An Association Probability Based Noise Generation Strategy for Privacy Protection in Cloud Computing

Gaofeng Zhang[1], Xuyun Zhang[2], Yun Yang[3,1,*], Chang Liu[2], and Jinjun Chen[2]

[1] Faculty of Information and Communication Technologies
Swinburne University of Technology, Hawthorn, Melbourne, Australia 3122
{gzhang,yyang}@swin.edu.au
[2] Faculty of Engineering and Information Technology
University of Technology, Sydney, Broadway, NSW, Australia 2007
(Xuyun.Zhang,Chang.Liu,Jinjun.Chen)@uts.edu.au
[3] School of Computer Science and Technology, Anhui University, Hefei 230039, China

Abstract. Cloud computing allows customers to utilise IT services in a pay-as-you-go fashion to save huge cost on IT infrastructure. In open cloud, 'malicious' service providers could record service data from a cloud customer and collectively deduce the customer's privacy without the customer's permission. Accordingly, customers need to take certain actions to protect their privacy automatically at client sides, such as noise obfuscation. For instance, it can generate and inject noise service requests into real ones so that service providers are hard to distinguish which ones are real. Existing noise obfuscations focus on concealing occurrence probabilities of service requests. But in reality, association probabilities of service requests can also reveal customer privacy. So, we present a novel association probability based noise generation strategy by concealing these association probabilities. The simulation comparison demonstrates that this strategy can improve the effectiveness of privacy protection significantly from the perspective of association probability.

Keywords: Cloud service, Privacy protection, Noise generation, Association probability.

1 Introduction

Cloud computing is positioning itself as a promising and market-oriented service platform for delivering information infrastructures and resources [1]. Customers can use these services in a pay-as-you-go fashion while saving huge capital investments on their own IT infrastructures. However, cloud customers may concern about their privacy since they do not have much direct control inside the cloud [2]. So, privacy protection about service is critical as one of the most important research issues in cloud computing.

In cloud environments, there are many organisations which operate under various regulations to protect their customers' privacy. Meanwhile, many 'malicious' and

* Corresponding author.

C. Liu et al. (Eds.): ICSOC 2012, LNCS 7636, pp. 639–647, 2012.

unknown service providers could exist in these open and virtualised environments. Such service providers may collect service data from customers, and then deduce customers' privacy for unauthorised utilisation. And openness and virtualised features make it hard to distinguish them from these complex processes, especially in automated service compositions [3].

Therefore, cloud customers have to take certain technical actions to protect their privacy automatically at client sides without involving service providers. Compared to existing service-side privacy protection approaches [4, 5], noise obfuscation can match the scenario in this paper. It can inject noise service data into real service data automatically to conceal real data on customers' own, such as service requests, communication logs and so on. A historical probability based noise generation strategy (*HPNGS*) can improve the efficiency of noise obfuscation by past occurrence probabilities [6]. And time-series patterns [7] can improve the effectiveness of privacy protection based on *HPNGS*. In general, current noise obfuscation primarily focuses on concealing occurrence probabilities, and the goal of existing noise obfuscation is that the variance of all occurrence probabilities is as small as possible.

But in reality, privacy is of different varieties. In cloud, there could be various kinds of sensitive data which can be deduced from service data as customer privacy, for example, not only 'real' service requests in noise injected service request queues, but also association rules among 'real' service requests. If two requests are associated by association rules: after one request sent by one customer, then the other has a high probability to be sent sequentially. It could be a distinctive behaviour pattern of this customer, and customers' behaviour patterns or their identities could be revealed accordingly. Hence, it is a serious privacy risk.

In this paper, the main goal of noise obfuscation is that the variance of all association probabilities among service requests is as small as possible. So, we need to analyse association probabilities of past real service requests, and generate noise service requests which can conceal association rules by making association probabilities about the same, and these 'novel' noise service requests can protect customers' privacy as an improvement of privacy protection. Based on this, we present a novel association probability based noise generation strategy (*APNGS*).

The remainder of the paper is organised as follows. In Section 2, we present the association probability based noise generation strategy (*APNGS*). Then, in Section 3, we perform a simulation to demonstrate that *APNGS* can improve the effectiveness of privacy protection significantly. Finally in Section 4, we conclude our contributions and point out future work.

2 Novel Association Probability Based Noise Generation Strategy

Concealing association probabilities is the goal of privacy protection in this paper, and *APNGS* focuses on how to model, analyse and conceal these association probabilities. In this section, we firstly discuss association probability based noise injection model to support *APNGS*. Then, we present association probability model for noise generation. After that, we discuss two key issues of noise generation—noise generation probabilities and noise injection intensity. Lastly, we propose *APNGS*.

2.1 Association Probability Based Noise Injection Model

The noise injection model is based on other existing representative noise injection models [6-8] with some modifications to support *APNGS* as depicted in Fig. 1.

Q_R: queue of customer's real service requests to be protected.
Q_N: queue of 'noise' service requests to be injected in Q_R.
Q_S: queue of final service requests composing of Q_R and Q_N.
Q : a common set of Q_R, Q_S and Q_N. And $Q=\{q_1,q_2,\ldots,q_i,\ldots,q_n\}$.
ε: probability for injecting Q_N into Q_R, $\varepsilon \in [0,1]$. It is the noise injection intensity.

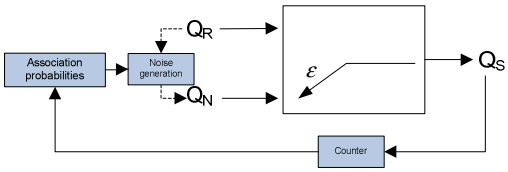

Fig. 1. Association probability based noise injection model

'Association probabilities': they are the basis of this strategy and guide noise generations. 'Noise generation': its function is to generate Q_N. We use 'Association probabilities' and 'Counter' to get noise generation probabilities in *APNGS*.

The overall working process of the model is to inject Q_N into Q_R based on ε, and Q_S is the result. The model can be described as follows: the customer generates a real service request queue Q_R to be sent. The noise service request queue Q_N is generated by *APNGS*. To obtain Q_S, a switch function is: the next service request in Q_S comes from Q_N on the probability of ε, and from Q_R on the probability of 1-ε. Suppose q_i is an item of Q and $P(Q_R=q_i)(t)$, $P(Q_N=q_i)(t)$ and $P(Q_S=q_i)(t)$ are occurrence probabilities of q_i in Q_R, Q_N and Q_S at time t, respectively.

2.2 Association Probability Model for Noise Generation

In this part, we investigate how to obtain association probabilities from service request queues in this association probability model for noise generation.

To define these association probabilities, we have:

$$AP = f_{AP}(Q_R) \tag{1}$$

In equation (1), AP denotes association probabilities, and it is an $n \times n$ matrix. Each item in this matrix $AP[i,j]$ is association probability between q_i and q_j.

Sliding window is the key and widely used approach to analyse information in a data stream or queue [9]. In this paper, we use a minimised and fixed sliding window to generate association probabilities. As a basic form of sliding windows, a minimised and fixed sliding window can aid to analyse data streams in terms of basic features. So, we obtain the association probability model for noise generation:

$$AP[i,\ j] = \frac{Con^{i,j}(Q_R)}{t-1} \tag{2}$$

In equation (2), $Con^{i,j}(Q_R)$ is the number of events that q_i and q_j are sent together in Q_R. Under minimised and fixed sliding windows, this event is that q_j is immediately next to q_i as a consequential relation in Q_R. And we use time length *t-1* as the denominator to normalise the equation. In some specific privacy protection situations, sliding window can be dynamic to withstand some specific privacy risks, such as side channel knowledge on it, or particular timestamps in request queues. In other words, sliding window is the changing key in noise obfuscation to protect privacy in cloud environments. And as a basic form, minimised and fixed sliding windows adopted in this paper can be easily modified to match specific noise obfuscations.

2.3 Association Probability Based Noise Generation

In this part, we discuss two key issues in noise generation—noise generation probabilities and noise injection intensity. Suppose noise generation probabilities are $P(Q_N = q_i)(t), \forall i \in [1,n]$ which means that for $\forall q_i \in Q$, probabilities of Q_N being q_i at time *t*, respectively. Noise injection intensity is ε which is introduced earlier.

1) Noise generation probabilities

In *HPNGS* [6], noise generation probabilities are:

$$\forall i, P(Q_N = q_i) = \frac{M - P(Q_R = q_i)}{n \times M - 1} \tag{3}$$

From equation (3), in $\forall i, P(Q_R = q_i)$, the highest one is $M = MAX\{P(Q_R = q_i), \forall i\}$, which is historical and accumulative data from past Q_R in practice as depicted in Fig. 1, just like $P(Q_R=q_i)$. And *n* is the number of q_i.

Besides, from Section 1 and [6], existing strategies have the same noise generation goal: $\forall i, P(Q_S = q_i) = 1/n$. So, on the basis of equation (2), we get the noise generation goal in *APNGS*:

$$\forall i, j, t, P(Q_S, t+1, i, j) = P[(Q_S = q_i)(t+1) \mid (Q_S = q_j)(t)] = 1/n \tag{4}$$

In equation (4), the noise generation goal is a family of conditional probabilities to express the probability of Q_S being q_i at time *t+1*, on the precondition of Q_S being q_j at previous time *t*. Besides, we have $i, j \in [1,n]$ and $t \in [1,T]$, and *T* is the time length of the overall process.

To realise equation (4), we can utilise new noise generation probabilities in equation (5) on the basis of equation (3):

$$P[(Q_N = q_i)(t+1) \mid (Q_S = q_j)(t)] = \frac{M(t,j) - P(Q_R, t, i, j)}{n \times M(t,j) - 1} \tag{5}$$

In equation (5), we have two components: equation (6) and equation (7):

$$P(Q_R, t+1, i, j) = P[(Q_R = q_i)(t+1) \mid (Q_S = q_j)(t)] \tag{6}$$

$$M(t+1, j) = MAX\{P[(Q_R = q_i)(t+1) \mid (Q_S = q_j)(t)], \forall i\} \quad (7)$$

In equation (6), $P(Q_R,t+1,i,j)$ is a family of conditional probabilities to express the probability of Q_R being q_i at time $t+1$, on the precondition of Q_S being q_j at time t.

In equation (7), $M(t+1,j)$ is the highest value, for every i, in a family of conditional probabilities to express the probability of Q_R being q_i at time $t+1$, on the precondition of Q_S being q_j at previous time t.

It is clear that equation (6) is the basis of equation (7). So, we only need to focus on equation (6) for association probabilities among service requests introduced before.

To obtain equation (6), we design a process on the basis of equation (2): this is an accumulative process. 3-dimension matrix $Matrix(i,j,t)$ has three parameters: t is time parameter, i is from $(Q_R=q_i)(t+1)$ which means an event that the *ith* request in the set Q will appear in Q_R at time $t+1$, j is from $(Q_S=q_j)(t)$ which means an event that the *jth* request in Q appears in Q_S at time t. So, $Matrix(i,j,t)$ means all past association relations among service requests q_i and q_j at time t. At a specific time, 2-dimension array $C[i][j]$ can replace $Matrix(i,j,t)$. We should collect all requests from time *1* to time t, and get accumulative $C[i][j]$. So, $P(Q_R,t,i,j)=Matrix(i,j,t)/SUM$, Where SUM is the number of all association relations among past requests. This presents the implementation of association probability model.

2) Noise injection intensity

According to the association probability based noise injection model defined earlier, to operate noise injection processes, ε is a necessary parameter.

From noise injection model and [6], we can get the relation among Q_S, Q_N and Q_R:

$$P(Q_S, t, i, j) = (1-\varepsilon) \times P(Q_R, t, i, j) + \varepsilon \times P(Q_N, t, i, j) \quad (8)$$

There are two components in equation (8): $P(Q_N,t,i,j)$ and $P(Q_R,t,i,j)$. So, we have equation (9) based on equations (6) and (4):

$$\forall i, j, t, P(Q_N, t+1, i, j) = P[(Q_N = q_i)(t+1) \mid (Q_S = q_j)(t)] \quad (9)$$

In equation (9), it is clear that the conditional probability of Q_S being q_i at time $t+1$ on the precondition of Q_S being q_j at previous time t is decided by the conditional probability of Q_R being q_i at time $t+1$ on the precondition of Q_S being q_j at previous time t , the conditional probability of Q_N being q_i at time $t+1$ on the precondition of Q_S being q_j at previous time t, and ε. So, we get ε by equations (8) and (4):

$$\varepsilon(t+1) = \frac{\frac{1}{n} - P(Q_R, t+1, i, j)}{\frac{M(t+1, j) - P(Q_R, t+1, i, j)}{n \times M(t+1, j) - 1} - P(Q_R, t+1, i, j)} = 1 - \frac{1}{n \times M(t+1, j)} \quad (10)$$

It is obvious that equation (5) and equation (10) can make the whole strategy to reach its goal—equation (4). It can address the serious risk identified in Section 1.

2.4 Association Probability Based Noise Generation Strategy

Title: Association probability based noise generation strategy
Input: Q_R is the queue of real service requests
Output: Q_S is the queue of final service requests

Step 1: Collect data and compute association probabilities
Initialise n*n+1 counters: $C[1][1],\ldots\ldots,C[1][n]\ldots\ldots C[n][1],\ldots\ldots C[n][n]$, S to record numbers of associations among each service data item and the sum;
Receive service data from all past Q_R, update the correspondent $C[i][j]$++ and the sum S++;
Compute previous association probabilities :

$$P(Q_R, t+1, i, j) = P[(Q_R = q_i)(t+1) \mid (Q_S = q_j)(t)] = \frac{C[i][j]}{S}$$

Step 2: Compute noise generation probabilities
Generate noise generation probabilities by equation (5):

$$P[(Q_N = q_i)(t+1) \mid (Q_S = q_j)(t)] = \frac{M(t,j) - P(Q_R, t, i, j)}{n \times M(t,j) - 1}$$

by $P(Q_R, t, i, j)$ and $M(t,j) = MAX\{P(Q_R, t, i, j), \forall i\}$;

Step 3: Compute noise injection intensity
Compute equation (10): $\varepsilon(t+1) = 1 - \frac{1}{n \times M(t,j)}$ to get the noise injection intensity;
We have noise requests queue
$Q_N\{P[(Q_N = q_i)(t+1) \mid (Q_S = q_j)(t)], \varepsilon(t+1)\}$;

Step 4: Noise injection process
Generate a noise N by:
$Q_N\{P[(Q_N = q_i)(t+1) \mid (Q_S = q_j)(t)], \varepsilon(t+1)\}$
Inject N into Q_R on the probability of ε to get Q_S ;
Update $P(Q_R, t, i, j)$ with counters.

Algorithm 1. *APNGS*: Association probability based noise generation strategy

In this part, we present *APNGS*. In Algorithm 1, we utilise *n×n+1* counters to record the matrix and the sum of association relations among requests in Step 1. From equation (5), we can generate noise generation probabilities in Step 2. About noise injection intensity, equation (10) can obtain it in Step 3. At last, we can execute the noise injection processes in Step 4.

In summary, we can find out that the major improvement between *APNGS*'s noise generation goal updating: replacing $\forall i, P(Q_S = q_i) = 1/n$ by $\forall i, j, t, P(Q_S, t, i, j) = 1/n$. Hence, *APNGS* can perform better in privacy protection situations considering association probabilities than existing noise generation strategies.

3 Evaluation

In this section, we perform an experimental simulation in our cloud simulation system called SwinCloud [10]. The simulation process is primarily to compute and compare the effectiveness of privacy protection between *APNGS* and *HPNGS* directly, and we discuss comparisons about other existing strategies: *TPNGS* [7] and random generation [8], too. Before the simulation, we generate a service queue as the real service queue from a set with a size of 10 randomly to operate two strategies.

We use a function: *Var_Ass(Strategy, t)* to express the main effectiveness of privacy protection on noise obfuscation to compare two strategies. And *Var_Ass(Strategy, t)* means that the variance of association probabilities in Q_S under *Strategy* protected at time *t*. In other words, time *t* can express the size of Q_R. Besides, we also use *Var(Strategy, t)* to denote the variance of occurrence probabilities in Q_S

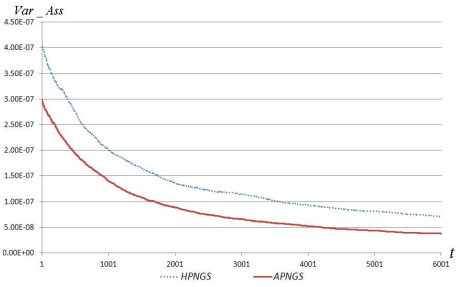

Fig. 2. Effectiveness comparison on association probability between *HPNGS* and *APNGS*

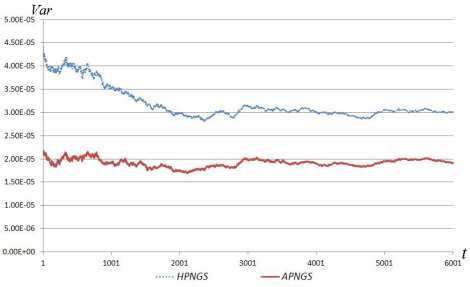

Fig. 3. Effectiveness comparison on occurrence probability between *HPNGS* and *APNGS*

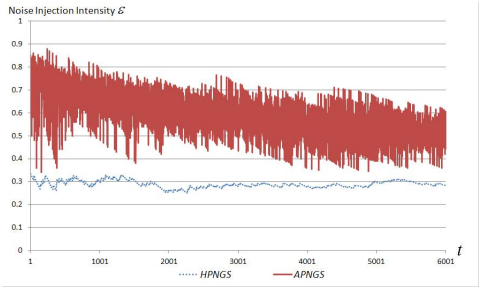

Fig. 4. Comparison on noise injection intensity between *HPNGS* and *APNGS*

with *Strategy* operating at time *t*. It denotes another aspect of the effectiveness of privacy protection.

In Fig. 2, the horizontal coordinate is time *t*, and *t* has a range of [1, 6001]; the vertical coordinate is *Var_Ass*. And if *Var_Ass* is lower, the effectiveness of privacy protection is better, so customer privacy is better kept. We can find out the overall trend being: with time passing, both of them keep on decreasing. Obviously, *Var_Ass(APNGS,t)*is about ¼ to ½ of *Var_Ass(HPNGS,t)*. So, *APNGS* achieves a significant improvement on the effectiveness of privacy protection on noise obfuscation over *HPNGS*, in terms of association probability.

In Fig. 3., we find out that *Var(APNGS,t)* can keep within a low level of about 2.00e-05, and is lower than *Var(HPNGS,t)* which is about 3.00e-05. So, *APNGS* has better effectiveness of privacy protection on noise obfuscation in terms of occurrence probability than *HPNGS*. That is because association probabilities can consider more details of data queues than occurrence probabilities to keep service requests balance.

At last, the cost of noise obfuscation should also be considered by the noise injection intensity. In Fig. 4, ε of *APNGS* is about 1.5 to 1 times more than ε of *HPNGS* with time passing. It means that *APNGS* costs more than *HPNGS*. So, to obtain better effectiveness of privacy protection, customers have to pay more. It is a trade-off depending on customers' demands.

About other typical noise generation strategies, such as random generation [8] and *TPNGS* [7], *APNGS* also performs well: *HPNGS* has already improved efficiency of

privacy protection from random generation with similar effectiveness. *TPNGS* still focuses on fluctuations of occurrence probabilities, and association probabilities are not incorporated.

In summary, *APNGS* can significantly improve the effectiveness of privacy protection on noise obfuscation in terms of association probability over existing representative strategies, with good effectiveness of privacy protection on noise obfuscation in terms of occurrence probability, at a reasonable extra cost.

4 Conclusions and Future Work

In this paper, we focused on privacy protection and noise obfuscation in cloud computing. To withstand some malicious service providers which are interested in association information among service requests, we analysed association probabilities in request queues and concealed them by noise obfuscation. Hence, we proposed a novel association probability based noise generation strategy (*APNGS*). The evaluation demonstrated that *APNGS* could significantly improve the effectiveness of privacy protection on noise obfuscation in terms of association probabilities, at a reasonable extra cost, in comparison to existing representative strategies.

Based on our current work, we plan to further investigate how to protect privacy in multiple malicious providers.

References

1. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems 25(6), 599–616 (2009)
2. Ryan, M.D.: Cloud Computing Privacy Concerns on Our Doorstep. Communications of the ACM 54(1), 36–38 (2011)
3. Ren, K., Liu, X., Chen, J., Xiao, N., Song, J., Zhang, W.: A QSQL-based Efficient Planning Algorithm for Fully-automated Service Composition in Dynamic Service Environments. In: 2008 IEEE International Conference on Service Computing (SCC 2008), pp. 301–308. IEEE Press (2008)
4. Evfimievski, A., Gehrke, J., Srikant, R.: Limiting Privacy Breaches in Privacy Preserving Data Mining. In: 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2003), pp. 211–222. ACM, Madison (2003)
5. Bu, Y., Fu, A., Wong, R., Chen, L., Li, J.: Privacy Preserving Serial Data Publishing by Role Composition. In: 34th Internationasl Conference on Very Large Data Bases (VLDB 2008), pp. 845–856 (2008)
6. Zhang, G., Yang, Y., Chen, J.: A Histrotical Probability based Noise Generation Strategy for Privacy Protection in Cloud Computing. Journal of Computer and System Sciences 78(5), 1374–1381 (2012)
7. Zhang, G., Yang, Y., Liu, X., Chen, J.: A Time-Series Pattern based Noise Generation Strategy for Privacy Protection in Cloud Computing. In: 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012), pp. 458–465. IEEE Press, New York (2012)

8. Ye, S., Wu, F., Pandey, R., Chen, H.: Noise Injection for Search Privacy Protection. In: 2009 International Conference on Computational Science and Engineering (CSE 2009), pp. 1–8. IEEE Press, New York (2009)
9. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002), pp. 1–16. ACM, Madison (2002)
10. Liu, K., Jin, H., Chen, J., Liu, X., Yuan, D., Yang, Y.: A Compromised-time-cost Scheduling Algorithm in SwinDeW-C for Instance-intensive Cost-constrained Workflows on a Cloud Computing Platform. International Journal of High Performance Computing Applications 24(4), 445–456 (2010)