ISSN 1870-4069

SecuredDW: A Homomorphic Schema to Securely Hosting Data Warehouse in the Cloud

Kawthar Karkouda, Ahlem Nabli, Faiez Gargouri

Sfax University, Miracl Laboratory, Tunisia

Abstract. Currently, cloud computing has become the most popular technologies in the area of IT enterprise. It has many advantages such as computing power, storage, network and software as a service. Moreover, many other benefits have made it attractive. In fact, it is easy to deploy, its technical infrastructure is adaptable to the volume of business activity and its cost is relative to consumption. Whereas building a data warehouse typically necessitates an important initial investment, with the cloud pay-as-you-go paradigm, BI system can benefit from this new technology. However, cloud computing brings its own risks in terms of security. For this purpose, before outsourcing sensitive data to the cloud, the owner must encrypt his data to keep secure. In the particular context of cloud data warehouse, privacy is of critical importance because it contains sensitive data. Cloud provider proposes traditional security solutions to ensure the confidentiality of outsourced data. Unfortunately, those solutions are not practical in the case of data warehouse anymore because they induce a heavy overhead in terms of data storage and query performance. So, a new solution must be proposed for outsourcing data warehouse to the cloud that respects its specification and swings performance and security. In this paper, we propose (SecuredDW) a new sharing schema for securing and querying a data warehouse hosted in the cloud based in a homomorphic algorithm. The integrity of data is also addressed in this paper by proposing two new signatures to verify the correctness of data sent and received from the cloud. Theoretical results show the efficiency of Secured DW in terms of privacy and performance with respect to other solutions.

Keywords: cloud computing, data warehouse, Security, integrity.

1 Introduction

With the booming of cloud computing, people are encouraged to adapt BI system in their companies. Such new delivery model can mitigate the cost of deployment of a data warehouse thanks to its "pay as you go" paradigm. So, company pays just the used resource. It is true that the most attractive advantage of using the cloud is its profitable cost, but also there are many more. Indeed, it is easy to deploy and its technical infrastructure is adaptable to the volume of business activity. The only drawback that prevents the move to the cloud is security. In fact, there are several security issues related to cloud computing. Some of those issues emanated from the traditional

architecture such as network attack, confidentiality of data, availability, authentication and vulnerability exploitation. Because cloud computing evolves rapidly and the push to effective controls to protect data in the cloud is nascent, many security solutions for clouds are presented in the literature. The most used solution is the encryption of data before sending it to the cloud with the symmetric and the asymmetric algorithms. Homomorphic encryption is also used to secure data hosted in the cloud.

In the context of data warehouse, these security problems become tougher to resolve because the high volume of data stocked in the warehouse and because the nature of OLAP query. More precisely, encrypting data warehouse can affect the cost of using the cloud especially in the case of homomorphic encryption that produces a very high volume overhead. Furthermore, symmetric and asymmetric algorithm cannot be a suitable solution for data warehouse because the decryption of data in the cloud can affect the performance of OLAP query and the cost of using the cloud. More than that, such scenario is based on the trust between the owner and the cloud provider, which is not the case.

For this reason, in this paper, we propose SecuredDW as a new sharing schema adapted to the nature of data warehouse. Our proposal 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 by using data splitting, multi-cloud and perturbation value.

In addition to that, in this work, we enforce data integrity by providing two signatures to verify the correctness of data.

Moreover, when data is encrypted the original order of data is broken. Thus, all fetched data must be decrypted and querying at the owner by the trust tier before to be sent to the client. This operation can affect the performance of range query and some others query when ordering is necessary. For that we will propose a weighted method that reduces the time complexity of such kind of query.

According to my knowledge, this work is one of a few work that provide an environment that takes into account the specifications of a data warehouse while balancing performance and security. It should be noted that it is not our aim to propose a solution as secure as the state-of-the-art encryption algorithms. We rather suggest a technique that provides a considerable level of overall security strength with respect to some performance overheads.

The rest of this article is organized as follows. The second section introduces and discusses the previous research related to our proposal. Then, we present SecuredDW as a new homomorphic schema to securely host data warehouse in the cloud. In the fourth part, we deal with the theoretical and performing results. Finally, the paper ends with a conclusion.

2 Related Works

As encryption is the most used solution to secure data outsourcing to the cloud, we will start by introducing some traditional encryption algorithms that are used in the context

of the cloud. In fact, symmetric encryption is mainly the use of the same key in encryption and decryption. That is to say that he who encrypts the data must share the key with the receiver who decrypts the data. The two most important modern symmetric algorithms are the data encryption standard DES [2] and the advanced encryption standard AES [3]. In opposite to symmetric encryption, asymmetric encryption is by definition the use of two different keys for encryption and decryption. RSA [4], Rabin [5] and ElGamel [6] are the most practical asymmetric algorithms. However, the main threat of using those algorithms is that they are based on trust between the owner and the cloud provider, which is not the case because the cloud provider may not be trustworthy and can fetch into the sensitive data. Although the data and the keys are stocked in the same cloud provider, this scenario can make data subject to external attacks. So, if intruders break the security system of the provider, they can steal the data with the keys and decrypt it easily. The inefficiency of those security techniques is not only in terms of privacy but also in terms of performance. Indeed, decrypting data before processing query is not practical mostly in the case of data warehouse.

To overcome those problems, many works in literature propose running data in ciphertext in the cloud. In this context, homomorphic encryption is presented. Authors in [7,8,9,10,11,12] propose a solution based on a fully homomorphic encryption. The advantage of those algorithms is that they allow addition and multiplication of encrypted data in ciphertext. However, they suffer from high time complexity and high volume overhead.

To perform computations over attributes that are used in the calculation of max and min aggregation functions or attributes that are compared using relational operators, order preserving encryption [13,14] and multivalued Order preserving encryption MV-OPE [15] are proposed.

To apply the power of running data in ciphertext in the cloud, authors in [16] propose to encrypt data warehouse with several encryption techniques depending on the type of attributes. This way, analytical queries can be processed in ciphertext. Yet, this solution suffers from high time complexity of running query and high volume overhead.

The availability of data is also a challenge when entered in the cloud. For this reason, the cloud provider replicates data to ensure its availability. Another solution presented consists in using erasing codes. The advantage of such solution is that it can reduce the volume of replicated data. Facebook, for example, is using this algorithm to ensure the availability of its data warehouse with minimum volume overhead [17].

Multi clouds, cloud of cloud, or inter cloud, are by definition the use of many cloud providers for data storage such as DSky [18], inercloud [19], and NCloud [20]. Authors in [21] use erasure coding to divide the data and stock it in different cloud providers to ensure its availability and to reduce the volume of data. Their approach seems to be good in term of availability and in term of reduction of data volume, but it is not secure.

Authors in [22] present CHARM a multi cloud schema that guarantees the availability of data with minimum cost.

The secret sharing algorithm, when first presented in [23], is very useful in the cloud to ensure confidentiality and availability as in works [24,25,26,27]. The problem with using the secret sharing is the high volume overhead generated after encryption. That's why, authors in [28] try to solve this problem by proposing a new model for sharing

data warehouse inspired from secret sharing. The idea is to split the data into block before encrypting it with a random linear equation. However, this approach suffers from high time complexity of decryption steps. Another problem arises when using this approach is that it cannot resist to collusion attacks.

To work out such a problem, authors in [29] propose S4 as a new schema based on secret sharing for enforcing privacy in Cloud data warehouse. The idea is to store secrets at one single CSP instead of sharing secrets to n CSP's. The privacy in S4 relies on the fact that k-1 splits are stocked in the CSP and the K^{th} splits necessary for reconstructing the secret are stocked in the owner. This way, they can avoid the problem of collusion, but the processing of the query cannot be done totally in the cloud.

Authors in [30] propose HORNS as a sharing schema based on the Residue Number System (RNS) and multi-cloud. The idea in HORNS is to divide the data in small chunks with the RNS and stock those chunks in a multi-cloud. In this concern, time complexity of encryption and decryption steps is reduced. But this scheme suffers from redundant data and collusion attacks.

2.1 Discussion

As presented in the last section, many works try to solve the problem of security when using the cloud. Nowadays, the most appropriate solution is to use symmetric and asymmetric encryption algorithm to encrypt data before sending it to the cloud. Those solutions are based on the trust between the user and the provider. For this reason, they are not secure enough because the provider can fetch into sensitive data. That's why, processing data in ciphertext is improved and new solutions based on homomorphic encryption algorithm, secret sharing, Information Dispersal Algorithm IDA are proposed in the case of database in general and in the case of data warehouses in particular. The problem with those solutions is that they are not practical enough, mainly in the case of data warehouse because it stocked a high volume of data and because of the time complexity of an OLAP query. For example, many homomorphic encryption algorithms are proposed in the literature as described in the previous section. But those algorithms are not a good solution for outsourcing data in the cloud because of the high time complexity of processing data and because of the volume overhead generated after encryption data. So, the famous homomorphic encryption algorithms existing in the literature cannot meet the need for a heavy computing application like the data warehouse. Some authors propose to use the secret sharing for outsourcing data to the cloud. Their choice is based on secret sharing since it has an acceptable time complexity comparing with the homomorphic encryption algorithm. But the problem with adopting this technique in data warehouse is that it generates a high volume overhead. Accordingly, authors in [25] propose a new secret sharing that reduces the volume overhead generated when encrypting data. But this solution suffers from the high time complexity when decrypting data. Information dispersal algorithm (IDA) is also proposed as a solution to outsource data in the cloud. It is known for its low time complexity of encrypt and decrypt data and its low volume overhead. However, this algorithm suffers from its weak security. Therefore, in this paper, we propose a new schema that balances security and performance when outsourcing data warehouse in

the cloud. Our schema is based on the simple privacy homomorphism as described in [1]. The privacy homomorphism will be illustrated as 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 module 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).

We are 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 with regards to volume overhead. In fact, because the data will be divided in a small residue number, the storage space will be reduced. But when it comes to 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.

So, 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 module m. The two prime numbers p and q will be kept secret in the owner. The 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. Then, 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 cipher text without decryption. After processing the query in the cloud, the provider sends the result to the owner in a ciphertext. The owner decrypts his 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).

Unfortunately, this schema can be broken by the cloud provider because it has the two chunks of data and the secret module 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. Thus, a new way will be suggested in the second section which can reduce the risk of breaking the security parameters of our schema using a multi-cloud and perturbed data.

3 SecuredDW: A New Schema for Securing and Querying Data Warehouse Hosted in the Cloud

This section presents SecuredDW as a new homomorphic schema for hosting and querying data warehouse in the cloud securely. In this schema, we focus on ensuring the three levels of security CIA (confidentiality, integrity and availability) as it is described in figure 1.

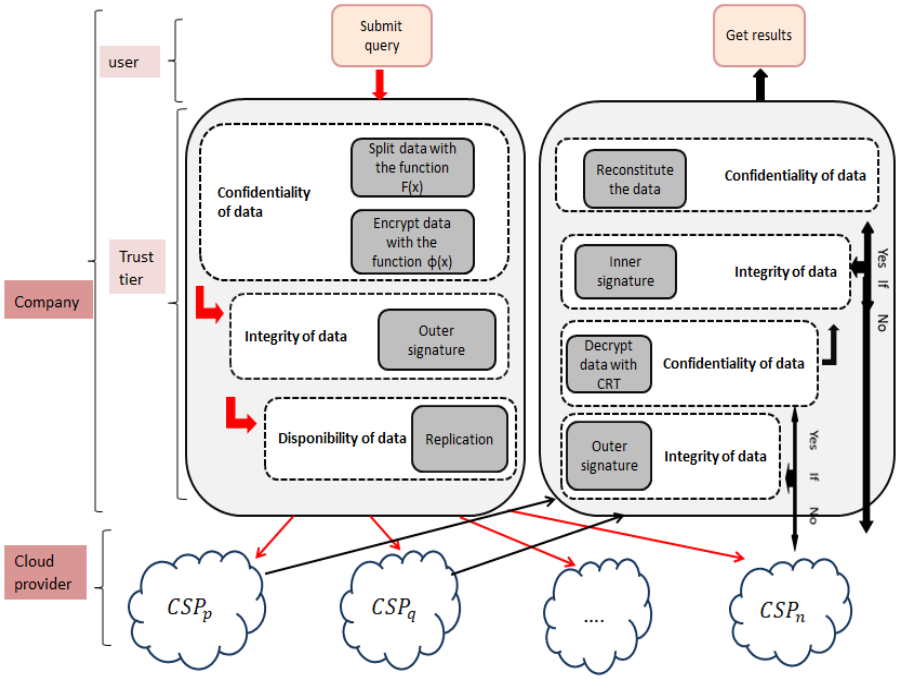

Fig. 1. Global Architecture of SecuredDW.

3.1 Enforcing the Confidentiality of Data

Authors in [28] describe the way how a cryptanalyst can infer the data and get the secret. They argue that this homomorphic privacy presented in [1] can be broken by a known plaintext attack. They illustrate the process of breaking the system as follows:

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 suppose that p' be the gcd$\{x_p - x$ for all data$\}$. In the same way, they suppose that q' be the gcd$\{x_q - x$ for all data$\}$. After that, it tests that $p=p'$ and $q= q'$and if this is the case, the cryptanalyst can decrypt all ciphertext. They prove that when ciphertext (x_p ,x_q) is specifically given, the cryptanalyst can find x' such as, $x' \equiv x_p \bmod p'$ and $x' \equiv x_q \bmod q'$.

So, it is clear that if the two chunks of data (x_p ,x_q) 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.

Therefore, as a first modification, we propose to split the secret data into a k small chunk with a random function F(x) like this:

$$F(x) = \sum_{i=1}^{k} x_i \quad \text{with} \quad x_i \in Z_m. \tag{1}$$

After splitting the original data with the random function F(x), we encrypt each chunk of data with the homomorphic function ϕ(x) :

$$\phi(x) = [\, x \bmod p \,,\, x \bmod q \,].$$

Splitting data and encrypting each chunk of data separately with the homomorphic function ϕ(x) is not enough to secure sensitive data. To achieve this, we propose to stock each share of secret data in a different cloud provider. In this way, we can reduce the probability of inferring data and breaking the encryption function because each provider has only one chunk of the secret data. So, the problem of intern risk will be decreased. Likewise, it is difficult for malicious users to break the security parameter of two cloud providers at the same time and get the chunks of secret data necessary to reconstruct the original data. Hence, the risk of breaking the system by an external intruder will be diminished. This way, our model will be more secure in terms of confidentiality towards the cloud providers as well as from external attack.

Besides, 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 $x \bmod m = x \bmod p = x \bmod q$ 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 secret data with two secret values r_p and r_q such that $r_p < p$ and $r_q < q$.

So our new homomorphic encryption function will be:

$$\begin{aligned} \phi(x) = (\, & [x_1 \times r_p \bmod p,\, x_1 \times r_q \bmod q\,], \\ & [x_2 \times r_p \bmod p,\, x_2 \times r_q \bmod q\,], \quad \text{with} \quad k \geq 2 \\ & \ldots\ldots \\ & [x_k \times r_p \bmod p,\, x_k \times r_q \bmod q\,]\,). \end{aligned} \tag{2}$$

After encrypting data with the homomorphic function ϕ(x), k pair of data will be produced (x_{kp}, x_{kq}) with $x_{kp} = x_k \times r_p \bmod p$ and $x_{kq} = x_k \times r_q \bmod q$. After that, the chunks of data $(x_{1p}, x_{2p,\ldots} x_{kp})$ will be sent in the CSP_p and $(x_{1q}, x_{2q,\ldots} x_{kq})$ will be sent in the CSP_q.

3.2 New Homomorphic Integrity Function

To ensure the integrity of data, we introduce in this paper new signatures named outer signature to verify the integrity of share and inner signature to verify the integrity of original data.

Outer signature

Outer signature is a new homomorphic function based on the modular approach.

For $c = x \bmod n$ is equivalent to $x-k= n*c$, k is the rest of division.

So the signature of each share designed " Sgn_x" will be computed with the homomorphic function as:

$$H_s(x) = x \bmod n = \text{Sgn\_x}, \tag{3}$$

and we will compute the key of each signature as:

$$key\_x = x - (Sgn\_x \times n). \tag{4}$$

In our case, to verify the integrity of our chunks of data, we will use the two equations (3) and (4). After that, each pair of data will be sent to a CSP.

In the cloud, each provider will verify the correctness of its share by computing:

$$Sgn\_x = share \bmod n,$$

and it verifies that the key received is:

$$key\_x = share - (Sgn\_x \times n).$$

Similarly, when the owner receives data from the CSP's, he can verify his data with the two equations (3) and (4).

The main advantage of our integrity function is in its homomorphic characteristic. This is very useful in the case of data warehouse when using the aggregation function.

Inner signature

To enhance the integrity of our schema we propose an Inner signature. The latter is a self-checked signature based on the homomorphic propriety of modular approach. It works by computing the equivalence between the chunks of data generated after splitting the original data with the random function F(x) and the share of data received from the cloud. Algorithm 1 is used to verify the integrity of the original data in the owner after decrypting the share of data x_{1p}, x_{1q}, x_{2p}, x_{2q},... x_{kp} ,$x_{kq}$with the CRT and getting the original chunk of data x_1, x_2,… x_k.

Algorithm 1

Inner signature $(x_1, x_2, x_k, x_{1p}, x_{2p}, x_{kp}, x_{1q}, x_{2q}, x_{kq})$

{

$S_p = |x_{1p} + x_{2p} + \ldots + x_{kp}|_p$

$S_q = |x_{1q} + x_{2q} + \ldots + x_{kq}|_q$

$Chunk_p = |x_1 + x_2 + \ldots + x_k|_p$

$Chunk_q = |x_1 + x_2 + \ldots + x_k|_q$

```
If ((S_p=Chunk_p) and (S_q=Chunk_q ))
 Than
     Write ("Data is correct ")
 Else
     Write ("Data is not correct ")
}
```

After computing the inner signature, whether it is correct that the trust tier reconstitutes the original data X with the function F(x), or else it asks the cloud provider to get other share of data.

3.3 Data Availability

In this section, we want to show the robustness of our solution if we use data replication as solution to ensure the availability of data. In this concern, we propose to replicate each chunk of data three times. So, nine chunks of data will be produced if k =3 is the number of data splitting. After that, the chunks of data (x_{1p},$x_{2p,...}x_{kp}$) will be sent to the CSP_p, (x_{1q},$x_{2q,...}x_{kq}$) will be sent in the CSP_q and the other replicated chunks will be sent to the two other clouds . This way, we can guarantee the availability of data if one or two cloud providers are not available. Moreover, we can eliminate the dependency of a single cloud provider.

With our method, data overhead cannot exceed the volume of original data warehouse twice. So, if we replicate the entire data warehouse three times, the volume overhead generated will be six times the volume of the original data warehouse. This is not very practical in the case of data warehouse, but it does not exceed the volume overhead generated with the other solutions.

4 Sharing Data Warehouse

In this section we will demonstrate how data warehouse will be shared in the cloud with our schema. Our new sharing model is based on two initial steps. The first step is a data sharing process, and the second step is a data reconstruction process. We will also delineate how the query will be processed and finally we will present a new method to process the range query in ciphertext.

4.1 Data Sharing Process

The data sharing process describes how original data will be processed before being sent to the clouds. It consists of the following steps:

- The trust tier person who is responsible for data security in the company proposes two secret prime numbers p and q and two secret values r_p and r_q, such that $r_p < p$ and $r_q < q$. He also calculates the module $m = pq$.

- He chooses the parameter k which is the number of chunks of data generated after splitting the original data.

- He affects each prime number p and q to a specific cloud provider. This is very important for maintaining the coherence of secret data.

- He splits the original data with the random function F(x) presented in equation (1).

- After that, he encrypts the data with the homomorphic function $\phi(x)$ presented in equation (2).

- The trust tier computes the signature of each chunk of data generated after encryption with the new homomorphic integrity function Hs presented in equation (3) and the keys of each signature with equation (4).

- Finally, the trust tier replicates all the chunk of data with its signature and its key and sends them to the corresponding cloud provider.

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

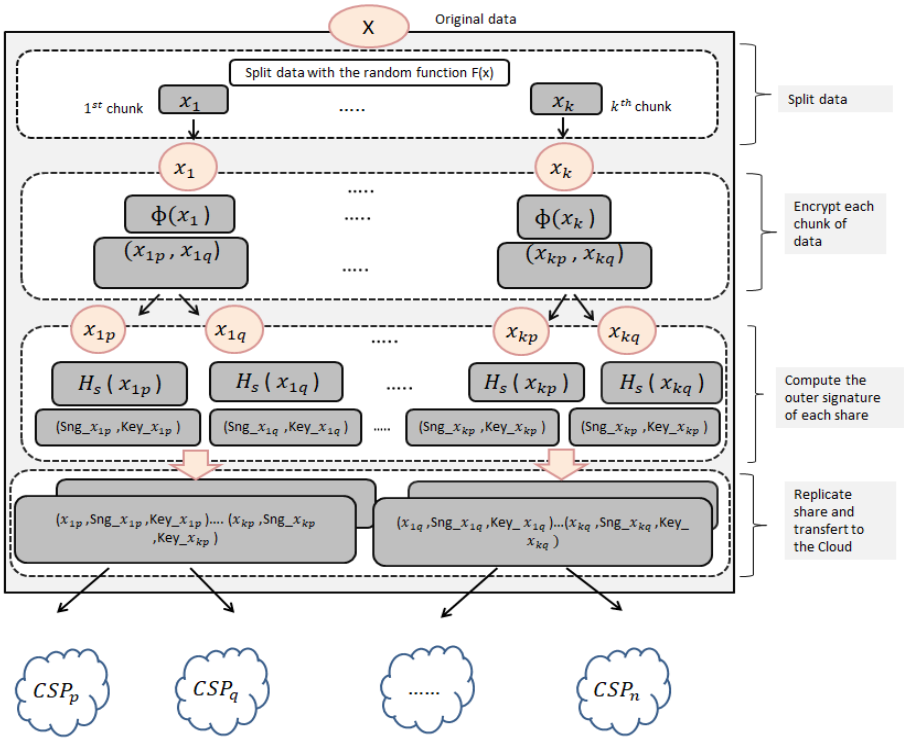

Fig. 2. Data sharing process.

4.2 Data Reconstruction Process

The data reconstruction process describes how original data will be reconstituted after being received from the clouds. It consists of the following steps:

-The trust tier asks two cloud providers CSP_p and CSP_q to get the shares of corresponding data X (x_{1p}, Sgn_x_{1p}, key_x_{1p}) ,(x_{2p} , Sgn_x_{2p} , key_x_{2p}) and(x_{kp}, Sgn_x_{kp},

key_x$_{kp}$) from CSP$_P$, (x_{1q} , *Sgn_x*$_{1q}$, *key_x*$_{1q}$) , (x_{2q} , *Sgn_x*$_{2q}$, *key_x*$_{2q}$) and (x_{kq}, *Sgn_x*$_{kq}$, *key_x*$_{kq}$) from CSP$_q$.

- He verifies the correctness of each share with the signature and the key. In case of errors, the trust tier can ask CSP to get a new share.

-He computes the scalar product of each share (x_{1p} , x_{1q}) , (x_{2p} , x_{2q}) and (x_{kp}, x_{kq}) by (r_p^{-1} *mod p*, r_q^{-1} *mod q*) to retrieve (x_1 *mod p*, x_1 *mod q*), (x_2 *mod p*, x_2 *mod q*) and (x_k *mod p*, x_k *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 shares of data (x_1 *mod p*, x_1 *mod q*), (x_2 *mod p*, x_2 *mod q*) and (x_k *mod p*, x_k*mod q*) .

- He verifies the integrity of the original data with the inner signature.

-If the inner signature is correct, the trust tier computes the original data with the function F^{-1}(x). Otherwise, he asks the cloud provider to get another share of data.

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

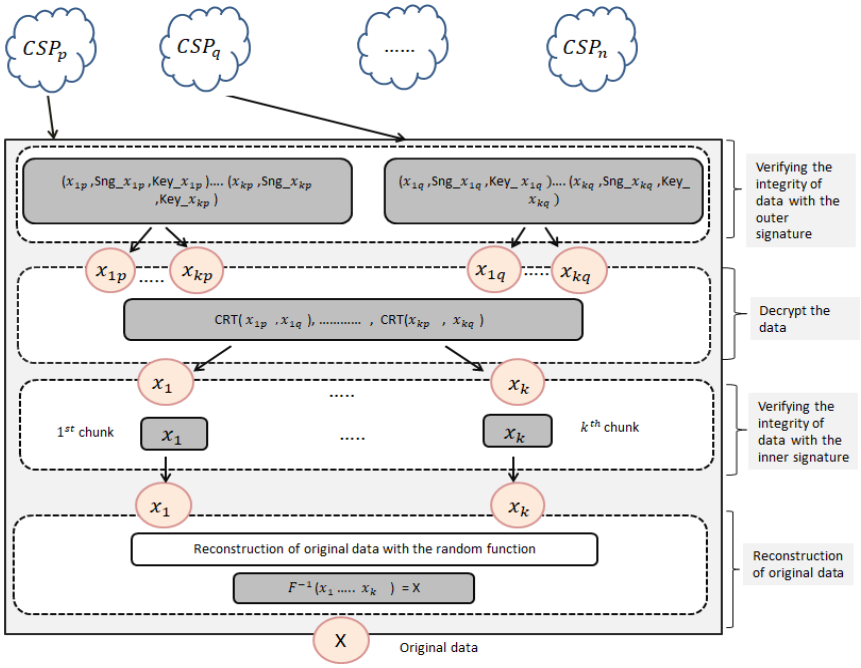

Fig. 3. Data reconstruction process.

To illustrate our schema, we present the example of data sharing and reconstruction process of the integer x= 17,

k = 3,

F(17) = 5+ 8 + 4 ,

P= 5, q= 7, r_p =3, r_q=2, m=35 , n = 2,

ϕ (5) = (5× 3 mod 5, 5× 2 mod 7),

ϕ (8) = (8 × 3 mod 5, 8 × 2 mod 7),
ϕ (4) = (4 × 3 mod 5, 4 × 2 mod 7),
ϕ (5) = (0, 3), ϕ (8) = (4, 2), ϕ (4) = (2, 1),

Compute the signature of each chunk of data as:

sign_x_{1p} = $H_s(0)$,sign_x_{1q} = $H_s(3)$,sign_x_{2p} = $H_s(4)$,sign_x_{2q} = $H_s(2)$,sign_x_{3p} = $H_s(2)$ and sign_x_{3q} = $H_s(1)$
sign_x_{1p} = $H_s(0)$ = 0 mod 2 = 0 and the key is key_x_{1p} = 0- (2* 0) = 0 ,
sign_x_{1q} = $H_s(3)$ = 3 mod 2 = 1 and the key is key_x_{1q} = 3 - (2* 1) = 1 ,
sign_x_{2p} = $H_s(4)$ = 4 mod 2 =2 and the key is key_x_{2p} = 4 - (2* 2) = 0 ,
sign_x_{2q} = $H_s(2)$ = 2 mod 2 =1 and the key is key_x_{2q} = 2- (2* 0) = 0 ,
sign_x_{3p} = $H_s(2)$ = 2 mod 2 =1 and the key is key_x_{3p} = 2- (2* 1) = 0 ,
sign_x_{3q} = $H_s(1)$ = 1 mod 2 = 0 and the key is key_x_{3q} = 1- (2* 0) = 1 ,

After that all data will be sent to the cloud provider.

CSP_p is identified with the secret parameter p and CSPq is identified with the secret parameter q.

For reconstruction the integer x =17,
The trust tier gets all data from the CSP's and verifies the correctness of data:
He verifies that
sign_x_{1p}= 0mod2 = 0 and thatkey_x_{1p} = 0 – (0×2) = 0
sign_x_{1q}= 3 mod 2 = 1 and that key_x_{1p} = 3 – (1×2) = 1
sign_x_{2p}= 4 mod 2 = 2 and that key_x_{1p} = 4 – (2×2) = 0
sign_x_{2q}=2 mod 2 = 1 and that key_x_{2q} = 2 – (1×2) = 0
sign_x_{3p}=2 mod 2 = 1 and that key_x_{3p} = 2 – (1×2) = 0
sign_x_{3q}=1mod 2 = 0 and that key_x_{3q} = 1 – (0×2) = 1

If it is the case, he computes:
P_p =7 , P_q= 5 ,b_p=3 ,b_q=3,x_{1p} = 0, x_{1q} = 3 ,x_{2p} = 4,x_{2q} = 2 ,x_{3p} = 2 ,x_{3q} = 1 , m=35
r_p^{-1} mod p = 3^{-1} mod 5 = 2,
r_q^{-1} mod q = 2^{-1} mod 7 = 4 ,

After that we compute: (0× 2 mod 5, 3×4 mod 7) = (0, 5) ,(4× 2 mod 5, 2×4 mod 7) = (3, 1) ,(2× 2 mod 5, 1×4 mod 7) = (4, 4).

Using the CRT we can compute:

x_1= CRT (0 , 5) = (0×7×3 + 5×5×3) mod 35 = 5
x_2= CRT(3 , 1) = (3 ×7×3 + 1×5×3) mod 35 = 8
x_3= CRT (4 , 4) = (4×7×3 + 4×5×3) mod 35 = 4

to verify the correctness of each chunk of data, we compute:

$S_p = |0 + 3 + 4|_5 = 2$
$S_q = |5 + 1 + 4|_7 = 3$
$Chunk_p = |5 + 8 + 4|_5 = 2$
$Chunk_q = |5 + 4 + 8|_7 = 3$
$S_p = Chunk_p$ and $S_q = Chunk_q$, so the chunks of data is correct.

After that, we compute the original data X = 5 + 4+ 8 = 17.

4.3 Sharing Data Warehouse

The whole table of a shared data warehouse is stored in a relational database at a given CSP's (two initial CSP's in our case) and each attribute value in each record is encrypted independently as described in the data sharing process except the primary keys and the foreign keys. Figure 4 gives an example of star schema data warehouse that is shared among two CSP's. Each shared model of data warehouse stands for the same schema as the original data warehouse, except that two other attributes are added to store signatures and keys.

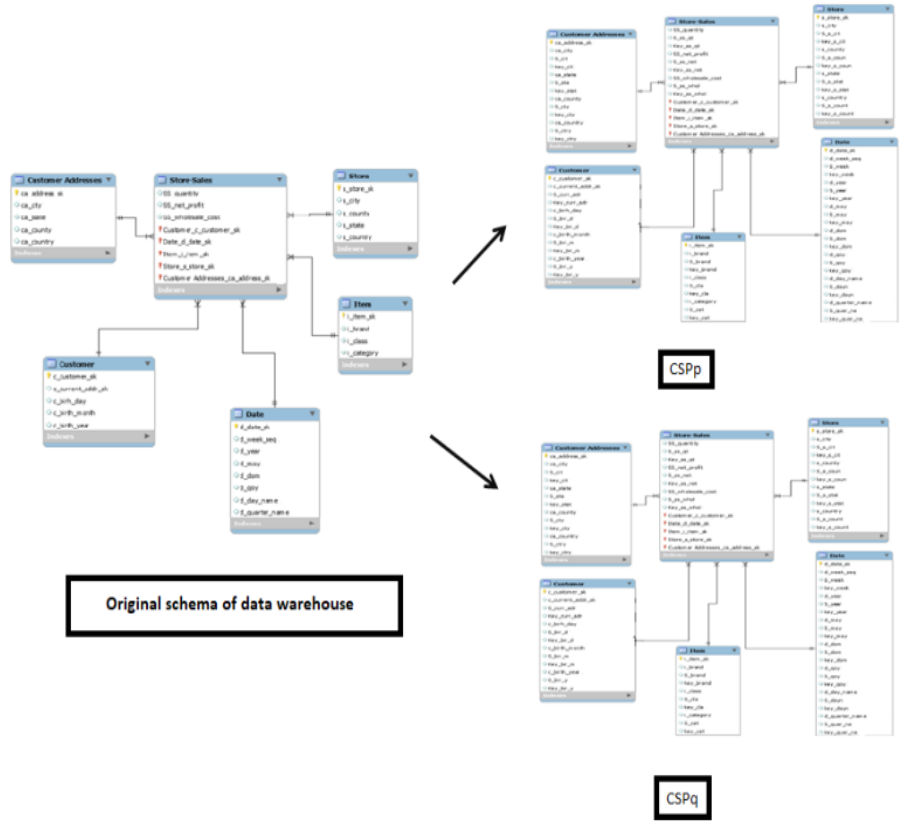

Fig. 4. Sharing schema of data warehouse in each CSP.

4.4 Querying Data Warehouse Hosted 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. Figure 5 describes the scenario of processing a select query.

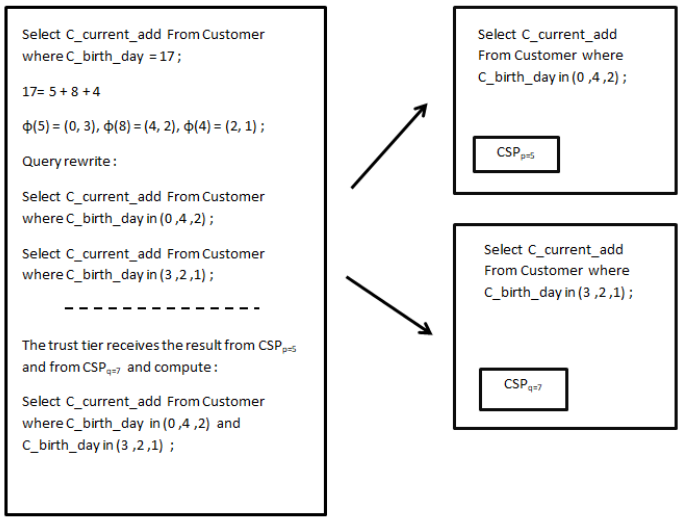

Fig. 5. Scenario of processing Select query.

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 error rows.

4.5 Weighted Method for Answering Range Query

When ordering is necessary, as in ORDER BY clauses and many comparison operators (>, <, ≥, ≤, BETWEEN…), it can no longer be applied since the original order is broken when sharing data. Thus, all fetched data must be decrypted and quered at the owner by the trust tier before being sent to the client. This operation is very expensive in terms of time computation. To infer the performance of such kind of query, we propose a weighted method that reduces the time complexity of running range query.

We will also propose two weights, the first one is to encrypt data with modulo p, while the second one is to encrypt data with modulo q, so:

$$Wp = \frac{|\ P^{-1}|_p}{p}, \tag{5}$$

$$Wq = \frac{|\ Q^{-1}|_q}{q}, \tag{6}$$

with $P = \frac{m}{p}$ and $Q = \frac{m}{q}$.

After that, for comparing two integer A ((a_{1p}, a_{1q}), (a_{2p}, a_{2q}), (a_{kp}, a_{kq})) and B ((b_{1p}, b_{1q}), (b_{2p}, b_{2q}), (b_{kp}, b_{kq})) encrypted with the homomorphic function we will compute :

$$Reslt_A = |a_{1p} \times Wp \times \frac{1}{r_p} + a_{1q} \times Wq \times \frac{1}{r_q} + a_{2p} \times Wp \times \frac{1}{r_p} + a_{2q} \times Wq \times \frac{1}{r_q} + a_{kp} \times Wp \times \frac{1}{r_p} + a_{kq} \times Wq \times \frac{1}{r_q}|_1, \tag{7}$$

$$Reslt_B = |b_{1p} \times Wp \times \frac{1}{r_p} + b_{1q} \times Wq \times \frac{1}{r_q} + b_{2p} \times Wp \times \frac{1}{r_p} + b_{2q} \times Wq \times \frac{1}{r_q} + b_{kp} \times Wp \times \frac{1}{r_p} + b_{kq} \times Wq \times \frac{1}{r_q}|_1. \tag{8}$$

After that, if Reslt_A>Reslt_B so A > B, else A < B. Consequently, using this method, there will be no need for decrypting all data when querying range query and the operation of comparison will be done in ciphertext in the company.

However, some range queries can be transformed and performed in the cloud if the comparison range is known. For example, the query "Select C_current_add From Customer where C_birth_day between 17 and 19" would be transformed to "Select C_current_add From Customer where C_birth_day in (0 , 4, 2) or in (3 , 0, 1) or in(3 , 0, 2) " at $CSP_{p=5}$, where(0 , 4, 2) , (3 , 0, 1) ,(3 , 0, 2) are the shares of 17, 18 and 19 at $CSP_{p=5}$, respectively for $CSP_{q=7}$.

5 Security Analysis and Performance Evaluation

This section is devoted to illustrate the relevance of our approach along two axes. The first axis is about the security features of our scheme, while the second axis is about the performance of our schema in terms of time complexity and volume overhead when using the pay-as-you go paradigm.

5.1 Security Analysis

In this section we will deal with the security analysis of our schema in terms of confidentiality and integrity. We will also show the efficiency of our proposal to overcome the problem of collusion.

Confidentiality of data

To protect data from plaintext attacks and from malicious cloud providers, we propose to split data into k chunks with the random function F(x) and encrypt each chunk with

the homomorphic function ϕ(x) presented in equation 2. Then, all encrypted chunks will be stocked in two cloud providers.

The objective is to keep minimal information about data and parameters among the cloud provider. As a result, we can reduce the risk of inferring the data and breaking the system.

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

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

$X = y \times p + xp$ and $X = y \times q + xq$ such that $x < m$.

We can conclude that even if the two secret parameters are discovered by the cloud provider, he cannot identify whether the chunks of data correspond to the parameters p or q or not. This ambiguity can disrupt the work of inferring the data. Furthermore, since factoring is hard, inferring the k chunks of data without known whether the chunks of data correspond to the parameters p or q cannot be done in the case of a huge volume of data as in the data warehouse.

In the worst case, if the cloud provider infers the data, decrypts the entire share and gets original chunk, the security of our original data cannot be breached because the cloud provider has just a chunk of data and not all the original data. So, we can deduce that splitting data is essential to the security of our schema. That's why the parameter k will be chosen carefully and should be $k \geq 2$.

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 to break the security of the two cloud providers and get the chunks of secrets at the same time. Even if he does this and decrypts the k chunks of data, he cannot reconstruct the original data because he does not know the random function F(x).

Integrity of data

Outer signature

In our schema, the processing of data is done in ciphertext. Thus, there is no need to use a complex integrity function to ensure the security of data. Our goal is to propose a simple method which allows the verifying of the data sent and received from the cloud with minimum time complexity.

Our integrity function Hs (x) is homomorphic. This is very practical in the case of data warehouse. There is no need to verify each chunk of data separately if we calculate the aggregations functions.

Inner signature

The inner signature is a self-checked signature. It is based on the homomorphic characteristic of arithmetic modular. So, if there are some mistakes that cannot be detected with the outer signature, the inner signature can detect it.

Collusion

With this schema the risk of collusion is small because data is split randomly with the random function F(x). Consequently, it is hard for the two clouds providers to predict how data is split if it colludes.

5.2 Performance Evaluation

Volume overhead

Our encryption function is based on the MOD operator which divides the data in a small residue number. So, there is no overhead volume when encrypting initial data. The original data is split into k chunks and each chunk will be encrypted with the Mod operator two times. The volume of the encrypted data cannot exceed twice the volume of the original data.

Temporal complexity

In our scheme for the encryption phase, we need just O(n) operation because the encryption function needs only modular operation.

The decryption function is based on the CRT. So we just need O (lglg n) operation for the decryption phase.

Comparing our schema with the existing related approaches

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

As it is described in table 1 the main advantage of our schema is that ensure three levels of security confidentiality, integrity and availability. More than that, it resists to collision attack.

Also, compared to other solutions our schema is very performing in term of time complexity of encryption and decryption phase. This makes it very suitable in the case of data warehouse. To all those benefits, are ensured with a reasonable volume overhead and a capacity of answering query processing totally or partially in the cloud.

Table 1. Synthesis of the characteristics of some approaches discussed.

		[24]	[31]	[28]	[27]	Our schema
Confidentiality		yes	yes	yes	yes	yes
Integrity	Outer signature	No	No	No	yes	yes
	Inner signature	No	No	No	yes	yes
Availability		yes	No	No	yes	yes
Collusion		yes	No	No	yes	No
OLAP query		yes	yes	No	yes	yes
Range query		No	yes	No	No	yes
Aggregation function		yes	yes	No	yes	yes
Data volume		6n	-	6n	3n	6n (three replicas)
Time complexity of encryption phase		O(n)	-	O(n)	O(n)	O(n)
Time complexity of decryption phase		$O(nlg^2 n)$	-	$O(nlg^2n)$	$O(nt^2)$	O(lglg n)

6 Conclusion

This paper presents SecuredDW as a system for securely hosting and querying data warehouse in the cloud. SecuredDW uses a homomorphic encryption algorithm to ensure the confidentiality of data. This homomorphic encryption algorithm reveals a serious weakness as it can be deciphered by ciphertext attacks. To overcome this weakness, we propose a new sharing method of using this homomorphic privacy based on splitting data, 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. Two new signatures are

suggested to ensure the integrity of data sent and received from the cloud inner signature and outer signature.

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's why, we propose a weighted solution to this situation in order to reduce time consumption in the decryption phase. Our schema SecuredDW can be a promising solution for hosting data warehouse in the cloud that balances security and performance.

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

References

1. Rivest, R.L., Dlemam, L.A., Dertouzos, M.L.: On data bank and privacy homomorphisms. Foundations of secure computation. Academia Press, pp 169–177 (1978)
2. National Bureau of Standards: Data Encryption Standard. U.S. Department of Commerce, FIPS Publication 46, Washington, D.C., January (1977)
3. Anderson, R., Biham, E., Knudsen, L.: Serpent: A proposal for the advanced encryption standard. AES proposal: National Institute of Standards and Technology (NIST). In: W.K. Chen (ed.), Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135 (1998)
4. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126 (1978)
5. Rabin, M.O.: Digitized signatures and public-key functions as intractible as actorization. Technical Report LCS/TR-212, MIT Laboratory for Computer Science (1979)
6. ElGamal, T.: A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, IT-31(4):469–472, July (1985)
7. Jyh-Haw Yeh. A Secure Homomorphic Encryption Algorithm over Integers for Data Privacy Protection in Clouds. SmartCom (2016)
8. Zhao, F., Li, C., Chun, F.L.: A cloud computing security solution based on fully homomorphic encryption. In: 16th international conference on advanced communication technology IEEE (2014)
9. Yu, Y., Niua, L., Yang, G., Mu, Y., Susilo, W.: On the securit of auditing mechanisms for secure cloud storage. Future Generation Computer Systems, vol. 30, pp. 127–132 (2014)
10. Wei, L., Zhu, H., Cao, Z., Dong, X., Jia, W., Chen, Y., Vasilakos, A.V.: Security and privacy for storage and computation in cloud computing. Information Sciences, vol. 258, pp. 371–386 (2014)
11. Lopez-Alt, A., Tromer, V., Vaikuntanathan, E.: On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. Proceedings of the forty-fourth annual ACM symposium on Theory of computing, pp. 1219–1234 (2012)
12. Brakerski, Z., Vaikuntanathan, E.: Efficient fully homomorphic encryption from (standard) LWE. SIAM Journal on Computing 43(2):831–871 (2011)
13. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Sigmod 2004, June 13-18 (2004)
14. Hore, B., Mehrotra, S., Canim, M., Kantarcioglu, M.: Secure multidimensional range queries over outsourced data. The VLDB Journal pages, pp. 333–358 (2012)

15. 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)
16. Cruz-Lopes, C., Cesário-Times, V., Matwin, S., Rodrigues Ciferri, R., Dutra de Aguiar-Ciferri, C.: Processing OLAP Queries over an Encrypted Data Warehouse Stored in the Cloud. In: DaWaK 2014, LNCS 8646, pp. 195–207 (2014)
17. Rashmi, K.V., Shah, N.B., Gu, D., Kuang, H., Borthakur, D., Ramchandran, K.: A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster. In: HotStorage'13. San Jose, CA, June 27-28 (2013)
18. Bessami, A., Correia, M., Quaresma, B., André, F., Sousa, P.: DepSky: dependable and secure storage in the cloud-of-clouds. In: Proceedings of the sixth conference on computer systems. ACM, pp. 31–46 (2011)
19. Caclin, C., Haas, R., Vukolic, M.: Dependable storage in the intercloud. IBM rechearch, vol. 3783, pp. 1–6 (2010)
20. Alsolami, F., Chow, C.E.: N-cloud: improving performance and security in cloud storage. In: high performance switching and routing (HPSR), IEEE (2013)
21. Xu, H., Bhalerao, D.: A Reliable and Secure Cloud Storage Schema Using Multiple Service Providers. In: ICSE. DOI: 10.18293/SEKE2015-045 (2015)
22. Zhang, Q., Li, S., Liy, Z., Xingz, Y., Yang, Z., Dai, Y.: CHARM: A Cost-efficient Multi-cloud Data Hosting Scheme with High Availability. IEEE Transactions on Cloud Computing (2015)
23. Adi, S.: How to share a secret. Communication of the ACM, vol. 22 (1979)
24. Pundkar, S.N., Narendra-Shekokar, D.J.: Cloud Computing Security in Multi-clouds using Shamir's Secret Sharing Scheme. In: International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (2016)
25. Chattopadhyay, A.K., Nag, A., Majumder, K.: Secure Data Outsourcing on Cloud Using Secret Sharing Scheme. International Journal of Network Security 19(6):912–921 (2017)
26. Butoi, A., Tomai, N.: Secret sharing scheme for data confidentiality preserving in a public-private hybrid cloud storage approach. In: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (2014)
27. Hadavi, M.A., Jalili, R., Damiani, E., Cimato, S.: Security and searchability in secret sharing-based data outsourcing. In: Int. J. Inf. Secur 14(6) (2015)
28. Varunya, A., Harbi, N., Darmont, J.: A novel multi secret sharing approach for secure data warehousing and On-Line analysis processing in the cloud. In: IGI (2015)
29. Moghadam, S.S., Darmont, J., Gavin, G.: S4: A New Secure Scheme for Enforcing Privacy in Cloud Data Warehouses. In: 7th International Conference on Information Systems and Technologies (ICIST 2017). Dubai, United Arab Emirates, pp. 9–16 (2017)
30. Mahadewan, A.G., Kamesh, T.: Horns: A homomorphic encryption schema for cloud computing using residue number system. In: IEEE 45th Annual Conference on Information Sciences and Systems (2011)
31. Rivest, R.L., Dlemam, L.A., Dertouzos, M.L.: On data bank and privacy homeomorphisms. Foundations of secure computation. Academia Press, pp. 169–177 (1978)