

Available from: http://dx.doi.org/10.1109/CCGrid.2012.82.

Copyright © 2012 IEEE.

This is the author’s version of the work, posted here with the permission of the publisher for your personal use. No further distribution is permitted. You may also be able to access the published version from your library. The definitive version is available at http://ieeexplore.ieee.org/.
A Time-series Pattern based Noise Generation Strategy for Privacy Protection in Cloud Computing

Gaofeng Zhang, Yun Yang, Xiao Liu  
Faculty of Information and Communication Technologies  
Swinburne University of Technology  
Hawthorn, Melbourne, Australia 3122  
{gzhang, yyang, xliu}@swin.edu.au

Jinjun Chen  
School of Systems, Management and Leadership  
Faculty of Engineering and Information Technology  
University of Technology, Sydney  
Broadway, NSW, Australia 2007  
Jinjun.Chen@uts.edu.au

Abstract—Cloud computing promises an open environment where customers can deploy IT services in a pay-as-you-go fashion while saving huge capital investment in their own IT infrastructure. Due to the openness, various malicious service providers may exist. Such service providers may record service information in a service process from a customer and then collectively deduce the customer’s private information. Therefore, from the perspective of cloud computing security, there is a need to take special actions to protect privacy at client sides. Noise obfuscation is an effective approach in this regard by utilising noise data. For instance, it generates and injects noise service requests into real customer service requests so that service providers would not be able to distinguish which requests are real ones if their occurrence probabilities are about the same. However, existing typical noise generation strategies mainly focus on the entire service usage period to achieve about the same final occurrence probabilities of service requests. In fact, such probabilities can fluctuate in a time interval such as three months and may significantly differ than other time intervals. In this case, service providers may still be able to deduce the customers’ privacy from a specific time interval although unlikely from the overall period. That is to say, the existing typical noise generation strategies could fail to protect customers’ privacy for local time intervals. To address this problem, we develop a novel time-series pattern based noise generation strategy. Firstly, we analyse previous probability fluctuations and propose a group of time-series patterns for predicting future fluctuated probabilities. Then, based on these patterns, we present our strategy by forecasting future occurrence probabilities of real service requests and generating noise requests to reach about the same final probabilities in the next time interval. The simulation evaluation demonstrates that our strategy can cope with these fluctuations to significantly improve the effectiveness of customers’ privacy protection.

Keywords—Cloud computing, Privacy protection, Noise obfuscation, Noise generation, Time-series pattern

I. INTRODUCTION

Cloud computing is positioning itself as a new and promising platform for delivering information infrastructures and resources as IT services [1]. 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 [2]. However, customers often have concerns about whether their private information can be protected when facilitating their IT services in the cloud since they do not have much control inside cloud [3]. Without it, customers may eventually lose the confidence in and desire to deploy cloud computing in practice [4]. Therefore, security and privacy are critical as one of the most concerned issues in cloud computing.

In cloud computing environments, there are many organisations, such as the banking and immigration sectors, which operate under various regulations and policies to protect their customers’ privacy. Meanwhile, a large number of unknown and malicious service providers may exist in open and virtualised cloud computing environments. Such service providers may collect service information from customers to analyse and deduce customers’ privacy without their permissions.

For service providers, it is a common phenomenon to collect their customers’ information, like service requests. From large to small firms, they often use them to analyse customers’ behaviours, habits, and other private information [5]. Most ethical ones have adequate self-control to use the information conforming to policies and regulations, but some others may misuse the information in unethical ways, especially in open cloud environments because their open and virtualised features make customers hard to distinguish which service providers are trustworthy. In this paper, we focus on customers’ privacy without some specific data structures or types. For instance, these service requests from customers to service providers may have some private information to fulfil service processes. And this information could be single data items from some sets, like in this paper.

Existing major security and privacy mechanisms have not considered this situation thoroughly, hence cannot aid customers to withstand such unethical services. With facing such a privacy risk, customers should be protected by taking certain technical actions for their privacy automatically at client sides without participation of service providers. Noise obfuscation is a promising and effective approach in this regard. For example, it injects noise service requests into real customer service requests automatically. These noise requests
are some extra requests just ‘like’ real ones, generated by noise generation strategies not customers’ operations. When final requests’ occurrence probabilities are about the same with those injected noise requests, service providers cannot tell which requests are real ones with high confidences, except other information sources existing. In other words, the goal of noise obfuscation on privacy protection is that the variance of all occurrence probabilities of requests is as small as possible. The key advantage is that this approach does not need cooperation or assistance from service providers. Hence, it seems promising to protect customers’ privacy due to the existence of ‘unethical’ service providers.

Currently, a historical probability based noise generation strategy (HPNGS) is proposed to improve the efficiency of privacy protection in a pay-as-you-go cloud environment [6]. Compared to conventional random noise generation, HPNGS generates noise requests based on their previous probabilities: If one request has a high occurrence probability of real service requests, it will be generated as noise requests with a low probability. So, all requests including noise ones and real ones can still reach about the same occurrence probabilities, but with far fewer noise requests than random noise generation. In the pay-as-you-go style of cloud computing, few noise requests mean less cost, hence more efficiency.

In reality, due to the dynamics of cloud computing environments, occurrence probabilities of real service requests may have some fluctuations. For the purpose of privacy protection, we try to make all probabilities to be similar in every local time interval. It means that customers’ privacy can be protected. However, the existing strategy (HPNGS) has not taken these fluctuations into consideration for noise service request generation because it utilises past probabilities as a whole to generate noise requests without considering local time intervals. In other words, HPNGS can have about the same probabilities in the entire time period, but may not be the case in all time intervals which compose the entire time period due to these probability fluctuations. As a result, final service requests including real ones and noise ones may have some significant fluctuations. Then, service providers would still be able to deduce customer private information from these fluctuations in those time intervals. This is a serious privacy risk. Besides, random noise generation [7] does not consider this privacy risk before, too.

To address this problem, we develop a novel time-series pattern based noise generation strategy (TPNGS) for privacy protection in cloud computing. In the strategy, we analyse all past probabilities, and deduce time-series patterns by time-series segmentation. Based on these past time-series patterns, we analyse current probabilities of real requests and forecast “future” probabilities of real requests with pattern matching. At last, we generate time-series pattern based noise requests to protect customers’ privacy. These noise requests can make final requests to reach the goal that all occurrence probabilities of final requests are kept about the same, even in some time intervals with probabilities’ fluctuations.

Let us take a weather service as a motivating example. One customer, who often travels to one city in Australia, like ‘Sydney’, checks the weather report regularly from a weather service in cloud environments before departure. The frequent appearance of service requests about the weather report for ‘Sydney’ can reveal the privacy that the customer usually goes to ‘Sydney’. But if a system aids the customer to inject other requests like ‘Perth’ or ‘Darwin’ into the ‘Sydney’ queue, the service provider cannot distinguish which ones are real and which ones are ‘noise’ as it just sees a similar style of service request. These requests should be responded and cannot reveal the location privacy of the customer. In such cases, the privacy can be protected by noise obfuscation in general. Given the privacy risk identified in this paper before, the customer could go to ‘Sydney’ in this month and ‘Perth’ in the next month. So, these probabilities of requests may have some fluctuations: ‘Sydney’ is high in this month and low in the next month; ‘Perth’ is low in this month and high in the next month. And in the view of the entire service period, both occurrence probabilities may be about the same already. But the itinerary of this customer still can be discovered by some unethical services: the person will go to ‘Sydney’ in this month and ‘Perth’ in the next month. So, these fluctuations are quite hard to be concealed by existing noise obfuscation. To address this, the ultimate goal of privacy protection in this paper is to keep occurrence probabilities of final requests to be about the same in every time intervals, instead of only in the entire time period. We can forecast these fluctuations by time-series patterns and generate noise service requests to achieve this goal. In this example, privacy is location information in service requests, not these actual requests. So, it is obvious that we can utilise this paper’s work in other conditions.

The remainder of the paper is organised as follows. In Section II, we overview the related work. Then, in Section III, we enhance some preliminary knowledge to support our work. In Section IV, we present our novel time-series pattern based noise generation strategy (TPNGS). In Section V, we perform a simulation to demonstrate that our noise generation strategy can improve the effectiveness of privacy protection significantly. Finally, in Section VI, we conclude our contributions and point out future work.

II. RELATED WORK

In this section, we overview some privacy protection approaches: such as privacy-preserving data mining (PPDM), privacy information retrieval (PIR), anonymity browsing and searching, noise obfuscation and insertion. We would discuss and analyse their features to form our solution for the privacy risk introduced in Section I. Besides, time-series pattern is an effective tool to forecast variables in time-series situations, which serves as a part of the basis for our TPNGS strategy.

Many and more researchers are starting to produce and/or have produced remarkable research on privacy protection related to the cloud environments. X. Huang et al. [8] discuss privacy protection in value-added context-aware cloud. K. Simonis et al. [9] present a biometric encryption system in the privacy protection of biometric search area. R. Neisse et al. [10] investigate trust in cloud environments and promote data security. These papers express that there are many privacy protection situations in cloud computing that should be considered and protected by many specific privacy
historical probability based noise generation strategy for noise injection in privacy-aware searching by formulating noise injection as a mutual information minimisation the characters of information [22].

Different from PPDM, PIR utilises another approach to protect privacy, which mainly prevents database operators from knowing users' interested records. B. Chor et al. [14] have a conclusion that, to get a perfect protection, a user has to query all the entries in database when dealing with a single server framework. A. Beimel et al. [15] and I. Goldberg et al. [16] apply information theories to dig deeply in PIR.

Proxies and anonymity networks to protect customers' privacy have been widely discussed. The major goal is to keep anonymity or "invisibility" in a complex or "dangerous" network condition. For example, onion routing [17] and its successor TOR [18] provide a more sophisticated privacy protection scenario, making it difficult for attackers to trace the customer via network traffic analysis. Narayanan et al. [19] present a framework for analysing privacy and anonymity in social networks, and develop a topology-based re-identification algorithm targeting at anonymous social network graphs.

As analysed in Section I, various malicious service providers may exist in cloud environments. They may record customers' service requests and collectively deduce customers' private information. Therefore, customers' privacy needs to be protected without service providers. This is the scenario we address in this paper.

PPDM is not an ideal choice to address the scenario because it is out of customers' control, hence not suitable for protecting customers' privacy in this paper. PIR mainly works at service provider sides, hence has the similar problem. Anonymity or proxy networks need service provider's cooperation to enable such access. Besides, their IP addresses can be traced in the end especially for normal common customers who often simply use cloud services straight from their computers.

Noise obfuscation and insertion is another widely adopted method for protecting information privacy. C. Ardagna et al. [20] focus on the location privacy protection in a mobile environment, and present a solution based on different obfuscation operators. E. Perron et al. [21] investigate noise utilisation in wireless conditions. Similar with above noise insertion in signal communication, noise injection builds on the ground of information theory to cover the characters of information [22]. S. Ye et al. [7] describe noise injection in privacy-aware searching by formulating noise injection as a mutual information minimisation problem. A trust based noise injection strategy for privacy protection in cloud computing is presented to discuss influences of complex relations in cloud environments on noise obfuscation scheme [23]. G. Zhang et al. [6] present a historical probability based noise generation strategy for privacy protection in cloud computing to improve the efficiency of current noise privacy protection and obtain a promising cost-saving in cloud environments, which forms the foundation of this paper. But it does not consider fluctuations of probabilities introduced before, so it has a shortcoming in the effectiveness of privacy protection. This is what we plan to address in this paper. Currently, some cryptograph methods [24, 25] have been discussed in multiple computing and is still trying to improve efficiency to make them into practical.

In the scenario discussed in this paper, noise obfuscation is utilised by customers at client sides, which is different from existing privacy protection approaches. And the efficiency of our approach is different that at server sides too. So, we compare the efficiency among noise obfuscation strategies at client sides, which is an important aspect of the evaluation.

About time-series pattern, E. Keogh et al. [26] use an online algorithm for segmenting time series in mining time series databases. X. Liu et al. [27] present a time-series pattern based algorithm to forecast duration intervals in scientific workflow activities. E. Shi et al. [28] investigate the aggregation of time-series data and present a group of PSA algorithms to protect each source's privacy, when the data aggregator is untrusted. Considering the problem in this paper, the time-series pattern is an effective tool to forecast "future" occurrence probabilities based on past probabilities in the situation with probability fluctuations. We can analyse and deduce several patterns from all past probabilities. Then, jointly with current occurrence probabilities, we can forecast persuasive "future" real request probabilities to guide noise generation. And the probability fluctuations can be foreseen and addressed.

III. PRELIMINARY ON TIME-SERIES PATTERN BASED NOISE GENERATION

In this section, we propose our time-series pattern based noise injection model on the basis of existing noise injection model at first. Then we investigate time-series patterns of past probabilities and present our time-series pattern based forecasting algorithm (TPF) for privacy protection on the basis of existing time-series pattern forecasting algorithms.

A. Noise injection model

Our time-series pattern based noise injection model is modified from [6] to fulfil our time-series pattern idea, and it is shown in Error! Reference source not found..

\[ Q_R : \text{queue of customer's real service requests to be protected.} \]
\[ Q_N : \text{queue of "noise" service requests to be injected in} Q_R. \]
\[ Q_S : \text{queue of final service requests composing of} Q_R \text{and} Q_N. \]
Segmenting and pattern generation of these patterns, we introduce an algorithm for pattern generation. Firstly, we introduce an algorithm for time-series pattern generation. Secondly, we introduce an algorithm for time-series pattern generation strategies to forecast fluctuations introduced on real service requests. Therefore, on the basis of this model, we will discuss them in Section IV to present our TPNGS strategy.

The overall working process of the model is to inject \( Q^*_N \) into \( Q_s \), based on \( \varepsilon \) so that we can get \( Q_s \). The model can be described as follows. 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 probabilities of \( q_i \) in \( Q_R \), \( Q_N \) and \( Q_s \) at time \( t \) respectively. \( P(Q'_R = q_i)(t) \) is the probability of \( q_i \) in past \( Q_R \) at time \( t \). Hence, \( q_i \) appeared in \( Q_s \) with a probability of \( P(Q'_s = q_i)(t) \) at time \( t \), and it will appear in \( Q_N \) with a probability of \( P(Q_N = q_i)(t+1) \) at time \( t+1 \).

As introduced before, to protect customers’ privacy, we need to achieve the state that \( \forall i, \sum P(Q_s = q_i)(t+1) \) are about the same. Therefore, if we forecast that \( P(Q_R = q_i)(t+1) \) has a high value by our strategy, then \( q_i \) will not be taken as noise in the next time so that \( P(Q_R = q_i)(t+1) \) will have a smaller value, and vice versa. This is the general process of generating noise requests based on time-series patterns of real requests. In other words, we consider one extra parameter—time \( t \) than existing noise generation strategies, to forecast fluctuations introduced before.

Besides, in this model, to get noise requests, noise generation probabilities and noise injection intensity are necessary to be analysed. Hence, on the basis of this model, we will discuss them in Section IV to present our TPNGS.

**B. Time-series pattern based forecasting algorithm for noise generation**

In this section, we present our TPF algorithm for noise generation. Firstly, we introduce an algorithm for time-series segmenting and pattern generation (TSPG). Then, on the basis of these patterns, we introduce an algorithm for pattern matching and forecasting (PMF). As last, to support our TPNGS strategy, our TPF algorithm is presented.

Similar to other data in time-series pattern based forecasting algorithms [26-28], occurrence probabilities have the characteristic of changing with time. That is the precondition of time-series pattern based algorithms. In this paper, past occurrence probabilities compose of various occurrence probabilities of various service requests, and each of them can be treated as an independent time-series pattern based forecasting process. Therefore, in one time-series pattern based forecasting process, we execute TSPG and PMF algorithms to derive several forecasting results. Then we combine these processes together and integrate these forecasting results to aid noise generation in the TPNGS algorithm. This is the main procedure of our time-series pattern based forecasting algorithm for noise generation.

1) **TSPG: Time-series segmenting and pattern generation algorithm**

Here we introduce the first part of time-series pattern based forecasting—the TSPG algorithm based on [27].

In brief, TSPG divides past occurrence probabilities and gets some time segments. Then it checks the validation of them and generates patterns. We utilise the bottom-up and top-down approaches to move windows in time-series to make sure that the variance of one segment is close to, but not more than a pre-set parameter as a maximum boundary of variance. Then we split the time-series queue into several time segments. Lastly, we validate them and set them as patterns by a pre-set parameter \( \text{Min}_\text{pattern length} \) which means the minimum boundary of a validated pattern’s length. The input of TSPG algorithm is past occurrence probabilities of real service requests: \( P(Q_R = q_i)(t), k \in [1, n], t \in [0, T] \), and the output is a group of time segments—\( \text{Patterns}_{[j]}, j \in [0, m] \). The function of TSPG algorithm is \( \text{Patterns}_{[j]}, j \in [0, m] = \text{TSPG}(P(Q_R = q_i)(t), k \in [1, n], t \in [0, T]) \).

Besides, every validated pattern has an attribute—\( \text{next value} \) which is the first value of the next pattern or time segment after this pattern in a whole queue. It is a key attribute for forecasting in TSPG described next.

2) **PMF: Pattern matching and forecasting algorithm**

Here we introduce the second part of time-series pattern based forecasting—the PMF algorithm based on [27], too.

In brief, PMF utilises patterns, resulted from TSPG, to match current probabilities. If we find a matched pattern, its forecasting attribute—\( \text{next value} \) can play a key role to forecast “future” probabilities. \( \text{Min}(|\text{Patterns}.\text{mean-CP}.\text{mean}|) \) denotes a function which returns one pattern with a minimum absolute difference of means between it and \( CP \) which denotes the current probabilities queue. This is the main part of PMF to find out the suitable pattern to match the current probabilities queue, and we utilise the means of patterns to evaluate this. So, the function of this algorithm is \( MP, FR = \text{PMF}(|\text{Patterns}_{[j]}|, j \in [0, m], CP) : i.e., one input is patterns we have got \( \text{Patterns}_{[j]} |, j \in [0, m] \), another input is current probabilities queue \( CP \); one output is the
matched pattern \( MP \), another output is the forecasting result \( FR \). Our forecasting result \( FR \) is a probability which denotes the future occurrence possibility of one real service request, and it is decided by the matched pattern \( MP \).

In the real process of pattern matching, the PMF algorithm takes the mean of current probabilities queue \( CP \) as the default value. If we cannot find out a suitable pattern, the mean is used as the forecasting result \( FR \) to guide noise generation.

3) TPF: Time-series pattern based forecasting algorithm

Here we present our TPF algorithm for noise generation.

In Algorithm 1, we detail the time-series pattern based forecasting algorithm for noise generation based on the TSPG algorithm which can be applied as a function named TSPG(), and the PMF algorithm which can be applied as a function named PMF(). We operate them for various probabilities of service requests and get various forecasting results. After that, we need to normalise these forecasting results. It is apparent that for a certain time interval, the sum of probabilities of all service requests is 1. Besides, we denote \( L \) for the length of current probabilities queue. In this paper, we set it to be equal to \textit{Min. pattern length} for the balance between effectiveness and efficiency of forecasting.

Title: Time-series pattern based forecasting algorithm for noise generation

Input: All past occurrence probabilities \( P(Q_t = q_i(t), t \in [0, T]), k \in [1, n] \)

The length of current probabilities queue \( L \).

Output: One group of future probabilities \( P(Q_t = q_i(t+1), k \in [1, n]) \)

for \( i = 1 \) to \( n \) do

\( \text{Execute forecasting process} \)

end

Algorithm 2 TPF: Time-series pattern based forecasting algorithm

In the TPF algorithm, we first utilise the time-series segmenting and pattern generation algorithm (TSPG) and the pattern matching and forecasting algorithm (PMF) to execute various time-series patterns based forecasting processes which are divided from an entire probabilities forecasting process. In each single process, we deduce time-series patterns by segmenting on past probabilities, and utilise these patterns to match the current probabilities to forecast probabilities in the next time interval. Then, we combine these results from these processes. At last, we normalise these results to integrate one group of “future” probabilities of real requests for noise generation. Compared to [27], TPF algorithm spends a lot of efforts on the utilisation of forecasting results and their normalisation for noise generation.

\[ \forall i, P(Q_{t+1} = q_i(t)) = \frac{\text{MAX}(\forall j, \text{TPFA}(P(Q_t = q_j(t)), t' \in [1, t-1])) - \text{TPFA}(P(Q_{t+1} = q_j(t)), t' \in [1, t-1])}{n \times \text{MAX}(\forall j, \text{TPFA}(P(Q_t = q_j(t)), t' \in [1, t-1])) - 1} \]

IV. NOVEL TIME-SERIES PATTERN BASED NOISE GENERATION STRATEGY

Based on the previous section, in this section, we first analyse our time-series pattern based noise generation with noise generation probabilities and noise injection intensity, and then we propose our novel time-series pattern based noise generation strategy for privacy protection in cloud computing (TPNGS).

A. Time-series pattern based noise generation

In this section, we introduce the two keys of our noise generation strategy—noise generation probabilities and noise injection intensity. In the process of noise generation, noise generation probabilities determine which kinds of noise requests should be generated and the noise injection intensity controls how many noise requests should be generated.

1) Noise generation probabilities

On the basis of [6], we can present our time-series pattern based noise generation probabilities. We add parameter \( t \) to denote the time attribute in noise generation processes. Then, we have noise generation probabilities in TPNGS.

\[ \forall i, P(Q_t = q_i(t)) = \frac{M(t) - P(Q_{t+1} = q_i(t))}{n \times M(t) - 1} \]

In equation (1), \( M(t) \) is that for every \( i \), the largest \( P(Q_t = q_i(t)) \) at time \( t \).

\[ M(t) = \text{MAX}(\forall i, P(Q_t = q_i(t))) \]

Based on equations (1) and (2), we can get \( P(Q_t = q_i(t)) \) which is an important part of noise generation probabilities.

\[ \forall i, P(Q_t = q_i(t + \Delta t)) = \text{TPFA}(P(Q_t = q_i(t')), t' \in [1, t]) \]

In equation (3), \( \text{TPFA}() \) denotes the function of our TPF algorithm in Algorithm 1. Hence, equation (3) is a key contribution of our paper: we use past requests' probabilities to forecast future requests' probabilities to aid noise generation by time-series patterns. We set \( \Delta t = 1 \). Then we have equation (4) below.

\[ \forall i, P(Q_t = q_i(t)) = \text{TPFA}(P(Q_{t+1} = q_i(t')), t' \in [1, t - 1]) \]

Combining equations (1), (2) and (4), we can get final noise generation probabilities in TPNGS by equation (5).
2) Noise injection intensity

To reach the goal of privacy protection discussed in Section I, we try to get final “indistinguishable” probabilities.

\[ \forall i, \forall t, P(Q_s = q_i)(t) = \frac{1}{n} \quad (6) \]

From the noise injection model in Section III, we have the following probabilities.

\[ \forall i, P(Q_s = q_i)(t) = (1 - \epsilon)P(Q_r = q_i)(t) + \epsilon P(Q_r = q_i)(t) \quad (7) \]

Combining equations (6) and (7), we can derive noise injection intensity to reach the goal privacy protection.

\[ \epsilon(t) = 1 - \frac{1}{n \times M(t)} \quad (8) \]

To realise equation (8), we have equation (9).

\[ \epsilon(t) = 1 - \frac{1}{n \times \max(TPF.A(\forall i, P(Q_r = q_i)(t'), t' \in [1, t-1]))} \quad (9) \]

Equations (5) and (9) can make the whole strategy to reach its goal, i.e. equation (6).

Compared to existing noise generation strategies, like HPNGS or random generation, TPNGS enhances the outcome of privacy protection from \( \forall i, P(Q_s = q_i) = \frac{1}{n} \) to equation (6). It can now address the serious privacy risk identified in Section I. Besides, it is clear that the goal of TPNGS, i.e. realisation of equation (6), is a sufficient condition of the goal of existing strategies: \( \forall i, P(Q_s = q_i) = \frac{1}{n} \). So, if occurrence probabilities are about the same in every local time intervals, these probabilities will be about the same in the overall time period.

B. Time-series pattern based noise generation strategy

In this section, on the basis of the former sections, we present our novel time-series pattern based noise generation strategy—TPNGS.

In Algorithm 2, we can see that the major improvement of TPNGS is to use \( \forall i, TPF.A(P(Q_r = q_i)(t'), t' \in [1, t-1]) \) as forecasting results in Step 1. As stated earlier, the TPF algorithm is our time-series pattern forecasting algorithm for noise generation which utilises time-series patterns to summarise past probabilities and forecast “future” probabilities. In our strategy, we use the TPF algorithm in the first step to forecast, and utilise the results of the TPF algorithm in later steps (Step 2 and Step 3)—computing noise generation probabilities and noise injection intensity. It is obvious that our strategy performs better in the privacy protection situation with fluctuations of probabilities than existing strategies, like HPNGS or random generation. In Step 4, noise injection processes have been executed. Besides, at an extreme condition without fluctuations, it is clear that TPNGS and HPNGS could perform similarly in noise generation, for there is no need to forecast.

In Section V, we evaluate TPNGS with other strategies.

Title: Time-series pattern based noise generation strategy

Input: \( Q_s \) is the queue of real service requests
Output: \( Q_i \) is the queue of final service requests and the privacy protection result

1. Collect all past probabilities and utilise the TPF algorithm
2. Compute noise generation probabilities
3. Compute noise injection intensity
4. Execute noise injection process

Algorithm 2 TPNGS: Time-series pattern based noise generation strategy

V. EVALUATION

In this section, we perform an experimental simulation in our cloud simulation system called SwinCloud [29] (Swinburne Cloud Simulation Environment). The aim is to simulate our novel TPNGS in order to demonstrate that TPNGS can improve the effectiveness of privacy protection significantly, compared to existing noise generation strategies, like HPNGS or random generation.

Besides, how to deal with distributed denial-of-service (DDoS) attacks has become a very serious issue concerned by servers. But in this paper, we omit the possibility of our noise being viewed as DDoS attack. In fact, the number of our noise is much less than a common DDoS attack which normally has millions of requests [30].

A. Simulation background and environment

SwinCloud is a cloud simulation environment [29]. It is built on the computing facilities in Swinburne University of Technology and takes advantage of existing SwinGrid system. In general, the functions of VMWare can offer unified computing and storage resources.

B. Simulation process

The simulation process is to compute and compare the privacy protection effectiveness of TPNGS with that of HPNGS. In this process, we choose HPNGS to compare with our TPNGS. In the end of this section, we demonstrate the comparison between HPNGS, TPNGS and random generation. Before the simulation, we generate a service queue as the real service queue from a set with the size of 10 randomly. So we can not design some fluctuations on purpose to facilitate TPNGS.
We set a function \( EPP(\text{Strategy}) = VAR(\text{Strategy}) \) to express the effectiveness of privacy protection to compare two strategies. As discussed in Section I, the variance of these probabilities of final requests is a suitable tool to evaluate the effectiveness of privacy protection. \( VAR(\text{Strategy}, t) \) means that the variance of all occurrence probabilities of requests in \( Q_2 \) is under \( \text{Strategy} \) at time \( t \). A low variance of all probabilities denotes that all occurrence probabilities of final requests are about the same, and unethical service providers cannot find out real ones as introduced in Section I. Therefore, the less \( EPP(\text{Strategy}, t) \), the better effectiveness of privacy protection that can be achieved.

At the worst of condition for \( TPNGS \) executing; one request is one pattern. Pattern generation is pre-computing. So, at noise generation processes, \( TPNGS \) only need to traverse all patterns or requests to get the matched one, and this cost could not influence noise generation processes significantly, compared to other existing typical strategies.

**C. Simulation results and analysis**

In this section, we derive \( EPP(\text{HPNGS}, t) \) and \( EPP(\text{TPNGS}, t) \) which denote the effectiveness of privacy protection under these two noise generation strategies at time \( t \), respectively. They are depicted in Figure 2. They change by \( t \) from 0 to 5000. In Figure 2, the horizontal coordinate is time \( t \). The vertical coordinate is \( EPP \) reflecting privacy protection effectiveness. If \( EPP \) is lower, the privacy protection effectiveness is better.

**Figure 2 Comparison between \( HPNGS \) and \( TPNGS \)**

We can see that with time \( t \) passing, both \( EPP(\text{HPNGS}, t) \) and \( EPP(\text{TPNGS}, t) \) keep a pattern of fluctuating in specific zones. \( EPP(\text{HPNGS}, t) \) fluctuates mainly between 0.00004 and 0.0001 while \( EPP(\text{TPNGS}, t) \) fluctuates between 0.00001 and 0.00004. Therefore, in general, \( EPP(\text{TPNGS}, t) \) is about 1/3 of \( EPP(\text{HPNGS}, t) \) from the figure. Therefore, we can conclude that our novel \( TPNGS \) significantly improves the effectiveness of privacy protection than existing \( HPNGS \).

Besides, in Figure 3, we could find out that in the whole simulation process, noise injection intensities of \( TPNGS \) are smaller than those of \( HPNGS \). They fluctuate in the levels of 0.45 and 0.7, respectively. Just like introduced before, noise injection intensity is the probability of noise requests in final requests, and it could express the number of noise requests with certain real requests. As mentioned in Section I, due to the pay-as-you-go style in cloud computing, the number of noise requests means the cost of noise requests. So, our \( TPNGS \) can save the cost of noise than existing \( HPNGS \) and improve the efficiency of noise obfuscation on privacy protection. In other words, \( TPNGS \) use small sliding windows to analyse time-series data, not like \( HPNGS \) use whole queues to be a sliding window. For one specific time, noise obfuscation only considers and obfuscates a piece of data, not the entire one. That is why \( TPNGS \) could get lower cost on noise than \( HPNGS \).

**Figure 3 Comparison on noise injection intensity**

In summary, our novel time-series pattern based noise generation strategy (\( TPNGS \)) could make a significant improvement on the effectiveness of privacy protection with a decreased noise service cost in comparison to historical probability based noise generation strategy (\( HPNGS \)).

About random noise generation [7], the effectiveness and efficiency of privacy protection have been discussed in [6]. So, it is obvious that our \( TPNGS \) can improve both effectiveness and efficiency of privacy protection from \( HPNGS \) which mainly improve the efficiency of privacy protection from random noise generation. Therefore, our \( TPNGS \) improve the effectiveness and efficiency of privacy protection from exiting typical noise generation strategies.

**VI. CONCLUSIONS AND FUTURE WORK**

In open cloud computing environments, customers' privacy protection is a challenge as malicious service providers may record customer service data and then collectively deduce customers' privacy. Noise obfuscation is an effective approach to deal with this. For example, it generates and injects noise service requests into real customer service requests to make sure that their occurrence probabilities are about the same. So service providers cannot distinguish which requests are real ones. However, existing typical noise generation strategies only focus on the entire service usage without probability fluctuations. In fact, such probabilities can fluctuate in different time intervals by which service providers may be able to deduce the customers' privacy for the time interval although they
cannot deduce the customers’ privacy for the entire period.
To address this problem, we developed a novel time-series
pattern based noise generation strategy (TPPNGS). On the
basis of past occurrence probabilities, we summarised time-
series patterns to describe fluctuations in past occurrence
probabilities. After that, these patterns could be utilised to
forecast occurrence probabilities in the next time interval.
Hence, based on these forecasting, noise generation could be
more accurate and effective in the next time interval. The
simulation evaluation demonstrated that our strategy could
cope with these fluctuations, i.e. significantly improve the
effectiveness of privacy protection in local time intervals.
Besides, the efficiency of privacy protection could be
improved by TPPNGS, too.
Based on TPPNGS, we plan to investigate how to protect
customers’ privacy in the scenario where multiple malicious
service providers may collaborate with each other. Besides,
these time intervals could be investigated in terms of
variability and dynamic to match some other attack models.

REFERENCES
[1] Rajkumar Buyya, Chee Shin Yeo, Srikrishna Venugopal, James
Broberg, and Ivona Brandic, “Cloud Computing and Emerging IT
Platforms: Vision, Hype, and Reality for Delivering Computing as
599-616, June 2009.
Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David A.
 Patterson, Ariel Rabkin, Ian Stoica, and Z. M. Tari, “Above the Clouds:
A Berkeley View of Cloud Computing,” Communications of the ACM,
Manager for Cloud Computing,” presented at the 1st International
Doorstep,” Communications of the ACM vol. 54, pp. 36-38, January
2011.
[5] Stefan Sackmann, Jens Ströker, and Rafael Accorsi, “Personalization
in Privacy-Aware Highly Dynamic Systems,” Communication of the
Probability based Noise Generation Strategy for Privacy Protection in
Cloud Computing,” Journal of Computer and System Sciences,
Elsevier, Published online: http://dx.doi.org/10.1016/j.jcss.2011.12.020 ,
for Search Privacy Protection,” presented at the 2009 International
Conference on Computational Science and Engineering, Vancouver,
Canada, 2009.
[8] Xin Huang, Yin He, Yifan Hou, Lisi Li, Lan Sun, Sina Zhang, Yang
Jiang, and Tingting Zhang, “Privacy of Value-Added Context-Aware
Service Cloud,” presented at the 1st International Conference on
[9] Koen Simons, Pim Tuyt, and Bart Preneel, “Privacy Weaknesses in
Biometric Sketches,” presented at the 30th IEEE Symposium on
Security and Privacy, Oakland, California, USA, 2009.
[10] Rezaeo Nereae, Dominic Heiling, and Alexander Pretschner,
“Implementing Trust in Cloud Infrastructures,” presented at the 11th
IEEE/ACM International Symposium on Cluster, Cloud and Grid
Computing (CCGrid2011) New Beach, California, USA, 2011.
[12] Alexandre Efremovskii, Johannes Gehrze, and Ramakrishnan Srikant,
“Limiting Privacy Breaches in Privacy Preserving Data Mining,”
presented at the 22nd ACM SIGMOD-SIGACT-SIGART Symposium
International Journal on Uncertainty, Fuzziness and Knowledge-
[14] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and M. Sudan,
981, November 1998.
Constructions for Information-Theoretic Private Information
Retrieval,” Journal of Computer System Science, vol. 71, pp. 213-247,
August 2005.
[16] Ian Goldberg, “Improving the Robustness of Private Information
Retrieval,” presented at the 2007 IEEE Symposium on Security and
Privacy, Oakland, California, USA, 2007.
[18] Dingding Renger, Nick Mathew, and Paul Syverson, “Tor: The
Second-Generation Onion Router,” presented at the 13th USENIX
Networks,” presented at the 30th IEEE Symposium on Security and
Privacy, Oakland, California, USA, 2009.
[20] Claudio A. Ardacli, Marco Cremoneini, Sabrina De Capitani di
Vermicari, and Pierangela Samarati, “An Obfuscation-Based
Approach for Protecting Location Privacy,” IEEE Transactions on
Wireless Network Secrecy,” presented at the 28th Conference on
Computer Communications (IEEE INFOCOM 2009), Rio de Janeiro,
Brazil, 2009.
Channel,” presented at the 1993 IEEE Symposium on Security and
Privacy, Oakland, California, USA, 1993.
[23] Gaofeng Zhang, Yan Yang, Dong Yuan, and Jinjun Chen, “A Trust-
Based Noise Injection Strategy for Privacy Protection in Cloud
Computing,” Software: Practice and Experience, Wiley. Published
presented at the 41st Annual ACM Symposium on Theory of Computing,
(STOC09), Bethesda, Maryland, USA, 2009.
of Computation using Multiple Servers,” presented at the 18th ACM
Conference on Computer and Communications Security, Chicago,
Illinois, USA, 2011.
[26] Eamonn Keogh, Selina Chui, David Hart, and Michael Pazzani, “An
Online Algorithm for Segmenting Time Series,” presented at the 2001
IEEE International Conference on Data Mining, San Jose, California,
USA 2001.
[27] Xiao Liu, Jinjun Chen, Ke Liu, and Yun Yang, “Forecasting Duration
Intervals of Scientific Workflow Activities Based on Time-Series
Patterns,” presented at the IEEE Fourth International Conference on
e-Science, Indianapolis, Indiana, USA, 2008.
[28] Elaine Shi, T-H. Hubert Chan, Eleanor Rieffel, Richard Chow, and
Dawn Song, “Privacy-Preserving Aggregation of Time-Series Data,”
presented at the 18th Annual Network & Distributed System Security
Symposium, San Diego, California, USA, 2011.
[29] Xiao Liu, Dong Yuan, Gaofeng Zhang, Jinjun Chen, and Yun Yang,
“SwiDcW-C: A Peer-to-Peer Based Cloud Workflow System for
Managing Instance Intensive Applications,” in Handbook of Cloud
Voeker, and Stefan Savage, “Inferencing Internet Denial-of-Service
Activity,” ACM Transactions Computer Systems, vol. 24, pp. 115-139,
May 2006.