May - June 2020
ISSN: 0193-4120 Page No. 10803 - 10810

A Study on Personalized Privacy Preservation Framework based Ontology Document

Hye-Kyeong Ko[1]

[1]Department of Computer Engineering, Sungkyul University, Anyang, Republic of Korea
ellefgt@sungkyul.ac.kr (Corresponding Author)

Article Info
Volume 83
Page Number: 10803 - 10810
Publication Issue:
May- June 2020

Article History
Article Received*: 19 November 2019*
Revised*: 27 January 2020*
Accepted*: 24 February 2020*
Publication*: 19 May 2020*

Abstract

This paper considers data-publishing, where the publisher needs to specify sensitive information that should be protected. If a document that contains such information is published carelessly, users could infer unauthorized information by exploiting common sense inference. In this paper, we propose a framework that uses encryption for preventing sensitive information from being exposed to unauthorized users. In this framework, sensitive data contained in ontology documents are encrypted separately, and then all encrypted data are moved from their original document to the protected information set and bundled with and encrypted structure index. Our experiments show that the propose framework prevents information leakage via data inference. Moreover, the experiment results show that our method demonstrates better query processing performance than the existing method.

Keywords; *data-publishing, privacy preservation, ontology document, unauthorized information*

I.INTRODUCTION

The amount of data available in digital form is ever-increasing, and almost invariably, the data are now near networks [1], [2]. Recent research on integration systems and peer-to-peer databases has created new ways for diverse groups to share and process data [3-5]. However, in most practical cases, complex constraints of trust and confidentiality exist between these cooperating and competing groups. As a result, in many cases data can be disseminated only when there are no security or confidentiality issues for any potential recipients [6]. OWL (Ontology Web Language) is a semantic web generation language for publishing and sharing ontologies that provide advanced web search, software agents and knowledge management functions on the web [7]. OWL defines propositions accumulated in the inference system as a language defining the web ontology and its related knowledge, and consists of set of classes and properties that can describe the relationship between a class and its members and enable logical inference on facts not defined syntactically [7], [8].Ontology document publishing with security requirements encounters many challenges because users can infer data using such common sense inference. The information leakage to the user of the type of information where a higher security-level authority exists. Therefore, information leakage could allow users to guess their own decryption capabilities compared with those of other users. Both an adversary and general user could obtain a sense of their privileges and those of other users. If the data owner publishes data insecurely, users could infer unauthorized information from the published document by common sense inference (e.g., “all patients in the same ward have the same disease”).

Motivation example. A hospital has ontology documents on its patients, physicians, and departments. Figure 1 illustrates part of an ontology document represented as a tree. A patient has private information such as name, disease, and ward. A physician has information such as name (represented as the pname element) and treat, and a patient is identified by the name information.

Published by: The Mattingley Publishing Co., Inc.

May - June 2020
ISSN: 0193-4120 Page No. 10803 - 10810

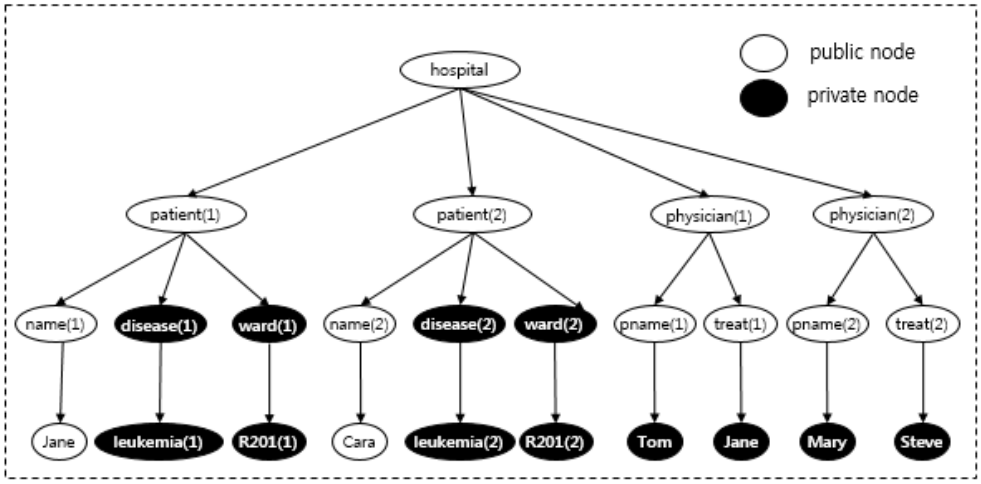

Fig 1. Original ontology document with sensitive information

In thisexample, Tom is treating patient Jane, who has leukaemia and lives in ward R201. The hospital provides documents similar to the form shown in Figure 1. Some data are sensitive and should not be inferred by unauthorized users. In particular, the hospital does not want departments to know the disease of patient Jane, in this example, leukaemia, for her privacy. A naïve approach would hide a shaded leukaemia node to solve this problem. However, if it is well known that patients in the same ward have the same disease, department users could infer that Jane has leukaemia from the information about patient Cara, who also lives in ward R201. Simply hiding the shaded leukaemia node cannot protect all sensitive information because of common sense inference. We can solve the information leakage problem in the following ways: the first is by hiding the leukaemia (1) node of the Jane element so that users cannot infer the disease information. The second is by hiding R201(1) for Jane and R201(2) for Cara so that users cannot use the ward information. The third is by hiding the disease nodes of both Jane and Cara. In addition, users could infer Jane's disease by the related information.

For example, users could guess the patient information in terms of the related branch physician(1)/pname(1)/Tom and physician(1)/treat(1)/Jane in Figure 1. Ontology document publishing with security requirements encounters many challenges because users can infer data using such common sense inference. If we remove sensitive nodes, such as Jane, leukaemia (1), and R201(1) that are related to the physician(1)/treat(1)/Jane branch, some related information from the sensitive nodes could still remain. For example, even after removing those sensitive nodes, the elements disease(1), ward(1), disease(2), ward(2), treat(1), and treat(2) still exist. A user could be aware of the existence of that part in the document that he/she is not allowed to access; this is information leakage to the user of the type of information where a higher security-level authority exists. Therefore, information leakage could allow users to guess their own decryption capabilities compared with those of other users. Both an adversary and general user could obtain a sense of their privileges and those of other users.

In this paper, we propose a framework for protecting sensitive information in published ontology documents from unauthorized users. In the framework, the data owner publishes an ontology document that is partially encrypted according to access rights. Each sensitive node is encrypted separately and all encrypted information is removed from its original document to the protected information set and bundled with the encrypted structure index that informs us of the structure information of the original document. The remainder of this paper is organized as follows. Section 2 surveys related work. Section 3 presents the proposed framework for secure data publishing and proves that our framework does not allow information leakage. Section 4 presents our experiment results, and Section 5 concludes the paper.

Published by: The Mattingley Publishing Co., Inc.

May - June 2020
ISSN: 0193-4120 Page No. 10803 - 10810

II. RELATED WORKS

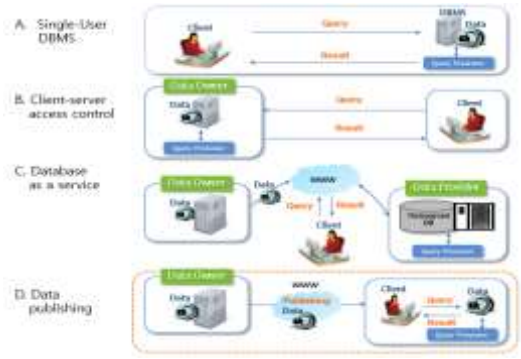

Fig 2. Figure2. Different architecture of database security based on trust domains [4]

Database security has been studied extensively in the past [8-10]. Recently, works have proposed methods for data publishing [6], [10], and [11]. Miklau and Suciu [6] showed a good approach for classifying different architectures for related studies based on trust domains. Figure 2 shows the four different database security architectures they identified. Architecture A has a single trust domain. Thus there are no security issues. Architecture B shows the client-server access control model. In architecture B, the server owns the data and controls the query execution. However, the server does not trust the client that submits queries [12-15]. Client-server applications and many web-based applications use this architecture. Much of the work in this architecture has focused on how to process user queries without disclosing protected data [9], [16], and [17]. In architecture C, the client owns the data and also issues queries. However, the client does not trust the server. In this case, the client would pay a trusted party to store data and execute queries Architecture D is for data publishing. In the data publishing architecture, once the data owner has published data such data can be downloaded, copied, disseminated, and redistributed. Both query generation and query processing are performed in the trust domains that are different from the domains for data [6]. Several security models have been proposed for data security [18], [19]. There are two traditional approaches for controlling access to data. The first maintains data on a secure server that authenticates users and enforces access policies, without publishing data [18]. The other publishes multiple views of data, one for each user [19]. However, these approaches have several limitations. First, the number of views can become very large. Second, users cannot further publish the data that they downloaded from the owner. Miklau and Suciu's work [6] contains an encryption-based approach to access control on publishing ontology documents. This method is essentially identical to publishing, with the exception of the specification for access control policies. In order to specify such access control policies, this method uses an extension of XQuery [20] to define sensitive data in a publishing document.

III.PROPOSED FRAMEWORK FOR PERSONALIZED PRIVACY PRESERVATION

Data publishing, once the data owner has published data, he/she loses control over the data [6]. Different users might have different viewing rights for different parts of the same document. The main problem with data publishing is that a user might infer the unauthorized sensitive information. Therefore, we propose a framework for secure data publishing of ontology documents that users encryption techniques. The key idea is to use different secret keys for encrypting different parts of an ontology document-based on the specified access control policies. The framework components are shown in Figure 3, where a partially encrypted ontology document is published on the Internet. Once data are published, the data owner does not have control over who downloads and processes the data, and thus published data should be properly encrypted to enforce access control policies. In the proposed framework, each user is required to register during the registration phase.

The data owner starts by annotating the ontology document according to the access rights of the users. In this registration phase, the authorization record returns specific information, called keys, to the user, such keys, are necessary to decrypt the relevant parts of the ontology source according to the user's access rights. Authorized users can access the data, depending on the keys they possess. In our framework, users do not need to decrypt the entire document; they can selectively access those parts of the published document that are predetermined by

Published by: The Mattingley Publishing Co., Inc.

May - June 2020
ISSN: 0193-4120 Page No. 10803 - 10810

the policy evaluator shown in Figure 3. In following paragraphs, we introduce the basic components of the policy evaluator.

Authorization Record

This is the access information recorded for authorizing transactions. The right of the users to access sensitive nodes is represented by XPath [21]. For example, some rights related to physician can be represented by //patient/name, //patient/ward.

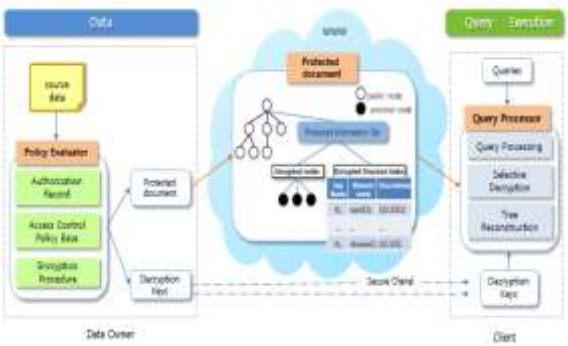

Fig 3. Example of proposed framework for personalized privacy preservation

Our framework also focuses on browsing privileges, that is, privileges for viewing information. Three different browsing privileges are supported: view, navigate, and browse-all. The view privilege allows a user to read all public documents. In contrast, the navigate privilege allows a user to view authorized document [22]. The browse-all privilege subsumes navigate and view privileges.

Key assignment

In our framework, each node is encrypted with a unique node key by the encrypted procedure shown in Figure 2. In order to send node keys to users, the node keys are grouped into sets of node keys. Different views of the document can be defined by choosing the appropriate sets of node keys.

3.1 Protected Information Set

In our framework, the system encrypts the sensitive parts of an ontology document. Encrypting such sensitive parts means that the selected parts of the original document structure are hidden from unauthorized users. The key idea of our framework is to prune sensitive nodes from the original document tree encrypt each sensitive node individually. Sensitive nodes are selected according to the authorization records, and moved to the protected information set. Then, the sensitive nodes are encrypted separately by the use of keys. These encrypted nodes are stored in the protected information set that consists of the encrypted nodes and encrypted structure index. After, decryption each sensitive node returns to its correct position in the original document using the encryption structure index.

The protected informationset (sensitive nodes) and public nodes are published to multiple users. Those nodes that are not secure are called “public nodes.” Figure 4 presents the example of an ontology document with sensitive nodes. The white and black circles denote the public and sensitive nodes, respectively. The sensitive nodes are only accessible to those users who won the matching keys. The system first selects the sensitive nodes according to the authorization record. Then, it labels the original ontology document. The labelling step has two cases: one labels the public nodes and the other the sensitive nodes. After labelling, all the sensitive nodes are pruned from the original document. Figure 4 shows an example of pruning the sensitive nodes (ward(1), disease(1), and so on). After pruning, the document includes only the public nodes. In the encryption step, each sensitive node is symmetrically encrypted under its node key, and the encrypted structure index is also encrypted. In our framework, the document is partitioned into public and sensitive nodes in order to support secure data publishing. The principal problem with query processing is to effectively find the positions of decrypted sensitive nodes. After encryption, all sensitive nodes are pruned from the original ontology document, and therefore the positions of these nodes in the document must be remembered. A secure and efficient labelling scheme is required for protecting the structure information of the sensitive nodes and for representing their positions. In this paper, we extend IBSL [23], to protected IBSL, which takes advantage of lexicographical order of binary strings. When sensitive nodes are pruned from the original document, an unauthorized user should not be able to infer other structure information using the labels of public nodes. Protected IBSL can hide the label values, assigned to sensitive nodes, and its labelling effectively separates the public nodes.The protected information set consists of the encrypted nodes and encrypted structure index. The pruning step prunes the sensitive nodes from the document while keeping the public nodes in the document without

Published by: The Mattingley Publishing Co., Inc.

May - June 2020
ISSN: 0193-4120 Page No. 10803 - 10810

label information. When it comes to encrypting the ontology document without label information.

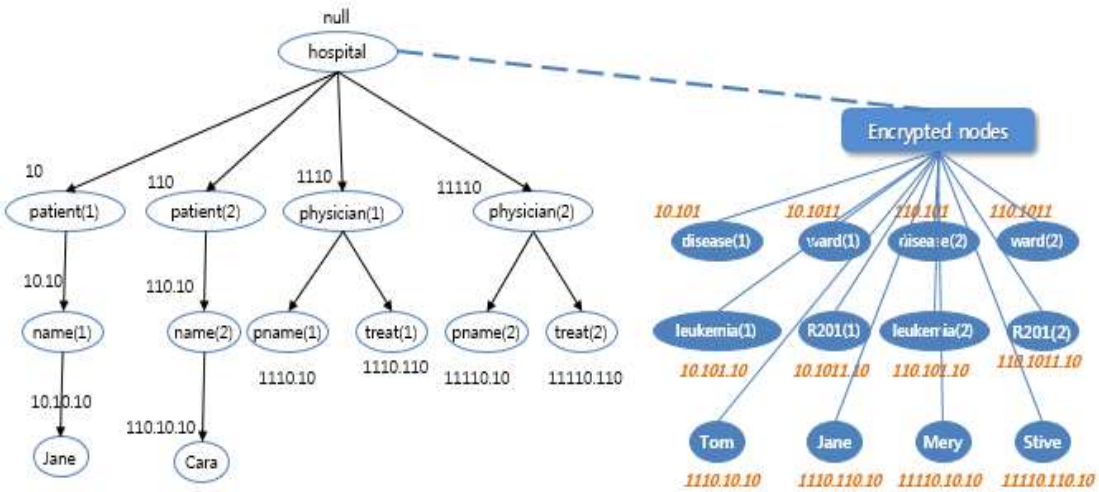

Figure 4. Example of a protected ontology document

When it comes to encrypting the ontology document, if the public nodes have labels, an adversary can guess the positions of the encrypted nodes through the label information and structure information of the public nodes. In order to protect sensitive information and for efficient query processing, the proposed framework utilizes the encrypted structure index, which contains the structure information of the original ontology document for identifying the positions of the decrypted sensitive nodes.

3.2Query Processing for Protected Ontology Document

The query processing algorithm takes a public document, the protected information set, a query, and the keys as input and places the decrypted nodes into the appropriate locations. We can place the child nodes of a decrypted node using the algorithm 1, which inserts the decrypted node into the pubic document. This demands for the public document to not be labelled, and for the structure information of the decrypt node to be known. Algorithm 1 identifies the parent node within the decrypted document and already has a child node in the decrypted node and the parent node,the algorithm 1 determines whether some of the decrypted node's siblings that should also be turned into children of the decrypted node.

Algorithm 1. Query processing

Input: (1) a public document
(2) the protected information set
(3) a query
(4) the keys that the user owns
Output: query results
Begin
01: process the query against the encrypted structure index
02: decrypt the encrypted nodes and encrypted structure
index using the keys
03: find those elements that satisfy the query against the
decrypted ontology data
04: place the elements in the public document
05: select an element and determine whether the label of the
element has some relationships with the labels stored in
the encrypted structure index
06: if it has a parent and the parent has a child in the public
document, select the first child of the parent node.
07: compare the label of the element and that of the first
child
08: append the element as a sibling of the first child

Published by: The Mattingley Publishing Co., Inc.

May - June 2020
ISSN: 0193-4120 Page No. 10803 - 10810

depending on the comparison result
09: else if it has a parent, but the parent does not have a child
10: append the element as a child to the parent
11: repeat the comparison in Lines 5 to 10 until all the
elements found in Line 3 are placed

If the parent has no child, the decrypted node is addeddirectly to the parent node. If the parent has a child, it should be determined whether some of the current children of the parent node should bechild of the decrypted node and where to insert the decrypted node in the list of child. If doing this, Algorithm 1 iterates over the children of the parent. If Algorithm 1 finds the first child, the decrypted node is inserted between the first and last child. If Algorithm 1 cannot find the first child, the decrypted node is added to the parent node.

IV.PERFORMANCE EVALUATION

We conducted experiments to evaluate our scheme and compare its performance with that of Miklau's scheme [6]. In the experiments, the data owner encrypts an ontology document and publishes the public document and protected ontology document to the user. Our scheme and Miklau's scheme were implemented using Java of the Advanced Encryption Standard (AES) [24] with 128-bit keys. The experiments were performed on a 3.20 GHz Pentium processor with 3GB of RAM that runs Windows 7. In selecting the nodes to be encrypted, XPath [21] was used. We conducted the experiments 20 times in order to obtain small confidence intervals.

Query processing time for protected documents

The number of nodes to be compared was measured to observe the relationship between nodes and sizes in the encrypted structureindex. Table 1 presents XPath expressions used to represent the nodes to be encrypted.

Table 1. Encrypted nodes

Encrypted node type	XPath expression
EN1	//Africa/*
EN2	//item/description/parlist[/listitem/text]
EN3	//item//parlist[.//mailbox]//text
EN4	//parlist/list/item/text
EN5	//*/text

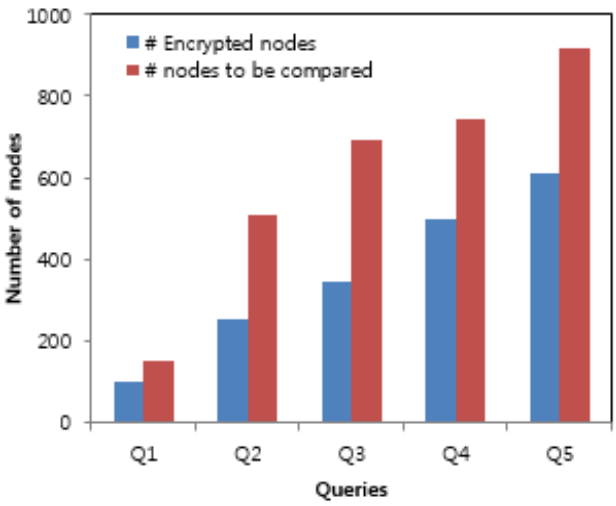

Fig.5. Query Processing time

In the experiment, we measured the decryption time of the encrypted nodes according to the queries. The encrypted node type listed in Table 1 is utilized as the queries. In order to compare the query processing time of the proposed scheme with that of Miklau's scheme [6], the number of encrypted nodes was observed when searching for a position in the ontology document, and the query processing time was measured. The results presented in Figure 5 indicate that the proposed scheme outperforms Miklau's scheme with respect to all the queries of 1.1 MB document. The number of nodes to be compared is shown in Figure 5. The results demonstrate that the number of nodes to be encrypted is related to the number of compared nodes, whereas identifying the position and query processing time is affected. According to the queries expressed by the XPath expression, the number of encrypted nodes and that of the nodes to be compared are different.

In the proposed scheme, labels are not compared to other nodes. This is because the proposed scheme labels the child nodes by extending the parent's label to represent the structural information of the ontology document. The results show that our scheme considerably outperforms Miklau's scheme on all the queries for the document.

With regard to the time required to replace node's position in the protected information set be decrypted in the public document. This is because protected IBSL can identify the relationship among nodes. In order to facilitate the determination of the relationship among nodes, in our scheme, the node

Published by: The Mattingley Publishing Co., Inc.

in a document are labelled such that the relationship between any two nodes can be established quickly. Hence, protected IBSL is crucial for efficient access control and fast query processing.

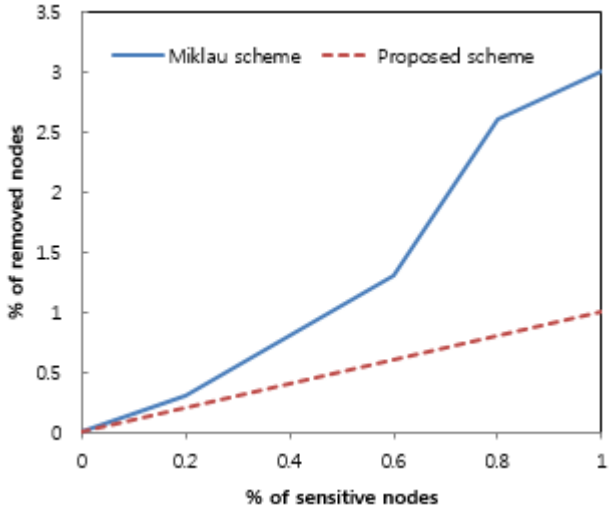

Fig.6. Comparison of number of nodes removed to prevent information leakage

Removing nodes to prevent information leakage
In this experiment, we measured how many nodes were removed to prevent information leakage. Figure 6 shows the result of removing nodes to prevent information leakage for 1.1 MB document. For the child/descendant constraints, when we traced their inference process, we had to choose a parent or ancestor to remove, and this could remove many child/descendant nodes. In Figure 6, the number of removed nodes increases with an increase in the number of sensitive nodes. When comparing the number of nodes removed with our scheme with those removed with Miklau's scheme, we can see that when the number of sensitive nodes increases, our scheme removes only 1/2 to 1/3 of the nodes compared with Miklau's scheme.

IV.CONCLUSIONS

Data owners publish data insecurely, and users can infer unauthorized information from the published document by common sense. Previous works in data publishing have considered the specification of access control policies and efficient query processing against encrypted document. In this paper, we proposed a novel framework that protects sensitive published information by encryption technique. The framework employs a protected information set and encrypted structure index to publish ontology documents without information leakage via common sense inference. In this framework, each unit of sensitive information is encrypted separately, and all encrypted information is moved from the original document to the protected information set and bundled with the encrypted structure index. For secure data publishing, the encrypted structure index summarizes the structure information of the original document. The encrypted elements to be contained in the query results are restored using the encrypted structure index. The query processing time of our scheme is 3~4times more efficient than Miklau's scheme. Our future work will focus on group key management for managing secure multi-level access control.

REFERENCES

[1] K. Lee, M. Kang, Y. Jung. (2018). Development of voice guide service for pharmaceutical information based on ontology, 7(4), 66-74.

[2] J. Choi, M. Koo. (2017). A study on the offering of the latest film information using XML parser. The Journal of Convergence on Culture Technology, 3(1), 19-23.

[3] Z. G. Ives, A. Y. Lyer, J. Madhavan, R. Pottinger, S. Saroiu, I. Tatarinov, S. Betzler, Q. Chen, E. Jaslikowska, J. Su, W. Yeung. (2003). Self-organizing data sharing communities with SAGRES, In SIGMOD Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (p.582).

[4] W. S. Ng, B. C. Ooi, K. L. Tan, A. Zhou. (2003). Peerdb: A p2p-based system for distributed data sharing. In Proceedings of the 19th International Conference on Data Engineering (pp. 575-586).

[5] M. Stonebraker, P. M. Aoki, W. Litwin. (1996). A wide-area distributed database system. VLDB Journal, 5(1), 48-63.

May - June 2020
ISSN: 0193-4120 Page No. 10803 - 10810

[6] G. Miklau, D. Sucu (2003). Controlling access to published data using cryptography. In Proceedings of the 29th International Conference on Very Large Data Bases (pp. 898-909).

[7] A. Algergawy, E. Schallehan, and G. Saake (2008). A sequence-based ontology matching approach. In proceedings of 18thEuropean Conference on Airificial Intelligence Workshops (pp. 26-30).

[8] Noy, N. F. (2004). Semantic integration: A survey of ontology-based approaches. SIGMOD Record, 33(4), 65-70.

[9] A. Brodskyand, C. Farkas, S. Jajodia. (2000). Secure databases: constraints, inference channels, and monitoring disclosures. IEEE Transaction on Knowledge and Data Engineering, 12(6), 900-919.

[10] S. Castano, M. G Fugini, G. Martella, P. Samarati. (1995). Database Security. Addison-Wesley & ACM Press.

[11] E. Bertino, S. Castano, E. Ferrari, M. Mesiti. (1999). In Proceedings of the 2nd International Workshop on Web Information and Data Management (pp. 22-27).

[12] J. G. Lee, H. Y. Whang. (2006). Secure query processing against encrypted XML data using query-aware decryption. Information Sciences, 176(13), 1928-1947.

[13] J. Kim. (2017). Study on semantic web for multidimensional data. The Journal of the Institute of Internet, Broadcasting and Communication, 17(3), 121-127.

[14] Y. Choi (2018). Design and implementation of video file structure analysis tool for detecting manipulated video contents. International Journal of Internet, Broadcasting and Communication, 10(3), 128-135.

[15] W. Kang. (2018). An extended access control with uncertain context. International Journal of Advanced Smart convergence, 7(4), 66-74.

[16] X. Yang, C. Li. (2004). Secure XML publishing without information leakage in the presence of data inference. In Proceedings of the 30th International Conference on Very Large Data Bases (pp. 96-107).

[17] S. Dawson, S. D. C. D. Vimercati, P. Lincoln. P. Samarati. (1999). In Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (pp. 114-125).

[18] E. Damiani, S. D. c. D. Vimercati, s. Paraboschi, P. Samarati. (2001). A fine-grained access control system for XML documents. ACM Transaction on Information and System Security, 5(2), 169-202.

[19] T. Yu, D. Srivastava, L. V. S. Lakshmanan, H. V. Jagadish. (2002). In Proceedings of the 28th International Conference on Very Large Data Bases (pp. 478-489).

[20] S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. R. J. Simeon. (2007). XQuery 1.0. http://www.w3.org/TR/2007/REC-xquery-20070123.

[21] J. Clark, S. DeRose. (1999). XML path language (XPath). http://www.w3.org/TR/1999/REC-xpath-19991116.

[22] H. Ko, S. Lee. (2007). On the efficiency of secure XML broadcasting. Information Sciences, 177(24), 5505-5521.

[23] H. Ko, S. Lee. (2010). A binary string approach for updates in dynamic ordered XML data. IEEE Transactions on Knowledge and Data Engineering, 22(4), 602-607.

[24] J. Daemen, V. Rijmen, The block cipher Rijndael. (1998). In Proceedings of the International Conference on Smart Card Research and Applications (pp. 277-284).

Published by: The Mattingley Publishing Co., Inc.