A New Schema for Securing Data Warehouse Hosted in the Cloud

Kawthar Karkouda, Ahlem Nabli, Faiez Gargouri

A new schema for securing data warehouse hosted in the cloud

Kawthar karkouda(✉),Ahlem nabli, Faiez gargouri

Miracl Laboratory, Higher Institute of Computer Science and Multimedia National,University of Sfax, Sfax, Tunisia

{Kawthar.karkouda,ahlem.nabli}@gmail.com,faiez.gargouri@isims.usf.tn

Abstract. Cloud computing is a new model in which computing infrastructure resources are provided as a service over the internet. Data owners can outsource their data in the cloud and benefit from on-demand high quality services. However, because cloud services and data owners are not in the same trusted domain, the outsourced data may be at risk. Therefore, data security is of critical importance in such a scenario. As a solution to this problem, data owners encrypt data before sending it to the cloud to ensure data confidentiality and cloud services decrypt it when processing queries. Such a scenario is not a suitable solution to data warehouse because of the large mass of data that will be processed .For that, this paper proposes a new schema based on the sample privacy homomorphism presented in [1]. This new schema improves the security of data warehouse hosted in the cloud, reduces storage space and processes encrypted data with minimum time complexity.

Keywords: Cloud computing, Data warehouse, Security, Integrity.

1 Introduction

Nowadays, Business Intelligence (BI) is a primordial factor for companies that are faced with large volumes of data. Indeed, the large volume of data accumulated over time is considered a rich source for decision makers. It allows them to have an overview on the various activities of the company and helps them make decisions. But building BI systems necessitates an important initial investment that can be a problem for adopting this technology.

On the other hand, cloud computing provides a new service that allows customers to create, maintain and query their data in the cloud using their internet connection. This new service delivery model is an important opportunity for companies because of its financial profitability, computing power and scalability. So, hosting a data warehouse in the cloud seems to be a good solution for BI.

But like each new technology, cloud computing brings its risks in terms of security. There are potential risks for sensitive data, for the latter are stored at an untrusted host. Consequently, before hosting data in the cloud, the owner must encrypt the data

with symmetric encryption or asymmetric encryption to ensure confidentiality. Such a scenario is not suitable for data warehouse because of the high volume of data stocked in the warehouse and processed in the OLAP query. This scenario cannot also guarantee the confidentiality of data over the cloud provider because it decrypts the data when processing queries. For this reason, the necessity of computing data in a cipher-text is proved with the homomorphic encryption function. Actually, this paper introduces a new model for sharing data warehouse in a multi- cloud. Our proposition is based on the homomorphic privacy presented in [1]. One serious deficiency of this homomorphic privacy is the possibility of being broken by clear text attacks. Thus, our contribution is to make this privacy homomorphism more robust and secure using multi-cloud and perturbation value. In addition, a new technique is proposed to verify the integrity of our data when received from the cloud. It should be noted that it is not our aim to propose a solution as secure as the state-of-the-art encryption algorithms. Rather a technique that provides a considerable level of overall security strength with respect to some performance overheads.
This article proceeds as follows: the section below surveys related works. The third section proposes a novel method for securely hosting and querying data warehouse in the cloud. The fourth part is then devoted to shed light on some theoretical results. Finally, the paper ends with a conclusion.

2 RELATED WORKS

The standard scenario for outsourcing data in the cloud is to encrypt data with symmetric encryption [2-3 ,4] or with asymmetric encryption [5] . After encrypting data, the owner sends this data to the cloud with the keys and stocks it in the data base of the provider . This scenario is not secure because intruders can break the security system of the provider and steal the data with the keys. Moreover, the owner cannot trust the provider.

The necessity of running data in cipher-texts is presented. Homomorphic encryption [6] allows computations on cipher-texts without decrypting them first. It is applied to attributes that are used in the computation of aggregation function such as sum and average. This technique is computationally expensive for practical use. Order preserving encryption[7][8]and multivalued Order preserving encryption MV-OPE [9] are used for performing computations over attributes that are used in the calculation of max and min aggregation functions, or attributes that are compared using relational operators. Those solutions are not practical in terms of time complexity and storage overhead. Besides, those encryption techniques and security protocoles are not sufficient to protect and process data in the cloud because such a solution is based on the trust between the owner and the provider. But, this constraint is not always true because the owner can disappear one day. That's why, researches in the literature have turned their attention to an alternative solution based on multi-cloud.

So, to maintain confidentiality, multi-cloud schemes such as DSky [10], inercloud [11], and NCloud [12] use symmetric encryption tools and distribute encrypted data over multiclouds. CloudStash [13] preserves confidentiality by applying a secret sharing scheme [14] directly on the file. It splits the file into shares and distributes

them over multiple clouds in parallel. The problem with this scheme is that it increases the storage overhead .

Several other works have used secret sharing to ensure the confidentiality and availability of data in the multi cloud. Authors in [15] propose dividing the secret in chunks of data to minimise the volume generated in the case of data warehouse. This is a practical solution to the data warehouse in terms of volume overhead. However, it creates high time complexity when decrypting the data. .

Information dispersal algorithm (IDA) [16] is also an encryption strategy that makes data available and secure. The idea of this algorithm is to distribute the data into insignificant shares like secret sharing .The advantage of this procedure is that the size of final data does not exceed (n/m) , with n is the total number of shares and m is the number of shares necessary for the reconstruction of the original data. Thus, with this algorithm the storage complexity is reduced, but the security is broke.

Authors in [17] try to highlight the advantage of the residue number system (RNS) and propose this schema HORNS based on the RNS system. In the same manner, as the works based on the secret sharing, HORNS proposes to divide the data in a small chunk with the modular arithmetic and stocks those residual numbers in a multi-cloud. This is an effective solution in terms of volume overhead and time complexity. Yet, this solution cannot be feasible in the case of collusion of cloud providers because the data will be decrypted immediately.

3 NEW SCHEMA FOR SECURING DATA WAREHOUSE IN THE CLOUD

3.1 Motivation

Our schema is based on the simple privacy homomorphism described in [1]. The privacy homomorphism will be illustrated as it is given in [1]:

Let p and q be two large secret primes and m= pq the product of such large secret primes. For that m is difficult to factor.

Consider the set of cleartext data $T = Z_m$, and the set of cleartext operation $F = \{+_m, -_m, \times_m\}$ consisting respectively of the addition, substraction and multiplication modulo m , with m=pq .

Let the ciphertext data set be $T' = Z_p \times Z_q$. Ciphertext operation F' is the component wise of these in F.

Define the encryption function $\phi(x) = [x \bmod p, x \bmod q]$. Given the two prime numbers p and q and the ciphertext $x_p = x \bmod p$ and the ciphertext $x_q = x \bmod q$, the secret x is decrypted using the Chinese remainder theorem (CRT).

One is motivated to use this privacy homomorphism because the latter is based on the modular arithmetic as it is described in the encryption function $\phi(x)$. This is very interesting in terms of volume overhead because the data will be divided in a small residue number so the storage space will be reduced. In terms of confidentiality, the

data will be encrypted with the two prime numbers p and q and will be computed in the range of m = pq. The decryption function of this schema is based on the use of Chinese remainder theorem. This technique is very practical and feasible because of its reasonable temporal complexity. Thanks to the homomorphic characteristic of encryption function, arithmetic operations can be done in a ciphertext in the cloud.

Hence, it is suggested that using this homomorphic privacy can be a promising solution for hosting data warehouse in the cloud.

3.2 Our proposition for hosting data warehouse in the cloud

The following section presents two scenarios for hosting data warehouse in the cloud.

Scenario 1

The simple scenario is to encrypt data stocked in the data warehouse with the encryption function $\phi(x) = [x \bmod p, x \bmod q]$. After that, the cipher text data $x_p = x \bmod p$ and the cipher text data $x_q = x \bmod q$ will be sent to the cloud provider with the modulo m. The two prime numbers p and q will be kept secret in the owner. Data stocked in the cloud will be processed modulo m. So, in this way, the cloud provider cannot decrypt the data with the modulo m because it is hard to factor. So, the data will be securely stocked in the cloud. Furthermore, with the homomorphic characteristic of modular arithmetic query, using arithmetic operation such that {+ ,-, x} will be done in the cloud in a cypher text without decryption. After processing the query in the cloud, the provider sends the result to the owner in a cyphertext. The owner decrypts their data with the two secret prime numbers p and q and the two chunks of encrypted data are received from the cloud using Chinese remainder theorem (CRT). Our proposition will be illustrated by introducing the two algorithms Alg-enc and Alg-dec:

```
Alg-enc
Input(x,  p , q)
Do
x_p =x mod p; x_q = x mod q;
Output (x_p  , x_q);
End;

Alg-dec
Input (x_p , x_q, p , q , m)
Do
  X = (x_p P_P b_P  +   x_q P_q b_q  ) mod m;
Output (x);
End;
```

With $P_p = m/p$; $P_q = m/q$; b_p is the multiplicative inverse of P_p modulo p end b_q is the multiplicative inverse of P_q modulo q. Those four constants can be precomput-

ed and stocked in the client and kept secret. This can reduce the temporal complexity of Alg-dec.

To ensure the processing of the query sent in the cloud, the owner will use the query rewriting.

As the data is encrypted with Alg-enc, the query sent from the client will be encrypted with Alg-enc and the order of the chunks will be kept.

To maintain the security of our model, a trust tier will be created; it can be the administrator of data warehouse or any trust tier in the company. The role of such tier is a middle tier between the client and the cloud. The owner will stock all their secret parameters at that trust tier. The role of the trust tier is to encrypt data with alg-enc and send it to the cloud and keeping the secret parameters p and q secret. This trust tier is also responsible for the decryption of data with Alg-dec after being received from the cloud using their secret parameters p and q. The important role of this trust tier is also rewriting the query when the client sends a query to the cloud. The trust tier rewrites the query as the data is encrypted in the cloud and sends the query to the cloud. After receiving the response from the cloud, the trust tier decrypts the result and sends it to the client. So, all the secret parameters of our method will securely be stocked in the trust tier and the client using the data warehouse has no information about how the data is encrypted and stocked in the cloud. In this way, our data warehouse will be more secure.

The figure below illustrates our first scenario for hosting and querying data warehouse in the cloud.

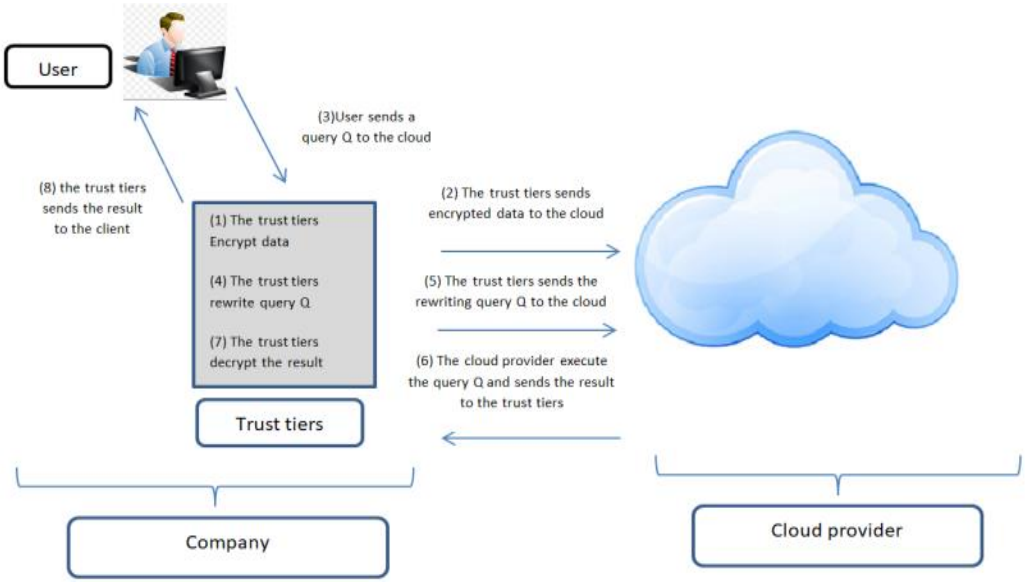

Fig. 1. First scenario for hosting and querying data warehouse in the cloud

Discussion

Our first proposed scenario for hosting data warehouse in the cloud is to encrypt all the data and stock them in one cloud provider. This solution seems satisfactory in terms of storage overhead and time complexity. Besides, the advantage of querying addition, subtraction and multiplication in a ciphertext is very important in the case of data warehouse because the nature of its OLAP query requires a massive volume of data.

Unfortunately, this schema can be broken by the cloud provider because it has the two chunks of data and the secret modulo m. It can infer the two chunks of data and get the two secret parameters p and q. Malicious intruders can also break the security parameters of the cloud provider, get the encrypted data and the modulo m from the cloud provider and decrypt it using the known cleartext attack as described in [18].

There are two factors that threaten the confidentiality of this schema: an internal factor being the cloud provider itself and an external factor being a malicious intruder. Consequently, a second scenario will be suggested which can reduce the risk of breaking the security parameters of our schema using a multi-cloud.

Scenario 2

Authors in [18] argue that this schema can be broken by a known plaintext attack. To illustrate, they present the way that cryptanalyst can infer the data and get the secret:

Suppose x is the integer that will be encrypted and presented by a pair (x_p, x_q), where $x_p = x \bmod p$ and $x_q = x \bmod q$. Assume that the cryptanalyst has the plaintext, ciphertext pair for some data. They supposes that p' be the gcd $\{ x_p - x$ for all data$\}$. In the same way, they supposes that q' be the gcd $\{ x_q - x$ for all data$\}$. After that it tests that p=p' and q= q', if this is the case, the cryptanalyst can decrypt all ciphertext. They proves that when specifically given ciphertext (x_p, x_q) ,the cryptanalyst can find x' such that $x' \equiv x_p \bmod p'$ and $x' \equiv x_q \bmod q'$.

So, it can be simply concluded that if the two chunks of data (x_p, x_q) are or one of them is kept secret from the cryptanalyst, the probability of inferring the data and breaking the system with the known plaintext attack will be reduced.

To this end, we propose to divide the two chunks of secret data and stock each chunk in a different cloud provider. As a result, this new model is based on two cloud providers; each of them stocks a chunk of secret data. In this way, we can reduce the probability of inferring data and breaking the encryption function because each provider has only one part of the chunk. So, the problem of intern risk will be decreased. Likewise, the risk of breaking the system from an extern intruder will be diminished because it is difficult for malicious users to break the security parameter of two cloud providers at the same time and gets the two chunks of secret data. So, with this new

sharing method, our model will be more secure in terms of confidentiality towards the cloud providers as well as external attack.

Also, when p, q, m = pq are very large integers, a small value x is very likely to have the same representation over Z_m, Z_p, and Z_q that is x mod m = x mod p = x mod q if x < min (p, q). This is an undesirable feature, because the homomorphic function $\phi(x)$ leaves the cleartext unencrypted (trivial ciphertext). To overcome this drawback we propose to multiply x with two secret value r_p and r_q such that $r_p < p$ and $r_q < q$.

Our new sharing model is based on two initial steps. The first step is data sharing process, and the second step is data reconstruction process.

Data sharing process:

. - The trust tier as mentioned in the first scenario will give the two secret prime numbers p and q and the two secret values r_p and r_q such that $r_p < p$ and $r_p < q$. Also, he calculate the modulo m= pq .

- He affects each secret prime number for a specific cloud provider. This is very important for maintaining the coherence of secret data.
- After that, he encrypts the secret data with the homomorphic function $\phi(x)$:

$$\phi(x) = [x \times r_p \bmod p,\ x \times r_q \bmod q] \tag{1}$$

and get the pair of data (x_p , x_q) with $x_p = x \times r_p \bmod p$ and $x_q = x \times r_q \bmod q$.

- The trust tier computes the signature of each chunk of data with the homomorphic function $H_s (A) = A \bmod B$ as:

$$sign\_x_p = H_s(x + x_p) \tag{2}$$

and

$$sign\_x_q = H_s(x + x_q) \tag{3}$$

- Finally, the trust tier sends each chunk of data with its signature to the cloud provider which corresponds, $(x_p , sign\_x_p)$ to CSP_p and $(x_q , sign\_x_q)$ to CSP_q .

The scenario of data sharing process is presented in figure 2:

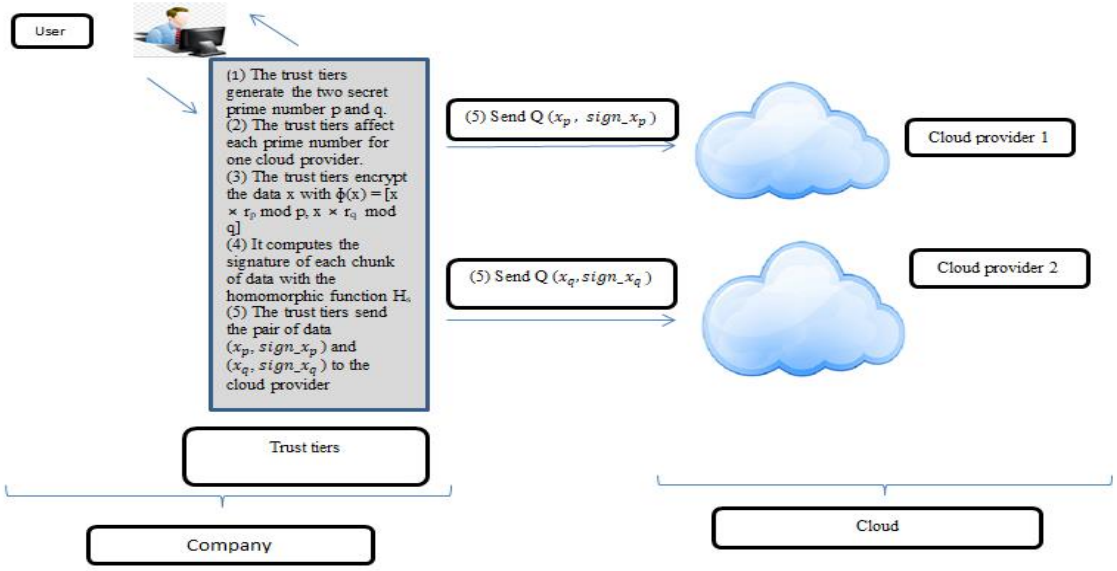

Fig. 2. Scenario of data sharing process

Data reconstruction process:

-The trust tiers ask each cloud provider to get the pair of data (x_p , sign_x_p) and (x_q , sign_x_q) .

-he compute the scalar product of (x_p , x_q) pair by (r_p^{-1} mod p, r_q^{-1} mod q) to retrieve (x mod p, x mod q) .

-After that, the trust tier decrypts the data using the Chinese remainder theorem with the two secrets parameters p and q and with the two chunks of data (x mod p, x mod q).

-Finally, he verifies the correctness of data with the two signatures. If sign_x_p = $H_s(x + x_p)$ and sign_x_q = $H_s(x + x_q)$ then data x is correct. In case of errors, the trust tier can ask CSP's to get a new pair's.

The scenario of data reconstruction process is presented in figure 3:

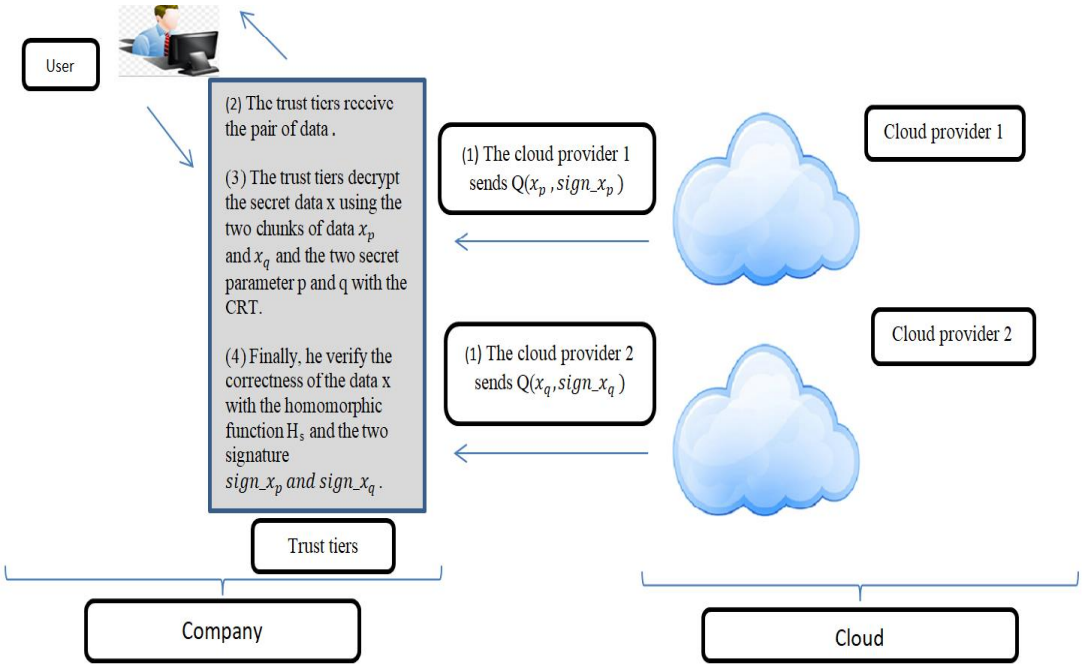

Fig. 3. Scenario of data reconstruction process

Example of sharing integer x= 17;
P= 5; q= 7; r_p =3 ; r_q =2; m=35;
ϕ (17) = (17 × 3 mod 5, 17 × 2 mod 7);
ϕ (17) = (1, 6); so the integer x= 17 is encrypted in the chunks of pairs (1; 6);
.

3.3 Querying data warehouse in the cloud

Our schema can directly support some basic OLAP operations at the CSP's through SQL operations and aggregation function. For example, simple select-from queries can be directly applied in the cloud. However, when expressing a condition in a where or having clause the trust tier must rewrite the query and post processing some operation in the company because the MOD operator is non- injective. Given that for X MOD Y = Z, the same output Z, considering Y a constant, can have an undetermined number of possibilities in X as an input which will generate the same value Z when applying the operator (e.g. 17 MOD 5=2, 22 MOD 5=2, 27 MOD 5= 2, etc).

For example, the query "SELECT ProdName FROM Product WHERE UnitPrice=17" would be transformed to the two queries "SELECT ProdName FROM Product WHERE UnitPrice=1" at CSP1, where 1 is the share of 17 at CSP1 and

SELECT ProdName FROM Product WHERE UnitPrice= 6" at CSP2, where 6 is the share of 17 at CSP2.

Or 1 can be also the result of $22 \times 3 \bmod 5 = 1$, $27 \times 3 \bmod 5 = 1$ etc.

So, the $CSP_{p=5}$ well return all the rows that correspond to 1 and that refer UnitPrice= 17, UnitPrice=22 and UnitPrice=27.

With the same manner, the $CSP_{q=7}$ well return all the rows that correspond to 6 and that refer UnitPrice= 17, UnitPrice=24 and UnitPrice=31.

After that, when the two queries are returned from the clouds, the trust tier must eliminate the erroning rows by a simple join between the two results of those queries before decrypting the final result.

This routine works for many comparison operators (=, ≠, EXISTS, IN, LIKE…) and their conjunction. Arithmetic operation and aggregation function such as sum, avg, count can be computed in ciphertext by the trust tier after eliminating the erroning rows.

But when ordering is necessary, as in ORDER BY clauses and many comparison operators (>, <, ≥, ≤, BETWEEN…), it can no longer apply since the original order is broken when sharing data. Thus, all fetched data must be decrypting and querying at the owner by the trust tier before to be sent to the client.

4 Security analysis and performance evaluation

4.1 Security analysis

Confidentiality of data

The confidentiality of data is our major focus in this paper. So, as described in section 3.2, the distribution of the two chunks of data in two cloud providers is a good solution for protecting data from plaintext attacks and from malicious cloud providers. The idea is to keep minimal information about data and parameters among the cloud provider.

So, the security parameter of our solution is based on the two parameters p and q and the two chunks of data x_p and x_q .

The role of trust tier as a middle tier between the user and the cloud is an effective solution that guarantees the confidentiality of the two secrets p and q.

The distributions of the two chunks of data are anonymous. Each cloud provider does not recognise if the chunk of data that is processes is about the parameter p or q.

Additionally, stocking each of the two chunks of data x_p , x_q in one cloud provider can reduce the risk of inferring the data and breaking the system.

Proof:

If the cloud provider predicts the p or gets the p (worst case), he can predict x as

$X= y \times p + x_p$ such that $x<m$;

The probability that the cloud can find x correctly is p/m . This probability is thus low and the cloud provider cannot also identify whether the chunks of data corre-

spond to the parameters p or q. This ambiguity can disrupt the work of inferring the data.

As a matter of fact, we can conclude that even if the two secret parameters will be discovered by the cloud provider, the latter cannot decrypt all the data directly because he doesn't have the second chunk of data. So, it is essential to do this operation on performing the system:

$X= y \times p + x_p$, $X= y \times q + x_p$ such that $x<m$;

This cannot be done in the case of a huge volume of data as in the data warehouse.

Similarly, it can be argued that the confidentiality of our schema is better with this new sharing strategy. In fact, even if the malicious intruder gets the two secret parameters p and q, it is hard for it to break the security of the two cloud providers and get the two chunks of secret at the same time.

Integrity of data

In our schema, the verifying phase is based on the correctness of two signatures. Hence, the risk of error does not exist. To reconstitute the integer x =17; the trust tier gets the two pairs of data from the CSP's (1,2) and (6,7) and we will suppose that there is a mistake when transferring the data from the cloud and the pair of data (1, 2) is transformed to (2, 2). The trust tier computes:

P_p =7 , P_q = 5 , b_p =3 , b_q =3 , x_p =4 , x_q = 3 ,m=35 ;

$r_p^{-1} \bmod p = 3^{-1} \bmod 5 = 2$;

$r_q^{-1} \bmod q = 2^{-1} \bmod 7 = 4$;

After that he compute: $(2 \times 2 \bmod 5, 6 \times 4 \bmod 7) = (4, 3)$;

Using the CRT he can compute:

$X= 129 \bmod 35$; so X = 24.

After that, the correctness of data is verified as:

$sign\_x_p = (24 + 2) \bmod 8 = 2$ and $sign\_x_q = (24 + 6) \bmod 8 = 6$;

So, the data x =24 is not correct because the signature $sign\_x_q$ is not correct.

Accordingly, since our solution is based on two verifying phases, this homomorphic function H_s is very secure and it does not reveal any information about the secret data x .

4.2 Performance evaluation

Volume overhead

In our solution there is no volume over head when encrypting initial data because our encryption function is based in the MOD operator that divides the data in a smile residue number.

Or this operation is done two times. For that, the volume of data in cyphertext cannot exceed twice the volume of original data.

Temporal complexity

Encryption, decryption, and homomorphic operations only need one/two modular operations in our schemas.

For the encryption phase, we need just O (n) operation; the decryption phase is based on the CRT. So we need just O (lg lg n) operation for decrypting phase.

Comparison of our schema to existing related approaches

In this section, we compare our schema with approaches presented in our state of the art with respect to security and performance. Table 1 synthesizes the features of all approaches discussed above.

Features	[13]	[15]	Our schema
Confidentiality	yes	yes	yes
Integrity	no	yes	yes
Range query	no	no	no
Aggregation function	yes	yes	yes
OLAP query	yes	yes	yes
Data volume	6n	3n	2n
Temporal complexity of encryption phase	O(n)	O(n)	O(n)
Temporal complexity of decryption phase	$O(n\lg^2 n)$	$O(nt^2)$	O(lg lg n)

5 Conclusion

This paper presents an original approach to share DW in the cloud that simultaneously supports data privacy and OLAP query with reasonable time complexity and minimum volume overhead. Our proposed solution is based on a homomorphic encryption algorithm that reveals a serious weakness as it can be deciphered by ciphertext attacks. For that, we propose a new method of using this homomorphic privacy based on multi cloud providers and perturbation values. With this new sharing schema, we can reduce the risk of breaking the security of the system by both cloud providers and malicious intruders. Also, a new technique is proposed to verify the integrity of our data when received from the cloud.The weakness of our schema lies in the processing of range queries in the owner after decrypting all the data. This operation can take a lot of time in the case of data warehouse because of the huge volume of data that will be decrypted before processing range query. That is why; we attempt to propose a solution to this situation in order to reduce time consumption in the decryption phase

in future work. We eventually endeavour to evaluate our schema in a real cloud provider.

References

1. Rivest, L.L., Adleman, L., Dertouzos, L.M. : On data banks and privacy homomorphisms . Massachusetts Institute of Technology Cambridge, Massachusett (1978)
2. Kadhen, H., Amagasa, T., Kitagawa, H. : A novel framework for database security based on mixed cryptography. ICIW ,Venise Italy163-170 2009.
3. Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H. : CryptDB :Processing queries on an encrypted database ".ACM 103-111 (2012).
4. Schneier, B. : Description of a new variable-length key,64-bit block cipher (blowfish).Fast software encryption ,Cambridge security workshop, London,UK page 191-204 (1993).
5. Liu, D. : Securing outsourced databases in the cloud .In security, privacy and trust in cloud systems, Springer,Heidelberg age 259-282 (2014).
6. Wang, P., Ravishankar, C.V. : Secure and efficient range queries on outsourced databases using Rp-Trees. 29th IEEE international conference on data engineering,Brisbane,Australia ,april 8-11,(2013).
7. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y. : Order preserving encryption for numeric data .Sigmod 2004 june 13-18 (2004)aris france.
8. Hore, B., Mehrotra, S., Canim, M., Kantarcioglu, M. : Secure multidimensional range queries over outsourced data . The VLDB Journal pages 333-358 (2012).
9. Kathen, H., Amagasa, T., Kitagawa, H. : MV-OPES : Multivalued-order preserving encryption schemes :A novel scheme for encrypting integer value to many different values . IEIC Trans(2010).
10. Bessami, A., Correia, M., Quaresma, B., André, F., Sousa, P. : DepSky :dependable and secure storage in the cloud –of-clouds .In processings of the sixth conference on coputer systems.ACM,(2011),31-46.
11. Caclin, C., Haas, R., Vukolic, M. : Dependable storage in the intercloud .IBM rechearch , vol.3783,1-6 (2010).
12. Alsolami, F., Chow, C.E. : N-cloud: improving performance and security in cloud storage . In high performance switching and routing (HPSR), (2013) IEEE.
13. Alsolai, F., Boult, T. : Cloud Stash: using secret sharing scheme to secure data, not keys, in multi-clouds. 11th international conference on information technology: new generations (2014).
14. Adi, S. : how to share a secret . Communication of the ACM novembre (1979),volume 22.
15. Varunya, A., Harbi, N., Darmont, J. : A novel multi secret sharing approach for secure data warehousing and On-Line analysis processing in the cloud .IGI (2015).
16. Rabin, M.O. : Efficient dispersal of information for security load balancing, and fault tolerance . Published by ACM (1989).
17. Mahadewan, A.G., Kamesh, T. : Horns : A homomorphic encryption schema for cloud computing using residue number system. IEEE , 45th Annual Conference on Information Sciences and Systems (2011).
18. Ervest, F.B., Yacov, Y. : on privacy homomorphism's .advances in cryptology eurocrypt,(1987).