

A Historical Probability based Noise Generation Strategy for Privacy Protection in Cloud Computing

Gaofeng Zhang, Yun Yang

Faculty of Information and Communication Technologies
Swinburne University of Technology
Hawthorn, Melbourne, Australia 3122
{gzhang, yyang}@swin.edu.au

Jinjun Chen

Faculty of Engineering and Information Technology
University of Technology Sydney
PO Box 123, Broadway, NSW 2007, Australia
Jinjun.Chen@uts.edu.au

Abstract: Cloud computing promises an open environment where customers can deploy IT services in pay-as-you-go fashion while saving huge capital investment in their own IT infrastructure. Due to the openness, various malicious service providers can exist. Such service providers may record service requests from a customer and then collectively deduce the customer private information. Therefore, customers need to take certain actions to protect their privacy. Obfuscation with noise injection, that mixes noise service requests with real customer service requests so that service providers will be confused about which requests are real ones, is an effective approach in this regard if those request occurrence probabilities are about the same. However, current obfuscation with noise injection uses random noise requests. Due to the randomness it needs a large number of noise requests to hide the real ones so that all of their occurrence probabilities are about the same, i.e. service providers would be confused. In pay-as-you-go cloud environment, a noise request will cost the same as a real request. Hence, with the same level of confusion, i.e. customer privacy protection, the number of noise requests should be kept as few as possible. Therefore in this paper we develop a novel historical probability based noise generation strategy. Our strategy generates noise requests based on their historical occurrence probability so that all requests including noise and real ones can reach about the same occurrence probability, and then service providers would not be able to distinguish in between. Our strategy can significantly reduce the number of noise requests over the random strategy, by more than 90% as demonstrated by simulation evaluation.

Keywords: Cloud computing, Privacy protection, Noise generation strategy, Historical probability

1. Introduction

Cloud computing is positioning itself as a new and promising platform for delivering information infrastructure and resources as IT services [1, 2]. Customers can then access these services to execute their business jobs in a pay-as-you-go fashion while saving huge capital investment in their own IT infrastructure [3]. However, Customers often have concerns about whether their privacy can be protected when facilitating services in the cloud, since they do not have much control inside the cloud [4, 5]. Correspondingly, privacy protection has become a critical issue and one of most concerning. Without it, customers may eventually lose the confidence in and desire to deploy cloud computing in practice [6, 7].

Although there are many organisations, such as the banking and immigration sectors which operate under various strict regulations and policies to protect customers' privacy, a large number of malicious and unknown service providers can exist in the open cloud environment. These service providers could record service requests from a customer and then collectively deduce the customer privacy information. Therefore, customers need to take certain actions to protect their privacy without the need for the service providers' cooperation. In this respect, obfuscation with noise injection is an effective approach that mixes noise service requests with real customers' service requests so that service providers would be confused about which requests are real ones if all request occurrence probabilities are about the same. For example, the frequent occurrence of service requests from a customer about the weather on "Gold Coast" tomorrow may help the weather service provider to guess that the customer might travel to "Gold Coast" tomorrow. However, if we inject some noise requests like "Perth" or "Darwin" into the "Gold Coast" request queue and make sure that their occurrence probabilities are about the same, it would be very difficult for service providers to deduce that "Gold Coast" is the real request.

Current obfuscation with noise injection uses random noise requests. Due to the randomness, we need a large number of noise requests to dilute the real ones so that all of their occurrence probabilities are about the same, i.e. service providers would not be able to distinguish in between. In a pay-as-you-go cloud environment, a noise request would cost the same as its real counterpart. Therefore, with a same level of confusion, i.e. for effective customer privacy protection, the number of noise requests should be kept as few as possible. Hence, in this paper we develop a novel historical probability based noise generation strategy (HPNGS).

Our strategy generates noise requests based on their previous occurrence probabilities, i.e. historical probabilities, so that all requests including those noise ones and real ones can reach about the same occurrence probabilities. This would confuse service providers. The simulation evaluation demonstrates that our strategy can significantly reduce the number of noise requests by over 90% compared with the random one.

The remainder of the paper is organised as follows. In Section 2, we detail related work and conduct problem analysis. In Section 3, we present our historical probability based noise generation strategy (HPNGS). In Section 4, we perform a simulation to demonstrate that our noise generation strategy can significantly reduce the number of noise requests than the random one. Finally in Section 5, we conclude our contributions and outline future work.

2. Related Work and Problem Analysis

We start with a general overview of the literature in this area in Section 2.1. Then in Section 2.2, we focus on the problem analysis and compare our work with others.

2.1 Related work

There are many research papers about privacy issues in cloud computing, privacy-preserving data mining, privacy information retrieval, anonymity browsing and searching, and obfuscation with noise injection. We review them in this section.

More and more researchers are starting to produce and/or have produced remarkable research on privacy protection related to the cloud environment. M. Smit *et al.* [8] describe a whole framework and methodology for managing privacy policy of an enterprise. L. Yan *et al.* [9] use hierarchical identity-based cryptography to realise mutual authentication in inter-clouds. X. Huang *et al.* [10] discuss privacy protection in value-added context-aware cloud. M. Hart *et al.* [11] develop privacy control in the Blogs situation by tags. F. Kerschbaum [12] presents the protocol by circuit encryption on a special business model with one service provider and many customers. K. Simoons *et al.* [13] present a biometric encryption system in the privacy protection of biometric search area. P. Golle *et al.* [14] present the trust problem between pollsters and respondents and use a “privacy-bond” to keep mutual trustworthy.

Privacy-Preserving Data Mining (PPDM) reveals a view of privacy leakage in the minutiae [15], and the bibliography [16] is a good introduction to this area. To protect privacy, Evfimievski *et al.* [17] use a randomisation operator to discuss the problem of association rule mining. Many privacy preserving methods focus on a topic small dataset and a particular family of data mining algorithms, but L. Liu *et al.* [18] inspect and design some approaches in the real world. Besides, K-anonymity [19] is a promising path to cope with this searching privacy preserving situation, particularly due to large scale and uneven distributions.

Different from PPDM, Privacy Information Retrieval (PIR) utilises another approach to keep privacy secret, which mainly prevents database operators from knowing users’ interested records. Based on information theory, B. Chor *et al.* [20] have a conclusion that, to get a perfect protection, a user has to query all entries in database when dealing with a single server framework. A. Beimel *et al.* [21, 22] and I. Goldberg *et al.* [23] apply information theories to dig deeply in PIR. Besides, E. Kushilevitz *et al.* [24] and C. Cachin *et al.* [25] allow servers with polynomial-time computational capabilities in most cases. S. Yinan *et al.* [26] use extended attribute based encryption by hierarchical relations.

Proxy or anonymity networks to protect users’ privacy have also been widely discussed [27]. The major goal is to keep anonymity or “invisible” in a complex or “danger” network condition. Onion routing [28] and its successor TOR [29] provide a more sophisticated privacy protection scenario, making it difficult for attackers to trace users via network traffic analysis. O. Berthold *et al.* [30] propose a comprehensive framework for generating anonymous real-time Internet access. A. Narayanan *et al.* [31] present a framework for analysing privacy and anonymity in social networks, and develop a topology-based re-identification algorithm targeting anonymous social network graphs. K. Hawkey *et al.* [32] research identity anonymity to protect privacy in Web 2.0.

Obfuscation with noise injection is another widely adopted method for protecting information privacy. It facilitates information theories to discover the characters of information [33]. S. Ye *et al.* [34] describe noise injection in search privacy protection by formulating noise injection as a mutual information minimisation problem. U. Hengartner *et al.* [35] design encoding policies as digital certificates and present an algorithm for the discovery of distributed certificates in access control to location privacy. K. Fukushima *et al.* [36] discuss obfuscation mechanism in mobile condition. G. Zhang *et al.* [37] discuss a trust mechanism in noise obfuscation scenarios in the cloud.

2.2 Problem analysis

As analysed in Section 1, various malicious service providers can exist in cloud computing environment. They may record customer’s service requests and collectively deduce customer private information. Therefore, customers need to take certain actions to protect their privacy without the need of cooperation from service providers. This is the scenario that we are addressing in this paper.

Existing research on privacy protection in cloud computing mainly focuses on service provider side and sometimes needs cooperation between customer side and service provider side such as encryption and decryption. Hence, they cannot be directly applied to the scenario stated above.

PPDM is not a good choice to address the scenario either because it is out of customers' control, hence not suitable for customer privacy protection. PIR is mainly working at service provider side, hence has the similar problem. Anonymity or proxy networks need service provider's cooperation to enable such access. Besides, their IP source can be traced in the end especially for normal common customers.

Obfuscation with noise injection seems naturally effective in protecting customer privacy because it injects noise service requests to confuse service providers and does not need cooperation from service providers. However, current obfuscation with noise injection generates noise requests randomly and due to the randomness, we need a large number of noise requests to mix with the real ones so that all of their occurrence probabilities are about the same, i.e. service providers would not be able to distinguish in between. As stated in Section 1, in a pay-as-you-go cloud environment, a noise request will cost the same as a real request. Therefore, with the same level of confusion, i.e. for effective customer privacy protection, the number of noise requests should be kept as few as possible. For this purpose, we develop a novel historical probability based noise generation strategy denoted as HPNGS that can significantly reduce the number of noise requests. We discuss HPNGS in detail in the following section.

3. Historical Probability based Noise Generation Strategy

We first discuss noise injection in Section 3.1. Then in Section 3.2, we investigate historical probability based noise generation. Finally in Section 3.3, we present our HPNGS.

3.1 Noise Injection

3.1.1 Basic noise injection model

Based on [34], we describe the basic noise injection model in Figure 1.

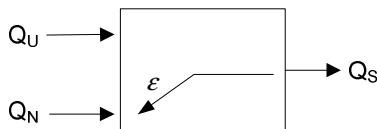


Figure 1. Basic noise injection model

Denotations in Figure 1 are listed as follows:

Q_U : a queue of customer's real service requests which are to be protected.

Q_N : a queue of noise service requests which are to be injected into Q_U .

Q_S : a queue of final service requests, which composes of Q_U and Q_N .

Q : a common set of values of Q_U , Q_S and Q_N . And $Q = \{q_1, q_2, \dots, q_i, \dots, q_n\}$. As we can see, a noise service request happens to be the same as a real one in Q_S for service providers.

ϵ : probability for injecting Q_N into Q_U , and $\epsilon \in [0,1]$. We call it noise injection intensity.

The overall working process of the model is to inject Q_N into Q_U based on ϵ so that we can get Q_S . Q_U is generated by the customer. Q_N is generated randomly independent of Q_U . ϵ plays a role like a switch. The probability for Q_N to become Q_S is ϵ while that for Q_U to become Q_S is $1 - \epsilon$.

3.1.2 Improved noise injection model

In the basic model, noise requests are generated randomly. As stated in Section 1 and Section 2.2, to protect the customer privacy we need to make the occurrence probabilities of all noise and real service requests about the same. To achieve this, random noise generation would need a large number of noise service requests because the same noise request may occur frequently. It then needs more injections of other noise service requests to balance the probability difference. Our idea is to use historical occurrence probabilities of various service requests. Simply speaking, those noise requests with higher historical occurrence probabilities are used less and vice versa. By this means, we can effectively dilute the occurrence of noise requests in a balanced fashion so that we can achieve about the same occurrence probabilities with fewer noise requests. As such, we improve the basic noise injection model with the one described in Figure 2.

The dash line in Figure 2 means that we utilise historical occurrence probabilities of service requests to generate new noise service requests.

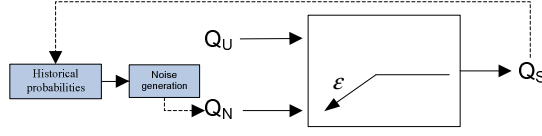


Figure 2. Improved noise injection model

The overall working process of the model is also to inject Q_N into Q_U based on ε so that we can get Q_S . The difference to the basic model is that we need to record historical probabilities of all service requests and then apply them to the noise generation. The model can be described as follows. Suppose q_i is a value of Q and $P(Q_U = q_i)$, $P(Q_N = q_i)$ and $P(Q_S = q_i)$ are probabilities of q_i in Q_U , Q_N and Q_S respectively. $P(Q = q_i)$ is the probability of q_i in past Q_S . Hence, q_i has appeared in Q_S with a probability of $P(Q = q_i)$. It appears in Q_N with a probability of $P(Q_N = q_i)$. To protect customer privacy, we need to achieve that $\forall i$, all $P(Q_S = q_i)$ are about the same. Therefore, if $P(Q = q_i)$ has a high value, then q_i will not be taken as noise this time so that $P(Q_N = q_i)$ will have a smaller value, and vice versa. This is the general process of generating noise requests based on historical probabilities of service requests. The details are presented in the next section.

3.2 Historical probability based noise generation

To achieve the outcome that if $P(Q = q_i)$ has a high value and hence q_i will not be taken as noise, we have designed the following strategy to generate our noise based on historical probabilities. Firstly, we should record all noise generation probabilities as historical probabilities, and then discuss the noise injection intensity ε .

3.2.1 Noise generation probabilities

To achieve noise generation probabilities $P(Q_N = q_i), \forall i$, we should record all $P(Q = q_i), \forall i$ in advance. In all of the probabilities of $P(Q = q_i)$, the highest one is $M = \text{MAX}\{P(Q = q_i), \forall i\}$. It is an accumulative record from previous $P(Q = q_i)$. These are historical probabilities. Accordingly, we get the noise generation probabilities by equation (1) below.

$$P(Q_N = q_i) = \frac{M - P(Q = q_i)}{\sum_j \{M - P(Q = q_j)\}}, \forall i \quad (1)$$

Equation (1) is based on the following idea: if $P(Q = q_i)$ has a high value, then the numerator must have a low value, and vice versa. $P(Q = q_i)$ is a historical probability and is within the range of $[0, 1]$. $P(Q_N = q_i), \forall i$ are also within the range of $[0, 1]$. It is the key of the whole strategy. The denominator is to ensure equation (2) below.

$$\sum_i P(Q_N = q_i) = 1 \quad (2)$$

Because of $\sum_j [M - P(Q = q_j)] = n \times M - 1 \neq 1$, the denominator is necessary to ensure that equation (2) is kept.

The generation probability of random noise is $P(Q_N = q_i) = \frac{1}{n}, \forall i$. With our strategy, we have $P(Q_N = q_i) = \frac{M - P(Q = q_i)}{n \times M - 1}, \forall i$. This is an improvement that saves the number of noise requests. We will further demonstrate this in Section 4 by comparison simulation.

3.2.2 Noise injection intensity

Equation (1) can express noise generation probabilities. According to the improved noise injection model in Section 3.1.2, ε is a necessary parameter for effective noise injection. It stands for the probability of all noises in the final service request queue. As indicated earlier, it is called noise injection intensity.

To reach the privacy protection discussed in Section 2.2, we need equation (3) below.

$$P(Q_S = q_i) = \frac{1}{n}, \forall i \quad (3)$$

We distinguish $P(Q = q_i)$ and $P(Q_S = q_i)$. The former is a historical probability from run-time historical data about Q_S ; the latter is a future probability with new Q_N about Q_S .

From the improved noise injection model described in Section 3.1.2, we can get the probabilities of Q_s which comprise of Q_U and Q_N based on equation (4) below.

$$P(Q_s = q_i) = (1 - \varepsilon)P(Q_U = q_i) + \varepsilon P(Q_N = q_i) \quad (4)$$

Combining equations (3) and (4), we get equation (5) below to obtain noise injection intensity ε .

$$\varepsilon(i) = \frac{\frac{1}{n} - P(Q_U = q_i)}{\frac{M - P(Q = q_i)}{n \times M - 1} - P(Q_U = q_i)}, \forall i \quad (5)$$

The final ε is the maximum one and is derived by equation (6) below.

$$\varepsilon = \text{Max} \{ \varepsilon(i) \} \quad (6)$$

Equations (1) and (6) ensure that the whole strategy reaches its privacy protection goal as stated by equation (3).

During operation, $P(Q = q_i)$ and M are updated in the process of noise injection. With recorded noise injections, equation (5) is kept for all cases. Then, we can get ε via equation (6) which is within the range of [0, 1]. But with random noise generation, we need $P(Q_N = q_i) = \frac{1}{n}, \forall i$ to retain equation (3). Therefore, with equation (4), we have $\varepsilon = 1$. This is not reasonable because it means that we never send out any real service requests.

3.3 Historical probability based noise generation strategy

In this section, we introduce the historical probability based noise generation strategy – HPNGS based on Section 3.2.

Input: Q_U is a queue of real service requests.	
Output: Q_s is the queue of output service requests mixed with noise requests.	
Step1: collect data items and initiate all historical probabilities $P(Q = q_i)$ from counters	Initiate $n+1$ counters $C_{[n+1]}$ to record numbers of each kind of service request and the sum; Record all real service-requests from previous Q_s and update corresponding counter $C_{[i]++}$ and the sum $C_{[n+1]++}$; Compute historical probabilities $P(Q = q_i) = \frac{C_{[i]}}{C_{[n+1]}}$;
Step2: compute every noise generation probability $P(Q_N = q_i)$	Generate every probability for noise from equation (1): $P(Q_N = q_i) = \frac{M - P(Q = q_i)}{\sum_j \{M - P(Q = q_j)\}}, \forall i$;
Step3: compute noise injection intensity ε and noise Q_N	Compute $\varepsilon = \text{Max} \{ \varepsilon(i) \}$ by $\varepsilon(i) = \frac{\frac{1}{n} - P(Q_U = q_i)}{\frac{M - P(Q = q_i)}{n \times M - 1} - P(Q_U = q_i)}, \forall i$ By now we get $Q_N[P(Q_N = q_i), \varepsilon]$;
Step4: noise injection	Generate a noise N from the final noise: $Q_N[P(Q_N = q_i), \varepsilon]$. Inject N into Q_U with the probability of ε to get Q_s . Update $P(Q = q_i), \forall i$ with counters $C_{[i]}$ and $C_{[n+1]}$. Go to Step 2.

Algorithm 1. Historical probability based noise injection strategy

During the working process of Algorithm 1, $P(Q = q_i), \forall i$, $P(Q_N = q_i), \forall i$ and ϵ are all changing with noise injections. We can find that $P(Q = q_i), \forall i$ will become more and more even with noise injections, which is our privacy protection objective. $P(Q_N = q_i), \forall i$ will also become more and more even with the changing $P(Q = q_i)$. ϵ will become lower and lower. This means that fewer noise requests will be used while achieving the same privacy protection outcome.

4. Comparison and Simulation

As analysed in Section 2.2, a random noise generation strategy needs a large number of noise requests to mix with real ones for customer privacy protection so that all of their occurrence probabilities are about the same, i.e. service providers would be confused, i.e. not being able to distinguish the noise ones from their real counterparts. As stated in Section 1, in a pay-as-you-go cloud environment, a noise request will cost the same as a real one. Therefore, with the same level of confusion, i.e. for effective customer privacy protection, the number of noise requests should be kept as few as possible. As presented in Section 3, our strategy of HPNGS focuses on the noise generation process. It generates noise requests based on their historical probabilities. Those noise requests which have higher probabilities will be used less. As such, we can use fewer noise requests to reach the privacy protection situation where all service requests will appear in about the same probabilities.

We now perform an experimental simulation in our cloud computing simulation environment called SwinCloud [38]. Our aim is to simulate the random noise generation strategy and our HPNGS to demonstrate that our HPNGS can significantly reduce the number of noise requests.

In Section 4.1, we describe the simulation environment. We then detail the simulation process in Section 4.2. In Section 4.3, we depict and analyse the simulation results to demonstrate the significant reduction of our HPNGS on the number of noise requests over the random noise generation strategy.

4.1 Simulation environment

SwinCloud is a cloud computing simulation environment. It is built on the computing facilities at Swinburne University of Technology [38]. In general, these functions of VMWare can offer unified computing and storage resources. Utilising the unified resources, we set up data centres that can host applications. In the data centres, Hadoop, which can facilitate the Map-Reduce computing paradigm and distributed data management, is installed. The architecture of SwinCloud is depicted in Figure 3.

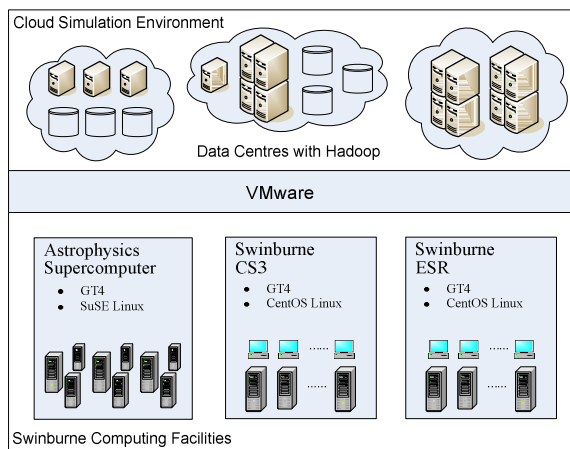


Figure 3. SwinCloud Infrastructure

4.2 Simulation process

In SwinCloud, we set two nodes to represent customer and service provider respectively. The customer node firstly generates a real service request queue from a set of requests. Then it generates a noise service request queue to inject into the real queue; they form a single request set. The service provider node receives the final service request queue and analyses the occurrence probabilities of those requests.

To demonstrate the advantage of our HPNGS strategy, we set up the simulation process to compare it with existing random noise generation strategy. In the process, we compare noise injection intensity ϵ of two strategies which reflect the number of injected noise requests.

We set a function $Var(Strategy, \epsilon)$ to denote the variance of probabilities of all final service requests including real ones and noise ones. To confuse the service provider for customer privacy protection, all probabilities should be about the same. That is to say, when $Var(Strategy, \epsilon)$ approaches 0, we are approaching the situation where customer privacy is protected. As such, the process needs to record and compare the changing noise injection intensities of both strategies.

4.3 Simulation results and analysis

Based on simulation process described in Section 4.2, we can derive $Var(HPNGS, \epsilon)$ and $Var(Random, \epsilon)$. They are depicted in Figure 4. They change by ϵ .

In Figure 4, the horizontal coordinate is Var . As Var approaches 0, the probabilities of all service requests are becoming about the same, i.e. privacy protection situation. The vertical coordinate is the noise injection intensity ϵ . When ϵ is smaller, there are fewer noise requests.

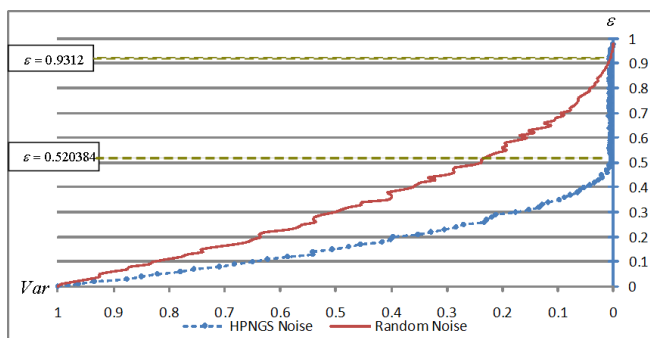


Figure 4. Comparison between our HPNGS and random noise generation strategy

From Figure 4, we can see the following: when Var approaches 0, the ϵ of our HPNGS is around 0.52038 while the ϵ of random noise generation strategy is around 0.9312. This means that with our strategy, we can achieve a 91.85% reduction in the number of noise requests over the random one. Therefore, we can say that our strategy can significantly reduce the number of noise requests when compared with the random counterpart.

5. Conclusions and Future Work

In an open cloud computing environment, various malicious service providers can exist. Such service providers may record service requests from a customer and then collectively deduce the customer private information. Therefore, customers need to protect their privacy. Obfuscation with noise injection, which mixes noise service requests with real customer service requests so that service providers will be confused about which requests are real ones, is an effective approach in this regard if their occurrence probabilities are about the same. However, current obfuscation with noise injection uses random noise requests. Due to the randomness, it needs a large number of noise requests to dilute the real ones to ensure that all of their occurrence probabilities are about the same. In a pay-as-you-go cloud environment, a noise request would cost the same as a real request. Hence, with the same level of confusion, i.e. for effective customer privacy protection, the number of noise requests should be kept as few as possible. In this paper, we have developed a novel historical probability based noise generation strategy named HPNGS. HPNGS generates noise requests based on their historical probability. Those noise requests with higher historical probability will be used less. The simulation has demonstrated that our HPNGS can significantly reduce the number of noise requests over its random counterpart by over 90%.

Based on this research, we plan to further investigate privacy protection issues in the situation where multiple service providers collaborate with each other to deduce customer privacy.

Acknowledgements

The authors are grateful for the simulation work of R. Zhu and the English proofreading by C. Fay.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, (2009), Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as The 5th Utility. *Future Generation Computer Systems*, 25(6), pp: 599-616, ISSN: 0167-739X, June 2009.

[2] A. Weiss, (2007), Computing in the Cloud *ACM Networker*, 11(4), pp: 16-25, ISSN: 1091-3556, December 2007.

- [3] UCB/EECS-2009-28, (2010), "Above the Clouds: A Berkeley View of Cloud Computing", M. Armbrust, et al. Berkeley, California, USA, <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>. Accessed on January 21, 2011.
- [4] S. Pearson, Y. Shen, M. Mowbray, (2009), A Privacy Manager for Cloud Computing, *Proceedings of The 1st International Conference on Cloud Computing*, Beijing, China, pp. 90–106, December 1-4, 2009. SpringerLink.
- [5] S. Pearson, (2009), Taking Account of Privacy when Designing Cloud Computing Services, *Proceedings of the 2009 ICSE(International Conference on Software Engineering) Workshop on Software Engineering Challenges of Cloud Computing*, pp: 44-52, Vancouver, Canada, May 23 - 23, 2009.
- [6] M. Ryan. (2011), Cloud Computing Privacy Concerns on Our Doorstep. *Communications of the ACM*, 54(1), pp: 36-38, ISSN: 0001-0782, January 2011.
- [7] C. Dwork. (2011), A Firm Foundation for Private Data Analysis. *Communications of the ACM*, 54(1), pp: 86-95, ISSN: 0001-0782, January 2011.
- [8] M. Smit, K. Lyons, M. McAllister, J. Slonim, (2009), Detecting Privacy Infractions in Applications: A Framework and Methodology, *Proceedings of IEEE 6th International Conference on Mobile Adhoc and Sensor Systems (MASS '09)*, pp: 694-701, Macau, China, October 12-15, 2009.
- [9] L. Yan, C. Rong, G. Zhao, (2009), Strengthen Cloud Computing Security with Federal Identity Management Using Hierarchical Identity-Based Cryptography, *Proceedings of The 1st International Conference on Cloud Computing*, pp:167-177, Beijing, China, December 1-4, 2009.
- [10] X. Huang, Y. He, Y. Hou, L. Li, L. Sun, S. Zhang, Y. Jiang, T. Zhang, (2009), Privacy of Value-Added Context-Aware Service Cloud, *Proceedings of The 1st International Conference on Cloud Computing*, pp:547-552, Beijing, China, December 1-4, 2009.
- [11] M. Hart, C. Castille, R. Johnson, A. Stent, (2009), Usable Privacy Controls for Blogs, *Proceedings of 2009 International Conference on Computational Science and Engineering*, vol.4, pp:401-408, Vancouver, Canada, August 29-31, 2009.
- [12] F. Kerschbaum, (2009), Adapting Privacy-Preserving Computation to the Service Provider Model, *Proceedings of 2009 International Conference on Computational Science and Engineering*, vol.3, pp: 34-41, Vancouver, Canada, August 29-31, 2009.
- [13] K. Simoens, P. Tuyls, B. Preneel, (2009), Privacy Weaknesses in Biometric Sketches, *Proceedings of the 30th IEEE Symposium on Security and Privacy*, pp: 188-203, Oakland, California, USA, May 17-20, 2009.
- [14] P. Golle, F. Mcsherry, I. Mironov, (2008), Data Collection with Self-Enforcing Privacy, *ACM Transactions on Information and System Security*, 12(2), Article 9, 24 pages, ISSN: 1094-9224, December 2008.
- [15] R. Agrawal, R. Srikant, (2000), Privacy-Preserving Data Mining, *ACM SIGMOD Record*, 29(2), pp: 439–450, ISSN: 0163-5808, June 2000.
- [16] K. Liu, "Privacy Preserving Data Mining Bibliography" (2007), http://www.cs.umbc.edu/~kunliu1/research/privacy_review.html. Accessed on January 21, 2011.
- [17] A. Evfimievski, J. Gehrke, R. Srikant, (2003), Limiting Privacy Breaches in Privacy Preserving Data Mining, *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp: 211–222, San Diego, California, USA, June 09 - 11, 2003.
- [18] L. Liu, M. Kantarcioglu, B. Thuraisingham, (2008), The Applicability of The Perturbation Based Privacy Preserving Data Mining for Real-World Data, *Data and Knowledge Engineering*, 65(1), pp: 5–21, ISSN: 0169-023X, April 2008.
- [19] L. Sweeney, (2002), K-anonymity: A Model for Protecting Privacy, *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5), pp: 557–570, ISSN: 0218-4885, October 2002.
- [20] B. Chor, E. Kushilevitz, O. Goldreich, M. Sudan, (1998), Private Information Retrieval, *Journal of ACM*, 45(6), pp: 965–981, ISSN: 0004-5411, November 1998.
- [21] A. Beimel, Y. Ishai, E. Kushilevitz, (2005), General Constructions for Information-Theoretic Private Information Retrieval, *Journal of Computer System Science*, 71(2), pp: 213–247, ISSN: 0022-0000, August 2005.
- [22] A. Beimel, Y. Ishai, E. Kushilevitz, J. Raymond, (2002), Breaking the $O(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval, *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science*, pp: 261–270, Vancouver, Canada, November 16-19, 2002.
- [23] I. Goldberg, (2007), Improving the Robustness of Private Information Retrieval, *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pp: 131-148, Oakland, California, USA, May 20-23, 2007.
- [24] E. Kushilevitz, R. Ostrovsky, (1997), Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval, *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pp: 364–373, Miami Beach, Florida, USA, October 19-22, 1997.
- [25] C. Cachin, S. Micali, M. Stadler, (1999), Computationally Private Information Retrieval with Polylogarithmic Communication, *Advances in Cryptology — EUROCRYPT'99*, pp: 402-414, ISSN: 0302-9743, ISBN: 978-3-540-65889-4, January 1999.
- [26] S. Yinan, Z. Cao, (2009), Extended Attribute Based Encryption for Private Information Retrieval, *Proceedings of IEEE 6th International Conference on Mobile Ad-hoc and Sensor Systems (MASS '09)*, pp: 702-707, Macau, China, October 12-15, 2009.

- [27] M. S. Olivier, (2005), Distributed Proxies for Browsing Privacy: a Simulation of Flocks, *Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, pp: 104–112, White River, South Africa, September 20-22, 2005.
- [28] D. Goldschlag, M. Reed, P. Syverson, (1999), Onion Routing, *Communications of ACM*, 42(2), pp: 39–41, ISSN: 0001-0782, February 1999.
- [29] R. Dingledine, N. Mathewson, P. Syverson, (2004), Tor: The Second Generation Onion Router, *Proceedings of the 13th Conference on USENIX Security Symposium*, pp: 21–21, San Diego, California, USA, August 9–13, 2004.
- [30] O. Berthold, H. Federrath, S. Köpsell, (2001), Web Mixes: A System for Anonymous and Unobservable Internet Access, *Designing Privacy Enhancing Technologies*, pp: 115–129, ISSN: 0302-9743, ISBN: 978-3-540-41724-8, January 2001.
- [31] A. Narayanan, V. Shmatikov, (2009), De-anonymizing Social Networks, *Proceedings of the 30th IEEE Symposium on Security and Privacy*, pp: 173-187, Oakland, California, USA, May 17-20, 2009.
- [32] K. Hawkey, (2009), Examining the Shifting Nature of Privacy, Identities, and Impression Management with Web 2.0, *Proceedings of 2009 International Conference on Computational Science and Engineering*, vol.4, pp:990-995, Vancouver, Canada, August 29-31, 2009.
- [33] I. James, W. Gray, (1993), On Introducing Noise into The Bus-Contention Channel, *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, pp: 90–98, Oakland, California, USA, May 24-26, 1993.
- [34] S. Ye, F. Wu, R. Pandey, H. Chen, (2009), Noise Injection for Search Privacy Protection, *Proceedings of 2009 International Conference on Computational Science and Engineering*, vol.3, pp: 1-8, Vancouver, Canada, August 29-31, 2009.
- [35] U. Hengartner, P. Steenkiste, (2005), Access Control to People Location Information, *ACM Transactions on Information and System Security*, 8(4), pp: 424–456, ISSN: 1094-9224, November 2005.
- [36] K. Fukushima, S. Kiyomoto, T. Tanaka, (2009), Obfuscation Mechanism in Conjunction with Tamper-Proof Module, *Proceedings of 2009 International Conference on Computational Science and Engineering*, vol.2, pp:665-670, Vancouver, Canada, August 29-31, 2009.
- [37] G. Zhang, Y. Yang, J. Chen. (2011), Trust-based Noise Injection Strategy for Privacy Protection in Cloud Computing, *Software: Practice and Experience*, Wiley, to appear, <http://www.ict.swin.edu.au/personal/yyang/papers/SPE-privacy-2010.pdf>, Accessed on January 21, 2011.
- [38] X. Liu, D. Yuan, G. Zhang, J. Chen, and Y. Yang, (2010), SwinDeW-C: A Peer-to-Peer Based Cloud Workflow System for Managing Instance Intensive Applications, in *Handbook of Cloud Computing*: Springer, ISBN: 978-1-4419-6523-3 pp: 309-332, 2010.