A Query Verification Scheme for Dynamic Outsourced Databases

Xiaoming Wang
Department of Computer Science, Guangzhou ,China
Email: wxmsq@eyou.com

Duobao Yuan
Department of Computer Science, Guangzhou ,China
Email: dby@eyou.com

***Abstract*—Recently, Kyriakos et al. proposed a Partially Materialized Digest scheme (*PMD*). Their scheme uses separate indexes for the data and the verification information to realize the authenticity and completeness verification in outsourced database. In this paper, we analyze *PMD* and show the drawbacks in *PMD* such as the slow update, the collusion and forgery attacks, and no content access control etc. In order to overcome these defects, the idea of designated verifier signature is introduced into query verification, and a secure, efficient query verification scheme for dynamic outsourced databases is proposed based on *PMD*. The analysis of security and performance shows that the proposed scheme has not only the same advantages with *PMD*, but also prevents the collusion and forgery attacks as well as enforces content access control. Furthermore, the update speed of the proposed scheme is faster than *PMD*, the storage cast of the proposed is lower than *PMD*.**

***Index Terms*—outsourced databases, query authentication, security, efficiency**

I. INTRODUCTION

With rapid developments of Internet applications and network technology, the outsourcing service is emerging as an important new trend called the "application-as-a-service". Outsourcing service is an operation mode that enterprises outsource their information systems or business processes to professional external agencies that can provide these services for much lower cost due to economies of scale. As a recent manifestation of this trend, there have been growing interests in outsourcing database services in both the commercial and the research community. In 2002, Hacigumus et al. first proposed database outsourcing in ICDE (International Conference on Data Engineering)[1]. The basic idea is that data owners delegate their database needs and functionalities to a third-party that provides services to the users of the database. Since a service provider is not always fully trusted, security and privacy of outsourced data are significant issues. These problems are referred to as data confidentiality, query verification, user privacy, and data privacy. Among them, the query verification takes a crucial role to the success of the outsourcing model. The database query verification enables users quickly and accurately to verify if the query results that the database server returned are correct and complete, in order to avoid wrong decisions based on false or incomplete data.

In 2009, Kyriakos et al.[2] proposed a Partially Materialized Digest scheme(*PMD*). Their scheme uses a main index (*MI*) tree solely for storing and querying the data, and a separate digest index (*DI*) tree that contains the *MHT*-based verification information in a compressed form. Because the two processes of the search query result and the verification information can be concurrently executed, therefore their scheme is high efficient. *PMD* can support the completeness and correctness of the query verification, fit to deal with massive data and better support to update the outsourced databases.

This paper analyzes *PMD* and shows the drawbacks in *PMD* such as slow update, the collusion and forgery attacks, and no content access control etc. For dealing with these shortcomings, the idea of designated verifier signature is introduced into query verification, and a secure and efficient query verification scheme for dynamic outsourced databases is proposed based on *PMD*. The analysis of security and performance shows that the improved scheme has not only the same advantages with *PMD*, but also conceals the owner's signatures during the query verification process, prevents collusion and forgery attacks, enforces a certain content access control, and reduces the storage overhead of the database server and improves the update speed etc. Therefore the proposed scheme can overcome the disadvantages of *PMD*.

II. *PMD* ANALYSIS

A. *PMD*[2]

Suppose there are N records in the database, and each record can be expressed in the form of $<key, A_1,\ldots,A_m>$ where *key* is an indexed attribute and $A_i(i=1,2,\ldots,m)$ is a non-indexed attribute. *PMD* uses *MI* tree for storing and querying the data, and *DI* tree for verifying information in a compressed form. *MI* tree is a standard B^+ tree on key, i.e., it does not store any verification information. *DI* tree is essentially an *MHT* with degree 2 and a composite tree. Figure 1[2] illustrates the *DI* tree in a scenario where the database contains N =16 records (r_1 to r_{16}), n = 4

doi:10.4304/jcp.7.1.156-160

external nodes. Where $H(r_i)=H(key|A_1|\ldots|A_m)$, $H_i = H(h_{i|j} \,||\, h_{k|l})$, $h_{i|j} = H(H(r_i)\,|\,H(r_j))$, $S(H_{1|2|3|4})$ is its signed root using ordinary signature scheme such as RSA. $H(.)$ is a hash function.

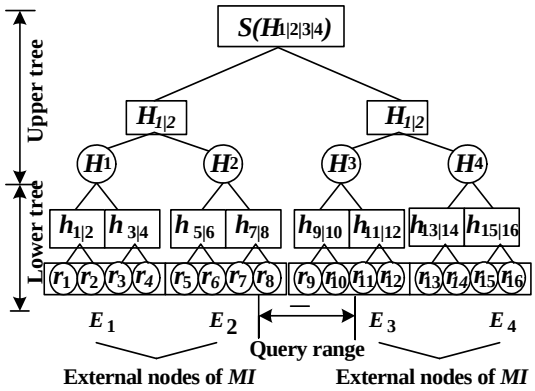

Figure 1 *DI* Digest index example

In order to reduce the storage overhead at the server and the I/O costs for VO (verification object) computation, *PMD* does not materialize the entire *DI* structure, that is, does not store the lower trees. Instead only when an external *MI* node is accessed during query processing and hash values is required from the corresponding lower tree of the *DI* in order to form the *VO*, the server computes them on the fly. Following the same principle, the server does not store the entire upper trees.

In particular, *PMD* divides the n upper tree leaves (i.e., the n lower tree roots) into groups of α. For each group, *PMD* builds a binary *MHT* (suppressed tree). In the second step, *PMD* constructs one binary *MHT* (explicit tree) on top of all the suppressed ones, as shows in Figure 2[2].

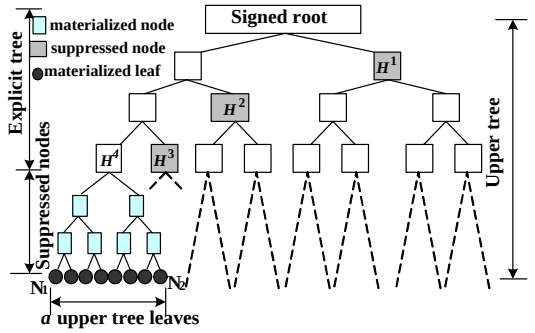

Figure 2 Upper tree compression

For every group (e.g., the group of α leaves shown in the Figure 2), *PMD* forms one *DI* page by inserting following messages (1) the signed *DI* root, and (2) its α hashes (the solid circles), and (3) the sibling digests to its path in the explicit tree (hashes H^1, H^2, H^3, shown as solid squares). In other words, the internal nodes of the suppressed tree (small rectangle) are not materialized, but they are computed on the fly (using the α leaves/solid circles) when needed. Overall, the *DI* contains n/α disk pages. Thus, *PMD* can compute *VO* for any range query with at most 2 disk accesses, one for each boundary object. If both c *left* and *right* are covered by the same suppressed tree, then only one page access is needed for their common *DI* block.

The *DI* works like an ordinary *MHT*. *PMD* considers the paths that lead to the left and to the right boundary objects as *left* and *right*. The *VO* for a range query contains (1) the signed *DI* root, (2) all left sibling hashes to the path of *left*, and (3) all right sibling hashes to the path of *right*. Upon receipt of the query result, the user combines it with *VO* components (2) and (3), to reconstruct the (missing) part of the *DI* between the paths of *left* and *right*. Then, the user verifies with the owner's public key whether the root of *DI* (i.e., component (1) of the *VO*) matches the locally computed root hash. If they match, the query results are deemed both complete and authentic; the collision-resistance of the hash function ensures that it is computationally infeasible for the server to tamper with the result and yet manage to produce hashes that match the original ones.

B. PMD analysis

Comparing with previous query verification scheme *PMD* has the following advantages: (1) there is more flexible architecture. When the user does not need verification, then *PMD* is similar to an ordinary and non-authenticated index without any performance degradation due to verification structure. Moreover, existing database can be directly authenticated, without having to be modified and re-uploaded. Therefore, the construction cost is significant savings. Also, query answering and *VO* generation can be delegated to different servers, thus the server's performance is full used; (2) separating the query processing and the *VO* computation tasks, this provides that the search result and the verification process can be implemented in parallel, so response is faster.

However, we analyze *PMD* and discover that *PMD* has the following shortcomings:

1) There exists the forgery attack in *PMD*. *PMD* uses an ordinary signature scheme such as RSA to sign *DI* root. When *PMD* deals with a user query, server returns the query results and the authentication objects including the data owner's signature to the user, thus exposes the data owner's signature. The ordinary signature is self-proven and can be passed. Anyone can verify the correctness of the signature and data as long as access to the signature. If an attacker intercepts the data and verification information that are uploaded to the server by data owner, he can impersonate the server to provide services for users or distribute the data owner's data and signatures. If a user already obtains the data and the authentication objects including the data owner's signature, he can impersonate the server to provide services for users or distribute the data owner's data and signatures. It is not permitted in practical applications such as payment services through the server.

2) There exists a collusion attack in *PMD* and it cannot achieve the content access control. When *PMD* deals with a user query, the server directly gives query results and the authentication objects including the data owner's

signature to a user, thus the users can make collusion attack together. For example, there are two legitimate users *A* and *B*, the data owner grants different privileges to *A* and *B* such as *A* can only query the index values in the range of $\alpha \le key \le b$ and *B* can only query the index values in the range of $b \le key \le c$. After *A* and *B* obtain their data and the authentication objects through legitimate query, they can collude and impersonate the server to provide services for other users such as accessing the data in the range of $\alpha \le key \le c$. However, *A* or *B* has no right to query the index values in the range of $\alpha \le key \le c$.

3) *PMD* can not well protect the privacy of data. When the user needs or only views some of these fields, the server must return all of these fields to the user in order to enable the user to verify the correctness and completeness of the data. Thus *PMD* leaks some unnecessary fields to the user and does not protect the privacy of data and implement the column-based access control, which is not allowed in many applications.

4) The update speed of *PMD* is slow. As long as the data is updated, the root node must be re-signed. Because each *DI* page stores a signed *DI* root, so each update (object insertion or deletion) requires modification of all *DI* pages, thus the cost of update is high. Although *Kyriako* et al.[2] proposed the dynamic *PMD*(*dPMD*) to resolve this problem, but *dPMD* still needs (2*d*-1) disk accesses (where *d* is the high of *DI* tree). When the database stores huge amounts of data such as $2d-1 >> 2$, *dPMD* needs to access the disk many times.

III. A NEW QUERY VERIFICATION SCHEME

In order to overcome the disadvantages in *PMD*, we improve *PMD* and propose a new query verification scheme for dynamic outsourced databases. We modify *PMD* as following:

1) Similar to *PMD*, our scheme uses *MI* tree for storing and querying the data, and *DI* tree for verifying information in a compressed form. But in order to improve the update speed, our scheme does not save the signed *DI* root in each *DI* page and stores the signed *DI* root in a separate disk block, which is different from *PMD*. When the owner inserts/deletes an object in the database, *PMD* requires to modify all *DI* pages and needs many I/O operations, however our scheme does not require to modify all *DI* pages, and needs the times of I/O operations much less than *PMD*. Therefore, the update speed of our scheme is fast than *PMD*.

Dealing with user query for the first time, server may read the signed *DI* root from the disk and save the signed *DI* root in memory until the signed *DI* root needs to be updated, thus our method guarantees the fact that server needs 3 disk accesses for only dealing with user query for the first time, and needs 2 disk accesses for each subsequent user queries, which just is the same as *PMD*. Therefore, the response speed of our scheme is as fast as *PMD*

Consider the examples in Figure 3 and Figure 4, let $|S|$ is the size of the signed *DI* root, $|H|$ is the size of hash value. Assuming that a disk block fits 7 hashes (i.e. $B=7|H|$), and $|S|=3|H|$.

In *PMD*, a *DI* tree showed in Figure 3 can be divided into 8 suppressed trees (H^1- H^8), i.e. $\alpha = 4$, because a *DI* page contains 4 hashes (the solid circles), and the sibling digests to its path in the explicit tree such as hashes H^2, H^9, H^{10} (shown as solid squares), in other words, the server needs 8 disk blocks to store the *DI* tree.

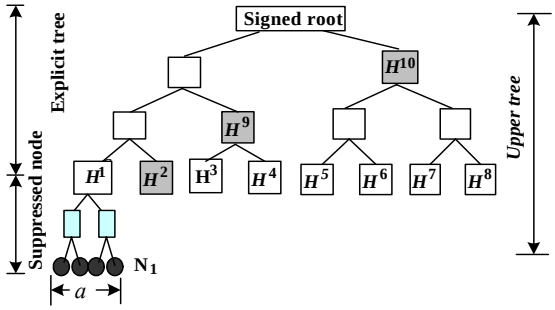

Figure 3 *DI* tree in *PMD*

If the owner inserts/deletes an object corresponding to the external leaf N_1, the updated value of this leaf causes the digest changes that propagate upwards until the root of the *DI* tree. Because each *DI* page materializes the signed *DI* root, therefore, server needs to modify all *DI* blocks and 8 I/O operations to reflect the update.

However, in our scheme, a *DI* page does not materialize the signed *DI* root, and only contains corresponding hashes, the signed *DI* root alone is stored in another a disk block. A *DI* tree showed in Figure 4 can be divided into 4 suppressed trees (H^1- H^4), i.e. $\alpha = 8$, server needs 4 disk blocks to store the *DI* tree.

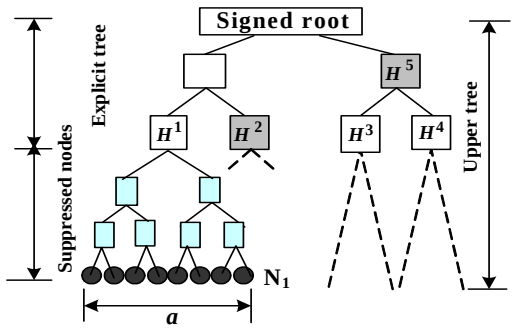

Figure 4 *DI* tree in our scheme

If the owner inserts/deletes an object corresponding to the leaf N_1, the updated value of this leaf causes digest changes that propagate upwards until the root of the *DI*. Because each *DI* page does not materialize the signed *DI* root, and only 3 *DI* pages relate with H^1hashes, therefore, server needs to modify 3 *DI* blocks and 3 I/O operations to reflect the update. Therefore, the update speed of our scheme is fast than *PMD*.

2) In *PMD* and some query verification technology [2~6,8~10], using ordinary signature such as RSA signs the root of *DI* tree, thus brought security issues such as forgery and collusion attacks as well as no content access

control etc. For dealing with these problems, we construct a new signature scheme combining with the idea of designated verifier, and use the new signature scheme to sign the root of *DI* tree, and build new query verification scheme of outsourced databases. When server deals with user queries, the server does not disclose the data owner's signature. Therefore, the new query verification scheme can prevent collusion and forgery attacks, and implement content access control.

New verification scheme is as following:

Let p_1, p_2, p_1' , p_2' and q be distinct large primes, and $p_1 = 2qp_1' + 1$, $p_2 = 2qp_2' + 1$,$n=p_1p_2$, $\varphi(n)$ =(p_1-1)(p_2-1) , g be the primitive element in $GF(n)$. $h(.)$ is a secure one-way hash function.

The data owner completes the following steps to sign root h_{root}.

1) The data owner computes two pair of secret keys (e_o, d_o) and (e_r,d_r) using RSA algorithm, and sends (e_r,d_r,g,n) to server.

2) The data owner chooses a random number $\sigma \in Z_n^*$, computes

$$Y_r = g^{e_r} \bmod n \text{ , } x = Y_r^{\sigma} \bmod n \text{ ,}$$
$$s = (\sigma d_o - h(h_{root})e_o) \bmod q$$

and sends (s, x) to server.

User verifies the query results:

1) Upon receiving the user access request, server chooses a random integer $\beta \in Z_n^*$ and computes $\gamma = \beta^{e_r} \bmod n$, and sends γ to the user.

2) The user chooses a random number $v \in Z_n^*$, and sends v to server.

3) On receiving v, server computes $Y_o = g^{e_o} \bmod n$ and

$$x^{d_r} = [(g^{e_r})^{\sigma}]^{d_r} = g^{\sigma} \bmod n \text{ , } z = \beta(g^{\sigma})^{v} \bmod n$$

and sends (z, s) and the query results to the user.

4) The user verifies the results of query. According to the principle of *MHT*, the user can calculate the hash value of the root based on the query results and *VO*, thus the user can verify

$$z^{e_r} = \gamma(g^s Y_o^{h(h_{root}^{'})})^{e_o v e_r} \bmod n$$

If it holds, then the query results are authentic and complete. Because

$$z^{e_r} = \beta^{e_r}(g^{\sigma})^{ve_r} = \gamma(g^s g^{h(h_{root})e_o})^{e_o v e_r}$$
$$= \gamma(g^s Y_o^{h(h_{root})})^{e_o v e_r} \bmod n$$

In the above query verification process, the user must cooperate with server to complete the verification of query results. Therefore, even if a attacker steals data files from the server or intercepts all the information on the process of the data owner uploading the data and verify information to server, the attacker can not impersonate the server to provide services for users since the attacker does not know the server's private key d_r and can not calculate

$$x^{d_r} = [(g^{e_r})^{\sigma}]^{d_r} = g^{\sigma} \bmod n$$

Therefore, our query verification scheme can resist the forgery attack existed in *PMD*. Moreover, because the server returned the different signature to the user each time, the user cannot use an existing signature to implement the collusion attack. So our scheme can also resist the collusion attack and achieve the content access control.

3) Aiming at the defect that *PMD* cannot realize the column-based access control, our scheme uses

$$H(r_i)= H(key||H(A_1)||\ldots||H(A_m))$$

to compute the message authentication code of the record r_i. When a user queries some fields, the server only returns the values of query fields and the hash values of other fields to the user, and does not return to the values of all fields, and hence does not leak unnecessary information to the user. Thus our scheme protects the privacy of data and implements the column-based access control. Therefore, our scheme can overcome the defect that *PMD* cannot implement the column-based access control.

PERFORMANCE AND SECURITY ANALYSIS

Our scheme, which is similar to *PMD*, uses *MI* tree for storing and querying the data, and *DI* tree for verifying information in a compressed form. But our scheme is different from *PMD*. Our scheme does not save the signed *DI* root in each *DI* page and stores the signed *DI* root in a separate disk block, thus improves update speed. Moreover, our scheme reduces the server's storage overhead since a signed *DI* root is stored.

After start, server immediately reads the signed *DI* root from the disk and computes (Y_o , g^{σ}), and then stored them in memory. In order to compute a *VO*, server needs 2 disk accesses for each user query, which just likes *PMD*. Therefore, the response speed of our scheme is as fast as *PMD*.

In order to verify the storage cost and update speed of our scheme, we perform experiments using Forest CoverType Data set that is provided by University of California, Irvine. Our experiments focus on compare between our scheme and *PMD*. The server storage cost simulation shows as Figure 5. The update speed simulation shows as Figure 6.

The experiments show that our scheme reduces the storage overhead of the database server and improves the update speed at the database server.

We introduce the idea of designated verifier to the query verification technology of outsourced databases, and construct a new signature scheme. We use the new signature scheme to sign the root of *DI* tree, rather than the ordinary signature scheme. Because the server returns to user different signatures when handling users query each time, the data owner's signature is hidden. Thus the attackers or users cannot impersonate the server to provide services for other users by using the existing signatures, and some users cannot make collusion attack using their own data and verification information. Therefore, our scheme overcomes the disadvantages of *PMD* such as collusion attack and forgery attack. The previous query verification schemes also use the ordinary signature scheme such as RSA signature scheme, but the

ordinary signature is self-proven and can be passed. Anyone can verify the signature as long as access to the signature. Therefore these schemes also exist above problems. We introduce the idea of designated verifier to the query verification technology of outsourced databases, and well resolve these security issues.

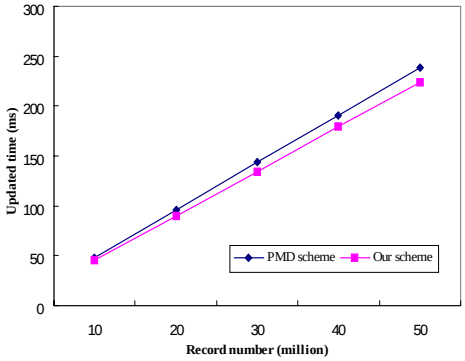

Figure 5 Server storage cost

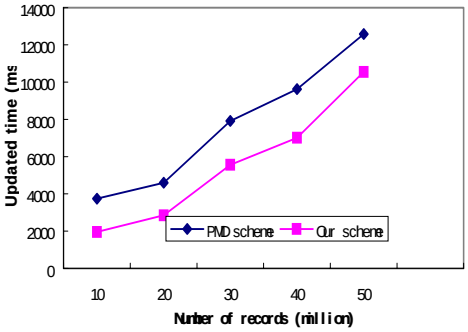

Figure 6 Server update speed

Because using $H(r_i)=H(key||H(A_1)||\ldots||H(A_m))$ to compute the message authentication code of the record r_i, server only returns the values of query fields and the hash values of other fields to the user when a user queries some fields. Therefore, our scheme can better support the projection queries, implement the content access control, and protect private data by the greatest extent.

V. CONCLUSION

In this paper, we improve *PMD*, and propose a new query verification scheme. We modify the storage contents of the *DI* page in *PMD* in order to improve the update speed. We also construct a new signature scheme combining with the idea of designated verifier signature, and replace the signature scheme of *PMD* with the new signature scheme to sign *DI* root in order to resist collusion and forgery attacks. We use $H(r_i)=H(key||H(A_1)||\ldots||H(A_m))$ to compute the message authentication code of the record r_i in order to protect the privacy of data and implement the column-based access control. We perform experiments on our scheme, and focus on comparing our scheme with *PMD*. The result of the experiments shows that the update speed of our scheme is faster than *PMD*, the storage cast of our scheme is lower than *PMD*.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China under Grant (61070164); Science and Technology Planning Project of Guangdong Province, China (S2011010001599); Natural Science Foundation of Guangdong Province, China (81510632010 0000 22).

REFERENCES

[1] H.,Hacigumus,S.Mehrotra,B.Iyer, et al. Providing database as a service. ICDE'02 Proceedings of the 18th International Conference on Data Engineering, IEEE Computer Society Washington, pp. 29-38.

[2] M.Kyriakos,S.Dimitris,P.HweeHwa. Partially Materialized Digest Scheme: An Efficient Verification Method for Outsourced Databases. The VLDB Journal,2009,18(1),pp. 363–381.

[3] P.Devanbu, M.Gertz, C.Martel, et al. Authentic third-party data publication. 14th IFIP 11.3 Working Conference in Database Security, The Netherland: Kluwer, 2000,pp.101-112.

[4] M.Charles, N.Glen, D.Premkumar, et al. A general model for authenticated data structures. Algorithmica.2004,39, pp.21-41.

[5] E.Mykletun, M.Narasimha, G.Tsudik. DSAC: integrity for outsourced databases with signature aggregation and chaining. ACM CIKM. New York, 2005,pp.235-236.

[6] L.Feifei,H.Marios, K.George,et al. Dynamic Authenticated Index Structures for Outsourced Database. Chicago, Illinois: ACM SIGMOD, 2006,pp.121-132.

[7] E. Mykletun, M.Narasimha, G.Tsudik. Authentication and Integrity in Outsourced Databases. ACM Transactions on Storage,2006,2(2), pp. 107-138.

[8] N.Maithili,T.Gene. Authenticated of Outsourced Databases Using Signature Aggregation and Chaining. Wuwongse: DASFAA. 2006, LNCS 3882,pp.:420-436.

[9] J.A.Mikhail,C.Y.Sun,K.Ashish. Efficient Data Authentication in an Environment of Untrusted Third-Party Distributors. ICDE 2008,pp.696-704.

[10] T.G.Michael,T.Roberto,T.Nikos.Superefficient verification of dynamic outsourced databases. CT-RSA 2008, LNCS 4964,pp.407-424.

[11] L.Bouganim, F.D.Ngoc, P.Pucheral,et al. Chip-secured data access: Reconciling access rights with data encryption. Proc. of the 29th International Conference on Very Large Database. VLDB Endowment, 2003,pp.1133-1136.

[12] L. Bouganim, C.Cremarenco, F.D.Ngoc, et al. Safe data sharing and data dissemination on smart device. Proc. of the 2005 ACM SIGMOD International Conference on Management of Data. New York: ACM Press. 2005, pp.888-890.

[13] Q Zhu, J.H. Chen, J.J.Le. Digital watermark-based authentication of queries in outsourced database. Journal of Computer Applications. 2008, 28(3),pp.605-608.

Xiaoming Wang received her Ph.D Degree from the College of Mathematics, Nankai University, China, in 2003. Currently she is a professor in the Computer Science Department, Jinan University. Her research interests include database security, network security, cryptography, etc.

Duobao Yuan received his M.S. Degree from Department of Computer Science, Jinan University, China, in 2010. Currently he is a engineer in Commercial Bank of China. His research interests include database security, network security,etc.