Esmeralda MIHANA

On The Freshness And Security Key Storage Problem Of Relational Databases Deployed On Cloud

M.S. 2015

FATİH UNIVERSITY

The Graduate School of Sciences and Engineering

Master of Science in Computer Engineering

ON THE FRESHNESS AND SECURITY KEY STORAGE PROBLEM OF RELATIONAL DATABASES DEPLOYED ON CLOUD

by

Esmeralda MIHANA

October 2015

ON THE FRESHNESS AND SECURITY KEY STORAGE PROBLEM OF RELATIONAL DATABASES DEPLOYED ON CLOUD

by

Esmeralda Mihana

A thesis submitted to

the Graduate School of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

October 2015
Istanbul, Turkey

APPROVAL PAGE

This is to certify that I have read this thesis written by Esmeralda MIHANA and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Computer Engineering

_____

Assist.Prof. Dr Ihsan H. AKIN

Thesis Supervisor

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Engineering

_____

Assist. Prof. Dr. Kadir TUFAN

Head of Department

Examining Committee Members

Assist.Prof.Dr. Ihsan H. AKIN _____

Assist.Prof.Dr. Kadir TUFAN _____

Assist.Prof.Dr. Tuğrul Yanık _____

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate School of Sciences and Engineering

_____

Prof. Dr. Nurullah ARSLAN

Director

October 2015

ON THE FRESHNESS AND SECURITY KEY STORAGE PROBLEM OF RELATIONAL DATABASES DEPLOYED ON CLOUD

Esmeralda MIHANA

M.S Thesis-Computer Engineering
June 2015

Thesis Supervisor: Assist.Prof.Dr. Ihsan Haluk AKIN

ABSTRACT

Relational Databases deployed on Cloud with semi-honest database administrators have cost and maintenance benefits. This deployment increased the deployment of databases into the cloud. While deploying the database reduces the general costs, security, privacy, row integrity, query completeness, and freshness issues are raised. Except the freshness, other problems are extensively searched and have proposed solutions in the literature. In this work, firstly, the freshness problem is under investigation. We provided the freshness by storing some local information from the cloud hosted database in the application server. As a second contribution, we investigated the on-line attacks on the multiplication server. The single and multiple applications servers case are investigated.

Keywords: Database Integrity, Relational Databases, Freshness, Cloud hosted Relational Databases, Application Servers

CLOUD ÜZERİNDE KONUŞLANMIŞ İLİŞKİSEL VERİTABANLARIN TAZELİĞI VE GÜVENLİK ANAHTAR DEPOLAMA SORUNU

Esmeralda MIHANA

Yüksek Lisans Tezi – Bilgisayar Mühendisliği
Eyül 2013

Tez Danışmanı: Assist.Prof.Dr. Ihsan Haluk AKIN

ÖZ

Yarı dürüst veritabanı yöneticileri ile Cloud üzerinde konuşlanmış İlişkisel Veritabanları maliyet ve bakım faydaları vardır. Bu dağıtım Cloud içine veritabanlarının konuşlandırılmasını arttı. Veritabanını dağıtma genel maliyetler, güvenlik, gizlilik, bütünlük, tamlık ve tazelik sorunları yükseltir. Tazelik dışında, diğer problemler literatürde yoğun çözümler buldu. Bu çalışmada, iki sorunlara konsantre oluyoruz. Ilk olarak, çok kullanıcılı uygulamalarda uygulama sunucuları saldırılmadı sürece, biz bulut barındırılan ilişkisel veritabanları tazeliğine sahip olduğunu gösteriyoruz. Ikinci olarak, biz uygulamalar sunucularının saldırı üzerinde araştırdık.

Anahtar Kelimeler: Veritabanı bütünlüğü, Ilişkisel Veri Tabanları, Tazelik, Bulut İlişkisel Veritabanları ev sahipliği, Uygulama Sunucular

ACKNOWLEDGMENTS

This thesis has been an important experience for me to develop my point of view on academic life. I would like to express my first words of gratitude to my supervisor, Assist.Prof.Dr. Ihsan H. AKIN, knowing that my thanks cannot compensate the enormous dedication and the long hours of discussion that he devoted to my work. His patience and sharpness of mind have been invaluable means for my understanding of the subject and for the improvement of my drafts. I also thank my jury members, Assist.Prof.Dr. Kadir TUFAN and Assist.Prof.Dr. Tuğrul Yanık for their comments and assistance.

I would also like to thank the staff members of Computer Engineering at Fatih University.

Finally, and above all, I would like to thank my parents and my husband parents for the sacrifices they made to give me the opportunity to follow my passions, and for, in the end, making it possible for me to complete it successfully. You, and your love, are always the key inspiration for me behind everything. Special thanks to my husband for supporting and motivating me not only during my thesis work, but also during my whole life.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF PROCESSES

PROCESS

LIST OF FIGURES

FIGURE

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOL/ABBREVIATION

AES	Advanced Encryption Standard
AS	Application Server
AS	Application Server
ASP	Active Server Pages
CBC	Cipher-Block Chaining Mode
CFB	Cipher Feedback Mode
CTR	Counter Mode
DaaS	Database as a Service
DB	Database
DBMS	Database Management System
DBPM	Database Provider Model
DES	Data Encryption Standard
DSAC	Digital Signature Aggregation and Chaining
DSAC	Digital Signature Aggregation and Chaining
ECB	Electronic Codebook Mode
HMAC	Keyed-Hashing for Message Authentication
HSM	Hardware Security Module
HSM	Hardware Security Module
IaaS	Infrastructure as a service
MAC	Message Authentication Code
MOOs	Modes of operation
NIST	National Institute of Standards and Technology
ODM	Outsourced Database Model
OFB	Output Feedback Mode
PaaS	Platform as a Service
PCBC	Plaintext Cipher-Block Chaining Mode

RDBMS	Relational Database Management System
RIC	Record Integrity Code
RSA	Rivest, Shamir, Adleman
SaaS	Software as a Service
SET	Secure Electronic Transaction
SQL	Structured Query Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
VOs	Verification Object

CHAPTER 1

INTRODUCTION

1.1 CLOUD COMPUTING

NIST definition: NIST (National Institute of Standards and Technology) is an all around acknowledged organization throughout the world for their standardization in the field of Information Technology. NIST characterizes the Cloud Computing architecture by describing five fundamental characteristics, three cloud service models and four cloud deployment models [9].

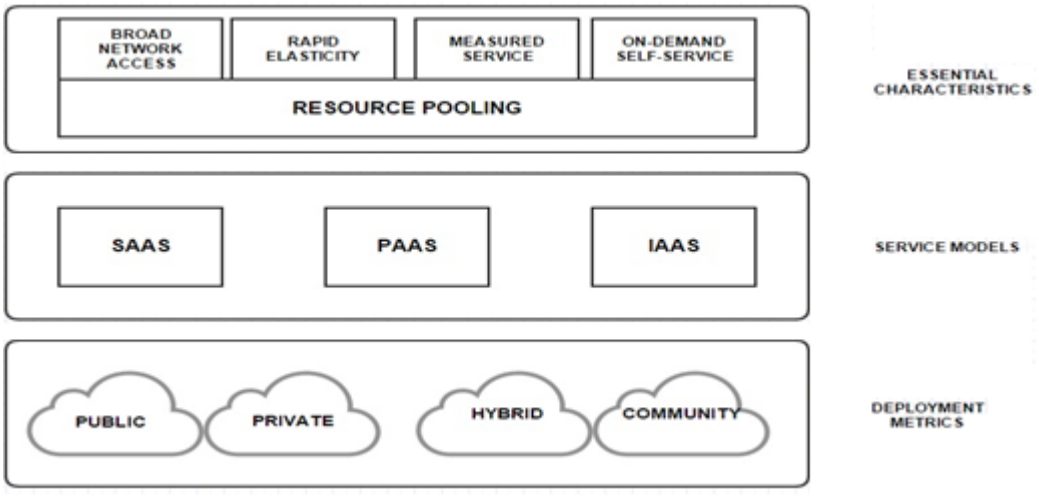

Figure 1.1 Definition of Cloud Computing by NIST

Cloud computing has ended up a standout amongst the most talked about innovation in late year and it is characterized as another sort of registering that depends on sharing figuring assets as opposed to having neighborhood servers or individual gadgets to handle applications [9]. Cloud computing means putting your information away and accessing the information and projects over the Internet rather than your PC's hard commute. The Cloud which is an illustration for the web as the administrations is given in the stage that is accessible for all.

The name cloud was propelled by the image that is frequently used to speak to the Internet in flowcharts and graphs.

Cloud computing is a term for the conveyance of facilitated administrations over the Internet. Cloud computing is equivalent another figuring sort; lattice registering where unused handling cycles of CPUS's over all PCs in a system are bridled to take care of issues seriously as opposed to utilizing stand-alone machine. Cloud computing guarantees a few alluring advantages for organizations and end clients like:

- ***Self-service provisioning*:** End clients can turn up figuring assets for a workload on-interest.
- ***Elasticity*:** Companies can scale up as the need of registering force increments and after that they can scale down when as interest abatements.
- ***Pay per use*:** Computing assets are measured at a granular level, permitting clients to pay just for the assets and workloads they utilize.

1.1.1 Types of Cloud Computing

The cloud computing, as indicated by Peter and Timothy, can be partitioned into five primary gatherings [9].

1. **On-demand self-service:** A customer can change server time and system stockpiling and other PC equipment and programming settings straightforwardly from the web without really connecting with the supplier to change. Along these lines, they have a reasonable picture of what is finished.
2. **Measured service:** Cloud frameworks control how much the customers utilized the administrations they gave to customers. The basic administrations are

capacity, CPU use, data transmission, and GPU utilization. Utilized assets are being checked and afterward a report is prepared so that the client get a diagram of the use and pay.

3. **Broad network access** : The system access is wide as far as it can be utilized over a ton of stages and gadgets that empowers simple correspondence between diverse hubs (e.g. Smartphone's, tablets, work-stations and portable PCs).
4. **Rapid elasticity :** The supplier's capacities can be provisioned and discharged flexibly and naturally. The capacities can be changed to fit the buyer requests and needs, and should be possible whenever.
5. **Resource pooling:** The supplier's assets are pooled and arranged at one area to give administrations to different purchasers through loads of virtual and physical assets that can be appointed and reassigned to fit the customers. The client cannot get to specifically to the supplier's assets. In any case, they have some information of where it is topographically. The assets of suppliers are capacity, system, transfer speed and servers.

1.1.2 Cloud Computing Services

Cloud computing services can be private, open, or hybrid. Indeed, even distributed computing has changed over the time, it has dependably been partitioned into three wide administration classes [9]: framework as an administration (IaaS), stage as an administration (PaaS) and programming as administration (SaaS).

1. **Infrastructure as a service (IaaS)**: Iaas is a sort of cloud computing in which an outsider supplier host virtualized computing resources over the Internet. In an IaaS model, an outsider supplier has equipment, programming, servers, stockpiling and other framework segments in the interest of its clients. IaaS suppliers additionally host client's applications and handle errands including framework upkeep, reinforcement and versatility arranging. IaaS is appropriate for workloads that are impermanent, test or change out of the blue.

2. **Platform as a service (*PaaS*)**: PaaS is a class of cloud computing that gives a stage and environment to permit engineers to assemble applications and administrations over the web. PaaS administrations are facilitated in the cloud and provided to clients over a web program. Stage as a Service permits clients to make programming applications utilizing instruments supplied by the supplier. PaaS administrations can comprise of preconfigured components that clients can subscribe to; they can decide to incorporate the elements that meet their necessities while disposing of those that don't.
3. **Software as service (SaaS)**: SaaS is a method for conveying applications over the Internet as an administration. Rather than introducing and looking after programming, you just get to it by means of the Internet, liberating yourself from complex programming and equipment administration. The other names of SaaS applications are Web-based programming, on-interest programming, or facilitated programming. SaaS applications keep running on a SaaS supplier's servers. The supplier oversees access to the application, including security, accessibility, and execution.

1.1.3 Cloud Computing Models

There are four diverse deployment models random to what sercice model that is utilized there are deployment models that can be connected to every one of them [9].

1. **Private cloud:** This kind of cloud structure is to be utilized to one and only association furthermore it is overseen by them. The framework could be taken care of by an outsider or themselves relying upon the administration understanding that exist.
2. **Public cloud**: Public cloud is accessible to the overall population frequently for free or with a payment. The administration can be given by an administration, organizations partnerships, or unions. A decent example is the free online stockpiles; icloud, dropbox, and Google drive are some surely understood cases.
3. **Community cloud**: This sort of cloud is utilized by groups that comprise of numerous association or clients. It might be possessed by the group or by an outsider to serve the group. The supplier's contribution relies on upon the administration display that is relevant to requests of the group. This is well

alternative when the organizations are framing associations over the assets they have.

4. **Hybrid cloud**: Hybrid cloud is a Private's blend, open and group cloud models. In spite of the fact that they are just limited together, we have half and half cloud as an arrangement model. A sample could be that an open cloud exists inside of an association for all workers and inside of this cloud is a private cloud available for supervisors.

1.2 DATABASE ON CLOUD

Cloud databases are a vital piece of the cloud framework. Database on cloud is a database that keeps running on a cloud computing platform and it is accessible to customers by getting to the cloud and delivered to clients on interest through the Internet from a cloud database supplier's servers. A cloud database is a sort of database administration that is assembled, conveyed and conveyed through a cloud stage [23]. Cloud databases could be either social or non-social databases. Contrasted with nearby databases, cloud databases are ensured higher adaptability and additionally accessibility and strength.

Because of the flexibility of cloud computing, hardware and software resources can be added to and expelled from the cloud without much exertion. Clients just need to pay for the expended asset while the costs for physical servers, organizing types of gear, framework upkeep and organization are shared among customers, along these lines decreasing the general expense. It is likewise conveyed as an administration, where the merchant specifically deals with the backend procedures of database establishment, sending and asset task undertakings.

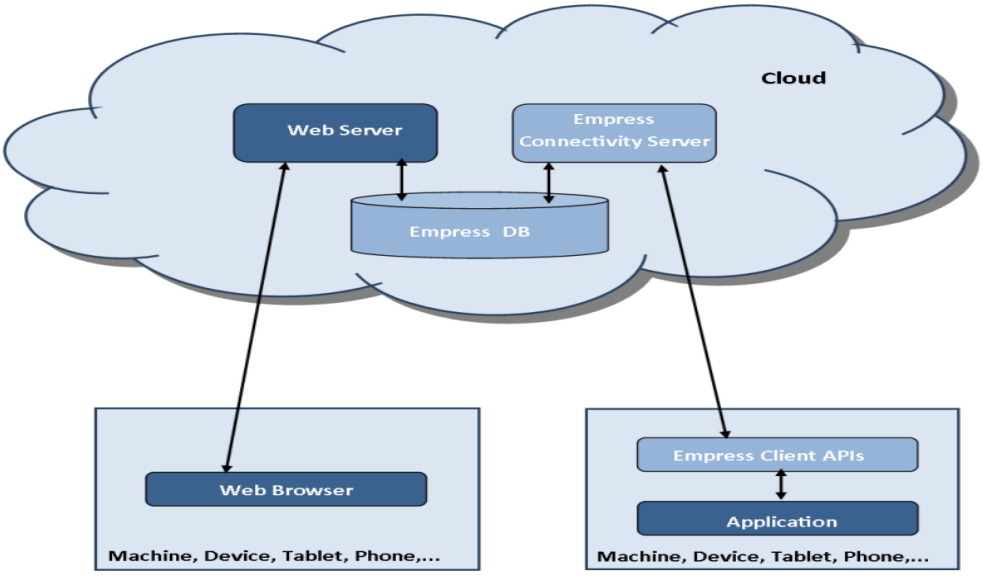

Figure 1.2 Database on Cloud

A cloud database acts as a standard database arrangement that is by and large executed through the establishment of database programming on top of cloud base. It might be straightforwardly accessed to through a Web program or a merchant gave API to application and administration combination. Unique in relation to a run of the mill database, a cloud database may be scaled on run-time, in which extra cases and assets of capacity and processing may be doled out quickly.

1.3 DATABASE ON CLOUD ISSUES

When the Database (DB) is deployed on the cloud, an attacker to can manipulate the DB to gain advantage even the DB is encrypted. As a result, the cloud DBs must be protected against malicious modifications.

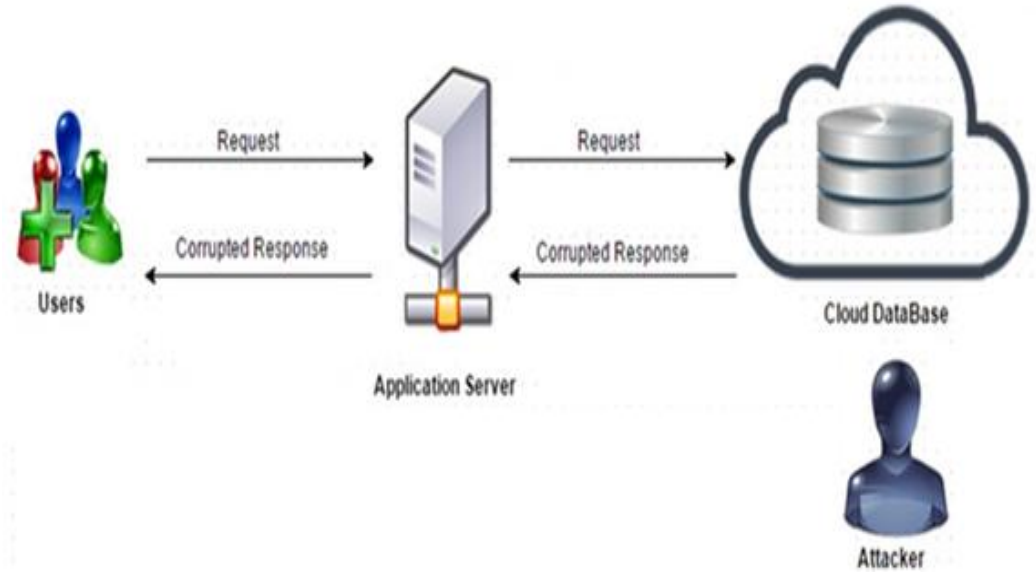

Figure 1.3 A common scenario of a Database Deployed on Cloud.

The hardware, maintenance, and management cost increased deploying data and processing into the cloud [12]. With this deployment, various problems of DBs arrived to the cloud; availability, privacy, and integrity are among them. Using multiple servers can be solving the availability [17]. The well-known encryption schemes can solve the privacy issues of the data [10]. Integrity of the DB is remains as one of the important issue in cloud computing [1-6].

Database (DB) is introduced under some trustworthiness limitations. Respectability requirements are a few properties that must dependably be fulfilled for the information in the DB to be predictable. At the point when the DB is conveyed over systems, a strategy to accept the data is required. The estimation of a DB is reliant upon a client's capacity to believe the fulfillment and soundness of the data contained in the information [4]. Guaranteeing the respectability in DB requires that the information ought to be secured from pernicious alterations. On the off chance that we can't guarantee this security we can say that the trustworthiness is lost [7].

DB integrity issues can be caused in different ways. An integrity problem can be caused by a user error, attack on the DB or systems where it is deployed, software problem and in addition by some hardware problems. The changes that can happen on

the DB when one of the previous issues occurs can be classified in malicious and non-malicious [3, 4, 16]. Malicious changes are made by malicious users (unauthorized users) that can access the DB by different secondary ways and their target is to make modifications on the DB on purpose. Non-malicious changes are made by non-malicious users (authorized users) that have access on the DB and accidently may corrupt the data.

There are various proposed methods to mitigate these problems. These methods include encrypting data, journaling, or using read-only storage in different situations. If we do not correct the loss of integrity, it can lead us to further damage, inaccuracy, or erroneous decisions [7]. This can affect not only the users but also the owners of the DB. Providing the integrity of DBs against malicious actions on the cloud is pioneered by Merkle by using the binary trees. It finishes up with some extra advantages that may be acknowledged with some of these routines and proposes an execution and security improvement for the DB frameworks.

1.4 DATABASE INTEGRITY

Integrity has gotten consideration from the examination group just as of late. Respectability is the confirmation that data must be gotten to or altered by those approved to do as such. Trustworthiness is connected with the exactness and consistency of put away information, demonstrated by a nonattendance of any adjustment in information between two upgrades of information record. [4-20]. Measures taken to guarantee trustworthiness incorporate controlling the physical environment of arranged terminals and servers, confining access to information, and keeping up thorough validation hones.

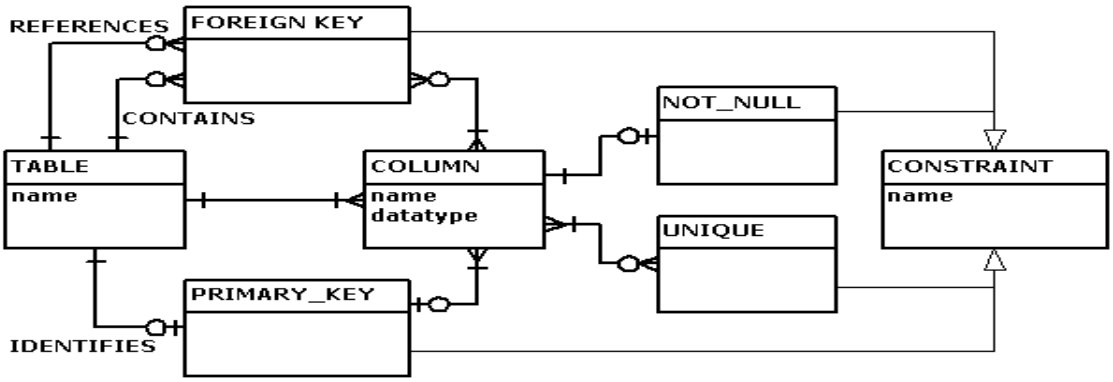

Figure 1.4 Database Integrity

1.4.1 Compromising Ways

Data Integrity can be traded off in various ways

- Human mistakes when information is entered
- Errors that happen when information is transmitted starting with one PC then onto the next
- Software bugs or infections
- Hardware breakdowns, for example, circle crashes
- Natural calamities, for example, flames and surges

1.4.2 Integrity Constraints

Integrity is upheld in both progressive and social database models. There are some trustworthiness limitations are utilized as a part of a social database structure to accomplish information honesty [19-21]:

- **Entity Integrity***:* This is worried with the idea of essential keys. The principle expresses that each table must have its own particular essential key and that each must be remarkable and not invalid.
- **Referential Integrity**: This is the idea of outside keys. The standard expresses that the outside key worth can be in two states. The principal state is that the

remote key worth would allude to an essential key estimation of another table, or it can be null.

- **Domain Integrity**: This expresses that all segments in a social database are in a characterized space.

1.5 CRYPTOLOGY

Cryptology, science worried with information correspondence and capacity in secure and generally mystery structure. It includes both cryptography and cryptanalysis. Cryptology is a blend of science and the utilization of recipes and calculations. With the goal information should be secured for capacity or transmission, it must be changed in such a way, to the point that it would be troublesome for an unapproved individual to have the capacity to find its actual which means. For this there are utilized numerical mathematical statements which are extremely hard to illuminate unless certain strict criteria are met. This trouble level is known as obstinacy.

The most vital things in cryptology are:

The Discrete Logarithm Problem: This is known as discrete exponentiation and is entirely easy to process. Then again, the inverse is genuine when we upset it.

The Integer Factorization Problem: This issue is the base for all mathematic ideas. They have been considering this idea for a long time and concurred that there is some dubious or unfamiliar law of arithmetic that disallows any alternate routes.

The Elliptic Curve Discrete Logarithm Problem: This is another cryptographic convention in view of an understood numerical issue. The properties of elliptic bends have been understood for a considerable length of time, however it is just as of late that their application to the field of cryptography has been attempted.

The development of cryptographic innovation has raised various legitimate issues in the data age.

1.5.1 Symmetric Cryptology

Symmetric Cryptography is the most familiar form of cryptography. In symmetric Cryptography two parties have the same key. In symmetric cryptography we assume that the two parties start with the same key and they can use the key for both encryption and decryption. These algorithms are very fast compared to asymmetric ciphers.

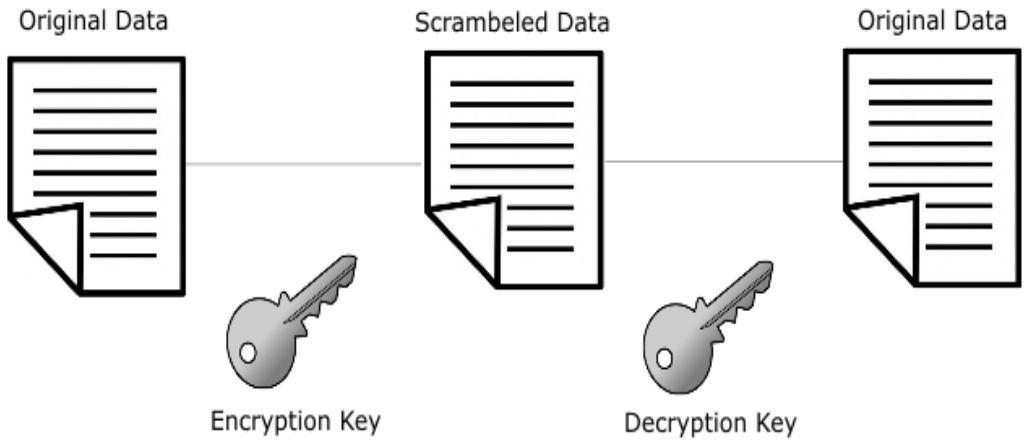

Figure 1.5 Symmetric encryption

This may be as basic as moving every letter by various spots in the letters in order. Both sender and receipt know the mystery key, they can scramble and decode all messages that utilization this key.

1.5.1.1 Examples of Symmetric Crypto Algorithms: DES, AES

DES: The Data Encryption Standard (DES) is the previous standard issues by NIST for symmetric-key encryption. DES utilizes the same key for encryption and decryption so the collector and the sender must have the same private key.

It was outlined at IBM in 1970. The straightforwardness of DES likewise saw it utilized as a part of a wide mixed bag of implanted frameworks, keen cards, SIM cards and system gadgets requiring encryption like modems, set-top boxes and switches.

In DES cryptographic key and calculation are connected to a piece of information at the same time instead of one piece at once. For encryption of a content DES bunches it into 64-bit pieces. Every square is enciphered utilizing the mystery key into a 64-bit figure content by method for stage and substitution. The procedure includes 16 adjusts and can keep running in four unique modes, encoding squares independently or making every figure piece reliant on all the past squares.

Decrypting is the latent of the encryption taking after the same steps yet turning around the request in which the keys are connected. DES utilizes a 64-bit key, however eight of those bits are utilized for equality checks.

AES: The Advanced Encryption Standard or AES is a symmetric key crypto algorithm. It is utilized to secure grouped data and is actualized in programming and equipment all through the world to scramble delicate information.

AES involves three different standard , AES-128, AES-192 and AES-256. Each of them encodes and decodes information in squares of 128 bits with cryptographic keys sizes 128-, 192-and 256-bits, separately. AES likewise DES utilizes the same key for enryption and decryption. There are 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys - a round comprises of a few preparing steps that incorporate substitution, transposition and blending of the info plaintext and change it into the last yield of figure content.

The main fruitful assaults against it have been side-channel attacks. Quite a few people have performed attack against reduced round variants of the Advanced Encryption Standard, and an examination paper distributed in 2011 exhibited that utilizing a procedure called a biclique attack could recoup AES keys speedier than a savage power attack by an element of somewhere around three and five, contingent upon the figure form. Indeed, even this attack, however, does not debilitate the down to earth utilization of AES because of its high computational many-sided quality.

1.5.1.2 Modes of operation

Methods of operation are formal depictions of the way that you utilize single encryption on a message that is bigger than the block size , *b*, of the cipher. The plaintext is divided into *b* sized blocks.

Methods of operation (MOOs) are discriminating in making compelling utilization of a piece figure. Especially for DES encryption, the standard MOOs can give privacy, or trustworthiness, however not both.

ECB (electronic codebook) Mode: It is the most clear strategy for encryption. Each plaintext piece is encoded or unscrambled autonomously. Permit parallel encryption and unscrambling. In this mode, the made ciphertext is not darkened, and can bring about recurrence attacks.

CBC (figure piece anchoring) Mode: The CBC technique for encryption was planned by IBM in 1976. The result is encoded using a figure's count as a piece of the standard way. Each following ciphertext square depends on upon the previous one. The principle plaintext square is added XOR to a subjective instatement vector (routinely suggested as IV). The vector has the same size as all plaintext squares.

PCBC (propagating cipher-block chaining or plaintext cipher-block chaining) Mode: The PCBC mode is similar to the officially depicted CBC mode. It moreover gives mixing bits in the midst of encryption coming about plaintext messages. Instead of the CBC mode, if one ciphertext bit is hurt, the accompanying plaintext square and other following ones will be hurt and not ready to scrutinize. In the PCBC mode both encryption and decryption can be performed using one and just string without a moment's delay.

CFB (cipher feedback) Mode: The CFB mode is similar to the effectively portrayed CBC mode. The rule differentiation is that one should encode mixed data from the past round (so not plaintext squares) and a while later add to plaintext bits. It doesn't impact the security quality anyway it results in using figure's encryption estimations (the same that were used for scrambling plaintext) in the midst of decrypting system.

OFB (output feedback) Mode: Algorithms that work in the OFB mode, make keystream bits that are used for encryption resulting data squares. In such way, the system for working of the square figure gets the opportunity to be similar to the technique for working of a regular stream figure..

CTR (counter) Mode: Utilizing the CTR mode makes piece figures technique for working like stream figures' system for working. As in the OFB mode, keystream bits are made paying little personality to substance of encoded data squares. In this mode, ensuing estimations of a growing counter are added to a nonce worth and the results are mixed of course. The nonce expect the same part as presentation vectors in the past modes.

1.5.2 Asymmetric Cryptology

Asymmetric cryptography or public-key cryptography will be cryptography in which a couple of keys is utilized to encrypt and decrypt a message with the goal that it arrives safely. To start with there is an open and private key pair from a testament power for the client.

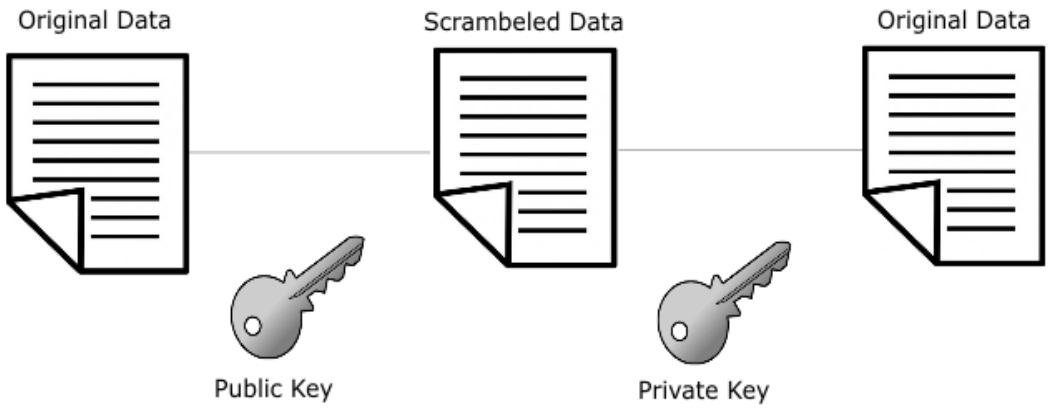

Figure 1.6 Asymmetric Encryption

In the event that a message needed sent in a secure way, sender can get the target public key from an open catalog. The public key encryption is performed with public key and after the encryption the message is sent to the beneficiary.

After the beneficiary gets the encrypted message, he uses his private key for decrypting t the message. Nobody else however the beneficiary ought to have the private key.

1.5.3 RSA

RSA is a cryptosystem for open key encryption, and is broadly utilized for securing delicate information, especially while being sent over an unstable system, the Internet. RSA was initially depicted in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman of the Massachusetts Institute of Technology.

This is otherwise called unbalanced cryptography. It utilizes two distinctive however numerically connected keys, one open and one private. People in general key is imparted to everybody except the private key must be mystery. The message in RSA can be encoded by both open key and private key. The inverse key from the one used to encode a message is utilized to unscramble it.

It gives a system for guaranteeing the secrecy, honesty, genuineness and non-reputability of electronic interchanges and information stockpiling. Numerous conventions like SSH, OpenPGP, S/MIME, and SSL/TLS depend on RSA for encryption and computerized capacities. It is likewise utilized as a part of programming projects.

The security of RSA depends on the computational trouble of figuring vast numbers. Encryption quality is specifically attached to key size, and multiplying key length conveys an exponential increment in quality, despite the fact that it weaken execution. RSA keys are ordinarily 1024-or 2048-bits in length, yet specialists trust that 1024-piece keys could be softened up the not so distant future, which is the reason government and industry are moving to a base key length of 2048-bits.

1.5.4 Digital signatures

A digital signature is a scientific method used to accept the realness and respectability of a message, programming or computerized record. Advanced marks can give the added affirmations of proof to root, character and status of an electronic archive, exchange or message, and in addition recognizing educated assent by the underwriter.

Digital signatures depend on open key cryptography. Utilizing an open key calculation, for example, RSA, one can create two keys that are numerically connected: one private and one open. To make a computerized signature, marking programming, (for example, an email system) makes a restricted hash of the electronic information to be agreed upon. The private key is then used to encrypt the hash. The explanation behind scrambling the hash rather than the whole message or archive is that a hash capacity can change over a discretionary data into a settled length esteem, which is normally much shorter. This spares time since hashing is much quicker than marking. The hash's estimation is extraordinary to the hashed information. Any adjustment in the information, notwithstanding changing or erasing a solitary character, results in an alternate worth.

A digital signature can be utilized with any sort of message whether it is encrypted or not just so the beneficiary can make sure of the sender's character and that the message arrived in place. Computerized marks make it troublesome for the underwriter to deny having marked something accepting their private key has not been traded off as the advanced mark is extraordinary to both the record and the endorser, and it ties them together.

1.5.5 HMAC

The Message Authentication Code (MAC) is a generally utilized method for performing message verification. HMAC (another way to say "keyed-Hashing for Message Authentication"), a minor departure from the MAC calculation, has risen as an Internet standard for a mixture of uses.

Message validation is a technique that permits conveying gatherings to confirm that got messages are bona fide. The two critical perspectives are confirming that the message's substance have not been adjusted and that the source is bona fide. The Message Authentication Code (MAC) is a generally utilized method for performing message validation. A minor departure from the MAC calculation has risen as an Internet standard for a wide mixed bag of uses

There have been different suggestions to join a secret key into a present hash count. HMAC got the most support. HMAC has been picked as the required to-complete MAC for IP Security, and is used as a piece of other Internet traditions, for instance, Transport Layer Security (TLS, soon to supplant Secure Sockets Layer) and Secure Electronic Transaction (SET).

HMAC Algorithm:

HMAC algorithm is expressed by Dobbs as follows [22]:

1. Append zeros to the left end of K to create a b-bit string K+ (for example, if K is of length 160 bits and b = 512, then K will be appended with 44 zero bytes 0x00).
2. XOR (bitwise exclusive OR) K+ with ipad to produce the b-bit block Si.
3. Append M to Si.
4. Apply H to the stream generated in Step 3.
5. XOR K+ with opad to produce the b-bit block So.
6. Append the hash result from Step 4 to So.
7. Apply H to the stream generated in Step 6 and output the result.

1.6 AUTHENTICATION

The idea of authentication is recognizable to very nearly everybody. Confirmation is the procedure of affirming somebody's personality with the supplied parameters like username and secret key. In security frameworks, verification is particular from approval, which is the procedure of giving people access to framework items in light of

their character. Approving that personality sets up a trust relationship for further communications [5,6].

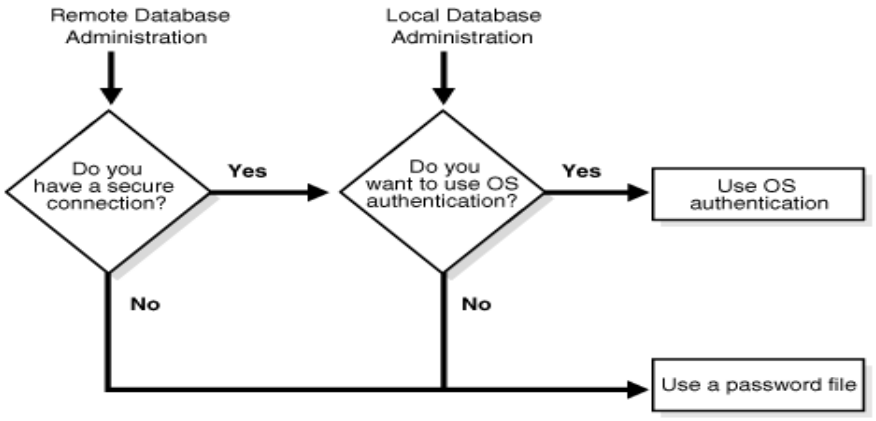

Figure 1.7 Database Authentications

Database authentication is the procedure or demonstration of affirming that a client why should endeavoring log into a database is approved to do as such, and is just concurred the rights to perform exercises that he or she has been approved to do. In the setting of databases verification gets one more measurement in light of the fact that it may happen at distinctive levels. It might be performed by the database itself, or the setup may be changed to permit either the working framework, or some other outer system, to confirm clients.

There are some confirmation sorts gave to verify into a Database framework (DBS):

- **Server:** This is the default security instrument. Verification happens on the server utilizing neighborhood working framework security. On the off chance that a client ID and secret word are indicated amid the association or connection endeavor, they are contrasted and the legitimate client ID and watchword blends at the server to figure out if the client is allowed to get to the occasion.
- **Server Encrypt**: Specifies that the server acknowledges encoded SERVER validation plans. In the event that the customer verification is

SERVER_ENCRYPT, the customer is confirmed by passing a decoded client ID and an encoded watchword to the server. On the off chance that the customer confirmation is SERVER, the customer is verified by passing a decoded client ID and a decoded secret word to the server.

- **Client:** Indicates that validation happens on the server/customer where the application is conjured utilizing working framework security. The client ID and secret key indicated amid an association or connection endeavor are contrasted and the legitimate client ID and watchword mixes on the customer hub to figure out if the client ID is allowed access to the case. No further verification will happen on the database server. On the off chance that the client performs a nearby or customer login, the client is known just to that neighborhood customer workstation. In the event that the remote occurrence has CLIENT validation, two different parameters focus the last verification sort: TRUST_ALLCLNTS and TRUST_CLNTAUTH.

CHAPTER 2

PREVIOUS WORKS

In this chapter, we present the previous works in the literature related to this work. Interested readers can access the works for more details by the references.

2.1 MERKLE'S TREE

When the data outsourced,an on-line attacker and malicious manager of the outsourced site can manipulate the data. In cloud setting it called the integrity of the deployed data on the cloud. The first solution to this is pioneered by Merkle in 1989,and his works open new direction to many [10]]. Merkle proposed using binary trees to check the integrity of the data.

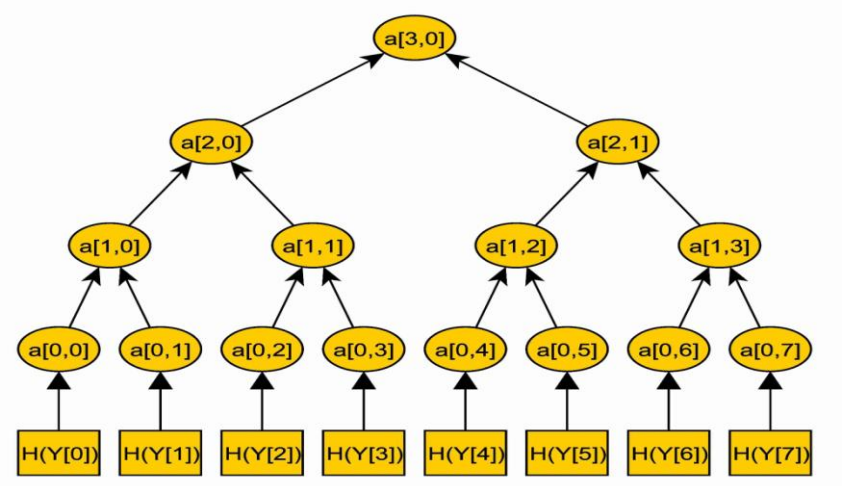

Figure 2.1 Merkle's Tree

In Merkle’s tree, every non-leaf node is labeled with the hash of the labels of its children nodes, see the Figure 2.1. Merkle's tree is extremely valuable in cryptography in light of the fact that they permit efficient and secure check for expansive information structures. This sort of tree can be utilized to confirm put away information, took care of and exchanged between two machines. The fundamental utilization of Merkle's tree is to verify that amid the information exchange the information squares are undamaged and unaltered.

2.2 A GENERAL MODEL FOR AUTHENTICATED DATA STRUCTURES

Martel,C., et. al (2004) have taken a shot at a model for social occasion credible information from the inquiries. Augmented Merkle thought into B+ tree (used the Merkle hash tree to give validation). Genuine distribution is a novel plan which permits un-trusted distributers to safely answer inquiries from customers in the interest of trusted logged off information proprietors.

In this work, they build up a novel, information model called Search DAGs and a summed up calculation for the development of check item (VOs). They demonstrate that the VOs along these lines developed are secure, reduced and proficiently calculable and irrefutable. They utilize seek DAGs to demonstrate the security of two new and a great deal more mind boggling information models for effective multi dimensional reach looks.

This permits reduced VOs to be registered (size O (logN +T)) for run of the mill 1-D and 2-D reach questions, where the inquiry answer is of size T and the database is of size N. They likewise how I/O proficient plans to develop the VOs . For a framework with circle squares of size B, they answer 1-D and 3-sided extent questions and register the VOs with O(logBN +T=B) I/O operations utilizing direct size information structures.

Authentic publication is a novel plan which permits un-trusted distributers to safely answer inquiries from customers in the interest of trusted logged off information proprietors. This methodology gives more prominent versatility (by including more distributers) and better security (on-line distributers don't should be trusted) [10].

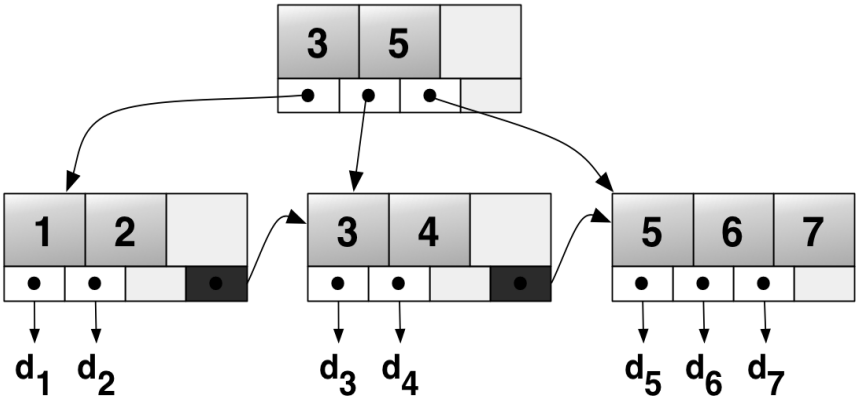

Figure 2.2 B+ Tree

2.3 ENCRYPTED DATABASE INTEGRITY IN DATABASE SERVICE PROVIDER MODEL

Hacigumus, H., Iyer, B. and Mehrotra, S. proposes an encrypted database uprightness conspire that permits the information's proprietor to guarantee the database's honesty facilitated at the administration supplier site, notwithstanding security of the put away information against pernicious assaults.

In their work, they take a gander at another imperative issue. In spite of the fact that the customer's information is shielded from the both outcasts and ASP it is misty by what means can the customer guarantee the database's uprightness and consequences of the inquiries reported back by the ASP.

They have exhibited this work by isolating the honesty issue in two measurements:

- When the customer gets a record from the server, in what capacity can the customer guarantee the record's uprightness? That is, by what method can the customer confirm that the information has not been changed in an unapproved way?
- When the customer gets set of answers for the question, by what method can the customer be sure that the greater part of the records, which are qualified by the inquiry, are incorporated into the outcomes?

They propose two-level encoded database trustworthiness plan, which comprises of Record Level Integrity and Table Level Integrity ideas, as an answer for this issue [10]. Their plan is joined with encrypted database stockpiling model. In resultant framework gives security of the put away information against vindictive assaults and in addition database honesty highlights, which guarantee the realness and legitimacy of the information put away at the administration supplier site.

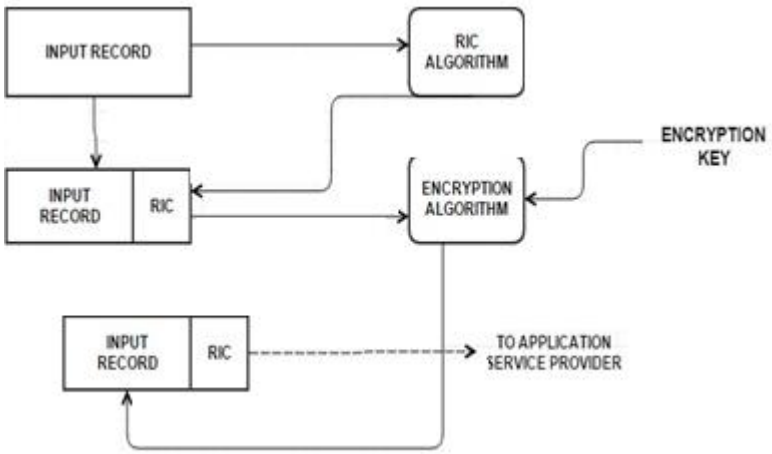

Figure 2.3 Record Level Integrity with RICs

2.4 EXECUTING SQL OVER ENCRYPTED DATA IN THE DATABASE SERVICE PROVIDER MODEL

Hacıgumus, H., Chen Li, Iyer, B and Mehrotra, S. have displayed a work on Executing SQL over Encrypted Data in the Database-Service-Provider Model [12]]. In

their work, they have considered a particular test about the proprietor's trust to the administration supplier.

They propose to store the information at the administration supplier in the wake of scrambling it so just the proprietor can unscramble. They say that the administration supplier can execute the question without unscrambling the information so the information can be more secure. There is utilized one strategy that permits the supplier to execute SQL at the supplier site. Just the proprietor can decrypt the outcome.

The method that they have utilized conveys a "coarse record" which permits fractional execution of SQL questions on the supplier side. The consequence of this inquiry is sent to the customer.

They proposed a procedure to work the SQL inquiry, and split it into a server question and a customer inquiry. The administration supplier holds the obligation to deal with the information's steadiness. This guarantees downright security to the customer and the expense of participating in inquiry execution with the administration supplier.

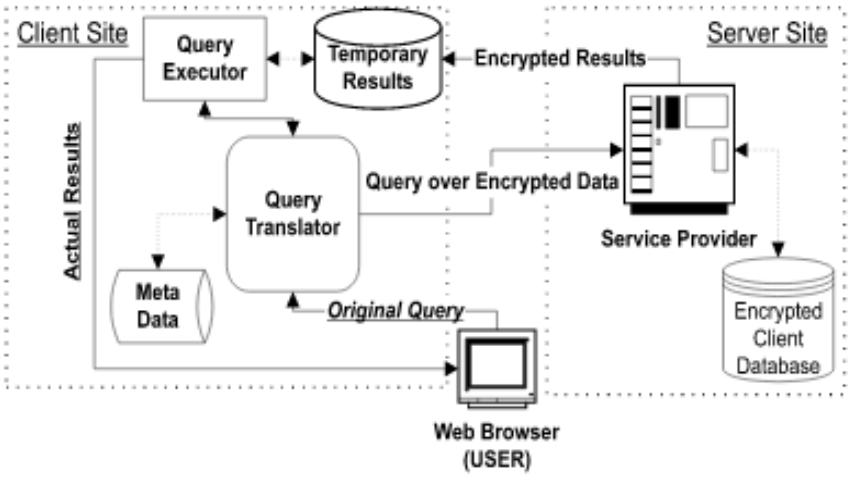

Figure 2.4 The Service Provider Architecture

2.5 AUTHENTICATION AND INTEGRITY IN OUTSOURCED DATABASES

Mykletun, E., Narasimha, M. and Tsudik,G. proposed a paper identified with Authentication and Integrity in Outsourced Databases[12]. This work gives components that guarantee the client that the inquiry results have not been messed with and are genuine as for the real information proprietor.

This work considered the issue of giving proficient information uprightness components in the outsourced database model. We introduced a novel methodology in light of mark accumulation procedures. In particular, we proposed a protected and viable Condensed-RSA plan which performs well in the Unified Client and additionally the Multi-Querier models.

In any case, since it doesn't give a way to total marks from diverse underwriters, we saw that it is not ideal in the Multi-Owner model. Then again, while the mark plan proposed by Boneh,et al. totals marks from unmistakable clients into one short mark, the computational many-sided quality of this plan was appeared to be very high. Thusly, as a piece of our future work, we need to concentrate on discovering effective and useful mark plans for the Multi-Owner model which can total marks produced by particular clients.

They research both security and productivity parts of the issue and builds a few secure and functional plans that encourage trustworthiness and legitimacy of question answers while bringing about low computational and correspondence costs.

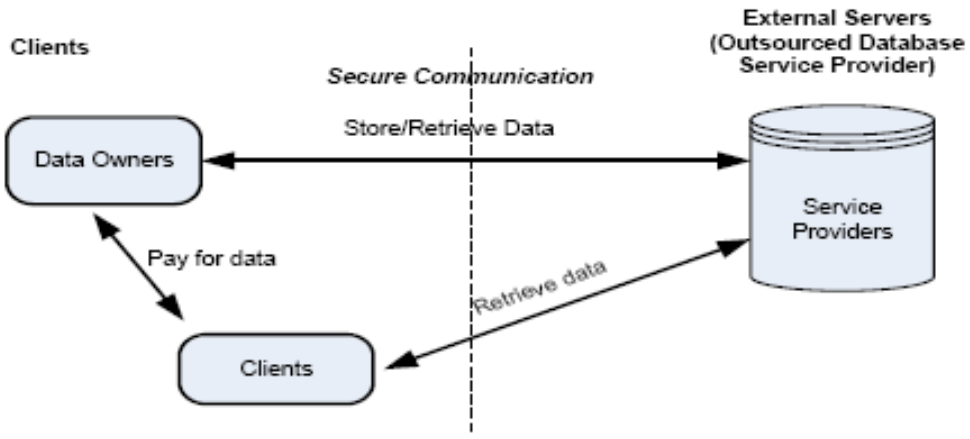

Figure 2.5 Outsourced Database Model (ODB) **[12]**

2.6 AGGREGATION QUERIES IN THE DATABASE AS A SERVICE MODEL

Einar Mykletun and Gene Tsudik propose a basic option for taking care of encrypted accumulation inquiries and depict its execution. They likewise consider an alternate kind of the DAS model which includes blended databases, where a few qualities are encrypted and some are left free. They demonstrate how range inquiries can be executed in this model [13].

In their introduced work they demonstrate that the homomorphic encryption plan in [10] is unreliable by exhibiting its defenselessness to a ciphertext-just attack.

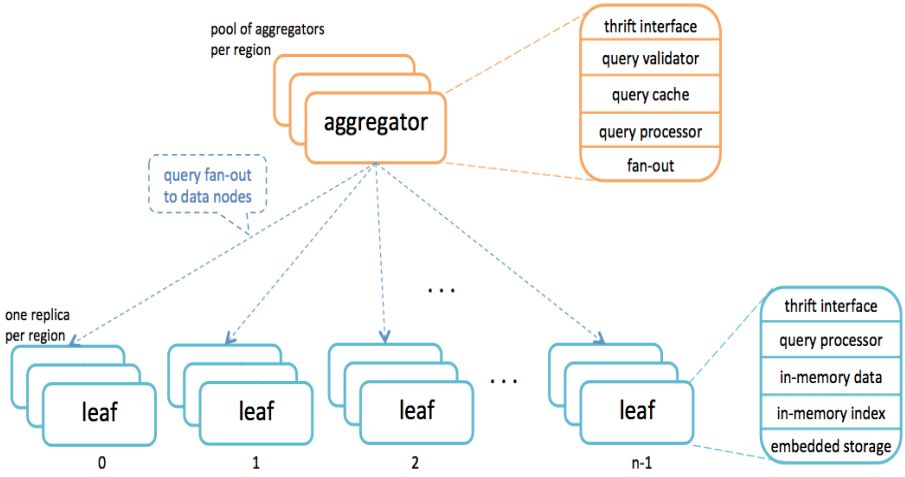

Figure 2.6 Aggregation Queries

Their proposed option does not include homomorphic encryption capacities. They further portray the conventions for detailing and executing questions and also upgrading encrypted tuples. They concentrate on a variation of DAS which has not been investigated up to this point: the purported blended DAS model, where a few properties are delicate while others are most certainly not.

2.7 AUTHENTICATION OF OUTSOURCED DATABASES USING DIGITAL SIGNATURE AGGREGATION AND CHAINING

Narasimha, M. and Tsudik, G. proposed authentication by utilizing Digital Signature Aggregation and Chaining (DSAC) ,by the customer [2]. They assert that

thusly is a productive for the most base-level questions, without requiring any perplexing information structures.

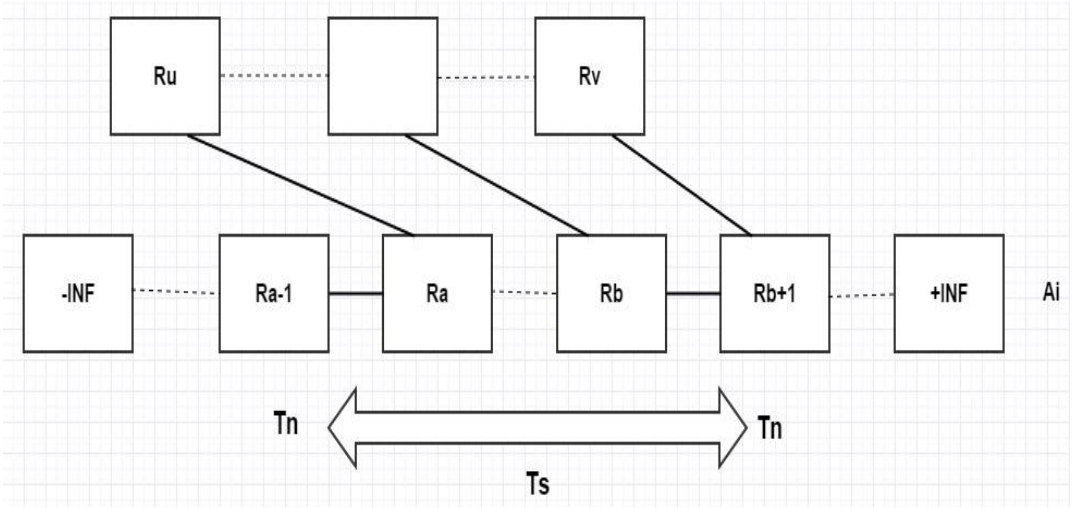

Figure 2.7 Signature Chain

By utilizing mark chain, see the figure 2.7, the server answers an extent question by discharging all coordinating tuples and the two limit tuples. The mark chain gives that the server has given back all tuples in the inquiry range. For reach (or correct worth) questions that outcome in no matches, the server makes, an unfilled verification by returning just the two limit tuples that subsume the non-existent esteem or range.

In DSAC, the proprietor signs each tuple before putting away into the outsourced database at the distributer/server's site. The server stores the tuple signature alongside each tuple. In light of an inquiry, the server basically sends the coordinating tuples and their marks to demonstrate respectability and credibility of the outcome.

2.8 EFFICIENT DATA INTEGRITY CHECKING FOR UNTRUSTED DATABASE SYSTEMS

Anderson Luiz Silverio et al, in their work consider the issue of guaranteeing information honesty and legitimacy in outsourced database situations. They give effective and secure method for guaranteeing information trustworthiness and

legitimacy while causing negligible computational overhead. They offer a few strategies in light of Message Authentication Codes (MACs) to identify pernicious and unapproved insertions, redesigns and cancellations of information [2].

Their systems concentrate on procedures for recognizing unapproved activities (insertions, cancellations and redesigns) from a helpless database server. Their systems offer two essential focal points:

- DBMS is free and can be effortlessly sent in existing situations.
- They concentrated on utilizing a more straightforward and effective cryptographic calculation to give the respectability check.

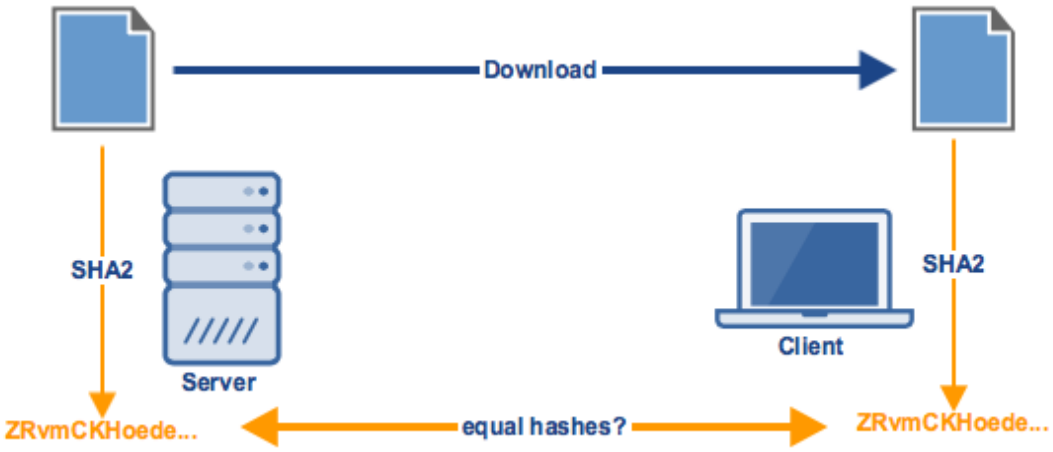

Figure 2.8 Efficient Data Integrity Checking

They trust that the straightforwardness and autonomy of our strategy makes it effectively deployable and good with genuine requests.

CHAPTER 3

THE FRESHNESS OF RELATIONAL DATABASES DEPLOYED ON THE CLOUD

3.1 THE FRESHNESS PROBLEM:

Data freshness has been recognized as a standout amongst the most imperative characteristics of information quality for information purchasers. A few reviews and exact studies have demonstrated that information freshness is connected to data framework achievement. At that point, accomplishing required information freshness is a test for the advancement of an expansive utilizations.

The freshness (rollback) problem of cloud hosted DBs occurs when a successful on-line attacker or a malicious client at cloud downloads at least all of a table from a database and in a later access they can replace the downloaded table with the current table. The integrity and authentication approaches cannot prevent this rollback attack. We begin with a series of theorems to show the case.

Theorem 1. *Let a honest data owner D has a single data storage S on the cloud. D, firstly, inserts data d_1 into S. Later, D uploads d_2 as an update of d1 into S, where $d_1 \neq d_2$. The D can't distinguish that the d_1 or d_2 is the last update of S as long as the D doesn't stores any information about the S.*

We prove this theorem by a game played with the data owner and a malicious data host.

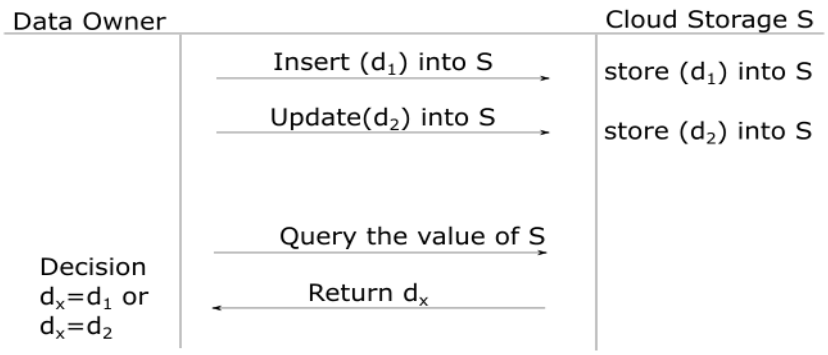

Figure 3.1 Proof of theorem 1

The previous theorem talks about raw that don't include any cryptographic security against modification of S by an on-line attacker or malicious administrators. Next theorem will show that hashing and singing will not prevent this.

Theorem 2. *Let a honest data owner D has a single data storage S on the cloud. D, firstly, hashes and digitally signs his data d_1 with a collusion resistant hash function as SHA-3 and computationally secure digital signature as DSA. Let d'_1 = d_1||sign(hash(d1)) Then D inserts data d^1_1 into S. Later, D uploads d'_2 = d_2||sign(hash(d2)) as an update of d'_1 into S, where d_1 != d_2 . The D can't distinguish that the d_1 or d_2 is the last update of S as long as the D doesn't stores any information about the S.*

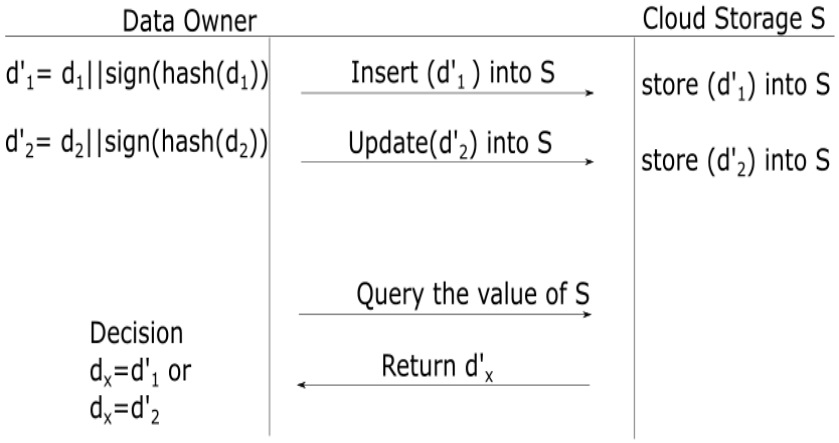

Figure 3.2 Proof of Theorem 2

The theorem 1 and 2, in short, tells us that without storing any information we cannot have freshness on outscored data.

Theorem 3. *Let a honest data owner D has a single data storage S on the cloud. D, firstly, hashes and digitally signs his data d_1 with a collusion resistant hash function as SHA-3 and computationally secure digital signature as DSA. Let $d'_1 = d_1||d_{1HS}$ where d_{HS}=sign(hash(d_1)). Then D inserts data d01 into S. Later, D uploads $d'_2 = d_2||d_{2HS}$ where d_{2HS} = sign(hash(d_2)) as an update of d'_1 into S, where d_1 != d_2. The D can distinguish that the d_2 is the last update of S if D stores d_{2HS}.*

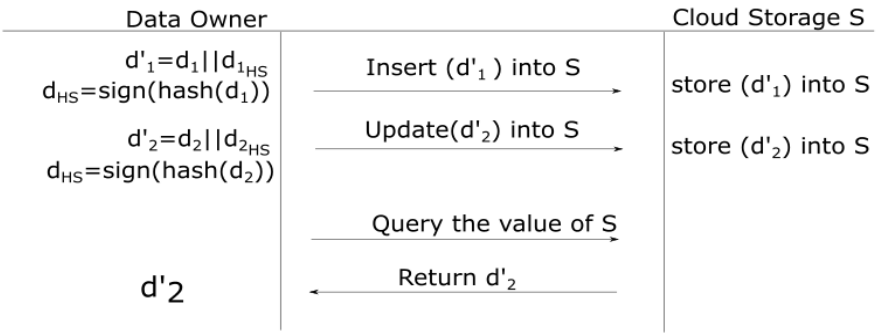

Figure 3.3 Proof of Theorem 3

3.2 THE SETUP AND SUPPORTED SQL QUERIES

To show that RDBMS-'s can have freshness we consider only one relation (table) R with n attributes (columns) a_1, a_2... a_n. In this work, we only consider INSERT, DELETE, UPDATE, and SELECT statements and we can list the queries as;

- ***SELECT*** ** FROM R WHERE* A_i *...*
- ***INSERT*** *INTO R VALUES (*a_1*,..* a_{n-1}*, HMAC)*
- ***DELETE*** *FROM R WHERE aj = :::*
- ***UPDATE*** *R SET col1=val1,col2=val2,... WHERE* a_j *= ...*

We have two approaches. First one is storing the HMAC'-s of the rows in a local table and the other one is storing the time of insert/update of rows as a time stamp as a column attribute on the cloud database and locally.

3.3 STORE THE HMAC ON AS

In order to provide the integrity and authentication of a row, Silverio et.al used the HMAC of a row. In the first approach, we stored the HMAC of the rows in a local database. We have selected SQL database as local database. We begin with insertion queries.

1) *Insertion Queries*: To guarantee that our local and cloud DB are synchronized, we also store the *last_rowid* in the application server. The steps of insertion query can be found on the Process 1. To simplify the insertion queries we also keep the last row's id. LID in the AS.

Process 1: PROCESSING INSERT QUERY

Input: Attributes for a_j for the insertion query q
Output: Inserted query freshness and Authentication.

1: $h = \textbf{HMAC}(a_1, a_2, \ldots, a_j)$
2: $q' = q \| HMAC(q)$
3: **Execute** query q' on the database server
4: **Insert** $LID, HMAC(q)$ into local table
5: **Increment** LID

Process 1.1 Processing Insert Query

2) Read Queries: For the freshness of the read query, the HMAC is compared with the local storage. The steps of read query can be found on the Process 2.

Process 2: PROCESSING READ QUERY

Input: Input query Q
Output: Query Result with freshness and Authentication.

1: T:= **Execute** query Q on the database server
2: T':= **Execute** query Q' on the local database
3: **for each** row in L **do**
4: **if** $HMAC(T_) \neq HMAC(row_{id})$
then return failure 1
5: **if** $HMAC(row_j) \neq HMAC(row_i)$
then return failure 2
6: **else continue**
7: **return success**

Process 1.2 Processing Read Query

In the process 2, the failure 1 is integrity and authentication failure. The failure 2 stands for freshness failure.

3) Update Queries: An update query is very similar to an insertion query. The main difference is the LID. The steps of an update query can be found on the Process 3.

Process 3: PROCESSING UPDATE QUERY

Input: Update query q with attributes a_j
Output: Updated Query with freshness and Authentication.

1: **Excute** a read query on the row.
2: **Check** HMAC
3: **Extract** the $rowID$
4: $h = \textbf{HMAC}(a_1, a_2, \ldots, a_j)$
5: $q' = q \| HMAC(q)$
6: **Execute** query q' on the database server
7: **Insert** $rowID, HMAC(q)$ into local table

Process 1.3 Processing Update Query

4) *Delete Queries*: The steps of an delete query can be found on the Process 4

Process 4: PROCESSING DELETE QUERY

Input: Delete query q
Output: Deleted row with table's freshness and authentication.

1: **Excute** a read query on the row.
2: **Check** HMAC
3: **Extract** the $rowID$
5: **Excute** delete query q on the database server
7: **Delete** row where $id = rowID$ from local table

Process 1.4 Processing Delete Query

3.4 STORING TIME STAMPS ON APPLICATION SERVER

Instead of storing the HMAC locally, another option is storing a time stamp which is a part of the row's HMAC. The process of read, insertion, update, and delete queries are almost same.

1) *Insert Queries*

Process 5: PROCESSING INSERT QUERY

Input: Attributes for a_j for the insertion query q
Output: Inserted query freshness and Authentication.

1: $h = \textit{TSTAMP}(a_1, a_2, \ldots, a_j)$
2: $q' = q \| \textit{TSTAMP}(q)$
3: **Execute** query q' on the database server
4: **Insert** LID, $\textit{TSTAMP}(q)$ into local table
5: **Increment** LID

Process 1.5 Processing Insert Query With Timestamp

2) *Read Queries*

Process 6: PROCESSING READ QUERY

Input: Input query Q
Output: Query Result with freshness and Authentication.

1: T:= **Execute** query Q on the database server
2: T':= **Execute** query Q' on the local database
3: **for each** row in L **do**
4: **if** $TSTAMP(T_) \neq TSTAMP(row_{id})$ **then return failure 1**
5: **if** $TSTAMP(row_j) \neq TSTAMP(row_i)$ **then return failure 2**
6: **else continue**
7: **return success**

Process 1.6 Processing Read Query with Timestamp

3) Update Queries

Process 7: PROCESSING UPDATE QUERY

Input: Update query q with attributes a_j
Output: Updated Query with freshness and Authentication.

1: **Excute** a read query on the row.
2: **Check TSTAMP**
3: **Extract** the $rowID$
4: h =**TSTAMP**$(a_1, a_2, \ldots, a_j)$
5: $q' = q \| \text{TSTAMP}\ (q)$
6: **Execute** query q' on the database server
7: **Insert** $rowID$, TSTAMP (q) into local table

Process 1.7 Processing Update Query with Timestamp

4) Delete Queries

Process 8 : PROCESSING DELETE QUERY

Input: Delete query q
Output: Deleted row with table's freshness and authentication.

1: **Excute** a read query on the row.
2: **Check** TSTAMP
3: **Extract** the $rowID$
5: **Excute** delete query q on the database server
7: **Delete** row where $id = rowID$ from local table

Process 1.8 Processing Delete Query with Timestamp

3.5 MERKLE'S TREE APPROACH

The previous approaches require a local DB and processing time. The number of the rows of local DB is the same as the cloud hosted database. A nice idea is using the Merkle's pioneering idea on the integrity of outscored data.

Binary trees can be mapped into a linear structure as in Heap data structure as long as there is no gap is allowed. However, it will require too much computation for the cloud. When the relational DB is frequently updated, for every update and insert it will require to download, check and update.

3.6 CONCLUSION FOR FRESHNESS OF CLOUD HOSTED DATABASES

If the companies really consider the rollback as an serious attack, we advice them not to deploy their database into the cloud.

CHAPTER 4

METHODS OF INVESTIGATION

We will separate the use cases of the DBs as Single user vs Multiple users where the ownership of the database belong to the users or not. With this separation, we will discuss the provided security and integrity of the proposed methods. Based on this separation, we will propose a framework that will not only contain the uses cases but also the supported queries and performance comparisons.

Secondly, each proposed work will be implemented. Finally, all methods will be heavily tested under our framework with the DB.

We will copy into a local table the row id and the HMAC in order to ensure the data freshness. After doing this we will check if the HMAC that we copied is the same with the HMAC in the database. After this, we will prepare a paper related to our research and publish it in order to be useful in the feature works to be used as a framework.

Let us list some Test Cases for Database freshness. First let's start with the insertion into the Database.

We make Insertion as follows:

1. Insert One Record Without Protection
2. Insert One Record With HMAC
3. Insert One Record With TimeStamp

Insert One Record Without Protection

The following C code is used to insert a row into database without any protection. This code snippet can be found in .c file in appendix.

```
long InsertOneRecordWithoutProtection()
{
        char* query = "INSERT INTO benchmark.user ("
        "name, "
        "email, "
        "password"
        ") VALUES ("
        "\'Esmeralda Mihana\', "
        "\mihanaesmeralda@gmail.com\', "
        "\'……..\' "
        ");";
        return DBbenchmark("InsertOneRecordWithoutProtection", query);
}
```

Insert One Record with HMAC

The following C code is used to insert a row into database with HMAC. This code snippet can be found in .c file in appendix.

```
long InsertOneRecordWithHMACAndECMAC()
{       char query[1000];
        long dbTime = 0;
        char* stat = "INSERT INTO benchmark.user ("
                        "name, "
                        "email, "
                        "password, "
                        "hmac, "
                        "history"
                        ") VALUES ("
                        "\'Esmeralda Mihana\', "
                        "\'mihanaesmeralda@gmail.com \', "
                        "\'……..\', "
                        "\'%s\', "
                        "\'%s\');";
char* message = "Esmeralda Mihana mihanaesmeralda@gmail.com……..";
        struct ByteArray* hmac = generateHMAC((unsigned char*)message);
        char chunk[2*(hmac->size) + 1];
        mysql_real_escape_string(connection, chunk, hmac->data, hmac->size);
```

1. Update One Record Without Protection
2. Update One Record With HMAC

Update One Record Without Protection

The following C code is used to update a row into database without protection. This code snippet can be found in .c file in appendix.

```
long UpdateOneRecordWithoutProtection(int id)
{
        char query1[1000];
        char query2[1000];
        long dbTime = 0;
        char* stat1 = "UPDATE benchmark.user SET "
        "name = \'bledar kazia \', "
        "email = \'bledi.kazia@gmail.com\',"
        "password = \'……….\'"
        "WHERE "
        "id = %d ;";
        snprintf(query1, 1000 , stat1, id);
        char* stat2 = "SELECT id, name, email, password FROM
        benchmark.user WHERE id = %d ;";
        snprintf(query2, 1000 , stat2, id);
        dbTime += DBbenchmark("UpdateOneRecordWithoutProtection1",
        query2);
        mysql_free_result(mysql_store_result(connection));
        dbTime += DBbenchmark("UpdateOneRecordWithoutProtection2",
        query1);
        return dbTime;
}
```

Update One Record With HMAC

The following C code is used to update a row into database with HMAC. This code snippet can be found in .c file in appendix.

```
long UpdateOneRecordWithHMAC(int id)
{
	long dbTime = 0;
//	unsigned int id = 1 + random()%9000;
	char query2[1000];

	char* stat2 = "SELECT id, name, email, password FROM
benchmark.user WHERE id = %d ;";
	snprintf(query2, 1000 , stat2, id);
	dbTime += DBbenchmark("UpdateOneRecordWithHMAC",query2);
	mysql_free_result(mysql_store_result(connection));

	char query[10]00];
	char* message = "bledar kazia bledi.kazia@gmail.com.....";

	struct ByteArray* hmac = generateHMAC((unsigned char*)message);

	char chunk[2*(hmac->size) + 1];
	mysql_real_escape_string(connection, chunk, hmac->data, hmac->size);

	char* stat = "UPDATE benchmark.user SET "
			"name = \'bledar kazia\', "
			"email = \'bledi.kazia@gmail.com\', "
			"password = \'……..\', "
			"hmac = \'%s\' "
			"WHERE "
			"id = %d;";

	snprintf(query, 1000 , stat, chunk, id);
	dbTime += DBbenchmark("UpdateOneRecordWithHMAC", query);
	return dbTime;
}
```

Now let us talk about the deletion in the Database.

1. Delete One Record Without Protection
2. Delete One Record With HMAC

Delete One Record Without Protection

The following C code is used to delete a row into database without protection. This code snippet can be found in .c file in appendix.

```
long DeleteOneRecordWithoutProtection(int id)
{
        char query[1000];
        char* stat = "DELETE FROM benchmark.user WHERE id = %d";
        snprintf(query, 1000, stat, id);
        return DBbenchmark("DeleteOneRecordWithoutProtection", query);
}
```

Delete One Record With HMAC

The following C code is used to delete a row into database without protection. This code snippet can be found in .c file in appendix.

```
long DeleteOneRecordWithHMAC(int id)
{
	char query[1000];
	char query2[1000];
	MYSQL_RES *result;
	MYSQL_ROW row;
	unsigned long *lengths;
	long dbTime = 0;

	char* stat = "DELETE FROM benchmark.user WHERE id = %d";
	snprintf(query, 1000, stat, id);
	dbTime += DBbenchmark("DeleteOneRecordWithHMACAndECMAC", query);
	char* stat2 = "SELECT hmac FROM benchmark.user WHERE id IN (%d, %d);";
	snprintf(query2, 1000, stat2, id-1, id+1);
	dbTime += DBbenchmark("DeleteOneRecordWithHMACAndECMAC", query2);
	result = mysql_store_result(connection);

	row = mysql_fetch_row(result);
	lengths = mysql_fetch_lengths(result);
	struct ByteArray* hmacPredecessor = (struct ByteArray*) malloc(sizeof(struct ByteArray));

	hmacPredecessor->data = (unsigned char*)row[0];
	hmacPredecessor->size = (unsigned int)lengths[0];
```

CHAPTER 5

EXPERIMENTAL RESULTS

We performed the experiments by using a Dell Pc. Intel® Core™ i5-450M Processor, 3M cache, 2.40 GHz. HDD 1TB, 2048 Mb Display memory and a RAM of 3 GB. We used a Windows 7 Ultimate 64 bit Operating System.

The DBMS used is MySQL database and the experiments were performed in a machine running both MySQL server and client application. We use MySQL Server 2008 and MySQL Client 2008 for the set up.

There is used SHA-1 hash function to calculate the HMAC with a 256-bit long key.

We perform the Insert, Delete and Update operation as follows:

1. *Insert Queries*

   From the table above we can see the timing for insert queries, without protection and with HMAC. We calculated timing 5 times and we get an average as seen in the table:

Table 1.1 Insert Queries Timing

Without Protection		**With HMAC**	
Nr.	**Timing (s)**	**Nr.**	**Timing (s)**
1	0.0457	1	0.0487
2	0.0445	2	0.0475
3	0.0439	3	0.0482
4	0.0452	4	0.0481
5	0.0449	5	0.0478
Avg	**0.04484**	**Avg**	**0.04806**

We have to mention that 90% of the timing is spent on server site and only 10% on client site.

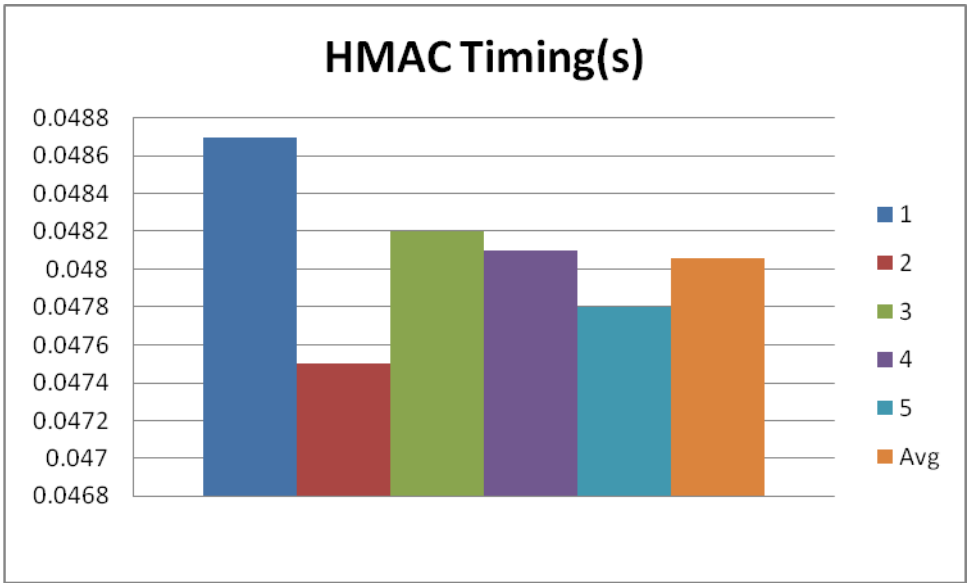

Figure 5.1 Insert Queries with HMAC

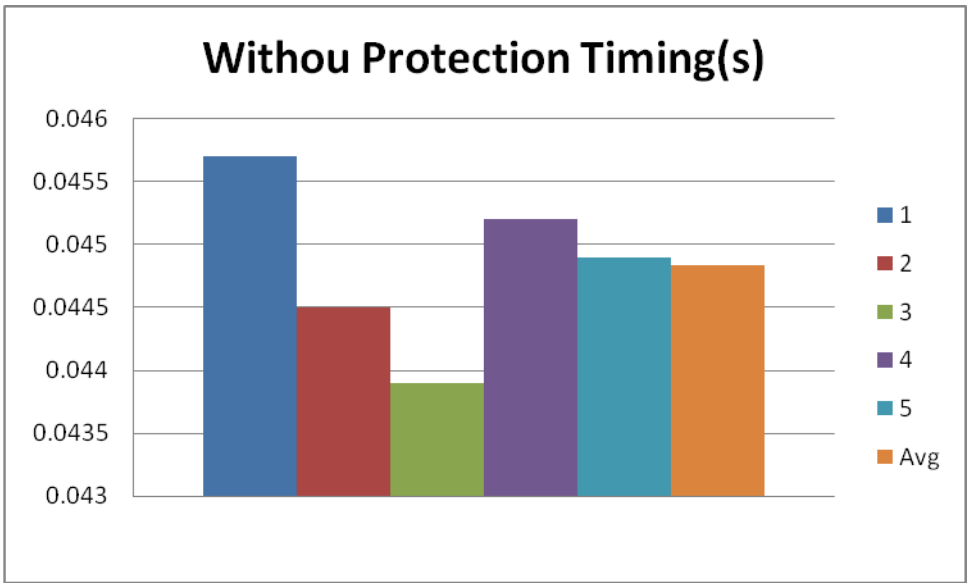

Figure 5.2 Insert Queries without Protection

2. *Delete Queries*

In the table below you will find the timing for the delete queries. We calculated the timing for the deleted query and got an average as in the table:

Table 1.2 Delete Queries Timing

Without Protection	
Nr.	**Timing (s)**
1	0.0532
2	0.0511
3	0.0541
4	0.0499
5	0.0544
Avg	**0.05254**

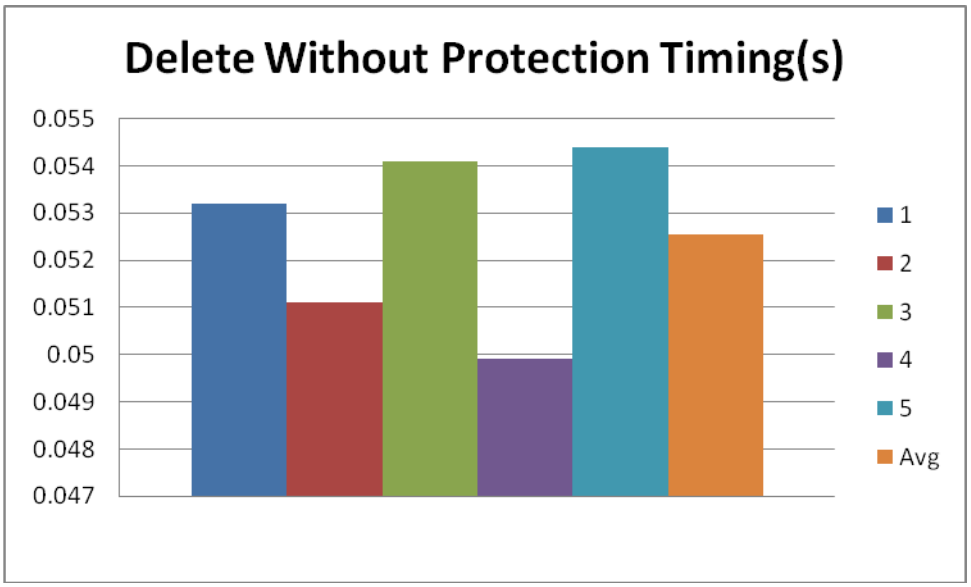

Figure 5.3 Delete Queries without Protection

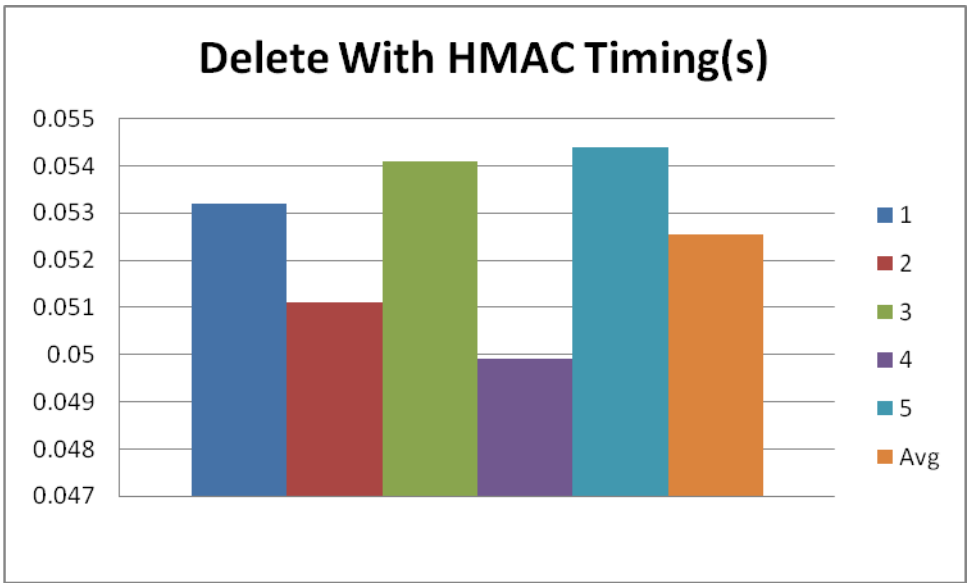

Figure 5.4 Delete Queries with HMAC

As we can view from the graph the deletion without protection and also with HMAC have the same value. This because when we perform Delete Queries it takes the same time for both because we do not have any extra operation at HMAC. The time spent on server site is 85% and the time spent on client size is 15%.

3. Update Queries

From the table above we can see the timing for insert queries, without protection and with HMAC. We calculated timing 5 times and we get an average as seen in the table:

Table 1.2 Update Queries Timing

Without Protection		**With HMAC**	
Nr.	**Timing (s)**	**Nr.**	**Timing (s)**
1	0.1312	1	0.1412
2	0.1298	2	0.1408
3	0.1330	3	0.1387
4	0.1352	4	0.1392
5	0.1348	5	0.1401
Avg	**0.1328**	**Avg**	**0.1400**

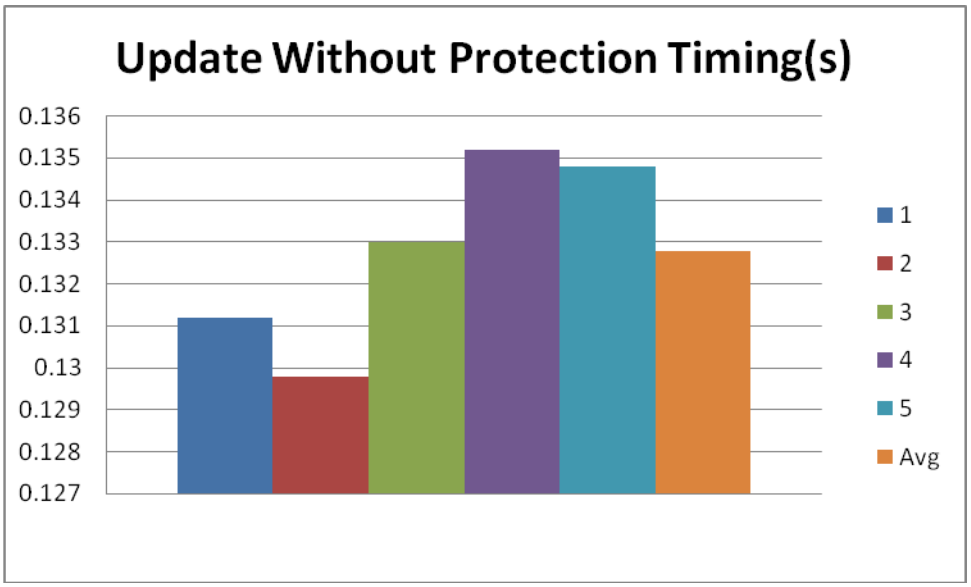

Figure 5.5 Update Queries without Protection

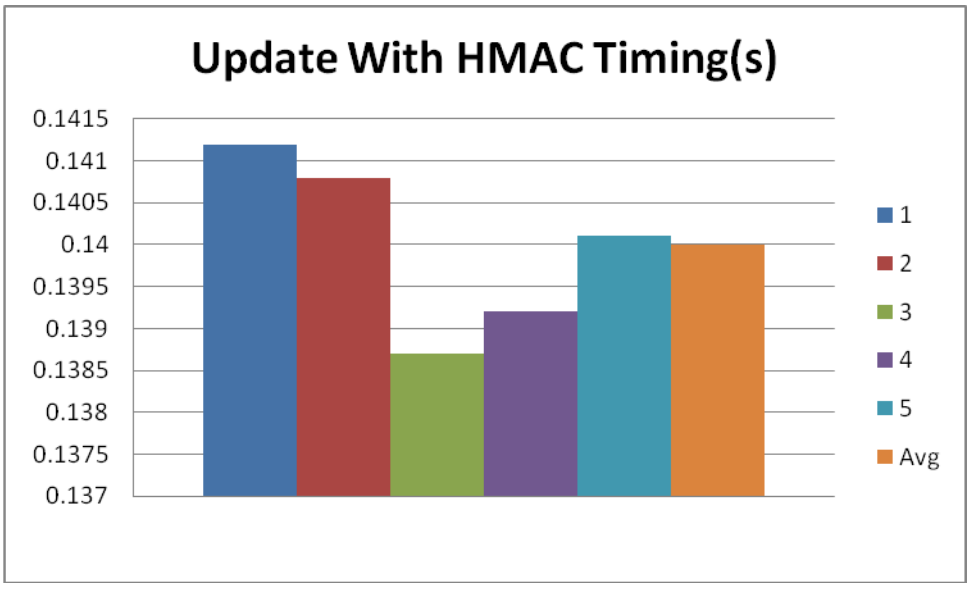

Figure 5.6 Update Queries with HMAC

The time spent on server site is 86% of update time and the time spent on client site is 14% of the update time.

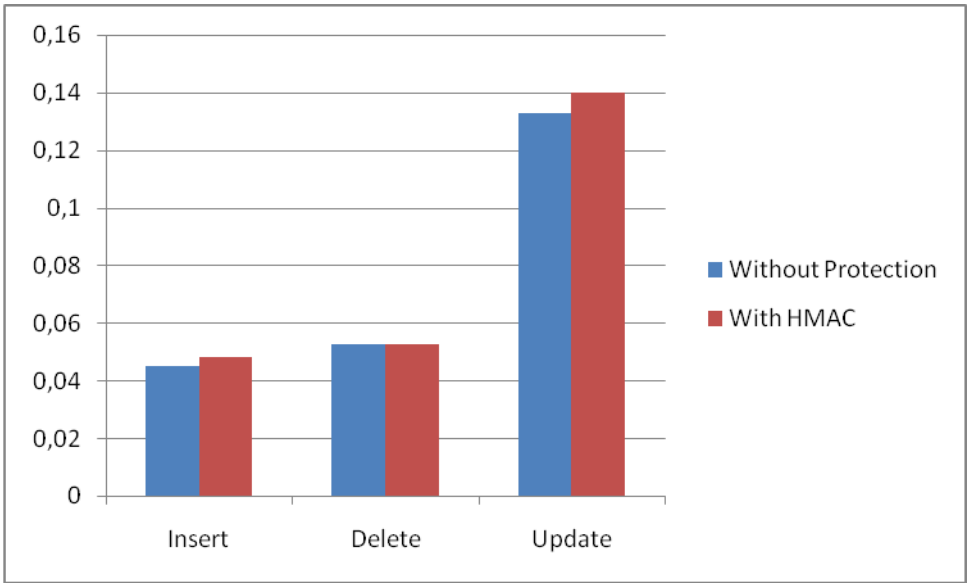

Figure 5.7 Queries Timing with HMAC and Without Protection

As is shown above when we perform insert query with HMAC we have 0.00322s more than without protection. For the delete query is the same because we delete the same thing in both cases and when we perform update query with HMAC we have 0.0072s more than without protection.

After performing these queries on the database we created a local table and save the last row id and the HMAC as follows:

```
int id = mysql_insert_id(con);
int HMAC = mysql_insert_HMAC(con);

printf("HMAC is: %d\n", id);
printf("The last inserted row id is: %d\n", id);

mysql_close(con);
exit(0);
```

int id = mysql_insert_id(con);

By using this command we can find the last rows id in order to save it to our local table.

int HMAC = mysql_insert_HMAC(con);

This command is used to find the HMAC.

fwrite(int id , fp);

fwrite(int HMAC,fp);

By using the above command we write the last row id and the HMAC locally.

Of course as you can understand from the solution there is some extra space locally that will be used to store the table that contains last row id and HMAC.

Table 2.1 HMAC Saved Locally

HMAC
F4905FA19DFCDD1309A27A6BB79B1252
11967649802F872E018BFD43E9AA5543
03D79DC4F96147E95D69682DF1A11AB9
B17EB0B070CBC70C69784A6DA97DADC1

The HMAC that is stored at the table below is called HMAC SHA1. We store this kind of HMAC because it needs less space so we can reduce the space needed locally.

After our calculation the space that will be needed for saving row id and HMAC is approximately *256 bits* for each table row.

We had to check if the HMAC stored is the same with the HMAC in the database.

```
if (table1.HMAC = = benchmark.user.HMAC)
printf("they are equal"\n)
else
printf("they are not equal"\n)
```

There is only some small possibility for the HMAC not to be equal. After the experiments we have done we found that HMAC cannot be equal only if at the message there is added another character.

CHAPTER 6

THE STORAGE OF CRYPTOGRAPHIC KEY PROBLEM of APPLICATION SERVERS

When the databases are deployed on the cloud they need to protect from malicious entities, on-line attackers or from the cloud host. All proposed and implemented techniques are relayed on cryptographic techniques that are requiring secret or private keys. The key must be stored somewhere so that the application server can access it easily whenever need it or there must be secure module that stores the key and performs cryptographic operations with the key. In this section, the security of key storage problem is discussed under various considerations. In this section it is assumed that the security, the integrity, the query completeness, and the freshness problems of cloud hosted databases are solved and they can answer the original queries required by application as in unprotected database.

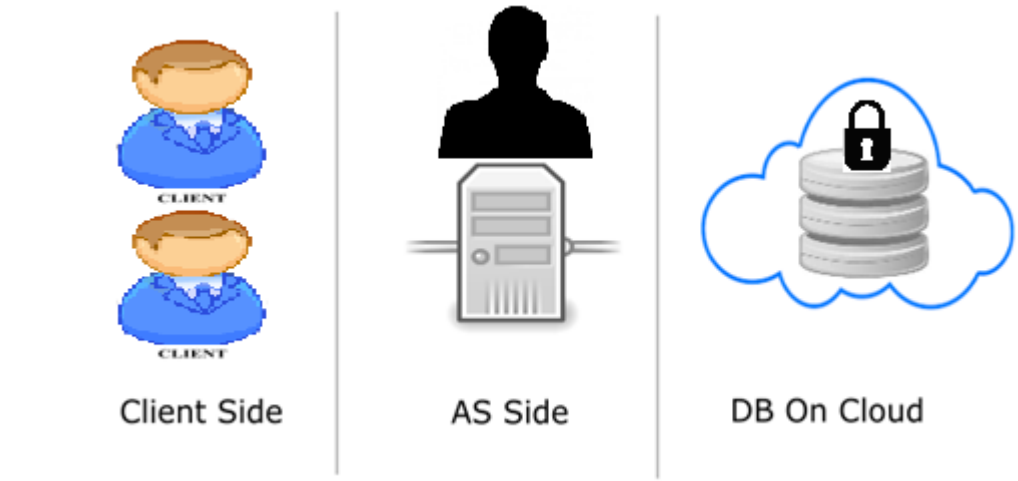

Figure 6.1 The database secure in the cloud but what about the crypto-keys

6.1 DB WITH SINGLE AS

The fundamental necessity of an AS incorporates information excess, high accessibility, burden adjusting, client administration, information or application security and a concentrated administration interface. Also, an application server may be associated by big business frameworks, systems or intranet and remotely got to by means of the Internet [18].

At the point when database scrambled, the Encryption Key (EK) must be put away in an ensured spot to keep the aggressors' entrance. The capacity can be a record or a registry key. The critical issue is confining access to it, permit just to the AS's client just. An application server executes and gives client and/or other application's entrance for the benefit of clients when using the introduced application's business or useful rationale.

A successful on-line attacker to the AS can steal the cryptographic keys. By another previous or later successful attack on the cloud hosted DB can have devastating results.

1. If the attack first applied on the AS, the keys, the database must be changed. This requires huge timing and labor work.

2. If the attack first occurred over the DB, the cryptographic keys used over the DB again must be changed. Otherwise a later successful attack will be result as in the next item.
3. If the attack occurred in both AS and DB, the attacker can reach all the secured information of the DB.
4. If the attack occurred in both AS and DB, they can manipulate the DB by performing operation over it. This is very dangerous, especially, if the attacks are not noticed.

In all the cases the most importing issue is the protection of the cryptographic

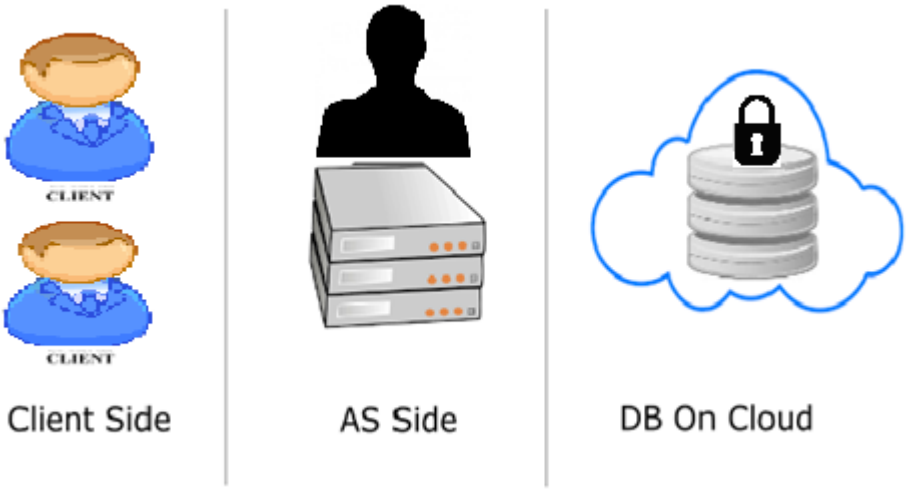

Figure 6.2 Application Server

6.2 COUNTERMEASURES

6.2.1 Third-Party

DB proprietor can store the marks. Third party is an individual association who is trusted by all gatherings included in the scrambled correspondences. It is the obligation of this association to keep its private key protected and mystery and to utilize that key to sign open keys of people it has checked. At the end of the day, in a bad position of meeting up close and personal to trade keys with each individual you wish to correspond with, you may draw in the administrations of a trusted outsider whose open

key you as of now need to go meet these people eye to eye. The outsider can then sign people in general keys and send them along to you, so you wind up with a checked duplicate without the inconvenience of trading every pair vis-à-vis. The points of interest of marking itself we will get to in a minute.

6.2.2 Hardware Security Module (HSM)

HSMs can be utilized in any application that uses computerized keys. Commonly the keys must be of high-esteem which means there would be a noteworthy, negative effect to the key's proprietor on the off chance that it were traded off.

The key is put away locally on simply the database server. A more secure choice is to utilize a Hardware Security Module (HSM) in either situation. This is an equipment gadget that is connected to the server and used to store the key as opposed to a confined organizer on the server.

HSM on AS can be a good candidate, however HSM's are passive and an attacker can use HSM to manipulate the DB [17].

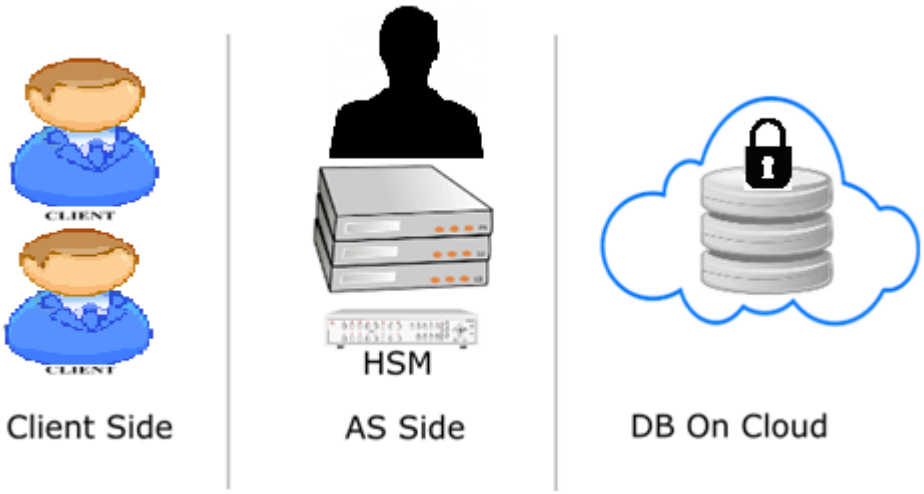

Figure 6.3 Hardware Security Modules

6.2 MULTIPLE ASs

Big companies and big websites uses more than one multiple AS to response their users as fast as possible. The number of ASs can excess 25 servers and one DB can execute their queries without any delay. In this case the attacker will have more than one AS to perform a successful attack.

A successful on-line attacker to the ASs as in the case of single AS can steal the cryptographic keys. And again as in single case. by another previous or later successful attack on the cloud hosted DB can have devastating results.

1. If the attack first applied on the ASs, the keys on the multiple ASs and the database must be changed. This requires more timing and labor work the single case.
2. As in the single case, if the attack first occurred over the DB, the cryptographic keys used over the DB again must be changed. Otherwise a later successful attack will be result as in the next item.
3. As in the single case, if the attack occurred in both ASs and DB, the attacker can reach all the secured information of the DB.
4. As in the single case, if the attack occurred in both AS and DB, they can manipulate the DB by performing operation over it. This is very dangerous, especially, if the attacks are not noticed.

In all the cases the most importing issue is the protection of the cryptographic keys and the solutions of the single case again can be used here.

CHAPTER 7

CONCLUSION

The defined framework will be a tool for who wants to protect their database systems deployed on the cloud. According to query requirements and application settings, they can use the results to apply their system or adjust to fit.

DB freshness is an important issue these days in the DB world. We think that this work will be useful for anyone or any company that will need to provide data freshness.

By using the local storage as we mentioned we can easily check the DB freshness by saving row id and HMAC and by checking if they are the same as in DB.

Of course there is a small disadvantage of our used method because it needs some extra local memory for saving the information locally. Using HMAC MD-5 for storing the information decreases a bit the storage needed.

This also is a secure model for the data stored in DB because locally there is only stored the HMAC and row id so the message cannot be decrypted locally as long as we don’t store locally the key.

In the feature we will prepare an article related to freshness and we will publish it. Also we will be part of some conferences with the freshness article.

REFERENCES

[1] Akin,I.H., Sunar,B. On the Difficulty of Securing Web Applications using CryptDB, PriSec 2014.

[2] Anderson L. Silverio and Ricardo F. Custodio, " Efficient Data Integrity Checking for Untrusted Database Systems " ,DBDKA 2014.

[3] Elisa Bertino, Chenyn Dai, Hyo-Sang Lim and Dan Lin, " High-assurance Integrity Techniques for Databases, BNCOD, Lecture Notes in Computer Science, Vol.5071, pp. 244-256, Springer, 2008

[4] Gopalan Sivathanu, Charles P. Wright, and Erez Zadok, " Ensuring Data Integrity in Storage:Techniques and Applications ", StorageSS'05, November 11, 2005, Fairfax, Virginia, USA.

[5] Hacigümüs, H and Iyer, B. R. Iyer and Mehrotra,S. Encrypted Database Integrity in Database Service Provider Model Certification and Security in E-Services, IFIP Conference Proceedings, Vol. 255, pp. 165-174, Kluwer, 2002.

[6] Hakan Hacigümüs and Bala Iyer and Chen Li and Sharad Mehrotra Executing SQL over encrypted data in the database-service-provider model Proceedings of the ACM SIGMOD International Conference on Management of Data, June 3–6, 2002, Madison, WI, USA, pp. 216-227, ACM Press, 2002

[7] James M. Slack and Elizabeth A. Unger, " A Model of Integrity for Object-Oriented Database Systems ", ACM, 1992.

[8] Charles Martel and Glen Nuckolls and Premkumar Devanbu and Michael Gertz and April Kwongand Stuart,G.Stubblebine © 2004 Springer-Verlag New York Inc. A General Model for Authenticated Data Structures

[9] Mell Peter, Grance Timothy (2009). The NIST definition of cloud computing. Retrieved February 25 2012 from http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[10] A.J.Menezes Handbook of Applied Cryptograph, CRC Press, 1996.

[11] Merkle, R. C. "A Digital Signature Based on a Conventional Encryption Function". Advances in Cryptology CRYPTO '87. Lecture Notes in Computer Science 293. p. 369.

[12] Einar Mykletun and Maithil Narasimha and Gene Tsudik. Authentication and integrity in outsourced databases TOS, 2(2), pp. 107-138, 2006.

[13] Einar Mykletun and Gene Tsudik, ” Aggregation Queries in the Database-As-a-Service Model ”, DBSec, Lecture Notes in Computer Science, Vol. 4127, pp. 89-103, Springer, 2006.

[14] Ajeet Ram Pathak, B. Padmavathi, “ Survey of Confidentiality and Integrity in Outsourced Databases ”, Vol. 2, pp. 122-128, 2013.

[15] R. R. Schell and D. E. Denning, “ Integrity in Trusted Database Systems ”, Proc. 9th NIST-NCSC National Computer Security Conference, pp. 30-36, 1986.

[16] W.Whitt Understanding the Efficiency of Multi–Server Service Systems Management Science, Vol. 38, :5, 708-723, 1992.

[17] Yao Chen and Radu Sion. 2011. To cloud or not to cloud?: musings on costs and viability. In Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '11). ACM, New York, NY, USA, , Article 29 , 7 page.

[18] TPC.Website;http://en.wikipedia.org/wiki/Transaction_Processing_Performance_Council

[19] http://en.wikipedia.org/wiki/Hardware_security_module

[20] http://en.wikipedia.org/wiki/Application_server

[21] http://en.wikipedia.org/wiki/Cloud_database

[22] http://www.drdobbs.com/security/the-hmac-algorithm/184410908