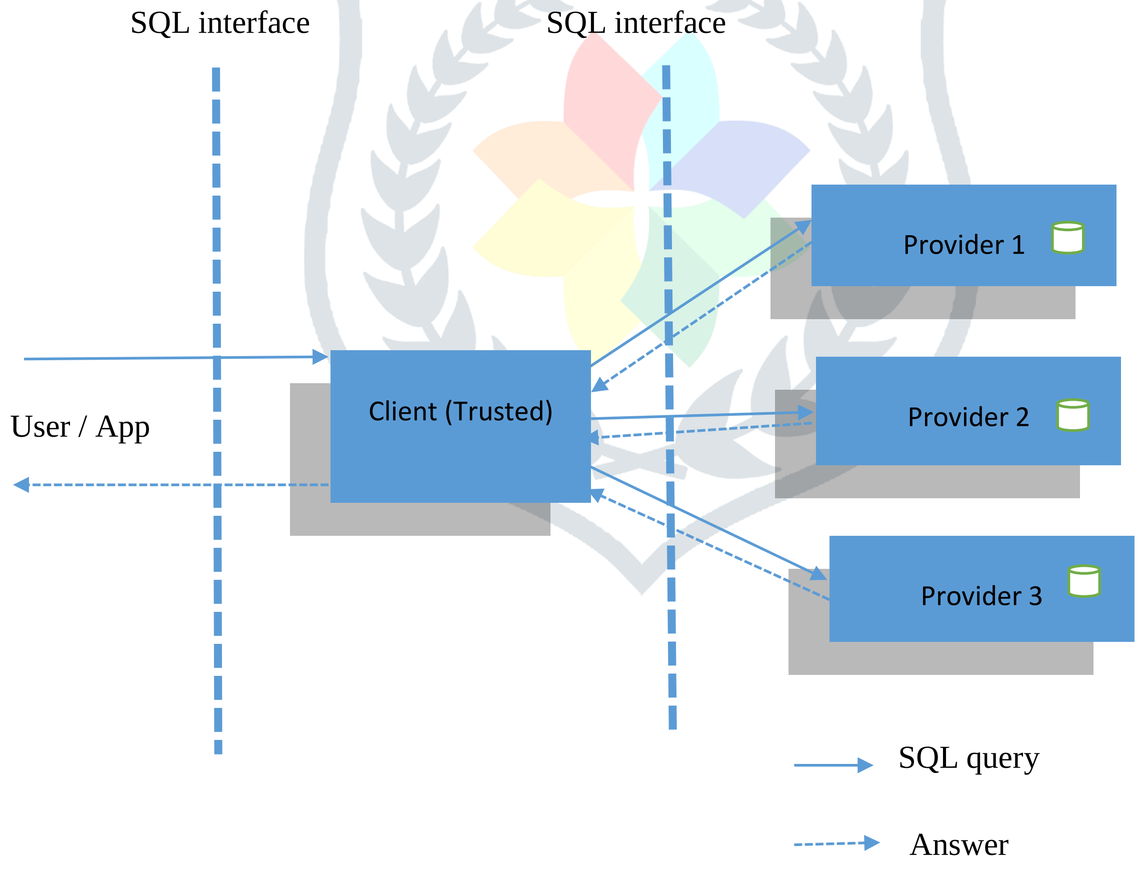

Figure 1: The System Architecture

The general architecture of a distributed secure database service, as illustrated in Figure 1, consists of a trusted client as well as two or more servers that provide a database service. The servers provide reliable content storage and data management but are not trusted by the client to preserve content privacy. The client wants to out-source the (high) costs of managing permanent storage to the service providers; hence, we assume that the client does not store any persistent data. However, the client has access to cheap hardware {providing processing power as well as temporary storage {which is used to provide three pieces of functionality:

1. Over a DBMS Front-End,the client exports a standard DBMS front-end to client-side applications, supporting standard SQL APIs.

2. Reformulate and Optimize Queries,the queries received by the client need to be translated into appropriate SQL sub-queries to be sent to the servers; such translation may involve limited forms of query-optimization logic, as we discuss later in the paper.

3. Post-process Query Results,the sub-queries are sent to the servers (using a standard SQL API), and the results are gathered and post-processed before being returned in a suitable form to the client-side application.

We take note of that each of the three bits of usefulness are genuinely shabby, in any event if the measure of post-handling required for questions is constrained, and can be performed utilizing modest equipment, without the requirement for ex-meditative information administration framework or faculty. Security Model As specified before, the client does not trust either server to save information protection. Every server is straightforward, yet conceivably inquisitive: the server may effectively screen every one of the information that it stores, and in addition the questions it gets from the client, in the expectation of rupturing security; it doesn't, in any case, demonstration perniciously by giving mistaken support of the client or by adjusting the put away information. The client looks after discrete, perpetual channels of correspondence to every server. We don't expect correspondence to be encoded; nonetheless, we accept that no busybody is fit for tuning in on both correspondence channels. The two servers are thought to be not able discuss straightforwardly with each other (delineated by the \wall" between them in Figure 1) and, indeed, require not know about each other's presence. Note that the client side is thought to be totally trusted and secure. There would not be much point in building up a secure database benefit if a programmer can basically enter the client side and straightforwardly get to the database. Anticipating client-side breaks is a conventional security issue random to protection safeguarding information stockpiling, and we don't worry about this issue here.

CONCLUSION

We have presented a novel distributed design for empowering security protecting outsourced stockpiling of information. We exhibited diverse strategies that could be utilized to break down information and clarified how inquiries might be streamlined and executed in this appropriated framework. We presented a meaning of protection in view of concealing arrangements of property estimations, showed how our disintegration strategies help in accomplishing security and considered the issue of recognizing the best security saving deterioration. Given the expanding occurrences of database outsourcing, and in addition the expanding noticeable quality of security worries and in addition controls, we expect that our design will demonstrate valuable both in guaranteeing consistence with laws and in lessening the danger of protection breaks.

REFERENCES

1. Advertiser charged in massive database theft. The Washington Post, July 22, 2004.
2. C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In Proc. CRYPTO, 2004.
3. California senate bill SB 1386. http: //info.sen.ca.gov/pub/01-02/bill/sen/ sb_1351-1400/sb_1386_bill_20020926_ chaptered.html, Sept. 2002.
4. G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. Technical report, StanfordUniversity, 2004.
5. H. Hacigumus, B. Iyer, and S. Mehrotra. Providing database as a service. In Proc. ICDE, 2002.
6. J.P. Morgan signs outsourcing deal with IBM. ComputerWorld, Dec 30, 2002.
7. Murat Kantarcioglu and Chris Clifton. Security issues in querying encrypted data. Technical Report TR-04-013, Purdue University, 2004.
8. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game { a completeness theorem for protocols with a honest majority. In Proc. STOC, 1987.
9. R. Agrawal and R. Srikant. Privacy-preserving data mining. In Proc. SIGMOD, 2000.
10. Y. Lindell and B. Pinkas. Privacy-preserving data mining. In Proc. CRYPTO, 2000.
11. Kalaivanan, M., and K. Vengatesan. "Recommendation system based on statistical analysis of ranking from user." International Conference on Information Communication and Embedded Systems (ICICES), , pp. 479-484. IEEE, 2013.
12. B.Narmadha, M.Ramkumar, K.Vengatesan, M.Srinivasan, "Household Safety based on IOT", International Journal of Engineering Development and Research (IJEDR), ISSN:2321-9939, Volume.5, Issue 4, pp.1485-1492, December 2017.