2009 International Conference on Multimedia Information Networking and Security

Data Privacy Preserving Mechanism based on Tenant Customization for SaaS

Kun Zhang, Yuliang Shi, Qingzhong Li[*], Ji Bian
School of Computer Science and Technology
Shandong University
Jinan, China
jackie_119@mail.sdu.edu.cn, liangyus@sdu.edu.cn, lqz@sdu.edu.cn, sdubj@yahoo.com.cn

***Abstract*—As a newly software delivery model, Software as a Service, SaaS for short, is the best way for small and medium enterprise to adopt the newly technology. However trustworthiness is greatest challenge in the wide acceptance of SaaS. In the absence of trustworthiness in SaaS applications, data privacy is the primary and the most important issue for tenants. How to protect the data privacy when software service and database are both hosted at the service provider's client is still an open issue. So based on the customization feature of SaaS applications and shared database shared schema storage model, this paper demonstrates the shared data storage model, defines three kinds of privacy constraints, and then proposes a customizable privacy constraints based approach for data privacy preserving by combing data encryption and information disassociation. This approach is proofed correctly and could be used for privacy preserving in SaaS applications in static scenarios.**

Keywords-Software as a Service; Privacy Preserving; Customization; Multi-Tenant

I. INTRODUCTION

SaaS, i.e. Software as a Service, is a newly software delivery model with the development of network and maturity of application software. For simplicity, SaaS could be defined as "Software deployed as a hosted service and accessed over the Internet". In SaaS, service providers take charge of the deployment, maintenance, upgrade and management of software service and charge tenants for services subscribed by tenants on demand. For small and medium enterprises, SaaS is the best way for them to adopt the newly technology. SaaS model reduce the cost in hardware, software, maintenance, upgrade and management for tenants. Service providers could use scale effect to provide one application instance to server a large number of tenants.

Though SaaS model is appealing to customers, especially for the small and medium enterprises, there are some hurdles preventing the wide acceptance and further development of SaaS, especially the trustworthiness of data. In SaaS, thousands of tenants store business data in the database of service provider and then lose control of these data. Due to the trustworthiness of service providers, data privacy and data integrity become the most important issues in SaaS applications. Service providers may be aggressive, they could alter, delete tenants' data or add fake data due to some purposes.

There has been a lot of research on data privacy preserving technology. In general, there are primarily two kinds of methods to protect data privacy, i.e. data encryption and information disassociation. However, these two approaches are not appropriate for SaaS. This paper proposes a novel customizable privacy constraints based privacy preserving approach in Software-as-a-Service applications. This approach combines the data encryption and information disassociation and defines three kinds of privacy constraints based on customization functionality in SaaS applications.

This paper organizes as follows. Section II presents the multi-tenant shared storage model as our basis. Section III defines the privacy constraints and introduces three kinds of privacy constraints. Section IV illustrates the customizable privacy preserving policy and the privacy constraints checking. Section V presents a novel privacy preserving approach based on privacy constraints. Section VI gives the related work. Section VII makes a conclusion.

II. MULTI-TENANT DATA STORAGE

To manage multi-tenant data, there are mainly three kinds of multi-tenant data architectures from the isolated extreme to the shared extreme. The simplest approach to manage multi-tenant data is storing tenant data in separate databases. The second approach to manage multi tenant data is shared database, separated schemas. The third approach to manage multi tenant data is shared database and shared schema.

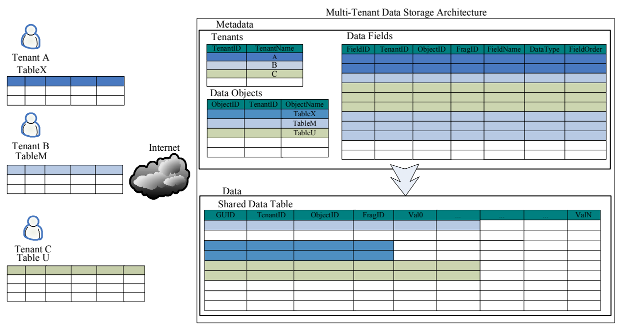


Figure 1. Multi-Tenant Data Storage Architecture

[*] Correspondent author: Qingzhong Li, lqz@sdu.edu.cn

Based on the three approaches for managing multi tenant data, we adopt the third approach, i.e. shared database, shared schema. We take the multi-tenant data architecture of Force.com and make an improvement by introducing a new column FragID which is used in data privacy preserving for information dissociation and reconstruction.

In the multi-tenant data storage architecture, there are two kinds of data, including metadata and data. Metadata, including Tenants, Data Objects and Data Fields, describes the data accordingly. Tenants could use metadata to extend their data model to meet their individual needs. Based on the metadata, tenants' data could be transformed between tenants' logical view and physical view.

III. Privacy Constraints

Based on the customization feature of SaaS, this paper proposes a privacy preserving approach based on customization privacy constraints.

For simplicity, this paper just considers a single relation in tenant's logical view. Given a relation R{a_1,a_2,…a_n} where $a_i(1 \leqslant i \leqslant n)$ is the ith attribute.

Definition 1: Privacy Constraints: Let A be a set of attributes, privacy constraint is a binary tuple PC{ AS, PP}. AS(Attribute Set) is the set of attributes which involved in this privacy constraints and AS is a subset of A, i.e. AS $\subseteq$ A. PP(Privacy Policy) is the customized privacy policy, which could be one of the three options {Compatible, Non-Compatible, Encryption}.

Based on the privacy policy in privacy constraints, there should be three different kinds of privacy constraints.

Definition 2: Compatible Privacy Constraints (cPC): compatible privacy constraint is one of three kinds of privacy constraints, i.e. cPC={AS, Compatible}. In compatible privacy constraint, values of attributes could be available together, i.e. the association between the values of attributes in AS is not sensitive and could be available to the service provider.

Definition 3: Non-Compatible Privacy Constraint (ncPC): non-compatible privacy constraint is one of three kinds of privacy constraints, i.e. ncPC={AS, Non-Compatible}. In non-compatible privacy constraint, values of attributes in the AS set should not be available together, i.e. the association between the values of attributes in AS is sensitive and therefore should not be released.

Definition 4: Encryption Privacy Constraints (ePC): encryption privacy constraint is one of three kinds of privacy constraint, i.e. ePC={AS, Encryption}. In encryption privacy constraint, values of attributes are sensitive and should not be available to service provider. The values of attributes in privacy constraint should be encrypted for privacy preserving.

IV. Privacy Preserving Policy

Based on the privacy constraints, tenants could customize specific policy for data privacy preserving according to their individual requirements.

A. *Customizable Data Privacy Preserving Policy*

Definition 5: Customizable Data Privacy Preserving Policy (CDPPP): Customizable Data Privacy Preserving Policy, CDPPP in short, is a triple tuple {CPC, NCPC, EPC}. In CDPPP, CPC is set of compatible privacy constraints (cPC), i.e. CPC={cPC1, cPC2, … cPCx}; NCPC is set of non-compatible privacy constraints (ncPC), i.e. NCPC={ncPC1, ncPC2, … ncPCy}; EPC is set of encryption privacy constraints (ePC), i.e. EPC={ePC1, ePC2, … ePCz}.

Note that in MEPC, CPC, NCPC or EPC set may be empty due to tenant's customization.

B. *Privacy Preserving Policy Checking*

To avoid the conflict and redundancy among the privacy constraints, this paper gives privacy constraints check up process to get a well defined privacy preserving policy.

For simplicity, we use the cPC, ncPC, ePC not only denote the privacy constraints but also the attribute set AS in the privacy constraints.

1) *Redundancy Checking*

Definition 6: Redundant compatible privacy constraints (rCPC): In CPC{cPC1, cPC2,…cPCx}, $\exists$ cPCi, cPCj $\in$ CPC, i$\neq$j, cPCi $\cap$ cPCj $\neq \Phi$.

Based on the SaaS application logic, we make the attribute appears once and only once in the customization of compatible privacy constraints to make sure the non redundant property of this kind of privacy constraints.

Definition 7: Redundant non-compatible privacy constraints (rNCPC): In NCPC{ncPC1, ncPC2,…ncPCy}, $\exists$ ncPCi, ncPCj $\in$ NCPC, i $\neq$ j, ncPCi $\subseteq$ ncPCj or ncPCj $\subseteq$ ncPCi.

Given the rMEPC, we reserve the relative small ncPC to reduce redundancy.

Definition 8: Redundant encryption privacy constraints (rEPC): in EPC {ePC1, ePC2,…ePCz}, $\exists$ cPCi, cPCj $\in$ CPC, i$\neq$j, $\exists$ ai$\in$ A, ai$\in$ cPCi and ai$\in$ cPCj, i.e. the value of attribute ai encrypted at least twice.

To reduce the redundancy of encryption privacy constraints, we confine cardinality of the attribute set of ePC is singleton, i.e. there exist only one attribute in the attribute set of ePC for encryption.

2) *Conflict Checking*

After reducing the redundancy among privacy constraints, we then face the conflict among privacy constraints, i.e. the conflicts between non-compatible privacy constraints and compatible privacy constraints.

Definition 9: Conflicted Privacy Constraints (ConPC): In nrCPC and nrNCPC, $\exists$ ncPCi $\in$ nrNCPC cPCj$\in$ nrCPC, ncPCi $\subseteq$ cPCj.

V. Customizable Privacy Preserving

A. *Data Privacy Preserving*

For the efficiency of data processing and management, this paper presents some principles about customizable privacy constraints based data privacy preserving approach.

Principle 1: The number of fragments of an original logical relation view should be as small as possible.

Principle 2: Those attributes, which could be processed and queried together frequently, should be put in the same fragment without violating the privacy constraints.

Principle 3: The number of attributes in plaintext of different fragment should be as even as possible.

Principle 4: The attributes which are not in the encryption privacy constraints should be placed in just one fragment in plaintext.

Principle 1 and 2 means that query could be processed in relatively large fragment which could reduce the probability of join operations between different fragments.

Principle 3 reduces the NULL waste of space given the appropriate fixed upper limit of the universal table layout.

Principle 4 means that there is no value of attribute in plaintext appearing at least two fragments, which could be used to violate data privacy by join operation between fragments.

```
CPCDPP
  Input:
        Relation A{$a_1,a_2,\ldots,a_n$}
        Customizable Data Privacy Preserving Policy {CPC, NCPC, EPC}
Output:
        Data Privacy Preserving Enforcement Policy F{$F_1,F_2,\ldots,F_k$}

  Main
        //Redundancy Checking
        nrCPC = RedundancyCheckingonCPC(CPC)
        nrNCPC=RedundancyCheckingonNCPC(NCPC)
        nrEPC=RedundancyCheckingonEPC(EPC)
        //Conflict Checking
        nConPC={nConCPC, nConNCPC,nConEPC}=ConflictChecking(nrCPC, nrNCPC, nrEPC)

        //initial fragment
        F={$F_i$ | $F_i$ is transformed from the well defined nConCPC}
        Num[$F_i$] is the number of attributes in fragment of $F_i$

        //set of attributes in plaintext to be solved
        Set ToBeSolvedPlainText = {$a_i$ | $a_i$ is the attribute in the plaintext form and $a_i$ is not in the attribute set of F}
        //set of attributes in ciphertext to be solved
        Set ToBeSolvedCipherText = {$b_i$ | $b_i$ is the attribute in the ciphertext form}

        //process the attributes in plaintext
        /*the main idea is to put the attribute non-solved into the fragment with minimum Num[$F_i$] without violate the
privacy constraints */
        While (ToBeSolvedPlainText != NULL)
             // select a attribute unsolved
              $a_i$= SelectFromSet (ToBeSolvedPlainText)
              ToBeSolvedPlainText = ToBeSolvedPlainText − { $a_i$}
              // reorder F
              Reorder the sequence of $F_i$ according to the Num[$F_i$] in an Ascending order
              newFragment=true
              for each $F_i$ in an ascending order
                    if ($F_i \cup \{a_i\}$ does not violate the non-compatible privacy constraints )
                          $F_i \cup \{a_i\}$
                          Num[$F_i$ ] +=1
                          newFragment=false
                          break
              // if the selected attribute are not allowed to put into the existing $F_i$, then generate a new fragment for it
               if newFragment = true
                    Generate a new Fragement $F_{i+1}=\{a_i\}$
                    Num[$F_{i+1}$]=1

         //process the attributes in ciphertext
        /* the main idea is to put the attribute non-solved into the fragment with minimum Num[$F_i$] */
         While (ToBeSolvedCipherText!=NULL)
              $b_i$= SelectFromSet (ToBeSolvedCipherText)
              ToBeSolvedCipherText = ToBeSolvedCipherText − {$b_i$}
              Reorder the sequence of $F_i$ according to the Num[$F_i$] in an Ascending order
              Select the $F_i$ with the minimum Num[$F_i$ ]
                    $F_i \cup \{a_i\}$
                    Num[$F_i$ ] +=1

         //Minimal Fragmentation Checking
         while ( there exist $F_i \cup F_j$ dose not violate the privacy constraints )
               then merge $F_i$ and $F_j$

         //Output the results
          Return F{$F_1,F_2,\ldots,F_k$}
```

Figure 2. Privacy Preserving Methods CPCDPP

Based on the principles, this paper gives the privacy preserving enforcement policy.

Definition 10: Data Privacy Preserving Enforcement Policy (DPPEP): Data Privacy Preserving Enforcement Policy is a fragmentation F{F_1, F_2,…, F_k}, in which F_i ($1 \leqslant i \leqslant k$)is a physical fragment that could be stored in the shared table. As we can see, these attributes could appear one and just one fragment for privacy preserving, or attackers would get some privacy information about a record by join operations.

Definition 11: Minimal Data Privacy Preserving Enforcement Policy (minDPPEP): For a fragmentation F{F_1, F_2,…, F_k}, $\forall F_i, F_j \in F$, $F_i \cup F_j$ would violate the non-compatible privacy constraints.

Given the above, we present the **C**ustomizable **P**rivacy **C**onstraints based **D**ata **P**rivacy **P**reserving approach in Figure 2, CPCDPP in short.

The process of CPCDPP is as follows. Firstly, check redundancy and conflicts among privacy constraints. Secondly, based on the well defined compatible privacy constraints, add the rest attributes into the minimal fragment without violating the non-compatible privacy constraints.

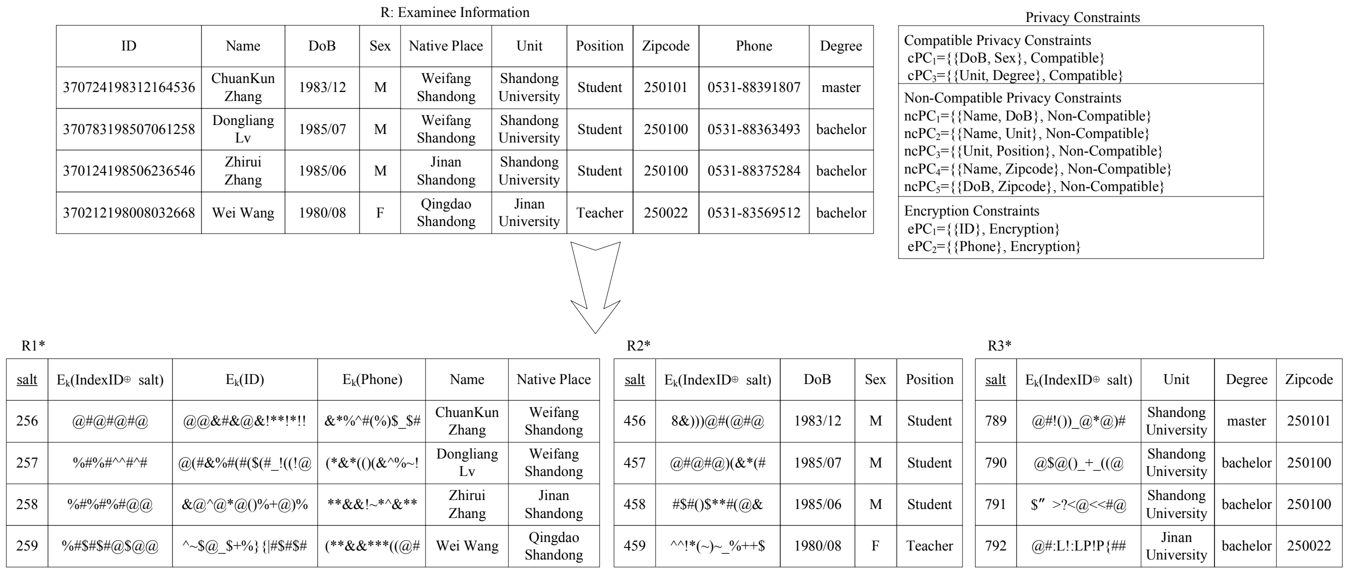
Theorem 1 (Correctness): CPCDPP terminates and finds a minimal data privacy preserving enforcement policy.

Proof (sketch). As the two attributes sets ToBeSolvedPlainText and ToBeSolvedCipherText is finite, the three while loops in CPCDPP terminate. Due to the while condition changes, the while loop is executed till the condition doesn't satisfy.

Redundancy checking reduces the redundancy on three kinds of privacy constraints. And then CPCDPP resolves conflicts among the privacy constraints. Based on the well-defined compatible privacy constraints, CPCDPP inserts the attributes in plaintext form not solved into the smallest fragment without violating the privacy constraints. The attributes in cipher text follow the same process to be inserted into the minimal fragment. To get a minimal fragmentation, CPCDPP checks out the fragmentation and then merges the fragments without violating privacy constraints. So CPCDPP gets a minimal fragmentation at last.

B. *Example*

An example for privacy preserving based on this approach is shown in Figure 3.



R: Examinee Information

ID	Name	DoB	Sex	Native Place	Unit	Position	Zipcode	Phone	Degree
370724198312164536	ChuanKun Zhang	1983/12	M	Weifang Shandong	Shandong University	Student	250101	0531-88391807	master
370783198507061258	Dongliang Lv	1985/07	M	Weifang Shandong	Shandong University	Student	250100	0531-88363493	bachelor
370124198506236546	Zhirui Zhang	1985/06	M	Jinan Shandong	Shandong University	Student	250100	0531-88375284	bachelor
370212198008032668	Wei Wang	1980/08	F	Qingdao Shandong	Jinan University	Teacher	250022	0531-83569512	bachelor

Privacy Constraints

Compatible Privacy Constraints
cPC_1={{DoB, Sex}, Compatible}
cPC_3={{Unit, Degree}, Compatible}

Non-Compatible Privacy Constraints
$ncPC_1$={{Name, DoB}, Non-Compatible}
$ncPC_2$={{Name, Unit}, Non-Compatible}
$ncPC_3$={{Unit, Position}, Non-Compatible}
$ncPC_4$={{Name, Zipcode}, Non-Compatible}
$ncPC_5$={{DoB, Zipcode}, Non-Compatible}

Encryption Constraints
ePC_1={{ID}, Encryption}
ePC_2={{Phone}, Encryption}

R1*

salt	E_k(IndexID⊕ salt)	E_k(ID)	E_k(Phone)	Name	Native Place
256	@#@#@#@	@@&#&@&!**!*!!	&*%^#(%)$_$#	ChuanKun Zhang	Weifang Shandong
257	%#%#^^#^#	@(#&%#(#($(#_!((!@	(*&*(()(&^%~!	Dongliang Lv	Weifang Shandong
258	%#%#%#@@	&@^@*@()%+@)%	**&&!~*^&**	Zhirui Zhang	Jinan Shandong
259	%#$#$#@$@@	^~$@_$+%}{|#$#$#	(**&&***((@#	Wei Wang	Qingdao Shandong

R2*

salt	E_k(IndexID⊕ salt)	DoB	Sex	Position
456	8&)))@#(@#@	1983/12	M	Student
457	@#@#@)(&*(#	1985/07	M	Student
458	#$#()$**#(@&	1985/06	M	Student
459	^^!*(~)~_%++$	1980/08	F	Teacher

R3*

salt	E_k(IndexID⊕ salt)	Unit	Degree	Zipcode
789	@#!())_@*@)#	Shandong University	master	250101
790	@$@()_+_((@	Shandong University	bachelor	250100
791	$″ >?<@<<#@	Shandong University	bachelor	250100
792	@#:L!:LP!P{##	Jinan University	bachelor	250022

Figure 3. An Example of CPCDPP

C. *Privacy Preserving Analysis*

Given that a single relation R with cardinality n and its fragmentation F{F_1,F_2,…,F_k} based on the customizable privacy constraint with cardinality n_1,n_2,…n_k respectively just considering the values in plaintext form. Like protection level in [9], we then give the privacy preserving level as follows.

Definition 12: Privacy Preserving Level (PPL): Privacy Preserving Level is the probability that insider attacker could not guess a whole record.

$$PPL=1-n/(n_1 \times n_2 \times \ldots \times n_k) \tag{1}$$

We could use the PPL to evaluate the protection level in the shared database of SaaS applications. Detailed analysis on the privacy preserving level is omitted by space consideration.

As we can see, CPCDPP is just appropriate for the static database of SaaS applications. When tenant insert a single record or a small amount of records to the database, our approach is useless in privacy preserving without help of fake tuples however customizable privacy preserving approach works. Attackers could find the association among the values of attributes in different fragment and get know the whole record. Note that, we must make assure that attackers could not differentiate real data from the faked data.

VI. Related Work

Microsoft gave the Software-as-a-Service Maturity Model based on scalability, multi-tenant efficiency and configuration. Frederick Chong in Microsoft analyzed three approaches to managing multi-tenant data, including separate database, shared database separate schema, shared database shared schema. Based on the shared database shared schema, Microsoft built a prototype system – Crab. IBM proposed a framework for native multi-tenancy application development and management by introducing the multi-tenancy enablement layer in [1]. IBM explored the configuration and customization issues and challenges in SaaS application in [2]. Reference [3] designed a multi-tenant framework for an Electronic Contract Management Application.

Reference [4] described three approaches to implement multi-tenant databases and made a comparison. Reference [5] made an analysis on different multi-tenant data model and proposed a chunk folding technology to improve performance on shared database shared schema database. Based on the shared tables and shared database instances (STSI), [6] proposed the bitmap interpreted tuple format to reduce the waste of NULL values in the shared tables and implemented a multi-tenant system – M-Store. Salesforce.com proposed the meta-data driven multi-tenant architecture Force.com to managing multi-tenant data.

For the privacy preserving, there are mainly two different approaches, i.e. data encryption and information disassociation. The general approach is to encrypt data. In outsourced database scenario, [7] used encryption.

Reference [8] proposed firstly storing data in plaintext with a series of privacy constraints. In [8], authors supposed data to be stored on two different servers, which belonged to two different service providers and never exchange information. Reference [9] proposed the blind custodian mechanism, in which all the fragments all stored at the same server and the association among fragments stored at the client for reconstruction the original. Reference [10] proposed an improvement based on custodian mechanism by introducing information association. In [11], the message should be compressed and split into multiple shares before it put into the lucky dip. Secure is achieved by combing the large number of shares, in which some are real and some are faked, for preventing attack guessing which shares reconstructing an original and truly message. Reference [12] introduced privacy constraints and generated a minimal fragmentation based on these privacy constraints.

VII. Conclusion

Combing data encryption and information dissociation, this paper builds a multi-tenant data architecture and proposes a customizable privacy constraints based approach for data privacy preserving in SaaS applications. However, this approach could be only applicable in the situation where attackers could get the snapshot of the shared database at some moment. To solve this problem, we would insert the fake tuples at the tenant's client for confusion. How to generate fake tuples, how many faked tuples should be inserted, how to distinguish between real tuple and fake tuple are the challenge we face in the near future. Further, this paper just consider privacy preserving for a single relation, approaches for the multiple relation could be studied and proposed later.

Acknowledgment

The research is supported by the National Natural Science Foundation of China under Grant No.60673130 and No.90818001, the Natural Science Foundation of Shandong Province of China under Grant No.Y2006G29, No.Y2007G24 and No.Y2007G38.

References

[1] Chang Jie Guo, Wei Sun, Ying Huang, Zhi Hu Wang, Bo Gao, "A framework for native multi-tenancy application development and management", 9th IEEE Internatioal Conference on E-Commerce Technology/ 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services(CEC/EEE 2007), Tokyo Japan, July 2007, pp551-558

[2] Wei Sun, Xin Zhang, Chang Jie Guo, Pei Sun, Hui Su, "Software as a Service: configuration and customization perspective", 2008 IEEE Congress on Service Part II, Hawaii USA,2008, pp18-24

[3] Thomas Kwok, Thao Nguyen, Linh Lam, "A Software as a Service with multi-tenancy support for an electronic contract management application", 2008 International Conference on Service Computing (SCC 2008), Hawaii USA, July 2008, pp179-186

[4] Dean Jacobs, Stefan Aulbach, "Ruminations on multi-tenant databases", BTW 2007, Aachen Germany, pp514-521

[5] Stefan Aulbach, Torsten Grust, Dean Jacobs, Alfons Kemper, Jan Rittinger, "Multi-tenant databases for software as a service: schema-mapping techniques", Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2008), Vancouver Canada, June 2008, pp1195-1206

[6] Mei Hui, Dawei Jiang, Guoliang Li, Yuan Zhou, "Supporting database applications as a service", Proceedings of the 25th International Conference on Data Engineering(ICDE 2009), Shanghai China, March 2009, pp832-843

[7] Hakan Hacigümüs, Sharad Mehrotra, Balakrishna R. Iyer, "Providing database as a service", Proceedings of the 18th International Conference on Data Engineering (ICDE 2002), California USA, March 2002, pp29-38

[8] Gagan Aggarwal, Mayank Bawa, Prasanna Ganesan, Hector Garcia-Molina, Krishnaram Kenthapadi, Rajeev Motwani, Utkarsh Srivastava, Dilys Thomas, Ying Xu, "Two can keep a secret: a distributed architecture for secure database services", Second Biennial Conference on Innovative Data Systems Research(CIDR 2005), Asilomar CA, January 2005, pp186-199

[9] Amihai Motro, Francesco Parisi-Presicce, "Blind custodians: a database service architecture that supports privacy without encryption", Data and Applications Security XIX, 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec 2005) , Storrs CT USA, August 2005, pp338-352

[10] Xianwei Li, Guohua Liu, Ying Yuan, Huidong Ma, "A database encryption method based on information dissociation and association", Computer Engineering & Science, National University of Defense Technology, School of Computer, Changsha, Hunan China, 2007, pp54-60

[11] Richard Brinkman, "Searching in encrypted data", doctoral dissertation, University of Twente, Holland, 2007

[12] Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, Pierangela Samarati, "Fragmentation and encryption to enforce privacy in data storage", 12th European Symposium on Research In Computer Security (ESORICS 2007), Dresden Germany, September 2007, pp171-186