Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2016, Vol. 12(7) 1–7
© The Author(s) 2016
DOI: 10.1177/1550147716657255
ijdsn.sagepub.com
SAGE

Privacy enhancement using selective encryption scheme in data outsourcing

Long Nguyen-Vu, Jungsoo Park, Minho Park and Souhwan Jung

Data outsourcing is a new, emerging data management paradigm in which the owner of data is no longer totally responsible for its management. Rather, a portion of data is outsourced to external providers who offer data management functionalities.

Abstract

In this article, we introduce a practical scheme that dynamically secures and outsources data on demand as well as propose a corresponding architecture to securely process data in database service provider. We also adopt the application of bring your own device in this scheme as an enhanced security solution. After studying over 1300 database models, we expect this scheme can be applied in production with justifiable result.

Keywords

Cloud privacy, bring your own device, data outsourcing, database as a service

Date received: 3 March 2016; accepted: 1 June 2016

Academic Editor: Fan Wu

Introduction

Nowadays, cloud services have been widely used in recent years, they brought convenience to the enterprise and they also brought new challenges: Data outsourcing becomes a headache as companies cannot guarantee the integrity and confidentiality of what they provided to the service providers. In terms of reliability, data could be modified accidentally or maliciously or could be leaked to the adversaries (attackers, malicious providers).[1–3] The demand of storing and processing data online grows quickly to adapt to the rapid change of business. It could lead to crisis if the cloud service provider is compromised and data of users are exposed to attackers in plain text.

Researchers have been working on a variety of techniques in order to limit the impact caused by data leak. Ones could be mentioned are access control, encryption or even extra layer of authentication and authorization leveraged by OAuth2.[4] The first step is always about securely encrypting data before storing them on the cloud. Even though processing data in encrypted format greatly enhances the security, it is not an easy task. Homomorphic encryption allows arbitrary computations on fully encrypted data, but this technique is limited to certain algebraic calculations, and the low performance makes it especially impractical to be applied in production.

After presenting our related work in the conference of Information and Communication Technology Convergence,[5] we realize bring your own device (BYOD) is immersing as a significant trend in corporate network. According to Forbes report[6], over 80% of enterprises now allow employees to use personal devices for daily tasks. In addition to 'data loss', 'stolen device' and 'unauthorized access' are frequent terms to express the concern of corporations regarding to BYOD security.[7] We saw the need of rigorous improvement over our previous evaluation analysis, as well as the adoption of BYOD should be made to be able to keep pace with the ever-changing technology. In a nutshell, we believe it is mandatory to keep data safe from end to end. As described in Figure 1, all requests (req(1), req(2)) and responses (res(1), res(2)) must be

School of Electronic Engineering, Soongsil University, Seoul, Korea

Corresponding author:
Souhwan Jung, School of Electronic Engineering, Soongsil University, Seoul, Korea.
Email: souhwanj@ssu.ac.kr

Creative Commons CC-BY: This article is distributed under the terms of the Creative Commons Attribution 3.0 License (http://www.creativecommons.org/licenses/by/3.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (http://www.uk.sagepub.com/aboutus/openaccess.htm).

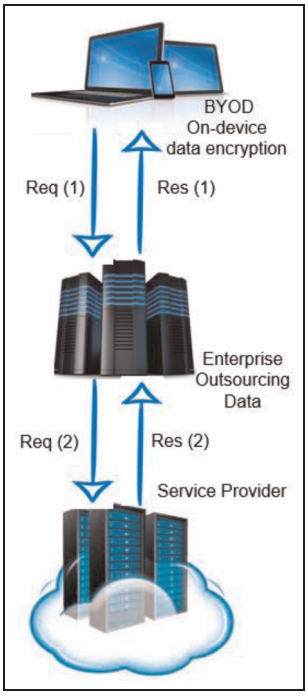

Figure 1. BYOD in the enterprise. BYOD: bring your own device.

protected by obfuscation to limit information revealed to third party or any interceptor, as in Man In The Middle attack.

To the best of our knowledge, there is no solution for operations on fully encrypted data that could be applied into real world scenarios. We take another approach with obfuscated data to limit the information exposed to other parties: Only data that need to be computed should be revealed to service provider, the rest is kept secret by encryption. In other words, the information to be processed appears in partially encrypted format. In terms of database scheme, we try to secure sensitive columns of a table before providing to the database service provider (DSP). Advantages and limitations of this approach will be described in the next sections.

In summary, our study in this article includes: (1) survey 50 categories of database, like banking, education, government, etc., with more than 1300 designed samples; (2) address the problem of data outsourcing with threat models and (3) propose an adoption of BYOD and service providers when outsourcing data to leverage the advantages of the cloud (network, resources, computation, etc.).

The rest of this article is organized as follows: section 'Related work' provides some related works. Section 'Threat models' considers some attack scenarios with real world implementations. We propose the scheme for data outsourcing in section 'Scheme for outsourcing data' and performance analysis in section 'Analysis and evaluation'. Conclusion and limitations will be discussed in section 'Conclusion'.

Related work

Secure meta mediator (SMM)[8] is a core component of secure outsourced database architecture proposed by Ahmed et al. SMM server stands between users and DSP, on one hand authenticates the users and the other hand encrypts and outsources databases to DSP. The role of SMM server is centralizing metadata of all users (databases and queries). While databases are stored at DSP, queries are prepared at SMM and then will be sent to DSP for execution phase. The problem of this model is that once SMM is compromised, attackers will get the information about all user data, and bottleneck could happen if SMM tries to process a large amount of queries at the same time before sending them to DSP. In this case, SMM server is a single point of failure. Our architecture, however, makes a distributed, on-demand approach which users will directly send requests to DSP, instead of going through such centralized server like SMM. Microsoft[9] also mentioned about column encryption. This technique uses symmetric key to encrypt columns in SQL Server, so that database can be more secure when transferred but cannot protect the data (in plain text) when processed.[10]

Threat models

Before proposing the countermeasure, we consider these scenarios:

Query interception

Assume that user wants to send a remote query to DSP to get some returned information. In the case that attacker is able to hijack the session, he could have some knowledge if the query is in plain text and then associates it with the returned result in this session.[11] The fastest way to solve the problem is to encrypt the query, but it leads to another issue since the user have to reveal the key to the DSP, so that they could be able to decrypt the query to perform an operation. Confidentiality can be achieved at both sides through homomorphic encryption, but as we have mentioned above, this mechanism is not applicable since it causes too much overhead.

Storage provider compromised

Databases are stored in storage service provider (SSP) servers, and it is a big trouble if those servers are compromised.[10] Encrypting database is mandatory, but sometime in the progress – database must be decrypted to be queried and the attack could occur at that moment when all data is in plain text. Moreover, exposing all of the sensitive information to the service provider is considered unwise (in case of semi-trusted provider).

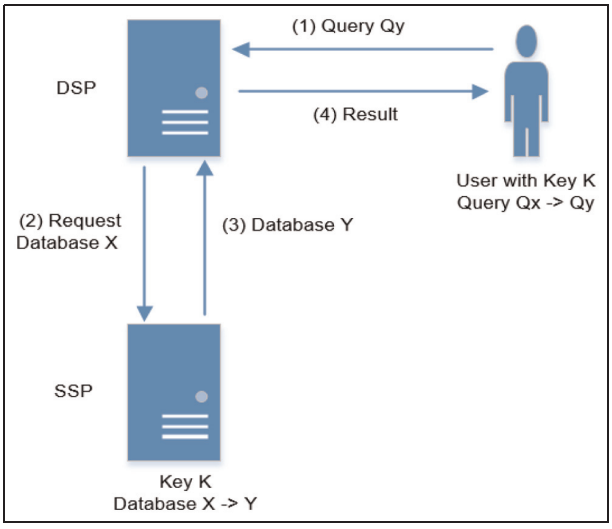

Figure 2. Outsourcing database scheme.

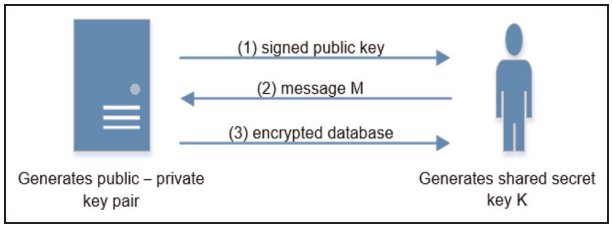

Figure 3. Key exchange between SSP and User. SSP: storage service provider.

Attacks on BYOD devices

To avoid unauthorized access to sensitive data in these devices, corporation may think about applying encrypting or obfuscating techniques carefully. In case employees lose their devices, the company can still assure that the data are somewhat ambiguous to the attackers. They can also take a step by remotely wiping the device.[12,13] Moreover, data loss may cause big impact to the business continuity, the availability of data should be brought in to consideration as well.[7] As the growth of different technologies, such issue is resolved quickly through cloud storage.

Scheme for outsourcing data

We assume there will be the presence of three parties: a user, a DSP and a SSP. SSP can either be an enterprise where the users (employees) work for or a trusted storage provider, while DSP is another service provider that performs computation (query) on databases. DSP is a semi-trusted provider, as it provides computing services for different types of users from different companies but cannot be fully trusted (sensitive information like client password, social security number, etc., must be kept secret)

(In Figure 2: An arrow '→' represents a *transformation*, for example:

- Query Qx → Qy is the transformation of query Qx into query Qy, some parts of the query Qx will be encrypted.
- Database X → Y is the transformation of database X into database Y, some parts of database X will be decrypted. Database Y becomes partially encrypted after the decryption. In database Y, sensitive fields in its tables will still remain encrypted. In SSP, database X is fully encrypted for better privacy and long-term storage.

Suppose X is a fully encrypted database. When the user wants to perform query Qx on database X, the first step is to transform the query Qx into Qy and send it to DSP (1). DSP needs database to perform a query, it then sends a request (for database X) to SSP (2). SSP authenticates DSP on behalf of the user. If successful, SSP will partially decrypt database X and respond to DSP with the corresponding database: Y

(3) In DSP, the query Qy will be applied on the database Y, a result will be sent back to the user in partially encrypted format.

(4) In most cases, the result is a series of rows in the database table. Notice that the property of database Y (which field should be decrypted) is up to the demand of the company and their users. We try to address all problems happened in this scenario and solve them in the next steps.

Set-up shared secret key and move data to SSP

In a set-up phase, the first step is to secure the data provided by the user, we have it encrypted at both ends: on user side and SSP side, using shared secret key. To achieve this, the secret key needs to be distributed at both sides. Firstly, user generates a secret key K. At the same time, SSP needs to generate a public–private key pair. The public key is sent to user (1), user encrypts the secret key K (into message M) using this public key and sends back to the SSP (2). After the SSP receives and decrypts message M using their own private key, SSP could retrieve the secret key K inside. User then uses the secret key K to encrypt whole database and transfer it to SSP (3). Note that public key need to be self-signed by SSP to guarantee non-repudiation, authenticity and integrity of shared secret key establishment process. Additional signing process could be made through public Certificate authorities like Symantec or Comodo (See Figure 3).

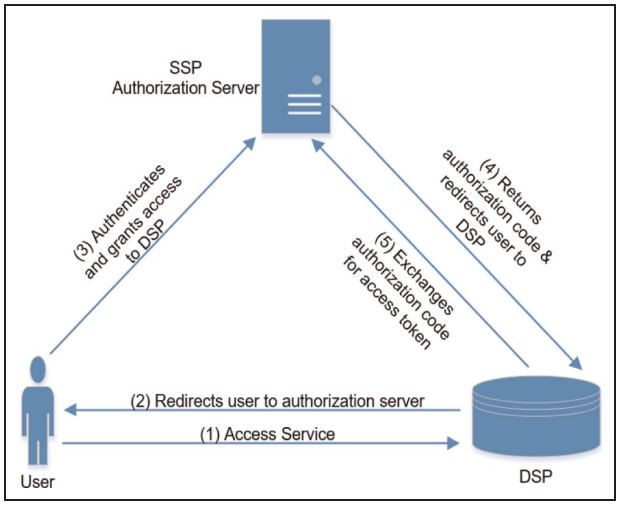

Figure 4. Resource authorization by using OAuth2.0.

Resource authorization

Resource belongs to the user is granted to DSP by access token, which can be deployed using OAuth2.0. For the completeness, we describe the flow of resource authorization in Figure 4.

SSP plays a role as an authorization server and it is also a resource server. To grant permissions for DSP, user accesses DSP (1) and is redirected to SSP (2). At SSP, user authenticates his identity and locates necessary resources that will be granted for DSP, user is redirected back to DSP, along with an authorization code (4). DSP then uses this authorization code to exchange for access token from SSP. From now on, DSP will have permission to access resource in SSP, on behalf of the user. Note that user here is the resource owner, and DSP is the third party service provider which is granted permission to access the resource in SSP (resource server).

Selective encryption

We had a thorough investigation on 1300 database models in different businesses,[14] each database model has variety of fields and their own attributes. For example, in banking database, sensitive fields can be *customer_id, credit_card_number* and *customer_name*. We realized that not all sensitive fields need to be encrypted unless they uniquely identify a customer. In this case, customer_name field can be remained in plain text. Assume that the attacker has this partially encrypted table, he would have no knowledge even if the names of customers are in plain text.

Additionally, we leave the decision to the user to decide which fields should be used for future queries and which fields could be left in unencrypted format. Moreover, that fields that need algebraic calculation are not sensitive in the sense that they do not reveal the identity of the identifier (person or thing in the database). For example, the balance of a bank account does not have much meaning unless the name field and other fields like birthdate and phone number are revealed (See Figure 5).

Results | Messages

	Customer_id	Customer_Name	Credit_card_number	Credit_card_number_encrypt
1	74112	Jack	2147-4574-8475	0x00433D8201CD98408F1283AEA54FDC21010000000C28ECF...
2	74113	Michael	4574-8475-2147	0x00433D8201CD98408F1283AEA54FDC2101000000CD14668...
3	74114	Peter	2147-8475-4574	0x00433D8201CD98408F1283AEA54FDC21010000000D87B941...
4	74115	Jack	2157-1544-8875	0x00433D8201CD98408F1283AEA54FDC21010000000FEBC04...

	Customer_id	Encrypted Credit Card Number	Decrypted Credit Card Number
1	74112	0x00433D8201CD98408F1283AEA54FDC21010000000C3DBB29...	2147-4574-8475
2	74113	0x00433D8201CD98408F1283AEA54FDC2101000000FDC93D...	4574-8475-2147
3	74114	0x00433D8201CD98408F1283AEA54FDC21010000000E0C1891...	2147-8475-4574
4	74115	0x00433D8201CD98408F1283AEA54FDC21010000000DDCA5B...	2157-1544-8875

Figure 5. Column encryption in a banking database table.

Query transformation

Queries also need to be transformed to be applied on (partially encrypted) databases. To achieve that, we encrypt the fields (column names) appear in a query as well as the 'value'. For example, the matching query can be present like this:

SELECT X, Y, Z FROM table_name WHERE X = value1 AND Y = value2

In this case, X, Y, Z are column names and value1, value2 are the values we want to query. While the syntax of SQL must be preserved, and suppose that X and Y are sensitive fields that need to be kept secret, the query can be transformed into:

SELECT ENC(X), ENC(Y), Z FROM table_name WHERE ENC(X) = ENC(value1) AND ENC(Y) = ENC(value2);

Similarly, other database actions can be achieved in the same way. We will discuss more about CRUD (Create – Read – Update – Delete) in the next section. For the briefness of database background explanation, these actions can be summarized as below:

- Update values in a row of a table:
  UPDATE *table_name*
  SET *column1* = ENC(*value1*), *column2* = (*value2*), …
  WHERE *some_column* = *some_value*;

- Delete a row of a table
  DELETE FROM *table_name*
  WHERE *some_column* = *some_value*;

- Create a table
  CREATE TABLE *table_name*(*column_name1 data_type*(*size*),

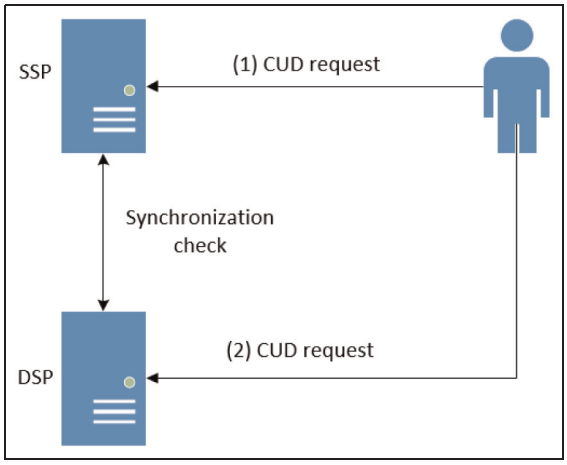

Figure 6. CUD request to both SSP and DSP. SSP: storage service provider; CUD: Create – Update – Delete; DSP: database service provider.

column_name2 data_type(*size*),
column_name3 data_type(*size*),
….);

Advanced actions such as INNER JOIN can also be applicable, as there is no difference between normal database table and partially encrypted database table:

SELECT *column_name(s)*
FROM *table1*
INNER JOIN *table2*
ON *table1.column_name* = *table2.column_name*;

In many cases, values in sensitive fields are not used for operations like adding, multiplying and so on but mostly used for searching. Otherwise, fields that are important (like account_balance, payment_method, etc.) *do not uniquely identify a single person*, that means they are not necessary to be kept secret by encryption so that they can be easily performed any operation in plain text format.

Update data in SSP and DSP

The flow of Update, Create and Delete requests must go to both SSP and DSP to make sure both resources are identical in terms of data integrity. The request flow is described in Figure 6. CUD stands for Create – Update – Delete, Read request is only sent to DSP for data retrieval.

Threat models consideration

Based on evaluating the proposed scheme, we again consider three scenarios mentioned in the previous section:

(1) Query interception: MITM attack may occur at user facing (communication between user and DSP) or SSP facing (communication between SSP and DSP). Refer to Figure 2, in both cases, the query of user and database X, Y are securely encrypted to avoid such attack.
(2) SSP compromised: In all cases, data in storage server remain in fully encrypted (or at least, partially encrypted format). The cost of being compromised could be minimized significantly.
(3) BYOD devices: The concern of data lost and stolen urges the enterprise to secure their data on the devices by encryption. In this scenario, the key must be kept secret by different implementations. Some could be mentioned are data lost prevention (implemented by Google on Android and Apple on iOS). It all up to the users to authenticate with their devices by using fingerprints, PINs or complex passwords. The discussion about these techniques is beyond this article.

Analysis and evaluation

Data

Our evaluation is implemented on MariaDB.[15] The purpose of partially encrypted database and transformed query is to limit the information that could leak to the DSP, while users could still be able to make use most of database services (by issuing transformed query on partially encrypted database), at the same time overhead could be avoided in some level. It complicates the computation and slows down the performance unnecessarily if operation happens on the whole encrypted databases. In this section, we evaluate the efficiency of both partial encryption scheme and full encryption scheme. The use case has been tested on virtual machine with these specifications: 4 GB of RAM, i7-2600 (3.40 GHz) CPU duo core. Measurement units include number database records per second both schemes can process (Performance) and the fraction of database sensitive columns (which is the percentage of sensitive fields in the table, from 0% to 100%).

Our experiment result can be roughly translated into line chart in Figure 7. As the percentage of sensitive fields increases too high, low performance may appear on partial encryption scheme because of the additional time needed to manage each encrypted field, but, in most cases, the performance is much better than full encryption scheme since the time to compute insensitive fields has been terminated. Note that the data must be defined (which parts are secret and which parts are not) to work with our scheme. The experiment result is stable in case of classified data, anomaly only happens due to the fact that there are some specific models that

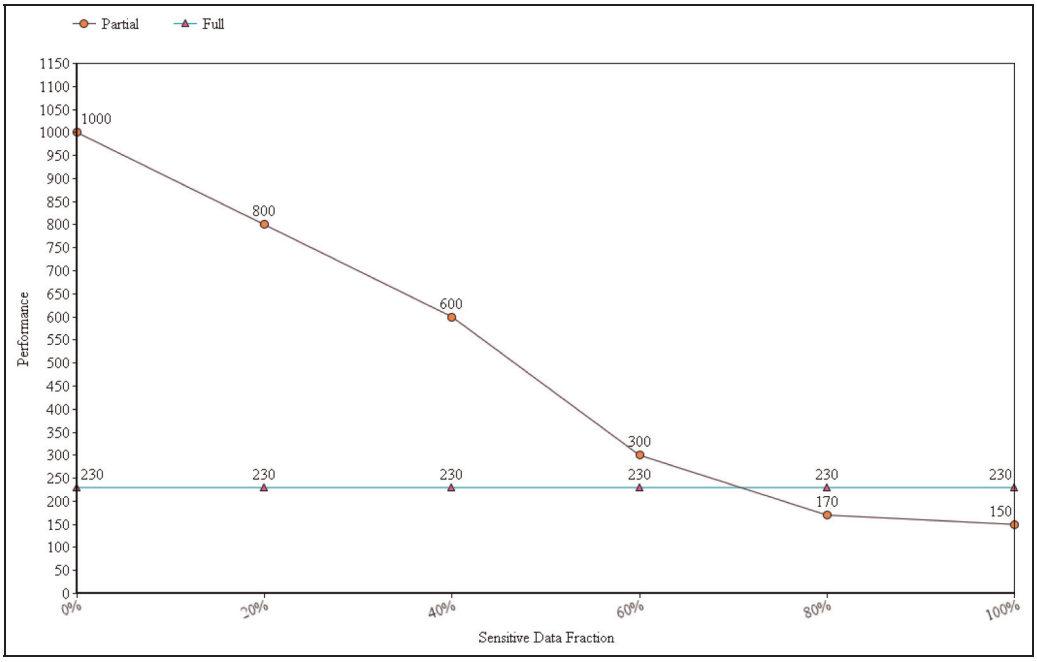

Figure 7. Measurement of partial and full encryption scheme when sensitive portion of data increase.

Table 1. Network benchmark (unit of measurement: Mbit/s).

	Cloud provider	Conventional server
Download	925.41	26.13
Upload	286.67	28.84

Table 2. Encrypting and decrypting benchmark with fully encrypted data.

	Encryption	Decryption
Real	9.041	7.339
User	6.884	4.928
Sys	0.220	0.144

contain too much confidential information, which is mentioned in the next section.

Network measurement

Cloud provider, in addition to its high performance, possesses great network resources compared to conventional servers or personal computer. We measure download/update speed by using SpeedTest API. This advantage help our scheme achieve high throughput of data transfer.

Encrypting and decrypting benchmark

We also take further examination by inspecting the time to encrypt and decrypt single files, one with 200 MB in size and one with 100 MB in size. The work was done on a quad-core server, Intel(R) C(TM) i5-3570 CPU @ 3.40GHz (See Table 1).

Table 3. Encrypting and decrypting benchmark with partially encrypted data.

	Encryption	Decryption
Real	8.059	6.599
User	3.480	2.472
Sys	0.112	0.064

Table 2 and 3 show the time for the encryption and decryption of both schemes. In details of Linux system, 'real' represents the time needed from start to finish of the call, which is the actual elapsed time; 'user' and 'sys' are the amount of CPU time spent in user mode and kernel within the process.

Conclusion

Although the set-up phase is simple, it takes some time to establish the shared secret key at both sides (SSP and user), key distribution on the two parties happens only once so it does not cause so much trouble. Our scheme works well with most common database models. In special models where sensitive fields (that must be encrypted) are gathered into one table, the process of encrypting and decrypting data could slow down, and query will encounter many limitations (because of

computation happens mostly on encrypted fields). The work around for this is to renormalize the database, so that sensitive fields are distributed evenly in all tables. Range queries on encrypted fields are limited for the time being, but we are considering to apply order preserving encryption technique. Moreover, we are improving the process between SSP and DSP when databases change, whenever there is a change in database, only the part was changed will be notify to SSP, the result will be updated in our future works.

In this article, we have proposed an approach to database outsourcing. While computation performed on fully encrypted databases is not a good choice in production, our selective encryption by assessment solve the requirements of confidentiality and performance. Encryption usually limits the accessibility of data as it keeps information in unreadable format, this technique gives the flexibility to queries and computations on server sides, therefore the time trade-off to process data could be affordable while privacy still can be guaranteed. The implementation is currently on relational database management system. In the future, we will extend the architecture to work with NoSQL,[16] first priority is to normalize the text field data in MongoDB to perform selective encryption.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Centre) support program (IITP-2016-H8501-16-1008) supervised by the IITP (Institute for Information & Communications Technology Promotion).

References

1. Mowbray M and Pearson S. A client-based privacy manager for cloud computing. In: *Proceedings of Fourth International ICST Conference on COMmunication System softWAre and middlewaRE (COMSWARE '09)*, New York, USA, Article 5, 8 pages, 2009. DOI: 10.1145/1621890.1621897.
2. Pearson S, Shen Y and Mowbray M. A privacy manager for cloud computing. In: *Proceedings of 1st international conference on Cloud Computing (CloudCom '09)* (eds MG Jaatun, G Zhao and C Rong), 22 November 2009, Vol.5931, pp. 9–106 Berlin, Heidelberg: Springer Verlag.
3. Khandelwal S (2015) *Anthem Data Breach - 6 Things You Need To Know*[Online], http://thehackernews.com/2015/02/anthem-databreach.html (accessed 16 April 2015).
4. OAuth2.0, [Online]. OAuth2.0 Homepage, http://oauth.net/2/
5. Nguyen-Vu L, Park M, Park J, et al. Privacy Enhancement for Data Outsourcing" in *Information and Communication Technology Convergence (ICTC)*, pp. 335-338, Oct. 2015.
6. BYOD Report [Online]. http://www.forbes.com/sites/elenakvochko/2015/08/14/has-byod-become-inevitable (accessed 2 January 2016).
7. Ketel M and Shumate T. Bring your own device: security technologies. In: *Proceedings of the IEEE SoutheastCon*, Florida, 2015.
8. Althneibat AMA, Hasan BEM, Hagazy AFA, et al. Secure outsourced database architecture. *IJCSNS* 2010; 10(5), http://paper.ijcsns.org/07book/201005/20100536.pdf (accessed 2 November 2015).
9. Encrypt a Column of Data [Online]. https://msdn.microsoft.com/enus/library/ms179331.aspx (accessed 2 November 2015).
10. Wang C, Wang Q, Ren K, et al. Ensuring data storage security in Cloud Computing. In: *International Workshop on Quality of Service*, Charleston, SC, 2009, pp. 1-9. https://msdn.microsoft.com/enus/library/ms179331.aspx
11. Man-in-the-middle Attack [Online]. https://www.owasp.org/index.php/Man-in-the-middle_attack (accessed 2 January 2016).
12. iCloud: Erase your device [Online]. https://support.apple.com/kb/PH2701?locale=en_US (accessed 2 January 2016).
13. Removing data from a mobile device [Online]. https://support.google.com/a/answer/173390?hl=en (accessed 2 January 2016).
14. Williams B, et al. *Database Models*[Online], http://www.databaseanswers.org/data_models/index.htm (accessed 16 April 2015).
15. Maria DB. https://mariadb.org (accessed 2 January 2016).
16. NoSQL. http://nosql-database.org (accessed 2 January 2016).
17. Sadeghi AR, Schneider T and Winandy M. Token-based cloud computing: secure outsourcing of data and arbitrary computations with lower latency. In: *Proceedings of the 3rd international conference on Trust and Trustworthy Computing (TRUST'10)* (ed A Acquisti, SW Smith and A-R Sadeghi), 21–23 June 2010, pp. 417–429. Berlin, Heidelberg: Springer-Verlag.
18. Pathak AR and Padmavathi B. Analysis of security techniques applied in database outsourcing. *IJCSIT* 2014; 5(1): 665–670.
19. Sendor J, Lehmann Y, Serme G, et al. Platform-level support for authorization in cloud services with OAuth 2. In: *2014 IEEE international conference on engineering (IC2E)*, 11–14 March 2014, pp. 458–465. Boston, MA: IEEE.
20. Zhifeng X and Xiao Y. Security and Privacy in Cloud Computing. *IEEE Communications Surveys & Tutorials* 2013; 15(2): 843–859.
21. D. Hardt, Ed., *The OAuth 2.0 Authorization Framework* [Online], http://tools.ietf.org/html/rfc6749, (accessed 2 January 2016). ISSN: 2070-1721, Microsoft, 2012.
22. Encryption Function [Online]. https://mariadb.com/kb/en/mariadb/encryption-functions/ (accessed 2 January 2016).