

A trust-based noise injection strategy for privacy protection in cloud

Gaofeng Zhang^{*,†}, Yun Yang, Dong Yuan and Jinjun Chen

*Faculty of Information and Communication Technologies, Swinburne University of Technology,
Hawthorn, Melbourne 3122, Australia*

SUMMARY

Cloud promises users that they can present and deploy IT services in a pay-as-you-go fashion in an open and virtualized cloud environment while saving huge capital investment in their own IT infrastructure. In this sense, protection of users' privacy is critical and has become one of the most concerned issues as otherwise users may eventually lose the confidence and passion of deploying cloud in practice. Under some special cloud circumstances, some users' privacy, such as plans or habits, could be induced from their service requests by service providers without permissions from users. In this regard, obfuscation strategy can protect this kind of privacy by injecting 'noise' service requests to confuse potential 'immoral' service providers. However, existing noise obfuscation strategies focus on single noise injection whereas investigation of noise injection architecture has been neglected. Especially, a common service pattern in inter-clouds environment, the cooperative service process including different service providers, makes the risk of privacy serious and uncontrollable by the spread of users' privacy. To address this, we present a novel trust-based noise injection strategy for privacy protection in cloud. To support the strategy, we describe our noise injection architecture in cloud which specializes in the relations between various service roles in inter-clouds based on our trust model. The simulation can demonstrate that our noise injection strategy could significantly improve the effectiveness of privacy protection. Copyright © 2011 John Wiley & Sons, Ltd.

Received 26 July 2010; Revised 29 October 2010; Accepted 6 December 2010

KEY WORDS: cloud; privacy protection; noise injection strategy; trust

1. INTRODUCTION

Cloud is positioning itself as a new promising platform for delivering information infrastructure and resources as IT services [1, 2]. Users can access these services for executing their business activities in a pay-as-you-go fashion while saving huge capital investment in their own IT infrastructure [3]. However, users often concern about whether their privacy can be protected when presenting their IT services or deploying others on cloud since they do not have much control of cloud [4]. Besides, privacy in cloud has a feature that the major part of its usage and management is on the service side which is also out of the direct control of privacy owners [5]. Therefore, privacy protection is critical and one of the most concerned issues in cloud. Without it, users may treat cloud as an unsafe technology and eventually reject it.

Although there are many organizations with Internet services, such as banks and real estates, which are under various and strict legislations and policies to protect their users' privacy, it cannot be ignored that a large number of service providers is or will be an inherent part of an open

*Correspondence to: Gaofeng Zhang, Faculty of Information and Communication Technologies, Swinburne University of Technology, Hawthorn, Melbourne 3122, Australia.

†E-mail: gzhang@swin.edu.au

cloud. Thus, such service providers may and could record service requests from a user and then collectively induce the user's privacy information without permissions from the user. Facing this serious risk, users should take some measures to aid them to protect their own privacy without cooperation from service providers. Noise obfuscation strategy is one such measure which aids users to protect privacy at the client side. It injects 'noise' service requests into real users' service requests so that service providers cannot distinguish which requests are real ones if their occurrence probabilities are about the same. Regarding obfuscation, service providers are unable to distinguish whether a specific service request is a real one or a 'noise'. This is a key foundation of this paper. We use an example to illustrate the noise obfuscation strategy. For example, a user who often travels to 'Sydney' checks the weather report regularly from a weather service in cloud before departure. The frequent appearance of service requests about the weather report for 'Sydney' can reveal the privacy that the user usually goes to 'Sydney'. But if a system aids the user 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'. It just sees a similar style of service request which should be responded and cannot reveal the location privacy of the user. In such cases, the privacy can be protected by the noise obfuscation strategy.

Currently however, the existing noise obfuscation strategies focus on the noise usage for a single-service process and there is only a single noise injection. In other words, there is only one service provider to answer users' service requests which may include both real and 'noise' service requests. Actually, users' service requests may need more than one service provider to answer them [6]. It is a cooperative service process with various service providers. In cloud, the style of inter-clouds with public clouds and privacy clouds is easily accepted for its flexibility on the balance of security and efficiency [4]. Especially in this inter-clouds environment, a cooperative service process could bring several service providers from different clouds together, and the risk of privacy increases with more and more service providers taking part in the process due to the spread of privacy. In brief, the service feature of inter-clouds is beyond the area of existing noise obfuscation strategies, bringing more challenges to privacy protection. These cooperative service processes are invisible to users as that is the cloud's feature of virtualization. In general, we have to protect users' privacy during the entire cooperative service process. In this paper, we use 'users' to denote real people who have privacy to be protected in cloud, and 'clients' to denote 'virtualized' clients which communicate with service providers in cloud. Hence, to some extent, a 'client' is a network terminal which applies noise obfuscation strategies to protect the privacy of the 'user' that utilizes the 'client' to get services in cloud. It is the computer which a 'user' uses. In the rest of this paper, we will utilize the 'clients' to associate with service providers to distinguish from 'users' who are real people. Clients delegate users to communicate with services providers. Thus, our strategy is part of 'clients' and is a kind of automatic aid for users. In this paper, we also name the noise obfuscation strategy as noise injection strategy.

To address the noise injection problem as introduced above, we investigate noise injection architecture for entire cooperative service processes in cloud. It specializes in various single-service processes with service roles in cloud based on a trust model. The trust model and privacy risk are basic supporting functions to fulfil the architecture. Based on this, we present our trust-based noise injection strategy (TNIS) for privacy protection in cloud, which protects users' privacy during the entire process of services' cooperation. In the strategy, we use 'noise' service requests to protect users' privacy in a cooperative service process by not only clients, but also other service providers. The trust model and noise injection model are bridges to connect clients and services as a whole for noise injection architecture. The noise injection architecture is used to aid in describing cooperative service processes. It provides a supporting environment for our TNIS. Thus, our strategy focuses on the procedure of cooperative service processes and protects users' privacy during entire cooperative service processes.

As regards inter-clouds, IBM is ambitious to provide an entire business solution on cloud's deployment, including servers, network and software, especially services in inter-clouds [7]. Pearson *et al.* [4] have started their work in inter-clouds. Yan *et al.* [8] use hierarchical identity-based cryptography to realize the mutual authentication in inter-clouds. It is noted that the inter-clouds

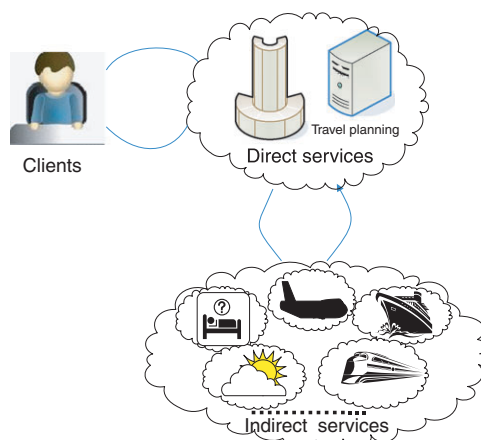


Figure 1. A cooperative service process in cloud.

architecture has been accepted widely, and it is going to take a significant position in business applications. Hence we emphasise on cooperative service processes in inter-clouds in this paper.

From the above we can see that our strategy in this paper requires service providers to protect users' privacy by cooperating with clients. Service providers have the motivation to keep users' privacy safe during the services between themselves and other service providers by noise injection strategy. For instance, service requests sent by this service could also leak users' privacy, just like users' classification and/or distribution of information. This could be a weapon for its business competitors. To address this, service providers could protect their users' privacy by facilitating noise injection strategy. Thus, our strategy is feasible to be applied in practice. For simplicity, we use 'service' to replace 'service provider' in this paper.

Let us take a travel planning service as an example. In Figure 1, a client sends a service request to a travel planning service, and this service sends its service requests to other services in different clouds to get travel information, such as flight service and hotel service. These services answer service requests back to the travel planning service which analyses all information from the answers to provide one or several travel plans and responds back to the client. It is clearly a cooperative service process. Obviously, some users' privacy may be leaked by some potential 'immoral' services in the travel planning process which needs several different services to respond, such as location information and travel itinerary. And it can be associated with one user's identity from other public information to break the user's identity privacy protection. Hence, the risk of privacy is serious. Besides, the travel planning service is provided by a cooperative service process among direct services and indirect services in inter-clouds architecture where direct services provide services for clients directly, and they have trust relations with clients for users' judgements from other non-technical factors, such as privacy terms in business contracts; indirect services provide services for clients indirectly, and they do not have trust relations with clients for the same reason, but they have trust relations with direct services. We will describe these services in inter-clouds in detail in Section 3.1.2. As discussed before, we should use noise injection strategy to protect privacy by a cooperative method in inter-clouds. This is the motivation of our TNIS for privacy protection in cloud.

This paper proposes our novel TNIS for privacy protection in cloud. It focuses on privacy protection in cooperative service processes, and utilizes our trust model to guide noise generation and injection during entire cooperative service processes. We present our noise injection architecture to aid in describing cooperative service processes from the perspective of noise protection, and utilize single-service processes to simplify the realization of this process in our TNIS. Thus, users' privacy can be fully protected in entire cooperative service processes.

The remainder of the paper is organized as follows. In Section 2, we overview the related work. In Section 3, we introduce our noise injection model, trust model and noise injection architecture

step by step. Then we present our novel TNIS for privacy protection in cloud in Section 4. In Section 5, we make evaluations by describing our simulation to demonstrate that our noise injection strategy can improve the effectiveness of privacy protection significantly. Finally in Section 6, we conclude and point out the future work.

2. RELATED WORK

There are many papers about privacy-preserving data mining (PPDM), privacy information retrieval (PIR), anonymity browsing and noise obfuscation in privacy protection.

PPDM reveals a view of privacy or security leakage in the minutiae [9], and the bibliography [10] is a good introduction to this area. To protect privacy, Evfimievski *et al.* [11] use a randomization operator to discuss the problem of association rule mining. Liu *et al.* [12] inspect and design some approaches to apply it into the real world instead of a typical small data set or a particular family of data mining algorithms.

Different from PPDM, PIR looks at another angle to keep privacy safe, which is primarily in preventing database operators from knowing clients' interested records. From the background of information theory, Chor *et al.* [13] conclude that to get a perfect protection, a client has to query all the entries in the database when dealing with a single server framework. Beimel *et al.* [14, 15] and Goldberg *et al.* [16] prevail on the information theoretical to research in PIR. Cachin *et al.* [17] allow servers to use polynomial-time computational capabilities in most cases to keep privacy safe. Yinan *et al.* [18] use the extended attribute-based encryption to research the hierarchical relations in PIR.

In another privacy-preserving area, proxies and anonymity networks have been widely discussed to protect user's privacy [19]. The major goal is that customers keep anonymity or 'invisibility' in a complex or 'dangerous' network situation. Onion routing [20] and its successor TOR [21] provide a sophisticated privacy protection scenario, which makes it difficult for attackers to trace users via network traffic analysis. Narayanan *et al.* [22] present a framework for analysing privacy in social networks, and develop a topology-based re-identification algorithm targeting at anonymous social network graphs. Hawkey *et al.* [23] research the identity anonymity to protect privacy in Web 2.0.

Regarding obfuscation and noise injection strategies, Ardagna *et al.* [24] focus on the location privacy protection in a mobile environment, and present a solution based on different obfuscation operators. Ficco *et al.* [25] protect the location privacy by a hybrid positioning system to obfuscate adversaries. Noise injection is a widely used strategy to protect information privacy and security, and it builds on the ground of the information theory to cover the characters of information [26]. Ye *et al.* [27] describe the noise injection in search privacy protection, with formulating the noise injection problem as a mutual information minimization problem. These existing noise injection strategies however, only consider one service's request response processes, and neglect the real complex service processes in cloud. This is what we address in this paper.

There are some other privacy protection strategies which should be discussed: PPDM is not suitable for this situation, and the major reason is that PPDM specializes in the data mining area. Besides, it is almost impossible for clients to be sure of the availability of PPDM in service providers. The PIR is a kind of schema dividing an entire request response service process into service providers and clients, and making sure that each part of the process cannot induce the entire privacy. But in a virtualized cloud's environment, clients are unlikely to locate all service providers in cooperative service processes. Hence, it is almost impossible to be applied in this situation. Based on the same reason, anonymity networks are also unsuitable.

3. NOISE INJECTION ARCHITECTURE FOR PRIVACY PROTECTION

To present our TNIS, we need to present our noise injection architecture for the strategy. To introduce the noise injection architecture, we need our trust model to connect all single-service processes and noise injections. In this section, we introduce our trust model in Section 3.1. In

Section 3.2, we describe noise injection models. We present our noise injection architecture in Section 3.3.

3.1. Trust model in cloud

Trust model is the foundation for the work described in this paper. It is based on other mature work with improvements. We need to utilize the model to generate and inject ‘noise’ service requests to protect users’ privacy.

There are many research papers about trust models in different applications. Boursas *et al.* [28] use multidimensional dynamic trust management to deal with the situation of large-scale distributed system. Squicciarini *et al.* [29] work on trust negotiation to improve privacy preserving. Bacon *et al.* [30] discuss access control and trust model in widely distributed services. Etalle *et al.* [31] use integrity constraints to maintain control in trust delegating. Golle *et al.* [32] present the trust problem between pollsters and respondents and use a ‘privacy-bond’ to keep mutual trust. Li *et al.* [33] use the trust model to improve the interoperability of cloud. They are all valuable references for our trust model to support our TNIS.

3.1.1. Trust relation. In this paper, we use the ‘trust’ between two service roles: a service request initiator and a service request respondent in view of single-service processes. A service request initiator could be a client or a service, and a service request respondent is solely a service. We will describe these service roles in detail in Section 3.3.1. Now we introduce trust relation based on [34].

Trust relation: as a major part of trust model, Trust Relation is a 9-tuple $(P, Q, C, T, D, t, v, p, n)$ which asserts that entity P trusts entity Q , where

- P and Q are the subsets of the set (Ω) of all the entities in a cloud system. In this paper, P and Q always have different role attributes. We will introduce these in Section 3.3 in detail.
- C is defined as the set of $\{auth, exe, code\}$, denoting trust classes for authentication, execution and code.
- T is the set of $\{direct, recommended, derived\}$ to denote different trust types. In this paper, we only consider the type of ‘direct’.
- D is the set of domains of $\{<dn, dt >\}$, dn denotes the name of domain; dt is the type of the domain and $dt \in DN = \{inter, inter\}$ denoting that the trust relation is intra-domain or inter-domain, respectively. We will detail these in Section 3.1.2.
- t is the time constraint when the relation is thought to be valid.
- v is the trust evaluation on this trust relation.
- p is the number of positive experiences associated with this trust relation.
- n is the number of negative experiences associated with this trust relation.

3.1.2. Trust’s domains in cloud. In Section 3.1.1, we have introduced the trust model primarily. We explain domain D of trust relation in this section. It is a basis for our TNIS, and an innovative improvement of our trust model in this paper. In Section 4, domain D will play an important role for noise generation and injection in our TNIS.

As introduced before, there are three members in cooperative service processes: clients, direct services and indirect services. In Figure 2, direct services are services that receive service requests from clients, and have to accept some cooperation from other services to answer clients’ service requests. Indirect services are services that are out of clients’ ‘view’, and may be in other clouds. In general, clients only send service requests to direct services which are ‘trustworthy’ to these clients, and direct services may ask other direct services or indirect services which are ‘trustworthy’ for direct services to help to accomplish the function of service process. These indirect services are invisible to clients. There are three kinds of single-service processes between them: ‘client-direct’, ‘direct-indirect’ and ‘direct-direct’.

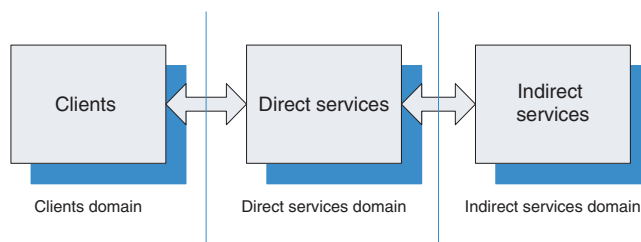


Figure 2. Cooperative service processes with clients, direct services and indirect services.

In this paper, we assume that an indirect service is the last step of a cooperative service process, and it does not send service requests to other indirect services to get cooperation. Thus, we do not have 'indirect-indirect' single-service processes.

In this paper, we consider three domains of trust, 'clients' domain, 'direct services' domain and 'indirect services' domain. They come from three members in cooperative service processes, respectively, hence, we can discuss these trust domains from the perspective of service processes.

The trust between a client and a direct service: As a client, direct services are the only kind of services which it can 'see' and send its service requests to, and the trust from a client to a direct service is a sectional trust from the client to the direct service excluding all other services belonging to cooperative service processes in this or other clouds. Thus, it is not an overall evaluation. In general, this trust has a domain type with an inter-domain, and the domain of this trust is $\langle clients, inter \rangle$. As a direct service, the trust from a direct service to a client belongs to user management and is beyond the scope of this paper.

The trust between a direct service and an indirect service: As a direct service, indirect services are all cooperative resources. The trust from a direct service to an indirect service has an evaluation for distributing tasks to these indirect services in other clouds. In general, this trust has a domain type with an inter-domain, and the domain of this trust is $\langle direct, inter \rangle$. As an indirect service, the trust from an indirect service to a direct service belongs to user management and is also beyond the scope of this paper.

The trust between one direct service and another direct service: As one direct service, other direct services are cooperative resources. When this direct service needs some helps, it may send some requests to other direct services. As a request initiator, the trust is a sectional trust to all other services which take part in this service process, too. In general, this trust has a domain type with an intra-domain, and the domain of this trust is $\langle direct, intra \rangle$. As the request respondent, trust belongs to user management and is also beyond the scope of this paper.

In summary, we have presented our trust model in cloud based on other mature research work and emphasized domain types of trust relations which could be utilized to manage noise generation strategies and noise injections in Section 4. They are essential issues of our TNIS.

3.2. Noise injection models in cloud

First of all, we should note that noise injection models in this paper focus on single-service processes with single noise injections while the noise injection architecture will be introduced in Section 3.3.

3.2.1. Basic noise injection model. In this section, we introduce the existing basic noise injection model from [27]. In Figure 3, this black box model is a suitable description of the process of noise injection.

Q_U : is the queue of the client's real service requests. It is the privacy-protect target.

Q_N : is the queue of 'noise' service requests which is to inject in Q_U and make it confused. It is the privacy-protect tool.

Q_S : is the queue of final service requests which is composed of Q_U and Q_N . It is the privacy-protect result.

Q : is the common set of Q_U , Q_S and Q_N . And $Q = \{q_1, q_2, \dots, q_i, \dots, q_n\}$.

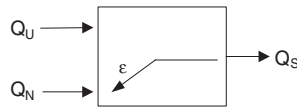


Figure 3. Basic noise injection model.

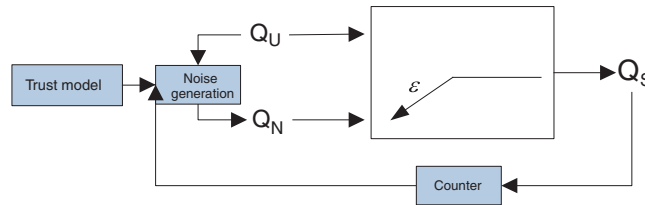


Figure 4. Improved noise injection model.

ε : is the probability of that Q_N can be sent as Q_S , and $\varepsilon \in [0, 1]$. It can also be named the ‘noise injection intensity’.

The major process of the model is to inject Q_N into Q_U to get Q_S . The client generates a service request queue Q_U to be sent. The noise queue Q_N is generated randomly and it is independent of Q_U . To get Q_S , a switch function plays: the next service request in Q_S comes from Q_N on the probability of ε , and from Q_U on the probability of $1 - \varepsilon$.

3.2.2. Improved noise injection model. In this section, to support our TNIS, we introduce our improved noise injection model.

The grey parts in Figure 4 symbolize our improvements to the basic noise injection model overviewed in Section 3.2.1.

‘Trust model’: The trust model is the basis of our TNIS, and it guides noise generation and injection processes.

‘Counter’: it counts service requests in Q_S and aids to generate Q_N in ‘Noise generation’.

‘Noise generation’: its function is to generate Q_N . We use ‘Trust model’ and ‘Counter’ to compute noise generation probabilities in our strategy. In this part, $P(Q_N = q_i)$, $\forall i, i \in [1, n]$ denotes all noise generation probabilities. In other words, the service request in Q_N is q_i on the probability of $P(Q_N = q_i)$. Different strategies need different noise generation probabilities to generate noise. We will describe the different strategies in this generation process in Section 4.

The major process of the model is also to inject Q_N into Q_U to get Q_S . The client generates a real service request queue Q_U to be sent. The noise queue Q_N is generated from ‘Trust model’ and ‘Counter’. A switch function plays: the next service request in Q_S is from Q_N on the probability of ε , and from Q_U on the probability of $1 - \varepsilon$. Regarding ε , it can also be decided from ‘Trust model’ and ‘Counter’, and it will be introduced in our strategy in Section 4.

In summary, we improve the existing basic noise injection model by supporting our noise generation strategies to improve the efficiency of privacy protection.

3.3. Noise injection architecture in cloud

As discussed in Section 1, a cooperative service process may include several service providers from different clouds. Thus, we need a supporting environment for our TNIS which is totally different from existing single noise injection strategies.

With the trust model and improved noise injection model in cloud, the noise injection architecture in cloud is presented in this section based on single-service processes with noise injections.

3.3.1. Single-service process with service roles. As mentioned in Section 3.1.2, two service roles in a cooperative service process have to be classified: service request initiator and service request respondent. To introduce them, we utilize a single-service process to represent one step of an entire

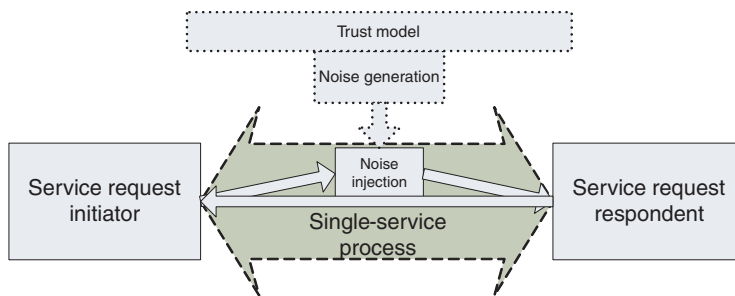


Figure 5. Single-service process with service request initiator and service request respondent.

cooperative service process with only these two members and their service process. We can use this process to cover different domains and clouds in terms of trust.

Service request initiator plays a role that we are mainly concerned in this paper. Trust model is a key part in service request initiator. On the one hand, it directs the real service requests' generation and distribution. On the other hand, it guides the 'noise' service requests' generation and injection by trust relation.

Service request respondent plays another role in the process. In this paper, it receives and responds service requests from the perspective of a service request initiator.

In Figure 5, it depicts a single-service process between a service request initiator and a service request respondent. From the perspective of a service request respondent, the process is very simple and just operates between itself and a service request initiator. But from the perspective of a service request initiator, a noise injection is an essential part of the service requests initiation and generation. The service request initiator protects privacy by noise injection which is invisible to its partner (the service request respondent). In general, the single-service process is a combination of the dual service roles with noise injection, and a process to execute these two service roles. Thus, we utilize single-service processes to model entire cooperative service processes.

'Noise generation' and 'trust model' are important issues to support single-service processes under our noise injection strategy.

In Section 3.1, we introduced the trust model in cloud. In that model, P and Q are subsets of the set (Ω) of all the entities in cloud. We use P to denote service request initiators, and Q to denote service request respondents in our strategy.

In the following section, we will use single-service processes with dual service roles to present noise injection architecture which is the supporting environment for our TNIS.

3.3.2. Noise injection architecture. In Section 3.2.1, we introduced single-service processes with dual service roles which are part of the noise injection architecture. In this section we present our noise injection architecture to support our TNIS. In Figure 6, we have three domains in this architecture: clients domain, direct domain and indirect domain which correspond to the trust domains mentioned in Section 3.1.2. The clients and direct domains are visible to users, and our noise injection strategy is deployed in these two domains to protect users' privacy. In the indirect domain, they are unknown and 'untrustworthy' from the perspective of users. It may be located in other clouds or public clouds.

From the virtualization perspective, we have three layers in the entire architecture: role layer, service layer and deployment layer which correspond to service roles environment, service environment and cloud environment, respectively.

What we focus on is the grey parts in the role layer. Based on single-service processes with service request initiators and service request respondents, noise injections have been highlighted as an essential part in these single-service processes with noise injections. To generate noises to protect privacy, we need noise generation strategies which have been highlighted to take part in noise injections. Besides, these strategies could also be executed by the trust model as highlighted. In Section 4, our novel TNIS will be proposed accordingly.

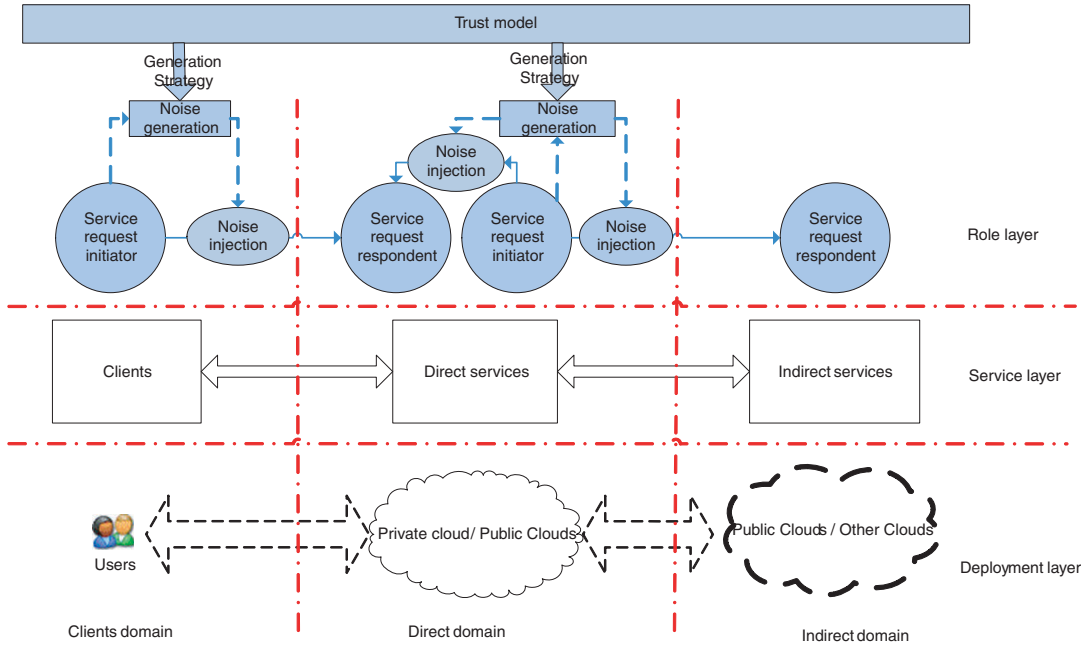


Figure 6. Noise injection architecture.

4. NOVEL TRUST-BASED NOISE INJECTION STRATEGY

In Section 3.3, the noise injection architecture in cloud was presented as a supporting environment to our noise injection strategy. In this section, we propose a novel TNIS. Our TNIS plays its role in single-service processes with noise injections which are the grey parts of noise injection architecture depicted in Figure 6. Every single-service process with noise injections and generations in this architecture utilizes TNIS to manage noise generation and injection to protect privacy.

In the TNIS, there are three noise generation strategies which execute noise generations. The three noise generation strategies have different goals for privacy protection. They come from different kinds of users' privacy.

We set three kinds of users' privacy: $pl \in \{direct\ privacy, probability\ distribution, interaction\ frequency\}$. The first one denotes original service requests without any analysis and induction. The second one denotes that this kind of privacy can be induced from the occurrence probabilities of original service requests. The third one denotes that the interaction frequency of original service requests could induce this kind of privacy. These three kinds of privacies reflect different attackers' goals, so that they can reflect the different goals of privacy protection.

It is easy to understand that each strategy sets one kind of privacy to its goal of noise privacy protection: the goal of the 'slight' noise generation strategy is to cover the $pl = 'direct\ privacy'$; the goal of the 'moderate' noise generation strategy is to cover the $pl = 'probability\ distribution'$; and the goal of the 'serious' noise generation strategy is to cover the $pl = 'interaction\ frequency'$.

Therefore, we can discuss the differences between the 'serious' noise generation strategy and the 'moderate' noise generation strategy. In Steps 3.3 and 3.4 in our TNIS, the two noise generation probabilities formulas are

$$P(Q_N = q_i) = \frac{\text{Max}\{P(Q_U = q_i)\} - P(Q_U = q_i)}{n * \text{Max}\{P(Q_U = q_i)\} - \sum_i P(Q_U = q_i)}$$

and

$$P(Q_N = q_i) = \frac{\text{Max}\{P(Q_S = q_i)\} - P(Q_S = q_i)}{n * \text{Max}\{P(Q_S = q_i)\} - \sum_i P(Q_S = q_i)}$$

Trust-based noise injection strategy: TNIS

Input: Service request queue: Q_U ,

Initial trust relations: $TR = \{tr_1, tr_2, \dots, tr_i, \dots, tr_m\}$,

Risk of privacy: d ,

Quality evaluation of this single-service process: e .

Output: Final service request queue Q_S ,

Updated trust relations: $TR = \{tr_1, tr_2, \dots, tr_i, \dots, tr_m\}$

Step 1: Evaluating trust relation

Input: Trust relations TR ;

Dual service roles in this noise injection: p and q

Output: v denotes the trust value between p and q in the range of $[0, 1]$

From Section 3.1, we have trust relations $TR = \{tr_1, tr_2, \dots, tr_i, \dots, tr_m\}$

where $tr_i = (P_i, Q_i, C_i, T_i, D_i, t_i, v_i, p_i, n_i)$.

Check all t_i in tr_i , remove all tr_i with unavailable t_i .

Check all i which can satisfy conditions: $P_i == p$ and $Q_i == q$.

If i does not exist, Dijkstra shortest path algorithm is used to find out an array of trusts, and a newly derived tr will be inserted into TR . Then this step will be re-executed.

If i exists, if i is unique, $v = v_i$.

if i is not unique, $v = v_j$ (where $j \in$ data set of i and $T_j == \text{'direct'}$)

Step 2: Evaluating risk of privacy in this single-service process with noise injection

Input: tr_i denotes trust relation in single-service process,

$d \in \{\text{'serious', 'moderate', 'slight'}\}$ which denotes initial level of privacy risk from user's judgement.

Output: $d' \in \{\text{'serious', 'moderate', 'slight'}\}$ which denotes final level of privacy risk.

Check D_i from tr_i .

If $(D_i == \langle \text{'clients', 'inter'} \rangle)$, $d' = d$.

If $(D_i == \langle \text{'direct', 'intra'} \rangle)$, $d' = d$.

If $(D_i == \langle \text{'direct', 'inter'} \rangle)$, if $(d == \text{'slight'})$, $d' = \text{'moderate'}$.

if $(d' == \text{'slight'})$, $d' = d$.

Step 3: Settling down noise injection intensity, and generating noise for injecting into service request queue Q_U

Input: Q_U, d', v

Output: Q_S , with Q_N denoted by $P(Q_N = q_i), \forall i$ and ε denotes noise injection intensity

3.1 Settle down the noise generation strategy.

If $(d' == \text{'slight'})$, go to Step 3.2, 'slight' noise generation strategy to be applied.

If $(d' == \text{'moderate'})$, go to Step 3.3, 'moderate' noise generation strategy to be applied.

If $(d' == \text{'serious'})$, go to Step 3.4, 'serious' noise generation strategy to be applied.

These three strategies will be introduced next, respectively.

3.2 The 'slight' noise generation strategy

We generate noise Q_N by noise generation probabilities:

$$P(Q_N = q_i) = \frac{1}{n},$$

and noise injection intensity:

$$\varepsilon = 1 - v$$

Go to Step 3.5.

3.3 The 'moderate' noise generation strategy

We generate noise Q_N by noise generation probabilities:

$$P(Q_N = q_i) = \frac{\text{Max}\{P(Q_U = q_i)\} - P(Q_U = q_i)}{n * \text{Max}\{P(Q_U = q_i)\} - \sum_i P(Q_U = q_i)},$$

and noise injection intensity:

$$\varepsilon' = 2(1 - v) \frac{n * \text{Max}\{P(Q_U = q_i)\} - \sum_i P(Q_U = q_i)}{n * \text{Max}\{P(Q_U = q_i)\}} \text{ and the final } \varepsilon = \text{Max}\{\varepsilon', 1 - v\}.$$

Go to Step 3.5.

3.4 The 'serious' noise generation strategy

We generate noise Q_N by noise generation probabilities:

$$P(Q_N = q_i) = \frac{\text{Max}\{P(Q_S = q_i)\} - P(Q_S = q_i)}{n * \text{Max}\{P(Q_S = q_i)\} - \sum_i P(Q_S = q_i)},$$

and noise injection intensity:

$$\varepsilon(t) = 2(1 - v) \frac{n * \text{Max}\{P[Q_S(t) = q_i]\} - \sum_i P[Q_S(t) = q_i]}{n * \text{Max}\{P[Q_S(t) = q_i]\}} \text{ and the final}$$

$\varepsilon = \text{Max}\{\varepsilon(t), 1 - v\}$ which changes with time t .

Go to Step 3.5.

3.5 Noise injection

We get the noise N from Q_N , and inject it into Q_U on the probability of ε , so we can get Q_S . In this step, we execute this noise injection process as our noise injection model described in Section 3.2.2.

Step 4: Evaluating quality of this single-service process, and updating trust relation in trust model about single-service process

Input: e denotes quality evaluation of this single-service process.
Output: Updated tr_i

4.1 Get a feedback e to denote the quality of the service
In this step, we have to collect feedback e from a service request initiator.

4.2 Update p_i and n_i in tr_i
If $(e \geq v_i)$, $p_i = p_i + 1$
If $(e < v_i)$, $n_i = n_i + 1$

4.3 Update v_i in tr_i
 $v_i = v_i + e \times (p_i - n_i)$

The difference between them means that the ‘*serious*’ noise generation strategy is more sensitive to time element t , and this is the foundation of the goal of ‘*serious*’ noise generation strategy to keep the interaction frequency at a stable level which is its method of privacy protection. As in the ‘*moderate*’ noise generation strategy, the method of privacy protection is to keep all occurrence probabilities of original service requests about the same.

Another issue to be clarified is $\varepsilon(t)$ and $Q_S(t)$ in the noise injection intensity formula

$$\varepsilon(t) = (1 - v) \frac{n * \text{Max}\{P[Q_S(t) = q_i]\} - \sum_i P[Q_S(t) = q_i]}{n * \text{Max}\{P[Q_S(t) = q_i]\}}$$

It is a supplement to fulfil the goal of ‘*serious*’ noise generation strategy by involving time element t .

The last issue is ε ’s generation in these three strategies. When the risk of privacy rises, it changes from $\varepsilon = 1 - v$, $\varepsilon = \text{Max}\{\varepsilon', 1 - v\}$ to $\varepsilon = \text{Max}\{\varepsilon(t), 1 - v\}$ by increasing step by step. It means that if the risk of privacy rises, the noise injection intensity rises with noise generation strategies changing.

In this strategy, we have an initial level of privacy risk d which is decided by users. The analysis of privacy risk is a specific area in privacy protection, which is beyond the scope of this paper. Thus, we use a user-defined initial level of privacy risk, and do some amendments for the final level of privacy risk in our strategy. Similarly, we use users’ quality evaluation to do the updating function of trust relation in Step 4. Trust relations in the trust model could be adapted, and the updating function realizes this for improving the adaptability of the trust model.

In summary, our TNIS for privacy protection in cloud is established on the trust model and noise injection model in cloud. It operates on the noise injection architecture with several single-service processes and dual service roles, and protects users’ privacy during entire cooperative service processes in an inter-clouds environment. We use our novel TNIS to protect privacy better than the existing single noise injection strategy (SNIS) as detailed in the following section.

5. EVALUATIONS

In this section, we introduce an experimental simulation in our cloud simulation system called SwinCloud (Swinburne Cloud Computing Simulation Environment). The simulation executes an instance about a cooperative service process between several services and a client. The aim is to simulate our TNIS in order to demonstrate that our strategy can improve the effectiveness of privacy protection significantly.

In Section 5.1, we describe the simulation environment. We then detail the simulation process in Section 5.2. In Section 5.3, we analyse the simulation results and illustrate the advantage of our strategy.

5.1. Simulation background and environment

SwinCloud is a cloud simulation environment [35]. It is built on the computing facilities in Swinburne University of Technology and takes advantage of the existing SwinGrid system [36]. In general, the functions of VMWare can offer unified computing and storage resources. The architecture of SwinCloud is depicted in Figure 7. In Section 5.2, we describe the simulation process.

In SwinCloud, we set several nodes to represent two service roles—service request initiators and service request respondents, respectively. The former first generate or forward ‘real’ service request queues where every request is from a request set with 50 items. Then they generate noise service request queues to inject into real queues, and they are the same request-sets. The latter receive the final service requests queues and analyse the effectiveness of privacy protection in the final service requests queues. Some nodes represent dual roles as the intermediate steps of cooperative service processes. That is the foundation of the simulation process.

5.2. Simulation process

To demonstrate the advantage of our strategy, we set the simulation process as a cooperative service process with the spread of privacy. In this process, we set the cooperative service process with all *Services* as a linear structure depicted in Figure 8. It is obvious that a linear service structure can express a cooperative service process clearly.

In Figure 8, $Client_1$ (it also can be viewed as $Service_0$) sends service requests to $Service_1$ with noise protection, then $Service_1$ operates this information and sends to the next service— $Service_2$ with noise protection too, and the same steps are repeated until $Service_9$ which is the last service to accomplish the client’s service requests.

Except the last service, all services and the client can inject noise to protect privacy according to some noise injection strategies. In the simulation process, we make a comparison between our TNIS and the existing SNIS. To evaluate the effectiveness of privacy protection, we set $r_i(TNIS)$

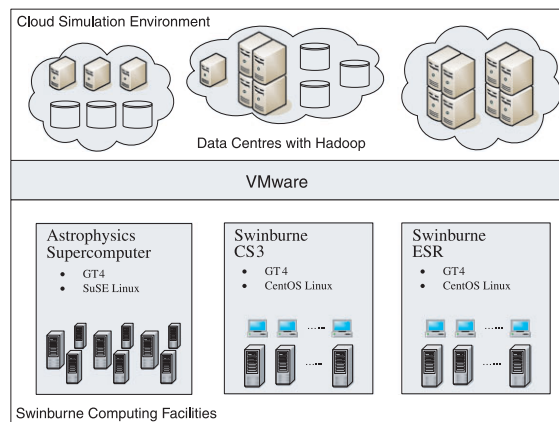


Figure 7. SwinCloud Infrastructure.

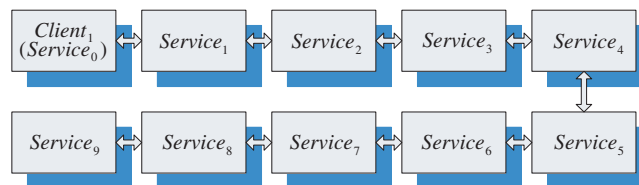


Figure 8. Linear service structure.

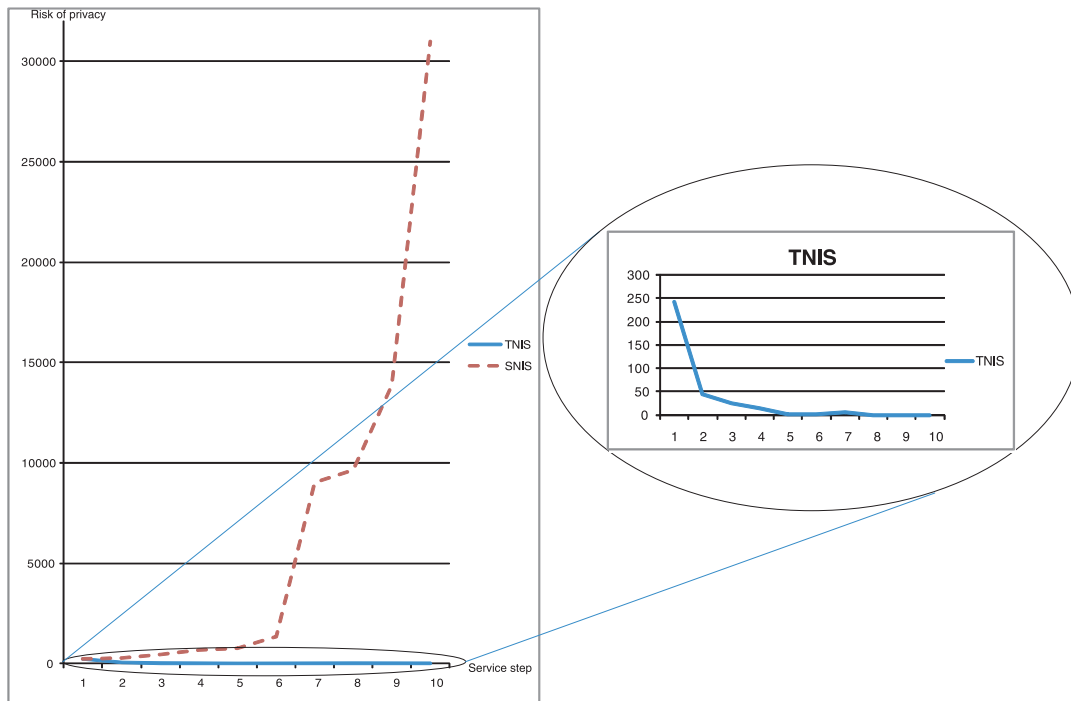


Figure 9. Comparison between $r_i(TNIS)$ and $r_i(SNIS)$.

and $r_i(SNIS)$ from $r_i = \prod_i (\varepsilon_i/v_i)$ which denotes the risk of privacy under the protection of one noise injection strategy in step i . ε_i is the noise injection intensity in the service step between $Service_{i-1}$ and $Service_i$. v_i is the trust value between $Service_{i-1}$ and $Service_i$. It is obvious that a lower r_i means a better effectiveness of privacy protection in step i .

Regarding $SNIS$, we utilize it to represent various existing various noise obfuscation strategies in the view of noise injection because they all focus on one single-service process with single noise injection [24–27], regardless of which kinds of noise generation strategies or methods they have.

This comparison based on simulation results is described in the following section.

5.3. Simulation results and analysis

Based on the simulation process described in Section 5.2, we have $r_i(TNIS)$ and $r_i(SNIS)$. They are depicted in Figure 9 and they change by the step of service process i .

In Figure 9, the horizontal coordinate is the step (i) in the service process. The vertical coordinate is the risk of privacy ($r_i = \prod_i (\varepsilon_i/v_i)$). If the risk of privacy r_i is lower, the effectiveness of privacy protection is better and the user's privacy is safer and vice versa.

Obviously, $r_i(TNIS)$ and $r_i(SNIS)$ have the same start, but later on they are extremely different. The former keeps the risk of privacy in a zone of moderate fluctuation and the latter increases rapidly. With the increasing of i , the disparity between $r_i(TNIS)$ and $r_i(SNIS)$ becomes more and more, and our strategy— $TNIS$ is more and more effective. At $Service_1$, these two privacy risks are the same. At $Service_5$, the disparity of two privacy risks is 614.301times. At $Service_9$, the disparity of two privacy risks is as high as 156 128.55 times. Thus, our strategy can improve the effectiveness of privacy protection significantly and keeps the risk of privacy in a very low zone, especially in complex cooperative service processes.

As a client, it could know the single-service process with noise injection with $Service_1$ instead of this entire service process as discussed in Section 1. That is why $TNIS$ and $SNIS$ start at the same level of privacy protection, however, with other services joining the service process, $SNIS$ could expose the disadvantage and have a much worse effectiveness of privacy protection due to neglecting the entire cooperative service process.

In summary, we can conclude that with our novel TNIS, we can decrease the risk of privacy significantly than existing single noise injection strategies in cooperative service processes in inter-clouds. In other words, we can improve the effectiveness of privacy protection significantly by our strategy in cloud.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel TNIS for privacy protection in cloud. In the strategy, we introduced the noise injection architecture in cloud which specialized in single-service processes with dual service roles in open and virtualized inter-clouds. They are all based on noise injection and trust models. Thus, users' privacy can be protected in every step of entire cooperative service processes by our TNIS. In general, the key work in this paper is to extend noise injection strategies to cooperative service processes for privacy protection in cloud.

In the future, we will improve our work in the following areas:

- Noise injection architecture: in this paper, we focus on the cooperative service process in cloud. But, it is known that cloud is made up of many kinds of service structures and styles. Thus enriching noise injection architecture is a future work.
- Noise generation strategies: in this paper, three noise generation strategies correspond to three kinds of privacies. It is clear that many kinds of privacies need to be protected by noise. In the future, we plan to consider some specialized privacy protection in some particular areas, such as healthcare privacy, to apply our solution in practice.
- Adversary model: in this paper, we have paid little attention to the adversary model. In practical situations, adversary is the guideline to analyse and evaluate privacy protection. To improve the work in this paper, an adversary model is an important tool to enhance privacy protection technology in the future.

REFERENCES

1. Buyya R, Yeo CS, 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* 2009; **25**(6):599–616.
2. Weiss A. Computing in the cloud. *ACM Networker* 2007; **11**(4):16–25.
3. Armbrust M. Above the clouds: A Berkeley view of cloud computing. *Communications of the ACM* 2010; **53**(6):50–58.
4. Pearson S, Shen Y, Mowbray M. A privacy manager for cloud computing. *Proceedings of the First International Conference on Cloud Computing*, Beijing, China, 1–4 December 2009; 90–106.
5. Pearson S. 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*, Vancouver, Canada, 23–23 May 2009; 44–52.
6. Luo X, Xu Z. Generation of similarity knowledge flow for intelligent browsing based on semantic link networks. *Concurrency Computation: Practice and Experience* 2009; **21**(16):2018–2032.
7. IBM Cloud Labs. IBM Perspective on Cloud Computing. Available at: ftp://ftp.software.ibm.com/software/tivoli/brochures/IBM_Perspective_on_Cloud_Computing.pdf, 2009 [5 April 2010].
8. Yan L, Rong C, Zhao G. Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography. *Proceedings of the First International Conference on Cloud Computing*, Beijing, China, 1–4 December 2009; 167–177.
9. Agrawal R, Srikant R. Privacy-preserving data mining. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, TX, U.S.A., 14–19 May 2000; 439–450.
10. Liu K. Privacy Preserving Data Mining Bibliography, 2007. Available at: http://www.cs.umbc.edu/~kunliu1/research/privacy_review.html [6 May 2010].
11. Evfimievski A, Gehrke J, Srikant R. Limiting privacy breaches in privacy preserving data mining. *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, San Diego, CA, U.S.A., 9–11 June 2003; 211–222.
12. Liu L, Kantarcioglu M, Thuraisingham B. The applicability of the perturbation based privacy preserving data mining for real-world data. *Data and Knowledge Engineering* 2008; **65**(1):5–21.
13. Chor B, Kushilevitz E, Goldreich O, Sudan M. Private information retrieval. *Journal of ACM* 1998; **45**(6):965–981.
14. Beimel A, Ishai Y, Kushilevitz E. General constructions for information-theoretic private information retrieval. *Journal of Computer System Science* 2005; **71**(2):213–247.

15. Beimel A, Ishai Y, Kushilevitz E, Raymond J. Breaking the $O(n1/(2k-1))$ barrier for information-theoretic private information retrieval. *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science*, Vancouver, Canada, 16–19 November 2002; 261–270.
16. Goldberg I. Improving the robustness of private information retrieval. *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, Oakland, CA, U.S.A., 20–23 May 2007; 131–148.
17. Cachin C, Micali S, Stadler M. Computationally private information retrieval with polylogarithmic communication. *Advances in Cryptology—EUROCRYPT'99*. Springer: Berlin, Heidelberg, January 1999; 402–414. ISSN: 0302-9743, ISBN: 978-3-540-65889-4.
18. Yinan S, Cao Z. Extended attribute based encryption for private information retrieval. *Proceedings of IEEE 6th International Conference on Mobile Ad-hoc and Sensor Systems (MASS '09)*, Macau, China, 12–15 October, 2009; 702–707.
19. Olivier MS. 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*, White River, South Africa, 20–22 September 2005; 104–112.
20. Goldschlag D, Reed M, Syverson P. Onion routing. *Communications of ACM* 1999; **42**(2):39–41.
21. Dingledine R, Mathewson N, Syverson P. Tor: The second generation onion router. *Proceedings of the 13th Conference on USENIX Security Symposium*, San Diego, CA, U.S.A., 9–13 August 2004; 21–21.
22. Narayanan A, Shmatikov V. De-anonymizing social networks. *Proceedings of the 30th IEEE Symposium on Security and Privacy*, Oakland, CA, U.S.A., 17–20 May 2009; 173–187.
23. Hawkey K. 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, Vancouver, Canada, 29–31 August 2009; 990–995.
24. Ardagna C, Cremonini M, De Capitani di Vimercati S, Samarati P. An obfuscation-based approach for protecting location privacy. *IEEE Transactions on Dependable and Secure Computing*, Preprint, June 2009.
25. Ficco M, Russo S. A hybrid positioning system for technology-independent location-aware computing. *Software: Practice and Experience* 2009; **39**(13):1095–1125. ISSN: 1097-024X.
26. James I, Gray W. On introducing noise into the bus-contention channel. *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, Oakland, CA, U.S.A., 24–26 May 1993; 90–98.
27. Ye S, Wu F, Pandey R, Chen H. Noise injection for search privacy protection. *Proceedings of the 2009 International Conference on Computational Science and Engineering*, vol. 3, Vancouver, Canada, 29–31 August 2009; 1–8.
28. Boursas L, Hommel W. Multidimensional dynamic trust management for federated services. *Proceedings of the 2009 International Conference on Computational Science and Engineering*, vol. 2, Vancouver, Canada, 29–31 August 2009; 684–689.
29. Squicciarini A, Bertino E, Ferrari E, Paci F, Thuraisingham B. PP-Trust-X: A system for privacy preserving trust negotiations. *ACM Transactions on Information and System Security* 2007; **10**(3):50.
30. Bacon J, Moody K, Yao W. Access control and trust in the use of widely distributed services. *Software: Practice and Experience* 2003; **33**(4):375–394.
31. Etalle S, Winsborough W. Maintaining control while delegating trust: Integrity constraints in trust management. *ACM Transactions on Information and System Security* 2009; **13**(1):27.
32. Golle P, Mcsherry F, Mironov I. Data collection with self-enforcing privacy. *ACM Transactions on Information and System Security* 2008; **12**(2):24.
33. Li W, Ping L. Trust model to enhance security and interoperability of cloud environment. *Proceedings of the First International Conference on Cloud Computing*, Beijing, China, 1–4 December 2009; 69–79.
34. Lin C, Varadharajan V, Wang Y, Pruthi V. Enhancing grid security with trust management. *Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04)*, Shanghai, China, 15–18 September 2004; 303–310.
35. Liu X, Yuan D, Zhang G, Chen J, Yang Y. SwinDeW-C: A peer-to-peer based cloud workflow system for managing instance intensive applications. *Handbook of Cloud Computing*. Springer: Berlin, 2010; 309–332. ISBN: 978-1-4419-6523-3.
36. SwinDeW-G Team. System Architecture of SwinDeW-G, 2008. Available at: http://www.swinflow.org/docs/System_Architecture.pdf, 10 May 2010.