Privacy Preserving Service Discovery and Ranking For Multiple User QoS Requirements in Service-Based Software Systems

by

Yin Yin

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2011 by the
Graduate Supervisory Committee:

Stephen S. Yau, Chair
Kasim Candan
Partha Dasgupta
Raghu Santanam

ARIZONA STATE UNIVERSITY

May 2011

ABSTRACT

Service based software (SBS) systems are software systems consisting of services based on the service oriented architecture (SOA). Each service in SBS systems provides partial functionalities and collaborates with other services as workflows to provide the functionalities required by the systems. These services may be developed and/or owned by different entities and physically distributed across the Internet. Compared with traditional software system components which are usually specifically designed for the target systems and bound tightly, the interfaces of services and their communication protocols are standardized, which allow SBS systems to support late binding, provide better interoperability, better flexibility in dynamic business logics, and higher fault tolerance.

The development process of SBS systems can be divided to three major phases: 1) SBS specification, 2) service discovery and matching, and 3) service composition and workflow execution. This dissertation focuses on the second phase, and presents a privacy preserving service discovery and ranking approach for multiple user QoS requirements. This approach helps service providers to register services and service users to search services through public, but untrusted service directories with the protection of their privacy against the service directories. The service directories can match the registered services with service requests, but do not learn any information about them. Our approach also enforces access control on services during the matching process, which prevents unauthorized users from discovering services.

After the service directories match a set of services that satisfy the service users' functionality requirements, the service discovery approach presented in this dissertation further considers service users' QoS requirements in two steps. First, this approach optimizes services' QoS by making tradeoff among various QoS aspects with users' QoS requirements and preferences. Second, this approach ranks services based on how well they satisfy users’ QoS requirements to help service users select the most suitable service to develop their SBSs.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all those who helped me during my PhD study and the writing of this dissertation. Without their help and support, the completion of my PhD degree and this dissertation would not be possible. First, I would like to gratefully thank my supervisor Professor Stephen S. Yau for his guidance and suggestions in my five academic studies. His insight thoughts in this research topic and great working enthusiasm have encouraged me and will further impact my future life. I also would like to thank the rest of my committee, Professors Raghu Santanam, Kasim Candan, and Partha Dasgupta, for their support and invaluable advice and comments on my research.

I am also indebted to Professors Nong Ye and Hessam Sarjoughian, and my colleagues in Distributed Parallel Software Engineering (DPSE) Laboratory and Information Assurance Center at ASU: Dazhi Huang, Ho G. An, Jing Huang, Mohammed Muqsith, Martineraranda Billibaldo, Su Seon Yang, Haishan Gong, Luping Zhu, Zhaoji Chen, Wei Gao, and Min Yan.

I would like to acknowledge the financial support from National Science Foundation (NSF) and DoD / ONR / Multidisciplinary University Research Initiative (MURI).

I would like to express my deep gratitude to my parents for their selfless love, and all my friends for their friendliness when I am living in a new country.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION AND MOTIVATION

Services in information and technology systems are usually referred to software programs or systems running in the background to perform tasks with or without user interactions. For example, the Apache service hosts web sites for users and waits for visitors, and the windows performance service automatically monitors system activities, such as CPU usage and memory usage. In service oriented architectures (SOA), services are generally defined as ”repeatable activities that can be characterized as capabilities or the access to capabilities” and ”SOAs support thinking and organizing in terms of services with distributed capabilities which may be under the control of different ownership domains, and is an architectural style as well as a paradigm for business and IT architecture” [1].

The concept of SOA was first introduced by Alexander Pasik in 1994 when he was working on middleware [2], and was rapidly developed and accepted as an innovative paradigm for distributed computing, especially with the emergence of web services [3] and a set of related standardized protocols [4, 5, 6]. To develop a system with services, called *service-based system (SBS)*, developers first decompose the requirements of the whole system to a set of requirements of services. Each service provides partial functionalities of the whole system, and collaborates with other services through message based communications. This process is similar to the object-oriented paradigm which develops large systems as a set of small objects. However, in object-oriented paradigm, objects’ interfaces, message formats and structures are usually privately specified and implemented within the organization or heavily rely on the software/hardware configurations. In SOA, all services’ interfaces and message protocols are specified with platform independent standards, such as WSDL [4] and SOAP [5]. SOA makes it much easier for developers to reuse existing services in the new

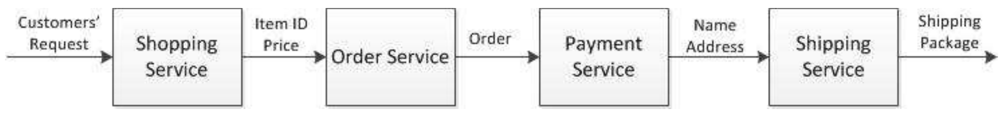

Figure 1.1: An online shopping SBS system with four services

systems rather than to develop the systems from scratch.

For example, an online shopping system can be decomposed to 1) a shopping service to provide an interface for customers to browse and search products, 2) an order service to generate orders for customers, 3) a payment service to charge customers, and 4) a shipping service to ship paid products to customers. All of these services are separate but collaborate together as a workflow showing in Figure 1.1.

SOA can accelarate the development of the above system by allowing developers to composing existing services rather than developing new services. The first phase of composing existing services is to find appropriate services that satisfy system requirements. This task is challenging because services may be developed and owned by different organizations and physically distributed in Internet. The developers of the onine shopping system can manually search services and find that Yahoo's Store Listings is a kind of shopping service, and the public payment gateway like authorize.net is a service that can be composed as a payment service. Then, the developers read documents and learn how to invoke these services in their own system. This manual service discovery process is time consuming and inefficient. An automatic service discovery is preferred.

1.1 Service Discovery and Matching

Service discovery is one of the key problems of SOA and SBS development. One kind of service discovery approach is to broadcast the request for the service in the whole network, such as the peer-to-peer service discovery [7] and WS-Discovery [8]. In such approaches, service providers manage their own services, and service users broadcast their service requests in the whole network whenever they want to find services. The service providers who receive the requests first check whether their own services can satisfy the requests. If yes, the service providers respond the service user (i.e., the SBS developer); otherwise, the service providers either forward the requests to other service providers or abandon the requests. Although this kind of broadcasting approaches can eventually find the requested services if they exist, it generates a lot of network traffic and is a slow process.

Another kind of service discovery approach requires a number of service directories, where service providers publish their services and service users find services. A centralized service directory is similar to a yellow book. However, while yellow books organize local business contact information and support search on business types and locations, centralized service directories need to organize services and support search on service functionalities, which are much more complex than contact information.

The general pattern of discovering services through service directories is illustrated in Figure 1.2 with three phases:

First, service providers (i.e., the developers or organizations who develop or host services) register services in the service directory. The information included in the service registrations depends on the implementation of the service directory. For example, in UDDI [6], a standard implementation of service directory, service registrations include the meta information about services and their providers, and a set of technical detail information called t-models.

Second, service users (e.g., SBS developers) look for services through the service directory by sending service requests to the service directory. The service directory will match registered services with the service requests, and forwards the matched services with their invocation information to the service users.

Third, after the service users find the needed service in the service directory, the

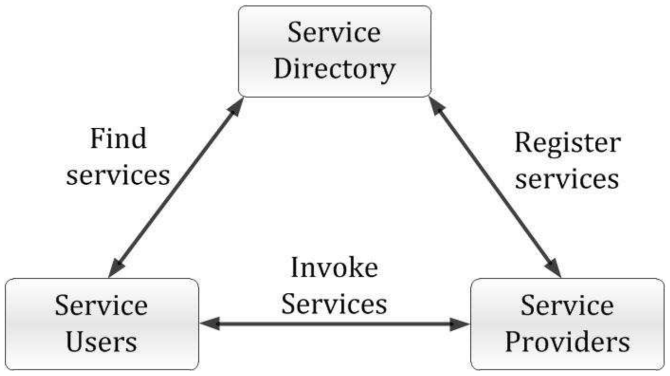

Figure 1.2: The pattern of discovering services through service directory

service users invoke the found services directly.

1.2 Challenges

In this section, we will discuss the major challenges in designing service discovery protocols with centralized service directories for SOA.

Untrusted Service Directory

Compared with broadcasting service discovery, service discovery with centralized service directories is more efficient but also needs to release a lot of information about services and service users to the service directories, which may reveal the privacy of service providers and users. Hence, when a service directory is not trusted by service providers or service users, they may refuse to register or lookup services through it. An untrusted service directory is harmful because of the following two problems:

1. The untrusted service directory may abuse the services or release the services to service users who have no permission to use these services.
2. Service registrations and requests may contain confidential information about service providers and requesters, which is required for service discovery to match services and requests but should not be revealed even to the service directory.

The first problem is to restrict access to services. Although a set of WS-Security standards [9] have been proposed to handle the authentication and authorization of service users, these standards require a trusted service authorizer to enforce these standards. Other secure service discovery protocols, such as SSDS [10] and UPnP security [11], also require a similar trusted authorizer as WS-Security standards. All of these do not work with the assumption that the service directory itself is untrusted. Furthermore, the first problem also requires that even the existence of services should be protected from unauthorized users. Otherwise, unauthorized users can launch denied-of-service attacks against these services.

The second problem is to protect the privacy of service providers and users. On the one hand, service providers rely on the service directory to store their services' information, and service users rely on the service directory to match registered services with their requests. On the other hand, service providers do not want the service directory to know which services they provided, and service users do not want the service directory to know which services they are looking for. These seeming incompatible requirements make the second problem challenging.

The Quality of Service

Besides the functionalities provided by services, the qualities of services (QoS), such as throughput, response time, reliability and security, are also critical for service users [12]. Service discovery should verify whether the registered services' QoS satisfy users' QoS requirements as well as functionality requirements, which requires the service directory to collect services' QoS information.

Several performance monitoring mechanisms [13, 14, 15] are proposed to monitor services' QoS either continuously or periodically. The monitored QoS information is reported to the service directory, and integrated with service registrations using the extension of WSDL [16, 17, 18], the extension of BPEL [19], or the extension of UDDI [20, 21, 22]. Furthermore, several QoS quality models and ontologies [23, 24, 25, 26, 27] are also proposed to organize services' QoS and incorporate service requesters' QoS requirements in the service requests. However, for some QoS aspects such as security, it is difficult to measure services' QoS on these aspects through monitoring. Although it is able to rate services' security as qualitative levels, such as very secure, secure, and not secure, a more precise quantitative metric is required to make sure that users' security requirements are satisfied by the services.

Second, even for the QoS aspects which QoS can be measured through monitoring, monitors can only passively collect services' QoS information, but cannot predict or adjust services' QoS, which restricts services' flexibilities in satisfying users' various QoS requirements. Developing quantitative metrics for those observable QoS aspects supporting QoS adjustments allow services to make tradeoff among various QoS aspects to better satisfy users' QoS requirements. For example, an adaptive encryption service should adjust its behaviors according to current situation to optimize its QoS on different aspects. In a friendly environment like business intranet where security is not critical, the service should use weak but cheap security mechanisms to improve its QoS on other aspects. But in an environment where a lot of attacking activities have been detected, the service should use strong security mechanisms even if QoS on other aspects is sacrificed.

QoS-based Service Ranking

The third challenge is also related to services' QoS. When there are more than one service that satisfy users' requirements on both functionalities and QoS, service users only need one service, and hence have to make a selection among all avaialble services. Service users alway prefer services that best satisfy their requirements. However, it is difficult to find the best service among services with equivalent functionalities, when service users have requirements on multiple QoS aspects.

First, when the selection among services needs to consider multiple QoS aspects simultaneously, services may perform good on one aspect but bad on another aspect. For example, there may be two services. One service has high throughput with high price, and another service has low throughput but with low price. In this case, the rank between these two services is related to users' specific QoS requirements and users' preferences on throughput and price. To compare services' QoS on multiple aspects, all of these QoS should be aggregated together with users' preferences on these aspects.

Second, because services' QoS on different aspects are measured with different units and value ranges. Services' QoS on different aspects is neither comparable nor be simply aggregated. Furthermore, while some QoS aspects such as throughput and delay which are easy to measure quantitatively as numbers, other QoS aspects such as security and usability are difficult to be measured quantitatively. These QoS aspects are usually measured qualitatively as coarse quality levels. It is not clear how to aggregate qualitative and quantitative QoS measurements.

1.3 Contributions

The overall goal of this dissertation is to help service providers and users to protect themselves from untrusted service directories, and still use service directories to find services that best satisfy their functionality and QoS requirements. The main contributions of this dissertation includes:

1. A privacy-preserving service discovery and matching protocol with untrusted service directory. This protocol encourages service providers and users to register and lookup services in public service directories without worrying about their privacy. Because public service directories are more open than private service directories, and can accept service registrations and service lookup requests from more organizations, they are helpful in increasing the number and diversity of available services, and hence promote the development of SBS systems.

2. A quantitative security metric, which measure services' security with security configuration vectors. This is the first quantitative security metric considering the vulnerabilities of the design of security algorithms and attackers' attacking power.

3. An approach to developing quantitative metrics for observable QoS aspects. The developed metrics allow services to measure their QoS on these aspects without monitors, but directly through their parameter values. All parameters are controllable by services. By adjusting parameters, services can provide customized services to users with tradeoff among QoS aspects.

4. A QoS-based service ranking algorithm. It helps users to select the best services when there are more than one service satisfying their functionality requirements. This algorithm ranks all available services based on the satisfaction of services' QoS on users' QoS requirements. The higher a service is ranked, the better the service satisfies users' QoS requirements.

1.4 Organization

In Chapter 2, the background and related work are discussed. In Chapter 3, the overall approach of discovering and selecting services with untrusted service directory is presented. In Chapter 4, a privacy-preserving service discovery and matching protocol is presented to protect both service providers' and users' privacy from untrusted service directory. The privacy-preserving service discovery and matching protocol only return-s services that both satisfy users' functionality requirements and are accessible for the users who send requests. For services that the user has no permission to access, even the existence of the service is protected from the user. In Chapter 5, a set of QoS aspects and their metrics are discussed. This chapter also discusses how to specify users' QoS requirements in their service requests. In Chapter 6, quantitative metrics are developed for security and observable QoS aspects. All of these metrics measure services' QoS with a set of parameters, which allow services to provide differential service for their users by making tradeoff among various QoS aspects. In Chapter 7, the QoS-based service ranking algorithm is presented to rank the services returned by the service directories according to their QoS and users' QoS requirements, which help users to find the services that best satisfy their both functionality and QoS requirements. Chapter 8 concludes this dissertation and proposes some future research problems in this research path.

Chapter 2

BACKGROUND AND RELATED WORK

In this chapter, the background information about service and SOA is first introduced to answer some basic questions, such as ”What are service and SOA?”, ”Why do we need service and SOA and what are the advantages of service and SOA?”, and ”How to use service and SOA?”. Then, we briefly surveyed the literatures on service discovery and matching and service QoS.

2.1 Services and SOA

With the growing size and complexity of software systems, the complexity of interoperability among different system components may increase quadratically with the number of system components because every component needs to communicate with all other components. For large software systems, it quickly becomes uncontrollable and a disaster of system development and management. To improve interoperability, Enterprise Integrate Bus (EIB) and Enterprise Application Integration Bus (EAI bus) introduce a global bus which serves as a middleware among all system components. As shown in Figure 2.1, because all system components only communicate directly with the bus and the communications between any two system components are through the bus, the complexity of interoperability becomes proportional to the number of system components. When a new component is added to the system, only the communication between the new component and the bus needs to be handled. EIB and EAI bus simplifies the communication among system components, but also hides the interactions among them. In EIB and EAI bus, individual system components are independent, all business logics heavily rely on the global bus to transfer, exchange and integrate messages among individual components. While EIB and EAI bus aims at providing a communication middleware for system components, SOA also improves interoperability of components in large scale systems but also covers other aspects, such as flexible

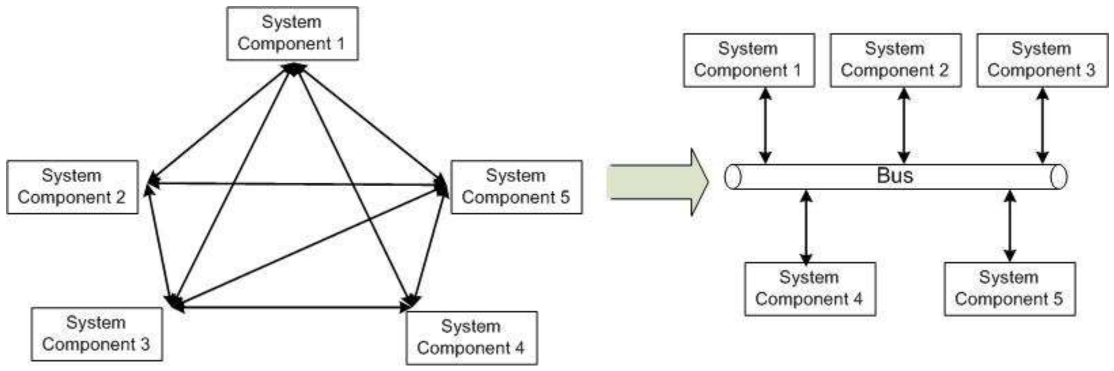

Figure 2.1: The interoperability of systems with and without bus

business process, reliability, and security.

Advantages of SOA

SOA has the following advantages

- Better interoperability. Large scale systems usually integrate some heterogenous system components from existing legacy systems or systems developed by other organizations. To handle the interoperability of such systems, data transformation is essential for transforming the data and messages among heterogenous components. EIB and EAI transform all components' data and messages to the format used by the global bus which is usually privately defined within the organization. From the perspective of SOA, each system component is a service or a set of services. SOA defines all services' interfaces with WSDL [4] and the messages among services with SOAP [5], both of which are XML-based specifications and open standardized. Hence, SOA enables interoperability not only within components from the same system but also among any services as long as they follow WSDL and SOAP, even though these services are distributed across networks, provided by different organizations, and/or implemented with different techniques. The better interoperability makes SOA a better choice in rapid development of large scale systems from existing services [28, 29].

- Loose coupling. SOA allows users to invoke services only based on services' interfaces without their actual access addresses. A mediator that is responsible to manage services will lookup appropriate services for users according to interfaces, and then forward users' requests to the access addresses of the found services. Hence, the service users and services are loosely coupled, and the services' interfaces and their underlying implementations are also loosely coupled. The benefits of loose coupling include but not limit to

  – Flexible business logics. Business logics can be developed based on services' interfaces rather than actual services. When the business logics change, only the compositions of services' interfaces are needed to be modified accordingly. The mediator will find services to satisfy the new business logics automatically. There is no change to services.

  – Fault tolerance. When a service fails, SOA can dynamically redirect users' requests to another service which supports the same interface [30]. The dynamical redirection means that 1) users' requests will be redirected to another service automatically without interruption, and 2) the destination of the redirection is dynamically determined during the run time.

  – Load balance. When there are more than one service providing the desired interfaces, SOA can distribute users' requests among all of these services to achieve the best quality and utilization of services. And, according to the system overload, system's capacity can be adjusted by adding more services or removing some existing services to save resources.

  – Flexible binding. Services' bindings are communication protocols between services and their users, such as HTTP and HTTPS. A service can support more than one bindings. With the help of interface-based invocation, users and business logics can decide which kinds of services will be invoked during the design time, but postpone the decision on communication protocols to the run time. For example, the business logic can switch between the HTTP and HTTPS protocols according to current transportation security requirements without changing the business logic.

- Security. Besides the security mechanisms provided by the network infrastructures such as SSL and Kerbores, SOA provides additional security mechanisms for the message layer, such as WS-Security [9], WS-Policy [31], XML-Encryption and XML-Signature [32]. Furthermore, SOA can implement security mechanisms as services, and reuse the security mechanisms whenever and wherever the security functionalities provided by the security mechanisms are required. For example, SOA can implement an AES encryption service to encrypt incoming traffics, which can be used anywhere when AES encryption is required.

Development Process of Service-based Systems

Services and SOAs enable rapid development of software systems from existing services, which generally includes the following three phases:

1. SBS Specification. The first phase of the development of SBS is to specify the SBS as a workflow composed with a set of abstract activities (i.e., services' interfaces) with specification languages such as BPEL4WS [33] and OWLS [24]. A good survey of SBS specification can be found in [34]. This process can be done manually by the developers [33, 35, 36, 37, 38] or automatically with AI planning techniques [24, 39, 40]. Each abstract activity in the workflow represents one operation, which takes the outputs of previous activities as inputs, processes the inputs, and sends the processed results to subsequent activities as outputs.

2. Service Discovery and Matching. In this phase, the SBS developers need to implement each abstract activity in the workflow with concrete services through centralized service directories [6, 8] or service discovery broadcasting in peer-to-peer systems [7].

3. Service Composition and Workflow Execution. When all abstract activities in the workflow have been implemented by concrete services, the workflow is ready for execution. After taking necessary initial inputs from users, the services are invoked in the order specified by the workflow. The data is shared and processed among all services to conduct the final results of the workflow. During the execution process, if one service fails, SBS can try to match another service for the corresponding activity, or to develop a new workflow to avoid that activity if no other suitable services for that activity are available.

2.2 Service Discovery and Matching

Service discovery and matching protocols are required to find services and verify if the founded services satisfy service users' requirements. Depending on where the protocols look for services, service discovery and matching protocols can be distributed [7] or centralized [6]. Distributed protocols have to broadcast the service requests to all service providers and match each service providers' services with the service requests, and centralized protocols match services in the centralized service directory, where all service providers register their services. Currently, most service discovery protocols are centralized, such as Service Location Protocol (SLP), Jini, UDDI, and Bluetooth [3]. Compared with distributed protocols, centralized protocols are more efficient, but also suffer from some new security challenges. First, because all service discovery and matching operations are through the centralized service directory, the service directory knows a lot of information about both service registrations and requests, which may be sensitive. Second, after service providers register their services in the service directory, the service directory has full control on the service registrations, and can expose these service registrations to any users even when the service providers do not want to reveal their service to these users.

Privacy Preserving Service Discovery and Matching

To protect services from malicious service users and vice verse, several secure service discovery protocols, such as SSDS [41] and UPnP security [11], have been developed to address the mutual authorization and authentication between services and service users. All of these protocols need a trusted service directory to honestly enforce the authentication and authorization, and have no protection for service and service users if the service directory itself is untrusted. The trustworthiness of service directories was first discussed in [42], where service registrations are protected from untrusted service directories with symmetric encryption. But this approach requires a trusted third party to maintain all symmetric encryption keys. In the same paper, they also proposed to store the hash values of the service registrations' keywords in service directories. Because hash functions are one way functions, the untrusted service directory cannot learn the service registrations' keywords from their hash values. However, hash functions are deterministic, and therefore the untrusted service can determine whether a service registration contains a specific keyword by comparing the hash values of service registrations' keywords with the hash value of the specific keyword. The PrudentExposure protocol [43, 44] protects service users' queries from service directories by wildcard search. To hide the service users' requests from the service directory, this protocol downloads all available service registrations in the service directory, which leads to huge communication overhead. So far, no service discovery protocol can search service directories without exposing the privacy information of both service providers and users to the untrusted service directory. This problem is challenging because it requires the service directory to match a service registration against a service request without knowing both of them.

Private Information Retrieval

Private Information Retrieval (PIR) schemes allow users to query databases without revealing their queries [45, 45, 46, 47, 48, 49, 50, 51, 52]. It is similar to the privacy p-reserving service discovery and matching protocols, which allow users to query service directories without revealing their requests. However, PIR schemes only guarantee that the untrusted database does not know which data the user is looking for, but have no protection on the privacy of data. That is, PIR schemes assume that the data is public but centrally stored in the database which maybe untrusted; the user, without a local copy, wishes to retrieve some data items from the database. A naive solution of PIR is to download the entire database and let users to retrieve the information from the local copy of data, which obviously preserves users privacy. However, this solution's communication complexity is equal to the size of the database, which is unpracticable for large databases. Thus, the main focus of the design of PIR schemes is to reduce the communication complexity.

The first non-trivial PIR scheme was first introduced by Chor *et. al.* [45] in 1995, in which they successfully reduced the communication complexity to $O(k \cdot n^{1/logk})$ with $k > 2$ copies of the database. Following his work, much effort has been devoted to further reduce the communication complexity of PIR schemes with multiple copies of the database. Currently, the best result is $n^{O(loglogk/klogk)}$ presented in [47].

Another research direction of PIR schemes is to achieve low communication complexity without copying the database. The first result in this direction is also presented by Chor *et. al.* [45]. They claimed that single database PIR does not exist, if the database has unlimited computing power. Thus, an interesting question is whether single database PIR exists if the database's computing power is restricted to probabilistic polynomial turning machine. In [46], Kushilevitz and Ostrovsky gave a positive answer to this question by constructing a single database PIR with communication complexity $O(2^{\sqrt{lognloglogN}})$. Kushilevitz and Ostrovsky showed that their scheme does not reveal the query to the database if the underlying Goldwasser-Micali public-key encryption scheme [53] is secure, whose security in turn relies on the discrete logarithmic assumption. Like the research about multiple database PIR, much works have been done to further reduce the communication complexity with various algebraic properties. The current best result $O(log^2n)$ is achieved by several groups [48, 49, 50].

Besides the effort of improving efficiency, the functionality of PIR is also extended to support more applications. The first extension is to support retrieving a block of data within one query [45, 52]. The second extension is to support keyword search in PIR [54], which allows the user to retrieve data from the database with keyword search. The third extension is to support private database [51], which only reveals the data what the user asks without any others. All of these three extensions are essential for the practical usage of PIR, but still do not provide sufficient support for privacy preserving service discovery and matching, which requires that the privacy of both the user and the data is protected.

Privacy Preserving Search

The first privacy preserving search schemes (PPS) was introduced by Dawn Xiaodong Song, David Wagner and Adrian Perrig [55]. Similar to PIR schemes, PPS schemes also protect data queries from untrusted servers, but they can support general data queries besides database queries. The fundamental problem of PPS schemes is how to store encrypted data in untrusted servers and enable the servers to search the encrypted data based on users' requests. According to the approaches of solving this fundamental problem, existing PPS schemes can be classified as following three categories.

- ***Searching based on special data structures.*** In this setting, the data stored in the public servers are organized with a special data structure. Peekaboo [56], describes the keywords as *(key, value)* pairs, and then stores the keys and the values in different servers separately. The confidentiality of the data is ensured when the servers storing keys and the servers storing values do not cooperate together to attack the system. SSE [57] organizes all documents in some linked lists. All documents containing the same keywords are linked together and encrypted by an encryption chain. To search the encrypted data by keywords, a look-up table is constructed to link the keyword and entrance of the according linked list together. Hence, whenever an entrance of the look-up table is identified, the database can locate the linked list and then successfully decrypts all related documents. Because the constructions of both the linked lists and the look-up table require the knowledge of the content of the data, all of them are constructed and submitted to the public servers by the data owner. This restriction implies that the user cannot search other users' data. In the setting of service discovery and matching, it means that service discovery and matching protocols developed with this kind of techniques can only allow service users to search services that provided by themselves, which significantly limit the reusability and availability of services.

- ***Searching based on trusted third parties.*** In this setting, the protection and the searching ability of the data is relied on the trusted third parties. An encrypted and searchable audit log is builded in [58] with the help of the trust third parties. The log is encrypted by its keywords as public keys, whose secret keys are preserved by the trusted third parties. The users search the encrypted data by first submitting their queries to the trusted third parties who will return a set of secret keys to the users. The security of this approach is heavily based on the trustiness of these third parties. Furthermore, the trusted third parties are involved in the process of each search request.

- ***Searching based on special encryption schemes.*** In this setting, the searching ability is implemented by encrypting the keywords with special encryption algorithms. While the normal encryption schemes only require that the users except secret key owners cannot learn any information from the ciphertexts, encryption schemes designed for keyword searching should allow users to check whether a ciphertext satisfies some conditionals, such as whether the ciphertext is the encryption of a given plaintext. The symmetric encryption scheme presented in [55] allows the secret key owners to search the encrypted data stored in the public servers without decryption. The asymmetric encryption schemes presented in [59, 60, 61] allows the public servers to search the data without the knowledge of the data content, when a special trapdoor is provided by the data owner. Except the exact keyword matching, the range search is allowed with the attributed-based encryption [62, 63] or hidden vector encryption [64]. Besides these encryption algorithms specially designed for keyword searching, homomorphic encryptions [65] are also wildly used to construct privacy preserving searching systems [66, 67, 68, 69].

UDDI Server

Universal Description Discovery and Integration (UDDI) is an XML-based specification language for describing service registrations and looking up services in service directories. UDDI is first proposed by OASIS as an open standard for service discovery and supported by many companies, including Microsoft, IBM, and SAP.

The core data structure of UDDI is shown in Figure 2.2 from `http://uddi.org/pubs/uddi-v3.0.2-20041019.htm`. Services (i.e., businessService) in UDDI server are organized with their providers (i.e., businessEntry). Each service has one or more access points (i.e., bindingTemplate), which may support different communication protocols. The tModels are used to describe services, such as their interfaces and categories. For example, a tModel can be used to specify a WSDL document [4]. If

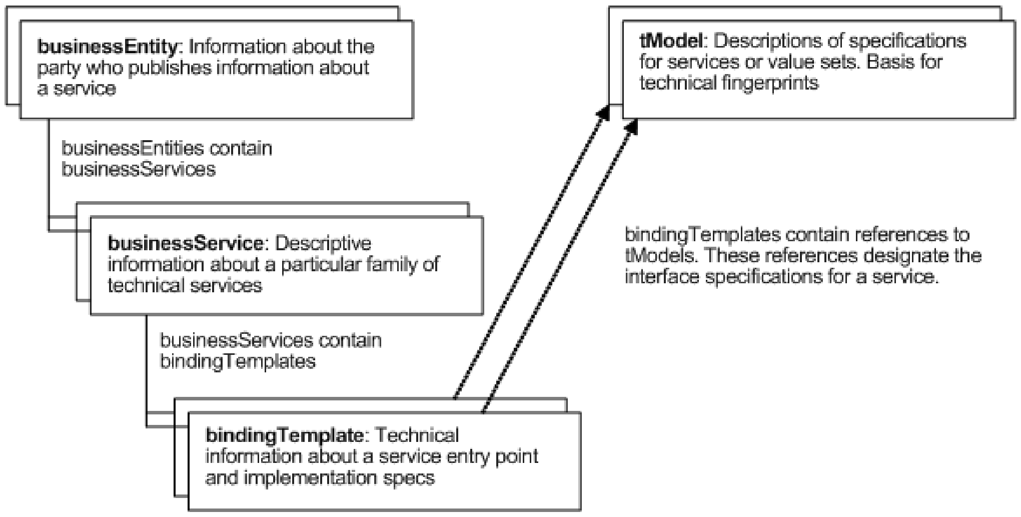
businessEntity: Information about the party who publishes information about a service
businessEntities contain businessServices
businessService: Descriptive information about a particular family of technical services
businessServices contain bindingTemplates
bindingTemplate: Technical information about a service entry point and implementation specs
tModel: Descriptions of specifications for services or value sets. Basis for technical fingerprints
bindingTemplates contain references to tModels. These references designate the interface specifications for a service.

Figure 2.2: The core data structure of UDDI

two services share the same tModel, it means that these two services provide the same functionalities defined in the WSDL document. Hence, a major advantage of using UDDI in service discovery is that we can search services with tModels (i.e., services' functionalities). The UDDI server will return a managed URL. Service users invoke services through this managed URL, but do not need to know which service is invoked. The UDDI server dynamically binds this URL with a service, and will automatically redirect the request sent to this URL to another service if the current service fails.

Besides searching services through tModel, the UDDI server also supports to search services with businessService, businessEntry, bindingTemplate, or their combinations.

2.3 Service QoS

Besides the functionalities, the quality of services (QoS) is another essential factor that should be considered in selecting services for the development of SBS.

Service QoS Specification

While services' functionalities can be modeled as operations with inputs and outputs [4], there is no universal pattern to model and specify services' QoS due to the variety of QoS aspects. Moreover, some QoS aspects, like security, are difficult to be quantitatively and accurately measured. Several QoS quality models and ontologies are proposed to formally organize and specify various QoS aspects with their metrics and interrelations, such as WSQM [23], OWL-S [24], WSMO [25, 26], and WS-Agreement [27]. These QoS specifications are then integrated with extended web services standards, such as the extension of WSDL [16, 17, 18], the extension of UDDI [20, 21, 22], and the extension of BPEL [19].

Because security is an important QoS aspect, a lot of researches have studied how to specify services' security requirements. BPEL4WS [33] integrates WS-Security [9] to protect the confidentiality and integrity of messages transmitted among services. The security requirements on how services deliver or process messages are supported by [42, 70]. [42] incorporates services' security capabilities and security constraints in the description of service contract, which enables the centralized service directory to match services based on both activity signatures (i.e., the inputs and outputs) and security requirements. This approach validates security requirements before the execution of workflow during the design time. To validate security requirement during the runtime, another extension based on λ-calculus is presented in [70], which validates security requirements by model-checking all services' behaviors. QoS requirements are addressed in [71] which first finds services matching the signature and then calculates the accumulated QoS of the workflow with found services. If the accumulated QoS already violates the QoS requirements, the service matching mechanism discards this service and continues to find other services. Otherwise, the service matching mechanism realizes the activity with the found service and continues to find services for proceeding activities.

When the requirements on some QoS aspects are difficult to be specified exactly. Moderated Fuzzy Discovery Method (MFDM) [72] and Intuitionist Fuzzy Sets (IFS) [73] are used to specify QoS requirements. For example, users can specify their requirement and preference as ”The service is very cheap” and ”The price of the service is very important” instead of ”The price of the service should be less than one dollar per hour” and ”The importance weight of the service’s price is 0.9”.

Service QoS Monitoring and Measurement

When services’ interfaces are designed during the design time and will not change during the running time, services’ QoS dynamically changes with services’ available resources, the number of users served, the running environments, and many other factors. In [15], a runtime QoS monitor was developed to continuously monitor services’ resources, users’ activities, and services’ QoS. While the monitoring on services’ resources requires access to the software and hardware platforms that host the services, there are other techniques developed to monitor and measure services’ QoS without the access to services’ platforms.

The concepts of user feedbacks and reputations are borrowed from the filed of trust management and social network to measure services’ QoS in [74, 75, 76]. In these works, services’ QoS is measured based on their users’ feedbacks. However, users’ feedbacks may be biased by their expectations or preferences, and be different even for the same service providing the same QoS. Furthermore, malicious users may provide false negative or positive feedbacks to affect the measurement of services’ QoS. Hence, an important problem in measuring services’ QoS with feedbacks is how to detect and remove inaccurate feedbacks. In [74], inaccurate feedbacks are detected based on deviations from the majority opinion. For example, if most users give good feedbacks to a service, a too bad feedback for this service is considered as fake, and will damage the feedback's provider's reputation. In [75], users' feedbacks are evaluated based on the usefulness of users' feedbacks, which is evaluated with Bayesian networks. Feedbacks from users with low reputation have small weights in measuring services' QoS. In [76], more factors are considered in the evaluation of users' reputation, including the social relations between the services' providers and users. Also, services provided by the same service providers are assumed to share similar feedbacks.

Data ming techniques are used in [77, 78] to measure services' QoS by analyzing their historic data, which assume that services' QoS satisfies some statistic properties and then predict services' future QoS from their historic QoS.

QoS Tradeoff among Various QoS Aspects

An adaptive service should be able to change its configurations or behaviors to provide customized QoS for different service users. Application level differentiated services [79, 80]control performance for different classes of service users. When the system resources are limited, fewer resources are allocated for normal users, and more resources are reserved to guarantee good performance for premium users. Furthermore, when resources are extremely limited and even the performance required by premium users cannot be fully satisfied, feedback controlled web services [81, 82] are proposed to adjust resource allocation to meet the most critical performance requirements, such as the delay in real-time systems.

These approaches require the control on the system resource allocation strategies, which are usually controlled by operating systems. For example, in [79], services provide two kinds of priorities by controlling process pool size and process priorities. In [82], services guarantee their delay by dynamically adjusting connection scheduling and process reallocation.

Security is another very important QoS aspect of services. There are a lot of researches on SOA security, including the protection of services from malicious consumers through authentication and authorization [83, 84, 85, 86], and the protection of messages through XML encryption and signature [32]. However, the impact of these security mechanisms on the systems' performance has not been well addressed.

In [87], the services made tradeoff between the usefulness of its data and the privacy protection of the data by selectively deleting some information from the data. When more information is deleted, the data contains less privacy information but also becomes more useless. This approach works only for applications that do not require complete data. The key length was used in [88] to control the tradeoff between security and delay. Once a service's delay exceeds a pre-defined threshold, the adaptive security policies defined in [88] would automatically reduce the key length to the next smaller value until the service's delay is reduced below the threshold again. While this approach only considers key lengths in the tradeoff between security and delay, a more precise security model QoSS was presented in [89] to control security strength. The QoSS model defines a set of security levels from high to low with parameters including the strength of cryptographic algorithm, key length, percentage of packets authenticated, security functions, confidence of policy-enforcement in remote login, and the robustness of authentication mechanism. The idea of classifying security to a set of discrete levels was also used in [90]. All of these approaches measure security qualitatively but not quantitatively, and hence cannot make precise tradeoff among security and other QoS aspects.

2.4 QoS-based Service Selection

QoS-based service selection is to select the service with the best QoS. If only one QoS aspect is considered, the service selection is easy by ordering services' QoS on this aspect. However, the service selection is challenging when more than one QoS aspect are needed to be considered.

First, different QoS aspects have different unites and value ranges. Services'

QoS on different QoS aspects have to be normalized before aggregation. Suppose there are m services $S_1, S_2, \cdots, S_m$ and their qualities on a specific QoS aspect are $q_1, q_2, \cdots, q_m$. In [91], each q_i $(1 \leq i \leq m)$ is normalized to $q_i/avg(q_1, q_2, \cdots, q_m)$, where $avg(q_1, q_2, \cdots, q_m)$ is the average quality of all services. In [92], q_i is normalized to $(q_{max} - q_i)/(q_{max} - q_{min})$ or $(q_i - q_{min})/(q_{max} - q_{min})$, where q_{max} and q_{min} are the best and worst qualities among all services. These normalization approaches only consider the QoS of all available services without users' QoS requirements and preferences on various QoS aspects. For example, if all services' qualities are too worse to satisfy users' requirements or all services' qualities perfectly satisfy users' requirements, the differences among their qualities on this QoS aspect cannot generate additional benefit or penalty to users, and hence should not affect the service ranking.

Second, the selection among services is not only based on services' QoS but also based on users' QoS requirements. To integrate users' QoS requirements in the normalization, Comuzzi and Pernici divided the range of services' QoS on each QoS aspect to several discrete levels [93]. Different levels represent users' different degree of satisfaction. All services' QoS within the same level are normalized to a same number. Hence, if two services' QoS have no difference from the perspective of users, the actual difference between these two services' QoS has no effect in the service ranking. This discrete normalization approach requires users to decide how to divide the range of services' QoS to discrete levels. Although mid-level splitting techniques can be used to help users [94], it still requires a lot of interactions with users. Another disadvantage of this approach is that services' QoS are normalized to a set of discrete levels coarsely, and hence cannot distinguish two services with similar qualities on all QoS aspects. Finally, it normalizes different QoS aspects independently, and does not consider users' preferences on different QoS aspects.

Third, services' QoS on different QoS aspects have to be aggregated together. Let service S_i's qualities on n QoS aspects be $(q_{i1}, q_{i2}, \cdots, q_{in})$, which are normalized to $(norm_{i1}, norm_{i2}, \cdots, norm_{in})$. Because each normalized QoS $norm_{ij}$ is within an unique range, services' qualities on different QoS aspects can be aggregated together for service selection. Typically, the aggregation is implemented by adding the normalized QoS together with weight factors [95]. That is, the service S_i will be ranked based on the value of $\sum w_j * norm_{ij}$, where the weight w_j represent users' preference on the j-th QoS aspect. Users assign larger weights for QoS aspects that are more important for them, and smaller weights for other QoS aspects. This aggregation approach ranks and selects services only based on services' QoS. To consider users' QoS requirements, the Euclidian distance between services' QoS and users' QoS requirements is used to rank services [92]. That is, the service S_i will be ranked based on the value of $(\sum(norm_j - norm_j^*)^2)^{1/2}$, where $(norm_1^*, norm_2^*, , norm_n^*)$ is users' normalized QoS requirements on each QoS aspect. In [93], a price model is used for aggregation of services' QoS on multiple QoS aspects. The price model converts a service's QoS qualities on each QoS aspect to a price, and adds all prices together as the service's total price, and ranks services based on their total prices.

Chapter 3

OVERALL APPROACH

This dissertation studies techniques about privacy preserving service discovery protocol with QoS-based service ranking. Compared with general service discovery protocols which look for services simply based on users' functionality requirements, the service discovery techniques developed in this dissertation have several desirable characteristics, which are discussed in Section 3.1. Then, the system infrastructure providing support for such service discovery and the overview of the service discovery procedure is presented in Section 3.2.

3.1 Characteristics of Privacy Preserving Service Discovery and Ranking for Multiple User QoS Requirements

A privacy preserving service discovery and ranking approach is a special kind of service discovery protocol. Besides helping service users to look for services in the service directory providing the required functionalities, it also has the following desirable characteristics:

- Service providers and users can use a public service directory to register services and find services without worrying about their privacy. All information about services stored in the service directory is encrypted, and all users' requests sent to the service directory are also encrypted. The service directory matches registered services with service requests without learning any information about both service registrations and service requests besides the matching results.

- The existence of services registered in the service directory is only exposed to users who are authorized to access these services, and is hidden from users who have no permission to access these services. If the service's provider does not authorize a service user to access the service, the service directory cannot reveal the service to the unauthorized user, and the service user cannot find the service in the service directory with any service request.

- Service users can specify their requirements on both functionalities and QoS in the service requests. Services that satisfy the functionality requirements but do not satisfy the QoS requirements will not be returned to the users.
- Services can provide differential QoS for their users with various QoS requirements. Services control their QoS through a set of parameters. By adjusting the values of these parameters, services make tradeoff among various QoS aspects with different tradeoff strategies.
- Services returned by the service directory are ranked based on the satisfaction of service users' QoS requirements. If the service directory returns multiple services, all of these services satisfy both functionalities and QoS requirements of the service user. And, the better the service satisfies the user's QoS requirements, the higher the service is ranked among all returned services. Hence, the top ranked service is the best service that the service user should use.

3.2 Overview of The Approach

The system infrastructure supporting privacy preserving service discovery and ranking is illustrated in Figure 3.1, which includes four entities, the service directory (**SD**), the QoS monitor (**QM**), the trusted third party (**TTP**), and the service ranker (**SR**). In this system infrastructure, service providers (**SP**) register services and service users (**SU**) discover services with the following four major steps.

Step 1) Service Matching on Functionality. In this step, services are matched only based on their functionalities.

Step 1.1) Key Initialization. **SP** and **SU** initialize encryption and decryption key pairs and access keys with **TTP**. The encryption and decryption key

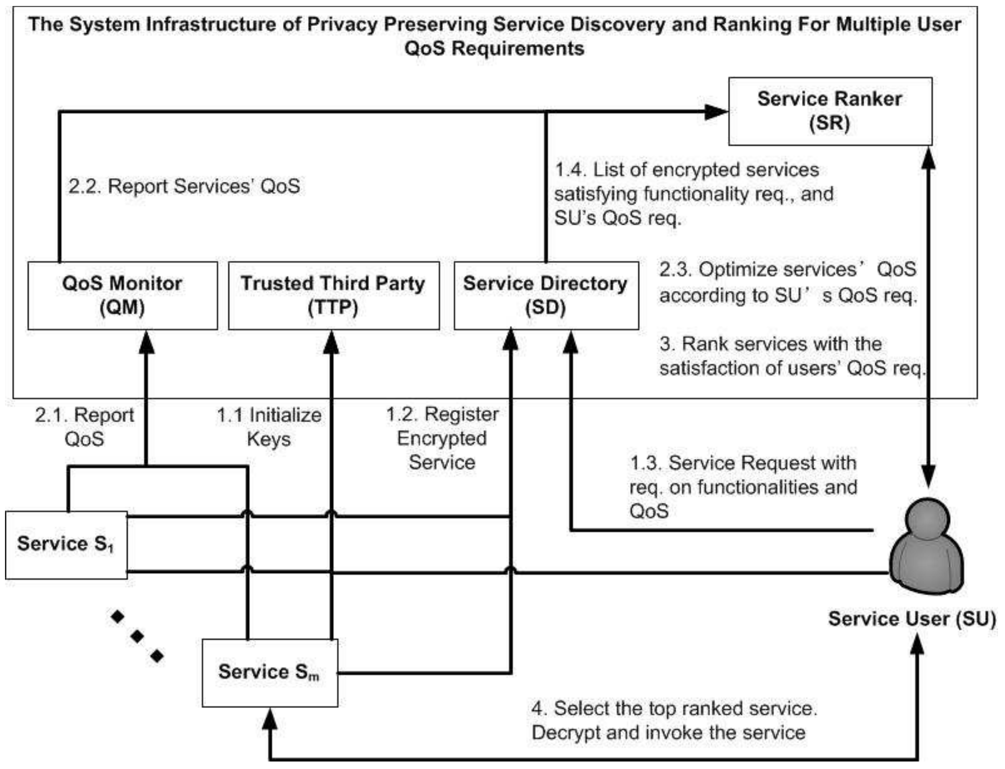
The System Infrastructure of Privacy Preserving Service Discovery and Ranking For Multiple User QoS Requirements
Service Ranker (SR)
2.2. Report Services' QoS
1.4. List of encrypted services satisfying functionality req., and SU's QoS req.
QoS Monitor (QM)
Trusted Third Party (TTP)
Service Directory (SD)
2.3. Optimize services' QoS according to SU's QoS req.
3. Rank services with the satisfaction of users' QoS req.
2.1. Report QoS
1.1 Initialize Keys
1.2. Register Encrypted Service
1.3. Service Request with req. on functionalities and QoS
Service S1
Service Sm
Service User (SU)
4. Select the top ranked service. Decrypt and invoke the service

Figure 3.1: The system architecture and procedure of privacy preserving service discovery and ranking for multiple user QoS requirements

pairs are used to encrypt and decrypt services' registration information stored in **SD**. The access keys are used to control the access to services. **SU** can search and access a service if and only if he/she possesses the correct access key to this service.

Step 1.2) Service Registration. **SP** encrypts services' information including the meta information describing services' functionalities, and technical details about service invocation. All of these information is encrypted with the encryption key from **TTP** before submitted to **SD**. Without the decryption key, **SD** cannot learn any information about the registered service.

Step 1.3) Service Request. Once **SU** wants to find services in **SD**, **SU** prepares a service request, which includes his/her requirements on both services' functionalities and QoS. The service request is encrypted with the encryption key from **TTP** in Step 1.1) before sent to **SD**. Without the decryption key, **SD** cannot learn any information about the submitted service request.

Step 1.4) Service Matching. When **SD** receives a service request from **SU**, it matches all registered services' functionalities with the functionality requirements of **SU**. All matched services are forwarded to **SR** as well as the QoS requirements of **SU**. During the matching process,**SD** never decrypts any encrypted services or encrypted service requests. Hence, all services forwarded to **SR** are kept encrypted. Furthermore, if **SU** cannot access a service, the service will not be forwarded even it satisfies **SU**'s functionality requirements.

Step 2) QoS Tradeoff. In this step, all services that satisfy **SU**'s functionality requirements are ranked based on their QoS.

Step 2.1) QoS Monitoring. Once **SP** registered services in **SD**, the registered services are monitored by **QM**. Services' QoS are periodically reported to **QM**.

Step 2.2) QoS Reporting. To help **SR** to enforce **SU**'s QoS requirements and rank all satisfied services, **QM** reports the QoS information of all services forwarded by **SD** in Step 1.4) to **SR**.

Step 2.3) QoS Tradeoff. With the list of encrypted service from **SD** and their QoS information from **QM**, **SR** checks whether these services satisfy **SU**'s QoS requirements. All services that do not satisfy the QoS requirements are removed from the list. For all services remained in the list, **SR** optimizes their QoS by making tradeoff among various QoS aspects according to **SU**'s QoS requirements.

Step 3) Service Ranking. **SR** ranks all optimized services based on how well the service satisfy **SU**'s QoS requirements. The better the service satisfies QoS requirements, the higher the service is ranked. At the end, the list of ranked and encrypted services is returned to **SU**.

Step 4) Service Invocation. From the list returned by **SR**, **SU** selects the top service or several services. **SU** decrypts the selected encrypted services and starts to use the service by invoking it.

Chapter 4

PRIVACY PRESERVING AND CONTROLLED ACCESS SERVICE DISCOVERY AND MATCHING

This chapter presents a privacy preserving and controlled access service discovery and matching protocol, which allows the untrusted service directory to 1) accept encrypted service registrations from service providers, 2) accept encrypted service requests from service users, 3) match encrypted service registrations with encrypted service requests through keywords, and 4) enforce the access control policies of services. Meanwhile, the untrusted service directories cannot learn any information about service registrations and requests besides the matching results.

4.1 Problem Statement

A privacy preserving and controlled access service discovery and matching protocol is an augmented service discovery and matching protocol, which also involves three kinds of participants: service providers **SP**, service users **SU** and service directories **SD**. There may be multiple distributed **SD**s where **SP** register services and **SU** search services. For the sake of simplicity, we consider only one **SD**, which is untrusted and tries to learn some information about services and requests. Furthermore, **SU**s' searching capability is controlled. That is, **SU** can only search services that he/she has the permission to access. The privacy preserving and controlled access service discovery and matching protocol is defined as follows.

Definition 4.1 (privacy preserving and controlled access Service Discovery and Matching)**.** Let **SP** be the provider of a service registration s consisting with meta information and technique details about service invocation. **SP** encrypts s with a set of keyword *Keyword* and a set of access keys *AccessKey*, and registers the encrypted s in the service directory **SD**. The service user **SU** searches services by submitting a service request to

SD. The service request is encrypted with an access key ak and a keyword k. The privacy preserving and controlled access service discovery and matching protocol matches the encrypted s with the encrypted service request, which satisfies the following three funtionality requirements:

1. **SU** can decrypt the encrypted s, only when s matches his/her service request.
2. If **SU** can access s, i.e, $ak \in AccessKey$, **SD** matches the encrypted s with **SU**'s service request successfully if and only if $k \in Keyword$.
3. If **SU** cannot access s, i.e, $ak \notin AccessKey$, **SD** cannot match the encrypted s with **SU**'s service request no matter whether $k \in Keyword$.

Furthermore, the privacy preserving and controlled access service discovery and matching protocol also satisfies the following three security requirements:

1. The encrypted s does not reveal any information about s and its *AccessKey* and *Keyword*.
2. The encrypted service requests do not reveal **SU**'s access keys and keywords to **SD** and other service users.
3. **SU** as well as **SD** cannot match the encrypted s if they do not have the correct access key.

$\square$

In the Definition 4.1, the set *Keyword* can be used to describe services' functionalities and the set *AccessKey* can be used to specify access policies to services. Hence, the service discovery and matching is based on services' functionalities.

The first two security requirements protect **SP**'s registrations and **SU**'s requests from the untrusted **SD**, and the third security requirement controls the access to services. The first two security requirements are satisfied if s and the service request are encrypted with semantic secure encryption algorithms. In this chapter, we first encrypt s and service request with a semantic secure symmetric encryption scheme and a random encryption key k, which ensures that **SD** cannot learn any information about the service and the service request from the encrypted s and the service request without the encryption key k. To enable **SU** to decrypt the encrypted s, we hide k with *AccessKey*. When **SU** has the correct access key, **SU** can recover k and then decrypt s.

4.2 Preliminaries

In this section, we present the basic techniques used in the construction of our privacy preserving and controlled access service discovery and matching protocol.

Semantic Secure Encryption Algorithms

An encryption algorithm generally consists with an encryption key ek, an decryption key dk, an encryption operation E and an decryption operation D. For symmetric encryption algorithms such as DES and AES, the encryption key and the decryption key are the same, *i.e.*, $ek = dk$. For asymmetric encryption algorithms such RSA and ElGamal, the encryption key and the decryption key are different, *i.e.*, $ek \neq dk$. The encryption operation E encrypts a message m to a ciphertext C with ek. That is, $E(m, ek) = C$. And, the decryption operation D decrypts the ciphertext C back to the message m with dk. That is, $D(C, dk) = m$.

An encryption algorithm is semantic secure if and only any adversary cannot learn any information about m from C without the knowledge about dk. If we give C to the adversary with another random message R with the same length of C, the adversary cannot distinguish C from R within polynomial time. Formally,

$$|\Pr[\mathscr{A}(C,I) = 1] - \Pr[\mathscr{A}(R,I) = 1]| < \varepsilon$$

where $\mathscr{A}$ is the adversary, ε is a negligible number, and I is used to model additional information available for $\mathscr{A}$. According to I, semantic security can be classified as chosen-plaintext semantic security, chosen-ciphertext semantic security, and others. More information about semantic secure encryption algorithms and their design techniques can be found in the Chapter 5 of [96].

Pairing

Pairing is a basic concept in ellipse curves. Since the construction of the first pairing-based key exchange protocol [97], pairing has been widely used to construct secure algorithms and protocols. A pairing is a mapping e from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two groups of the same prime order q. The bilinear mapping has three properties:

1. (**Bilinearity**) For any $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}$, we have $e(P^a, Q^b) = e(P, Q)^{ab}$
2. (**Non-Degeneracy**) There exists two elements $P, Q \in \mathbb{G}_1$ satisfying $e(P, Q) \neq 1$
3. (**Computability**) For any $P, Q \in \mathbb{G}_1$, there exists efficient algorithm to compute $e(P, Q)$

There are a lot of security assumptions about bilinear mapping. In this chapter, we will use the assumption of *Bilinear Diffie-Hellman Problem (BDH)*, which assumes that no probabilistic polynomial algorithm can compute $e(P, P)^{abc}$ from (P, P^a, P^b, P^c) with non-negligible probability.

Homomorphic Encryption

Homomorphic encryption schemes are special cryptosystems supporting operations on the ciphertexts directly without decryption. Let C_1 and C_2 be two ciphertexts of two messages m_1 and m_2 encrypted with a homomorphic encryption scheme correspondingly. There exists an efficient algorithm to compute the encryption of $m_1 \oplus m_2$ from

C_1 and C_2 without decryption. According to the operation $\oplus$, there are two types of homomorphic encryption schemes: additively homomorphic encryption schemes such as Pailler [65] and multiplicative homomorphic encryption schemes such as RSA and ElGamal. For example, let (e,n) be the encryption key of RSA, and m_1 and m_2 be two messages, RSA can compute the encryption of $m_1 m_2$ as

$$(m_1 m_2)^e = (m_1)^e (m_2)^e (\text{mod } n)$$

Most of existing homomorphic encryption schemes can only support either addition or multiplication but not both. Hence, an interesting problem is how to design a fully homomorphic encryption scheme to support both addition and multiplication. This problem is well recognized in the cryptography communities since the development of RSA. Until 2005, the first encryption scheme supporting both operations was designed by Boneh, Goh, and Nissim [98], named as BGN encryption scheme. BGN encryption is designed to compute the encryption of $m_{1,1}m_{1,2} + \cdots + m_{n,1}m_{n,2}$ from the encryptions of $m_{1,1}, m_{1,2}, \cdots, m_{n,1}, m_{n,2}$, and can support unlimited additive operations but only one time multiplicative operation on each encryption. In 2009, Craig Gentry in IBM presented the first fully lattice-based homomorphic encryption scheme which supports any numbers of additive and multiplicative operations in any orders [99]. However, the performance of this encryption scheme is unpractical.

The parameters of the BGN encryption scheme include two cyclic groups $\mathbb{G}, \mathbb{G}_1$ with the same order $n = q_1 q_2$, where q_1 and q_2 are two large primes. Let $g, h = u^{q_2}$ be two generators of the group $\mathbb{G}$. The BGN scheme encrypts the data m as the element $g^m h^r$ in the group $\mathbb{G}$, where r is a random number. Under the subgroup decision assumption [100], the BGN scheme provides semantical security, and has the following two properties:

1. (**Verifiable**) Given the encryption $g^m h^r$ and the data m', it is efficient to verify whether $m = m'$ by checking whether $(g^m h^r)^{q_1} = g^{q_1 m'}$.

2. (**Homomorphic**) Given two encryptions $g^{m_1}h^{r_1}$ and $g^{m_2}h^{r_2}$, it is efficient to compute the encryption of $m_1 + m_2$ as $g^{m_1}h^{r_1} \cdot g^{m_2}h^{r_2}$, and the encryption of $m_1 m_2$ as $e(g^{m_1}h^{r_1}, g^{m_2}h^{r_2})$, where e is a bilinear map from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_1$.

BGN encryption can only support small plaintexts if the decryptions on the encryption of these plaintexts are required, because BGN's decryption operations need to find m from g^m (i.e., solve the discrete logarithm problem) which is believed to be hard for large m and g in a high order cyclic groups $\mathbb{G}$. In this chapter, we will use the BGN encryption scheme to construct the privacy preserving and controlled access service discovery and matching protocol. Because we just need to verify whether a ciphertext is the encryption of one given data, and never decrypt any encryption, the high complex decryption operation of BGN is avoided.

Two-Party Private Intersection Predicate Evaluation

The two-party private intersection evaluation (PIPE) problem is introduced by Kiayias [101], where two parties (A, B) evaluate whether the intersection of their set S_A and S_B is empty. PIPE has the following properties:

1. (**Correctness**) For any two sets S_A and S_B, the output of $PIPE(S_A, S_B)$ belongs to $\{0, 1\}$, and $PIPE(S_A, S_B) = 1$ if and only if $S_A \cap S_B \neq \varnothing$.
2. (**Security**) Neither A nor B nor a third party can get more information about the sets S_A and S_B by executing the PIPE protocol than the output of $PIPE(S_A, S_B)$.

The first PIPE protocol is shown by Freedman *et. al.* [66] through representing set's elements as polynomials' roots. For example, the set S with n elements $\{s_1, s_2, \cdots, s_n\}$ can be represented as n roots of a n-degree polynomial as

$$f(x) = (x - s_1)(x - s_2) \cdots (x - s_n)$$

With this polynomial representation, we can verify whether the element a is in the set S through polynomial evaluation. If $f(a) = 0$, $a \in S$; otherwise, $a \notin S$. This idea is further extended by Kissner *et. al.* [68] to support more set operations including union and element reduction, and is used to design a privacy preserving search scheme in [67], which however is unpractical due to oblivious pseudorandom functions used in the scheme, whose construction requires $m\binom{2}{1}$ - OT protocols [102, 103] and one exponentiation.

4.3 Building Blocks

In this section, we present two useful building blocks in the construction of privacy preserving and controlled access service discovery and matching protocol.

Controlled Search

Controlled search is to control the search capability on one set by another set. We design this algorithm with the polynomial representation of sets as [66, 68]. The set S with s elements can be represented as s roots of a s-degree polynomial $f(x)$. With this polynomial representation, we can search the element a in the set S through polynomial evaluation. If $f(a) = 0$, $a \in S$; otherwise, $a \notin S$. In this subsection, we construct a polynomial $F_{S_1,S_2}(x)$ to search (a,b) in $S_1 \times S_2$, where S_1 and S_2 are two sets. For all $a \in S_1, b \in S_2$, $F_{S_1,S_2}(a) + F_{S_1,S_2}(b)$ returns the index of a in the set S_1, which should be in the range $\{1, 2, \cdots, |S_1|\}$; otherwise, the value of $F_{S_1,S_2}(a) + F_{S_1,S_2}(b)$ is random. We call such polynomial $F_{S_1,S_2}(x)$ for the set S_1 and S_2 as the *controlled search polynomial* for S_1 and S_2 because anyone who knows at least one element of the second set S_2 can use it to search the elements of the first set S_1.

For two sets $S_1 = \{a_1, a_2, \cdots, a_s\}$ and $S_2 = \{b_1, b_2, \cdots, b_t\}$, we design the controlled search polynomial $F_{S_1,S_2}(x)$ for S_1 and S_2 as a $(s+t)$ degree polynomial, which

```
Input: The controlled search polynomial $F_{S_1,S_2}(x)$ for sets $S_1, S_2$, and two elements
       $ak, k$
Output: ak's index in the $S_1$, if $ak \in S_1$ and $k \in S_2$; 0, otherwise
Compute $F_{S_1,S_2}(ak) + F_{S_1,S_2}(k)$ ;
if $1 \leq F_{S_1,S_2}(ak) + F_{S_1,S_2}(k) \leq |S_1|$
then
    return $F_{S_1,S_2}(ak) + F_{S_1,S_2}(k)$
else
    return 0
```

Figure 4.1: The algorithm $ControlledSearch(F_{S_1,S_2}(x), ak, k)$.

satisfies

$$F_{S_1,S_2}(x) = \begin{cases} r+i, & x = a_i \in S_1 \\ -r, & x \in S_2 \end{cases} \tag{4.1}$$

where r is a random number. The random number r is integrated in the representation of the polynomial and will never be expressed explicitly.

The algorithm *ControlledSearch*($F_{S_1,S_2}(x), ak, k$) shown in Figure 4.1 verifies whether $ak \in S_1$ and $k \in S_2$ by evaluating $F_{S_1,S_2}(x)(ak) + F_{S_1,S_2}(x)(k)$.

The correctness of the algorithm *ControlledSearch* is stated as the following theorem.

Theorem 4.2. *Let $S_1 = \{a_1, a_2, \cdots, a_s\}$ and $S_2 = \{b_1, b_2, \cdots, b_t\}$ be two sets. The output of the algorithm ControlledSearch($F_{S_1,S_2}(x), ak, k$) satisfies*

1. *If $ak \in S_1$ and $k \in S_2$, the algorithm ControlledSearch($F_{S_1,S_2}(x), ak, k$) always returns the index of ak in the set S_1.*

2. *Given two random elements a and b, the probability of $F_{S_1,S_2}(a) + F_{S_1,S_2}(b) \in [1, s]$ does not exceed $s(s+t)^2|R|/|D|^2$, where s is the size of S_1, t is the size of S_2, D is the domain of the polynomial $F_{S_1,S_2}(x)$, and R is the range of the polynomial*

$F_{S_1,S_2}(x)$.

Proof 1. According to the definition of controlled search polynomial given in (4.1), for any $ak \in S_1$ and $k \in S_2$, $F_{S_1,S_2}(ak) + F_{S_1,S_2}(k) = r + i - r = i$, where i is the index of ak in the set *AccessKey*. The algorithm will return i.

2. Suppose the polynomial $F_{S_1,S_2}(x)$ is a function mapping from the domain D to the range R. First, we prove that $|\{x|F_{S_1,S_2}(x) = a\}| \leq s+t$ for any random element $a \in R$. Define the polynomial $F(x)$ as $F_{S_1,S_2}(x) - a$, which is also a $(s+t)$-degree polynomial. For every element $x' \in \{x|F_{S_1,S_2}(x) = a\}$, we have $F(x') = F_{S_1,S_2}(x') - a = 0$. That is, all elements in the set $\{x|F_{S_1,S_2}(x) = a\}$ are the roots of the polynomial $F(x)$. Because a $(s+t)$-degree polynomial has at most $s+t$ roots, we have $|\{x|F_{S_1,S_2}(x) = a\}| \leq s+t$. That is, for any random $a \in R$,

$$\Pr[F_{S_1,S_2}(x) = a|x \in D] \leq \frac{s+t}{|D|}$$

For the random elements a and b, the probability of $1 \leq F_{S_1,S_2}(a) + F_{S_1,S_2}(b) \leq s$ is estimated as

$$\begin{aligned}
& \Pr[1 \leq F_{S_1,S_2}(a) + F_{S_1,S_2}(b) \leq s] \\
= & \sum_{r \in R} \Pr[F_{S_1,S_2}(a) = r] \cdot \Pr[-r+1 \leq F_{S_1,S_2}(b) \leq -r+s] \\
\leq & \sum_{r \in R} \sum_{i=1}^{s} \Pr[F_{S_1,S_2}(a) = r] \cdot \Pr[F_{S_1,S_2}(b) = -r+i] \\
\leq & \sum_{r \in R} \sum_{i=1}^{s} \frac{s+t}{|D|} \cdot \frac{s+t}{|D|} \\
\leq & \frac{s(s+t)^2|R|}{|D|^2}
\end{aligned}$$

□

```
Input:
1) The encrypted coefficients $EC = \{E_{pk}(a_n), E_{pk}(a_{n-1}), \cdots, E_{pk}(a_0)\}$
2) The encrypted point $EP = \{E_{pk}(b^n), E_{pk}(b^{n-1}), \cdots, E_{pk}(b^0)\}$
3) The target value $c$
4) The encryption key $pk$ and the verification key $vk$ of the BGN scheme
Output: 1, if $f(b) = c$; 0, otherwise
Compute the encryption of $f(b)$ as $C = \prod_{i=0}^{n} e(E_{pk}(a_i), E_{pk}(b^i))$ ;
if $C^{vk} = e(g,g)^{vk \cdot c}$ then
    |
else
    returns 1
return 0
```

Figure 4.2: The algorithm *Verify*(EC, EP, c, pk, vk).

Privacy Preserving Polynomial Evaluation

Privacy preserving polynomial evaluation is to evaluate $f(a)$ without the knowledge of the polynomial $f(x)$ and the element a. We design this algorithm with the BGN encryption scheme [98].

Let E_{pk} be the BGN encryption algorithm with the encryption key pk, and q_1 be the verification key vk. From the BGN scheme's properties, we construct the algorithm *Verify* to securely verify whether $f(b) = c$ in Figure 4.2. The algorithm takes five inputs EC, EP, c, pk, vk, where EC is the encrypted coefficients of the polynomial $f(x)$, EP is the encrypted point b, c is the target value, pk, vk are the encryption key and verification key of the BGN scheme.

The correctness and privacy of the algorithm *Verify* are stated as the following theorem.

Theorem 4.3. *Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ be an n-degree polynomial. Let EC, EP, c, pk, vk be the five parameters defined in the Verify algorithm. We have*

1. *If $f(b) = c$, the algorithm Verify(EC, EP, c, pk, vk) returns* 1.

2. *Given EC, EP, pk, no adversary can distinguish the polynomial $f(x)$ from a random n-degree polynomial $R(x)$, or distinguish the element b from a random element r in polynomial time with non-negligible probability, if the BGN encryption scheme is semantically secure.*

3. *Given EC, EP, pk, vk and a random element r, the adversary cannot compute the polynomial $f(x)$ and the element b in polynomial time if the discrete logarithm problem is hard.*

Proof 1. Let $\mathbb{G}, \mathbb{G}_1, g$ and h be parameters of BGN encryption scheme introduced in Section 4.2, where $\mathbb{G}, \mathbb{G}_1$ are two cyclic groups with the same order $n = q_1 q_2$, and $g, h = u^{q_2}$ are two generators of the group $\mathbb{G}$. $vk = q_1$ is the verification key. For a data m, the BGN scheme encrypts m as $E_{pk}(m) = g^m h^r$, where r is a random number. Furthermore, because both g and h are generators of the same group, there exist $r_0 \in \mathbb{Z}$ that $h = g^{r_0}$.

According to the bilinearity of the mapping e from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_1$, we have

$$\begin{aligned}
C &= \prod_{i=0}^{n} e(E_{pk}(a_i), E_{pk}(b^i)) \\
&= \prod_{i=0}^{n} e(g^{a_i} h^{r_{i,1}}, g^{b^i} h^{r_{i,2}}) \\
&= \prod_{i=0}^{n} e(g^{a_i}, g^{b^i}) \cdot e(g^{a_i}, h^{r_{i,2}}) \cdot e(h^{r_{i,1}}, g^{b^i}) \cdot e(h^{r_{i,1}}, h^{r_{i,2}}) \\
&= \prod_{i=0}^{n} e(g,g)^{a_i b^i} \cdot e(g,h)^{a_i r_{i,2}} \cdot e(g,h)^{b^i r_{i,1}} \cdot e(h,h)^{r_{i,1} r_{i,2}} \\
&= \prod_{i=0}^{n} e(g,g)^{a_i b^i} \cdot e(g,h)^{a_i r_{i,2} + b^i r_{i,1} + r_{i,1} r_{i,2} r_0} \\
&= e(g,g)^{\sum_{i=0}^{n} a_i b^i} \cdot e(g,h)^{\sum_{i=0}^{n} a_i r_{i,2} + b^i r_{i,1} + r_{i,1} r_{i,2} r_0} \\
&= e(g,g)^{f(b)} \cdot e(g,h)^{r}
\end{aligned}$$

, where $r = \sum_{i=0}^{n} a_i r_{i,2} + b^i r_{i,1} + r_{i,1} r_{i,2} r_0$. If $f(b) = c$, we have

$$\begin{aligned} C^{vk} &= (e(g,g)^{f(b)} \cdot e(g,h)^r)^{q_1} \\ &= e(g,g)^{q_1 \cdot f(b)} \cdot e(g,u^{q_1 q_2})^r \\ &= e(g,g)^{vk \cdot f(b)} \cdot e(g,1_{\mathbb{G}})^r \\ &= e(g,g)^{vk \cdot c} \end{aligned}$$

Hence, the *Verify* algorithm returns 1.

2. Because the encryption of the BGN scheme is semantically secure which means that the adversary cannot distinguish the encryption from another random string with the same length, no information about $f(x)$ and b is revealed from CE, EP, pk.

3. When the adversary has the verification key vk, the adversary is able to determine whether a ciphertext is the encryption of the given data. It is because the verification vk enables the adversary to compute g^x from $E_{pk}(x)$. However, the adversary cannot compute the x from $E_{pk}(x)$ without solving the discrete logarithm problem. That is also the reason why the original BGN encryption scheme requires that the plaintext must be small. Hence, from EC, EP, pk, vk and r, the adversary cannot compute the $f(x)$ and b in polynomial time if the discrete logarithm problem is hard.

$\square$

4.4 Main Construction

In this section, the main construction of our privacy preserving and controlled access service discovery and matching protocol is presented.

Overview

Our privacy preserving and controlled access service discovery and matching protocol consists of five algorithms: *Initialization*, *ServiceRegistration*, *ServiceRequest*, *Ser-*

viceMatch, and *ServiceRetriever*. It also involves four entities: the trusted third party **TTP**, the service provider **SP**, the service user **SU**, and the service directory **SD**. These five algorithms match the Step 1.1), Step 1.2), Step 1.3), Step 1.4), and partial Step 4) in Figure 3.1. The details of the constructions of these five algorithms are described in the following:

- ***Initialization***: **TTP** initializes all parameters of the BGN encryption schemes, including the encryption key $pk = \{n, g, h, e, \mathbb{G}, \mathbb{G}_1\}$ and the verification key $vk = q_1$. **TTP** also initializes a secure symmetric encryption scheme *SE* and a hash function H which maps any string to the space of the encryption keys of E.

- ***ServiceRegistration(**m**,**$AccessKey$**,**$Keyword$**)***: When **SP** wants to share the service s with the access keys *AccessKey* and keywords *Keyword*, **SP** constructs the controlled search polynomial $F_{AccessKey,Keyword}(x)$ presented in Section 4.3 for the sets *AccessKey* and *Keyword*, and then encrypts s and the coefficients of $F_{AccessKey,Keyword}(x)$. The algorithm consists of the following steps:

  1. **SP** describes the technique detail information about invocation as s, and describes the service's functionalities as $Keyword = \{k_1, k_2, \cdots, k_t\}$. **SP** also shares a set of access keys ak with service users. If **SP** allows an user to access the service, **SP** puts the user's access key in the set *AccessKey*, denoted as $AccessKey = \{ak_1, ak_2, \cdots, ak_s\}$.

  2. **SP** generates a polynomial $F_{AccessKey,Keyword}(x) = a_{s+t}x^{s+t} + \cdots + a_1 x + a_0$ satisfying

$$F_{AccessKey,Keyword}(x) = \begin{cases} r + i, & x = ak_i \in AccessKey \\ -r, & x \in Keyword \end{cases}$$

     where r is a random number. The coefficients $a + s + t, \cdots, a_1, a_0$ can be computed with Newton interpolation formulae.

3. **SP** encrypts the coefficients of $F_{AccessKey,Keyword}(x)$ using the BGN encryption and the encryption key pk as

$$\begin{aligned} EC &= \{E_{pk}(a_{s+t}), E_{pk}(a_{s+t-1}), \cdots, E_{pk}(a_0)\} \\ &= \{g^{a_{s+t}} h^{r_{s+t}}, g^{a_{s+t-1}} h^{r_{s+t-1}}, \cdots, g^{a_0} h^{r_0}\} \end{aligned}$$

   where $r_{s+t}, r_{s+t-1}, \cdots, r_0$ are $(s+t+1)$ random numbers.

4. **SP** computes a random encryption key $rk = H(g^{r_m})$ for the symmetric encryption scheme SE, where r_m is a random number. The service information s is encrypted as $SE(s, rk)$.

5. To enable authorized users who possess an access key in *AccessKey* to decrypt the encryption of s, **SP** computes $C = \{g^{r_m/ak_1}, \cdots, g^{r_m/ak_s}\}$

6. **SP** registers the servicy by storing the encrypted service registration $S = (SE(s, rk), EC, C)$ in **SD**.

- ***ServiceRequest(**ak, k**)***: With a access key ak and a keyword k, **SU** generates the encrypted request Q and submits it to **SD**.

$$Q = \{E_{pk}(ak^{s+t} + k^{s+t}), E_{pk}(ak^{s+t-1} + k^{s+t-1}), \cdots, E_{pk}(ak^0 + k^0)\}$$

- ***ServiceMatch(**S, Q**)***: When **SD** receives the encrypted request Q from **SU**, **SD** matches the encrypted service registration S with the encrypted request Q through polynomial evaluation discussed in Section 4.3. This algorithm could be seemed as a secure version of the *Controlled Search* algorithm in Figure 4.1. The algorithm consists of the following steps:

  1. **SD** scans the encrypted service registration $S = (SE(s, rk), EC, C)$.

  2. For all $1 \leq i \leq s$, **SD** runs the algorithm *Verify(*EC, Q, i, pk, vk*)* described in Figure 4.2.

3. If there is an $i, (1 \leq i \leq s)$ that the algorithm *Verify(EC, Q, i, pk, vk)* returns 1, **SD** sends the encryption $SE(s, rk)$ to **SU** with the i-th element g^{r_m/ak_i} in the set C.

- ***ServiceRetriver(SE(s, rk), g^{r_m/ak_i})***: If **SU** searches service with (ak, k) and receives $SE(s, rk)$ and g^{r_m/ak_i} from **SD**, **SU** reconstructs the symmetric encryption key as $rk = H((g^{r_m/ak_i})^{ak})$ and then use rk to decrypt the encryption $SE(s, rk)$.

4.5 Correctness and Security

We prove that our privacy preserving and controlled access service discovery and matching protocol presented in Section 4.4 satisfies the three functionality requirements stated in the Definition 4.1.

Theorem 4.4 (Correctness). *The privacy preserving and controlled access service discovery and matching protocol presented in the Section 4.4 is correct.*

1. *If $SE(s, rk)$ and g^{r_m/ak_i} are returned correctly, the algorithm* ServiceRetriver *will compute the random encryption key rk correctly.*

2. *When $ak \in AccessKey$ which means that* **SU** *can access the service, the algorithm* ServiceMatch *will return the encrypted service if $k \in Keyword$; otherwise, the algorithm returns nothing.*

3. *When $ak \notin AccessKey$ which means that* **SU** *cannot access the access, the probability that the algorithm* ServiceMatch *returns the encrypted service is negligible for any keyword k.*

Proof 1. In the algorithm *ServiceMatch(S, Q)*, **SD** repeatedly calls the algorithm *Verify* with different parameters i. The algorithm *Verify* with the parameter i returns 1 only when $F_{AccessKey,Keyword}(ak) + F_{AccessKey,Keyword}(k) = i$. According to the definition of the function $F_{AccessKey,Keyword}(x)$, this means that i is the index of *ak* in the set *AccessKey*. That is, $ak = ak_i$. Hence, when the algorithm **ServiceRetriver** receives the inputs $SE(s, rk)$ and g^{r_m/ak_i} from **SD**, **SU** reconstructs the random encryption key *rk* correctly as

$$H((g^{r_m/ak_i})^{ak}) = H((g^{r_m/ak_i})^{ak_i}) = H(g^{r_m})$$

2. If $ak \in AccessKey$ and $k \in Keyword$, the definition of the controlled search polynomial ensures that $F_{AccessKey,Keyword}(ak) + F_{AccessKey,Keyword}(k) = i$, where i is the index of the *ak* in the set *AccessKey*. Hence, when **SD** uses the algorithm **Verify** in the i-th time, the algorithm **Verify(**EC, Q, i, pk, vk**)** will return 1 and then the algorithm **ServiceMatch(**S, Q**)** will return $SE(s, rk)$ and the i-th element g^{r_m/ak_i} in the set C to **SU**. This is ensured by the Theorem 4.3 and the following equations.

$$\begin{aligned}
& \prod_{i=0}^{n} e(E_{pk}(a_i), E_{pk}(ak^i + k^i)) \\
= & \prod_{i=0}^{n} e(E_{pk}(a_i), E_{pk}(ak^i)E_{pk}(k^i)) \\
= & \prod_{i=0}^{n} e(E_{pk}(a_i), E_{pk}(ak^i)) \prod_{i=0}^{n} e(E_{pk}(a_i), E_{pk}(k^i)) \\
= & \prod_{i=0}^{n} E_{pk}(a_i ak^i) \prod_{i=0}^{n} E_{pk}(a_i k^i) \\
= & E_{pk}(f(ak))E_{pk}(f(k)) \\
= & E_{pk}(f(ak) + f(k)) \\
= & E_{pk}(i)
\end{aligned}$$

3. Because **SU** does not have the correct access key and has no information about the set *Keyword*, **SU** can only generate a service request with a random chosen access key *ak* and a random chosen keyword *k*. According to the Theorem 4.2, for the random *ak* and *k*, the probability that **SD** returns services does not exceed $s(s+t)^2|R|/|D|^2$, where D, R are the domain and the range of the polynomial $F_{AccessKey,Keyword}(x)$. In our protocol, the domain D is $\{0,1\}^N$, where N is the

longest length of the access keys. The range R is the plaintext space of the BGN encryption scheme, whose size is $q_1 q_2$. Therefore, the error probability becomes negligible if N are large enough.

$\square$

In the following, we prove that our protocol also satisfies the three security requirements stated in the Definition 4.1.

Theorem 4.5 (Security)**.** *The privacy preserving and controlled access service discovery and matching protocol presented in the Section 4.4 is secure, and has the following properties:*

1. *Without the correct access key ak, the adversary including the unauthorized **SU** and **SD** cannot decrypt the encrypted s generated in the algorithm ServiceRegistration with non-negligible probability.*
2. *Without the correct access key ak, in the algorithm ServiceMatch, the adversary including the unauthorized **SU** and **SD** cannot determine whether $k \in Keyword$ with probability larger than $1/2$ non-negligibly.*
3. *Without the correct access key ak, the adversary including **SU** and **SD** cannot compute the access key ak and the keyword k from the output of the algorithm ServiceRequest without resolving the discrete logarithm problem.*

Proof 1. All information about the service stored in **SD** is the encryption $SE(s, rk)$ and the set $C = \{g^{r_m/ak_1}, \cdots, g^{r_m/ak_s}\}$. Because we assume that the symmetric encryption scheme SE is semantically secure, no information about s is revealed if the adversary cannot compute the encryption key $rk = H(g^{r_m})$, where r_m is a random number. If the adversary has no information about g^{ak} for any $ak \in \{ak_1, \cdots, ak_s\}$, the set C reveals no information about r_m. Then, the adversary cannot reconstruct the encryption key rk and decrypt the encryption $E(m, rk)$ with non-negligible probability.

In our protocol, two messages about the ak are available for the adversary. The first is the encrypted request Q from **SU**. The second is the encrypted polynomial $F_{AccessKey,Keyword}(x)$ stored in **SD**.

For $Q = \{E_{pk}(ak^{s+t} + k^{s+t}), E_{pk}(ak^{s+t-1} + k^{s+t-1}), \cdots, E_{pk}(ak^0 + k^0)\}$, the information about the access key ak is hidden by the keyword k. The probability that the adversary correctly remove the keyword k from the encrypted query is at most $1/|Directory|$, where the $Directory$ is the set of all possible keywords. However, even when the keyword k has been guessed correctly, the adversary only learns g^{ak}. According to the DDH assumption, the probability that the adversary computes g^{r_m} from $g^{r_m/ak}$ and g^{ak} is still negligible. Therefore, from Q, the probability that the adversary decrypts the encryption $E(m, rk)$ successfully is negligible.

For the encrypted polynomial $F_{AccessKey,Keyword}(x)$, the probability that the adversary finds a correct access key ak is less than the probability that the adversary finds two random elements a and b satisfying $1 \leq f(a) + f(b) \leq s$ times the probability that $b \in Keyword$. That is

$$\Pr[ak \in AccessKey] < \Pr[1 \leq f(a) + f(b) \leq s] \cdot \Pr[b \in Keyword]$$

According to the Theorem 4.2 and the discussion in the proof of the Theorem 4.4, we have

$$Pr[ak \in AccessKey] < \frac{N}{(q_1 q_2)^2} \times \frac{1}{t}$$

where $\{0,1\}^N$ is the space of the access key, q_1, q_2 are parameters of the BGN encryption scheme, t is size of the set $Keyword$. When the parameters q_1, q_2 are large enough, the probability that the adversary guesses the correct access key is negligible.

2. Similar to the access key, in our protocol, two messages about the *Keyword* are available for the adversary. The first is the encrypted request Q from **SU**. The second is the encrypted polynomial $F_{AccessKey,Keyword}(x)$ stored in **SD**.

   For Q, the information about the keyword k is hidden by the access key ak. The probability that the adversary correctly remove the access key ak from the encrypted query is negligible, when the space of the access key is large enough.

   For the encrypted $F_{AccessKey,Keyword}(x)$, to determine if a keyword $k \in Keyword$, the adversary has to verify whether $F_{AccessKey,Keyword}(k) = r$, where r is a random number used in the construction of $F_{AccessKey,Keyword}(x)$. Because the adversary does not know any access key in *AccessKey* and any keyword in *Keyword*, the adversary cannot distinguish r from any random numbers. Therefore, given a keyword k, the probability that the adversary successfully determine whether $F_{AccessKey,Keyword}(k) = r$ cannot be non-negligibly larger than random guessing, whose probability is $1/2$.

3. From the encrypted request Q, the adversary cannot learn more information than $\{g^{ak^i+k^i} | 1 \leq i \leq s+t\}$. The adversary cannot compute ak and k from these information due to the hardness of the discrete logarithm problem.

$\square$

4.6 Extension for Other Types of Matching

In this section, we will extend our privacy preserving and controlled access service discovery and matching protocol to support conjunction keywords matching and capability-based service matching.

Conjunction Keywords Matching

In the protocol presented in Section 4.4, the service matching will succeed if one keyword is matched. Single keyword matching is easy but may not be accurate enough.

For example, the service matching with the keyword *Print* will return all print services. With the support of conjunction keywords matching, it is able to specify the request of HP's print service with two keywords *HP* and *Print*.

Our privacy preserving and controlled access service discovery and matching protocol can be easily extended to support conjunction keyword matching. In the *ServiceRequest* algorithm, if **SU** wants to search services with n keywords $\{k_1, k_2, \cdots, k_n\}$, **SU** generates the encrypted request Q as

$$Q = \{E_{pk}(ak^{s+t} + \sum_{j=1}^{n} k_j^{s+t}/n), E_{pk}(ak^{s+t-1} + \sum_{j=1}^{n} k_j^{s+t-1}/n), \cdots, E_{pk}(ak^0 + \sum_{j=1}^{n} k_j^0/n)\}$$

Then, when the *ServiceMatch* algorithm invokes the *Verify* algorithm to match services, the *Verify* algorithm runs as

$$\begin{aligned} C &= \prod_{i=0}^{s+t} e(E_{pk}(a_i), E_{pk}(ak^i + \sum_{j=1}^{n} k_j^i/n)) \\ &= \prod_{i=0}^{s+t} e(g,g)^{a_i \cdot ak^i + a_i \cdot \sum_{j=1}^{n} k_j^i/n} \cdot e(g,h)^{a_i r_{i,2} + r_{i,1}(ak^i + \sum_{j=1}^{n} k_j^i/n) + r_{i,1} r_{i,2} r_0} \\ &= e(g,g)^{\sum_{i=0}^{n}(a_i \cdot ak^i + a_i \cdot \sum_{j=1}^{n} k_j^i/n)} \cdot e(g,h)^{\sum_{i=0}^{n}(a_i r_{i,2} + r_{i,1}(ak^i + \sum_{j=1}^{n} k_j^i/n) + r_{i,1} r_{i,2} r_0)} \\ &= e(g,g)^{f(ak) + \sum_{j=1}^{n} f(k_j)/n} \cdot e(g,h)^{r'} \end{aligned}$$

,where $r' = \sum_{i=0}^{n}(a_i r_{i,2} + r_{i,1}(ak^i + \sum_{j=1}^{n} k_j^i/n) + r_{i,1} r_{i,2} r_0)$.

When $\{k_1, k_2, \cdots, k_n\} \subseteq \mathit{Keyword}$, $f(k_i) = -r$, where r is the random number used in the construction of polynomial $f(x)$ as defined in (4.1). Hence, if $ak = ak_i \in \mathit{AccessKey}$ and $f(ak_i) = r + i$,

$$\begin{aligned} C &= e(g,g)^{f(ak) + \sum_{j=1}^{n} f(k_j)/n} \cdot e(g,h)^{r'} \\ &= e(g,g)^{r + i + \sum_{j=1}^{n} -r/n} \cdot e(g,h)^{r'} \\ &= e(g,g)^{i} \cdot e(g,h)^{r'} \end{aligned}$$

SD will returns $SE(s, rk)$ and g^{r_m/ak_i} to **SU** in the algorithm *ServiceMatch*, because

$$\begin{aligned} C^{vk} &= e(g,g)^{i \cdot vk} \cdot e(g,h)^{r' \cdot vk} \\ &= e(g,g)^{i \cdot vk} \end{aligned}$$

That is, the service successfully matches with the request with n keywords $\{k_1, k_2, \cdots, k_n\}$.

Capability-based Service Matching

Service matching based on services' capabilities was first presented by Paolucci *et al.* [104]. The motivation of capability-based service matching is to search services on the basis of what they provide. While UDDI describes services by their name and attributes, capability-based service matching describes services in terms of inputs, outputs, preconditions and effects.

In capability-based service matching, **SP** registers four categories of keywords to describe their services' inputs, outputs, preconditions and effects of the service. **SU** searches services with four separate keywords for inputs, outputs, preconditions and effects. These keywords will be matched together, and only services that match all four keywords from different categories are matched.

Similar to conjunction keyword matching, capability-based service matching also matches multiple keywords. While conjunction keyword matching only requires the number of matched keywords, capability-based service matching additionally requires that these matched keywords come from the four different categories. To separate keywords from different categories, we can add unique prefix for keywords in each category. For example, for all keywords describing services' input, we can add a prefix ”Input-”. Then the capability-based service matching can be implemented as the conjunction keyword matching shown in Section 4.6.

Another approach of capability-based service matching is to design four different controlled search polynomial for each categories of keywords, and **SU** submits four service requests to **SD** for inputs, outputs, preconditions and effects correspondingly. **SD** returns the encrypted service to **SU** only if the service matches all of four service requests simultaneously.

4.7 Performance Evaluation

In this section, we will evaluate the performance of the privacy preserving and controlled access service discovery and matching protocol. We will use the UDDI server introduced in 2.2 and shipped with Windows 2003 Server from Microsoft to organize services in the service directory, and implement a set of security services to support AES encryption and decryption operations. All security services share the same interface, but have different QoS and different tradeoff parameters. All services are hosted in a personal computer with E8400 3GHz CPU and 4G memory.

Integration of Additional Service Information in UDDI

UDDI does not specify the structure of the tModels, and hence makes it possible to integrate additional information with the services' registrations, such as services' categories or keywords [105], or services' QoS information [22].

In [22], the following three kinds of approaches are discussed to integrate services' QoS in UDDI.

- Type based approach. This approach creates a set of tModels for each QoS aspect, and specifies services' qualities on one QoS aspect as the corresponding tModel's value. Hence, if a service has QoS information on *n* QoS aspects, its UDDI registration needs to includes *n* tModels.
- Keyword based approach. This approach creates a general tModel for all QoS

aspects. All QoS aspects share the same tModel, but use two pairs of key name and key value to specify the name of the QoS aspect and the services' quality on this aspect.

- Ontology based approach. This approach describes the relations among QoS aspects with ontologies, which formally specifies the semantic of QoS description. Hence, by specifying the ontologies with tModels, this approach can ensure the consistency among all QoS descriptions integrated in the UDDI registation.

We simplify the second approach, and use it to integrate services' keyword sets and access key sets. We define the PPS tModel. To include services' keywords and access keys with their UDDI registration, services' registrations can include the PPS tModel in its instance information, and add the URL to an external document as the PPS tModel's overview document URL, which includes detail information about keywords and access keys.

Using separate documents to store additional information rather than merge all information in a huge documents have several advantages. First, it is much easier and faster to validate small XML documents than large documents. Second, the UDDI can selectively load the XML documents of additional information only when such information is required, which save UDDI server's resources. Third, separating services' additional information from their registrations makes service providers easier to update these additional information.

privacy preserving and controlled access Service Discovery and Matching

We implement the privacy preserving and controlled access service discovery and matching protocol with the GMP library (`http://gmplib.org/`) for high precision arithmetic and the PBC library (`http://crypto.stanford.edu/pbc/`) for the implementation of the privacy preserving polynomial evaluation presented in Section 4.3.

Table 4.1: The time of operations required by the privacy preserving and controlled access service discovery and matching protocol, where s is the size of the keyword set and t is the size of the access key set. The data is collected at a personal computer with E8400 3GHz CPU and 4G memory

Operation	**Times with different key size (ms)**			
	128	256	512	1024
Construction of controlled search polynomial	3.33	3.33	3.33	3.33
Encryption of BGN	8.43	41.02	254.28	1757.86
Additive homomorphic operation of BGN	0	0.01	0.12	0.217
Multiplicative homomorphic operation of BGN	12.18	74.79	552.53	4187.60

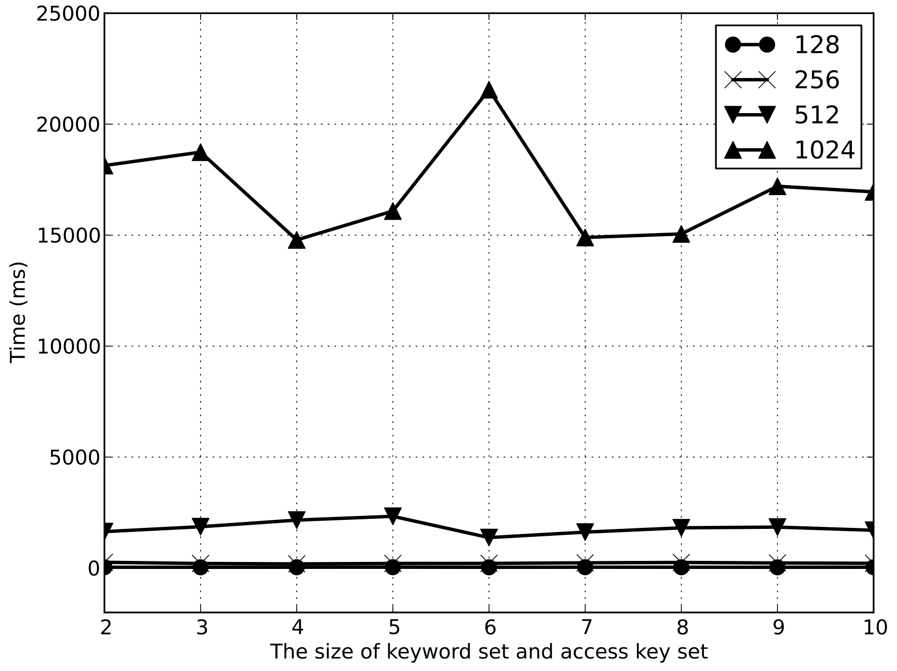

Figure 4.3: The time required by the algorithm *Initialization*

The basic operations required by the five algorithms presented in Section 4.4 are listed in the Table 4.1.

From the Table 4.1, we can see that the most expensive operations are the encryption and the multiplicative homomorphic operation of BGN. The encryption is required by the algorithm *ServiceRegistration* and algorithm *ServiceRequest*. The multiplicative homomorphic operation is required by the algorithm *ServiceMatch*. The time

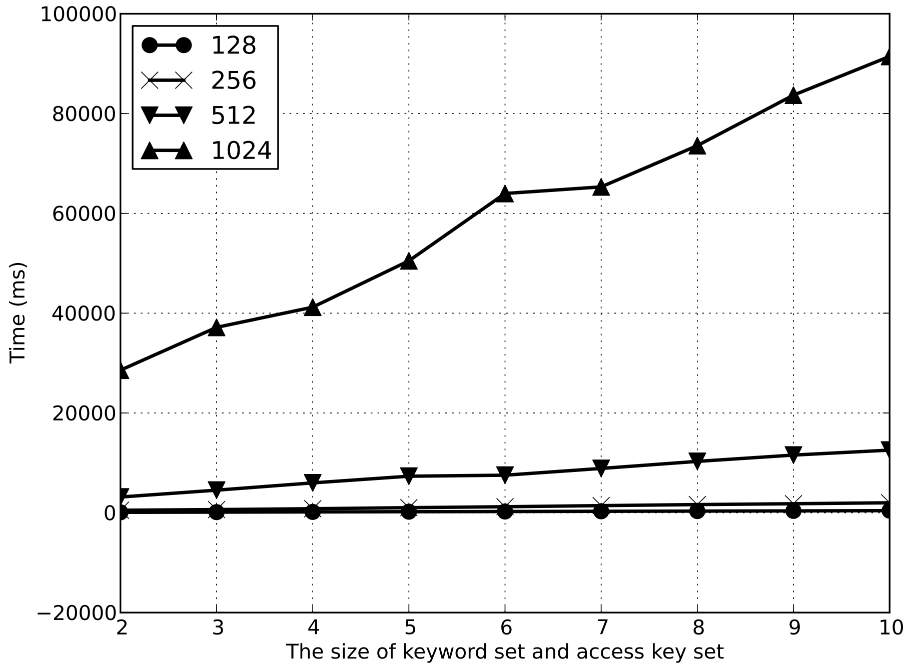

Figure 4.4: The time required by the algorithm *ServiceRegistration*

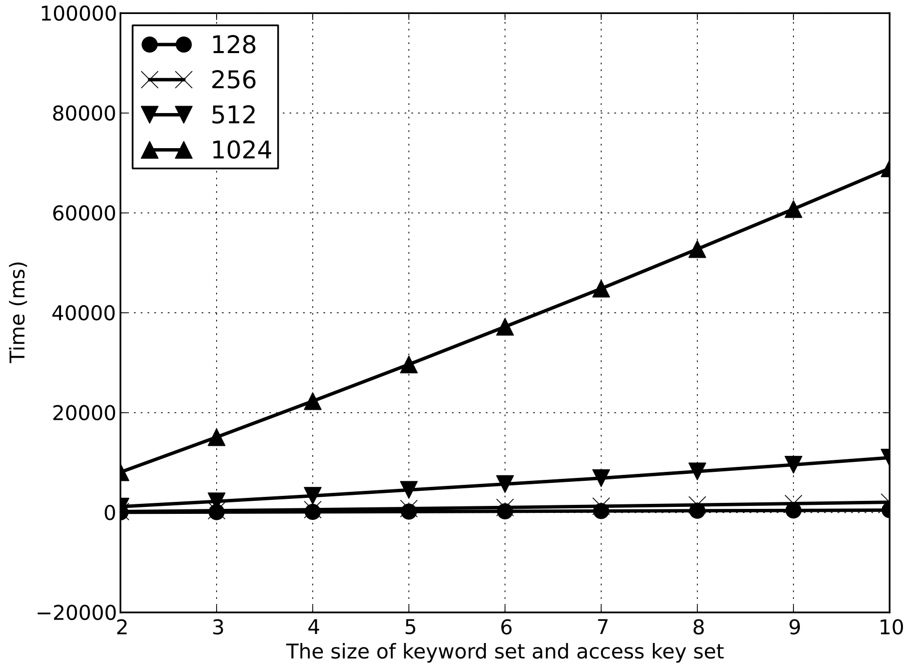

Figure 4.5: The time required by the algorithm *ServiceRequest*

Table 4.2: The time of algorithms in the privacy preserving and controlled access service discovery and matching Protocol

Algorithm	Key Length	**The size of keyword set and access key set**								
		2	3	4	5	6	7	8	9	10
Initialization	128	37	35	38	40	35	39	37	35	38
ServiceRegistration	128	90	128	170	212	250	297	337	374	427
ServiceRequest	128	42	86	133	87	244	302	364	415	486
ServiceMatch	128	90	160	220	287	357	424	487	539	625
ServiceRetrieve	128	0	0	0	0	0	0	0	0	0
Initialization	256	260	207	189	208	211	237	253	229	216
ServiceRegistration	256	509	643	814	1024	1210	1434	1641	1812	2004
ServiceRequest	256	197	381	580	794	1015	1264	1523	1792	2073
ServiceMatch	256	578	1001	420	1837	2233	2698	3109	3532	3995
ServiceRetrieve	256	0	0	0	0	0	0	0	0	0
Initialization	512	1642	863	2161	2332	1373	1619	1813	1846	1703
ServiceRegistration	512	3157	4532	5990	7327	7530	8884	10298	11561	12555
ServiceRequest	512	1185	2237	3346	4529	5701	6884	8231	9567	10989
ServiceMatch	512	4196	7203	10261	13293	16195	19191	22359	25461	28547
ServiceRetrieve	512	0	0	0	0	0	0	0	0	0
Initialization	1024	18137	18740	14785	16087	21554	14896	15056	17204	16951
ServiceRegistration	1024	28576	37156	41202	50491	63982	65306	73572	83692	91450
ServiceRequest	1024	8078	15095	22311	29658	37210	44844	52757	60772	68926
ServiceMatch	1024	31754	54621	77938	100840	124029	146312	170181	192236	215489
ServiceRetrieve	1024	0	0	0	0	0	0	0	0	0

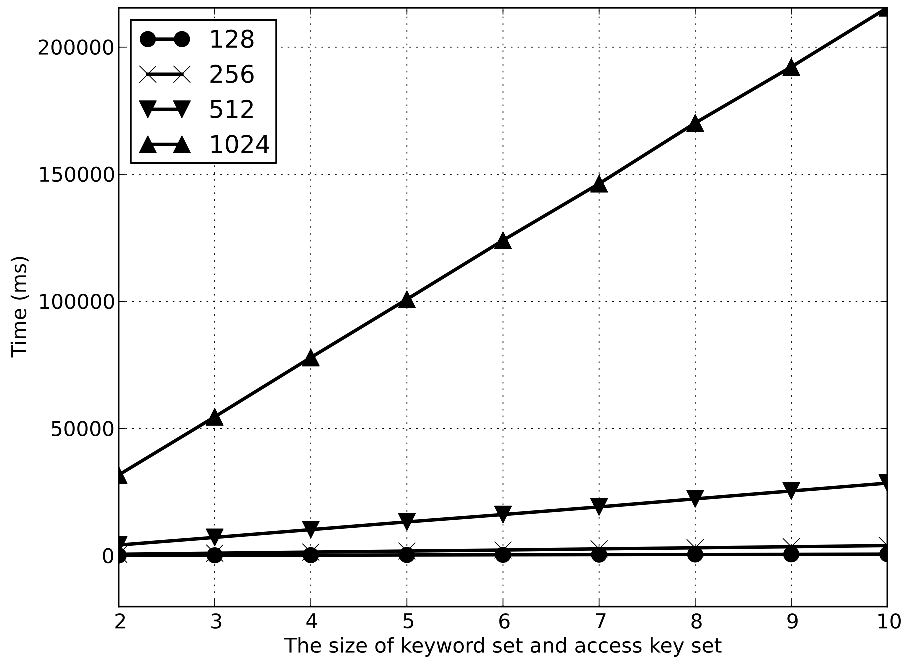

Figure 4.6: The time required by the algorithm *ServiceMatch*

required by all algorithms is listed in the Table 4.2, and shown in the Figure 4.3, Figure 4.4, Figure 4.5, and Figure 4.6.

- Algorithm *Initialization*. This algorithm initializes parameters. Especially, it needs to construct two cyclic groups $\mathbb{G}, \mathbb{G}_1$ with the same order $n = q_1 q_2$, where q_1 and q_2 are two large primes. This algorithm uses a random algorithm to find large primes which is the most expensive operation and dominates the time of the whole algorithm. The random algorithm generally needs more time to find longer primes but the actual time is not stable due to the randomness of the algorithm. Hence, from Table 4.2 and Figure 4.3, we can see that the time increases for longer key size and is independent with the size of keyword set and access key set. And, there is no clear relation between the time and the size of keyword set and access key set when the key size is fixed.

- Algorithm *ServiceRegistration*. This algorithm encrypts services' registrations with their keyword set and access key set. It constructs controlled search polynomial and encrypts the polynomial's coefficients. To register a service with s keywords and t access keys, this algorithm needs to encrypt $s+t$ coefficients. Hence, from Table 4.2 and Figure 4.4, we can see that the time increases linearly with the increasing of $(s+t)$. Because the encryption of BGN is slower for larger key size, the time also increases with the increasing of key size.

- Algorithm *ServiceRequest*. This algorithm encrypts the keywords and the access keys used by the service users in searching services in the service directory. To generate a valid service request for a service with a $(s+t)$-degree controlled search polynomial, the keywords and the access keys need to be encrypted $(s+t)$ times. Hence, from Table 4.2 and Figure 4.5, we can see that the time also increases linearly with the increasing of $(s+t)$. Let $T_{Enc,l}$ be the time of encrypting one message with BGN and the key size l. To generate a service request for all services in the service directory, the time required by the algorithm *ServiceRequest* is $O(N \cdot T_{Enc,l})$, where N is the largest degree of all services' controlled search polynomials, which is independent with the number of services in the service directory.

- Algorithm *ServiceMatch*. This algorithm matches service requests and services by evaluating services' controllable search polynomials with encrypted coefficients and points. The evaluation of a $(s+t)$-degree controlled search polynomial requires $(s+t)$ additive homomorphic operations and $(s+t)$ multiplicative homomorphic operations of BGN. Hence, from Table 4.2 and Figure 4.6, we can see that the time increases linearly with the increasing of $(s+t)$.

- Algorithm *ServiceRetrieve*. This algorithm decrypts the encrypted service registrations when the algorithm *ServiceMatch* matches the services successfully. It only requires one decryption operation of the symmetric encryption algorithm (i.e, AES algorithm in our experiment), one hash operation, and one discrete power operation, all of which are very fast. Hence, from Table 4.2, we can see that the time is 0 which means that the time is much shorter than one millisecond.

The evaluation results show that the privacy preserving and controlled access service discovery and matching protocol is very efficient with key size 128 and key size 256. With key size 512, the matching of a service can finish within half minutes, which is reasonable for small number of services. When the key size is 1024, the protocol needs half minute to four minutes to match a service, which is too long.

Chapter 5

QOS ASPECTS AND QOS REQUIREMENTS

All Step 1.3), Step 2) and Step 4) of the overall approach presented in Chapter 3 require services' QoS information and/or service users' QoS requirements. While services' functionalities can be modeled as operations with inputs and outputs [4], there is no universal pattern to model and specify services' QoS due to the variety of QoS aspects.

5.1 QoS Aspects

The following is a set of example QoS aspects.

- Delay. Delay is measured as the average waiting time of service users between sending a request message to the service and receiving a response message from the service. The unit of individual service request's delay is seconds, which may be very small like several microseconds, or infinitely large when the service request is banned or thrown away by the service. For a service, its delay is measured as the average delay of all requests, i.e., seconds per request.
- Price. Price is measured as the amount of money that service users have to pay for the usage of services, which may be 0 or very large. The unit of price is the same unit of money. A service can charge its users based on the time or the frequency of usage.
- Throughput. Throughput is measured as the maximum number of request messages that the service can accept per second. The unit of throughput is number per second, which may be 0 when the service does not accept any requests, or very large when the service is very powerful. There is no theoretical upper limit for the throughput.

- Capability. Capability is measured as the largest request message that the service can accept, which may be 0 for empty messages or very large. The unit of capability is bit. There is also no theoretical upper limit for the capability.

- Availability. Availability is measured as the percentage of time that the service is available for invocation since the registration of the service in the service directory. Availability is a number from 0% to 100%.

- Security. Service's security quality includes a set of QoS aspects, such as confidentiality, integrity, authentication and authorization. Unlike above QoS on performance, there is no nature metric for security. Possible security metrics are discussed in Chapter 6.

- Reliability. Reliability is measured as the percentage of users' requests that have been correctly processed and responded. Reliability is a number from 0% to 100%. Reliability is similar to availability but more restrict. When the service is unavailable, it definitely unreliable. But, even when the service is available, it still can be unreliable. Hence, services' reliability is always smaller than their availability.

- Reputation. Reputation is a special kind of QoS different from all above QoS aspects. While above QoS aspects are measuring services' performance, reputation is used to measure the accuracy of the measurement of other QoS aspects, especially when the QoS monitor relies on services to report their QoS. In this case, services with lower reputation may cheat the monitor by reporting better QoS. The unit of reputation is a number from 0 representing no reputation to 1 representing perfect reputation.

All of these QoS aspects are applicable for all kinds of services. However, for specific services, more QoS aspects may need to be considered. For example, for online

Table 5.1: Examples of QoS aspects for general services

Category	**QoS Aspect**	**Unit**	**Lower Bound**	**Upper Bound**
Cost QoS	Delay	seconds / request	0	∞
	Price	dollar	0	∞
Utility QoS	Throughput	# request / second	0	∞
	Capability	bits	0	∞
	Availability	percentage	0	100
	Security	percentage	0	100
	Reliability	percentage	0	100
	Reputation	null	0	1

video services, an important QoS aspect is the video resolution. For voice communication services, jitter is another very important QoS aspect. More QoS aspects for specific services can be found in [106]. Although there are a lot of QoS aspects, all QoS aspects can be generally classified as the following two categories:

- Cost QoS. For Qos aspects in this category, smaller values represent better qualities. Delay and price belong to this category.
- Utility QoS. For QoS aspects in this category, larger values represent better qualities. Throughput, capability, availability, security, reliability, and reputation belong to this category.

All QoS aspects listed above are summarized in Table 5.1.

5.2 QoS Requirement Specification

Suppose **SU** has requirements on n QoS aspects $\{c_1, c_2, \cdots, c_n\}$. For each QoS aspect c_j, **SU** specifies his/her QoS requirement of c_j as a tuple $req_j = \{l_j, u_j, w_j, r_j\}$ as follows

- The l_j and u_j are **SU**'s expected lower and upper bounds of the QoS on c_j.

- The w_j is the importance weight of the requirement req_j (i.e., **SU**'s preference on c_j), which can be specified as either a numeric value within $[0,1]$ or one of the fuzzy linguistic term from $\{$Don't care, Unimportant, Medium, Import, Very important$\}$ when **SU** does not know which weight value should be specified.
- The r_j is **SU**'s confidence in all the values of l_j, u_j and w_j specified in req_j, which is a value within $(0,1]$.

Expected QoS l_j and u_j

The l_j and u_j are **SU**'s expected lower and upper bound of services' qualities on the QoS aspect c_j. If c_j is a utility QoS such as delay, l_j represents the least acceptable quality, and u_j represents the expected best quality. If c_j is a cost QoS such as price, l_j represents the expected best quality, and u_j represents the least acceptable quality. If a service's quality of c_j is worse than the least acceptable quality, the service's quality does not satisfy **SU**'s requirements and will not be returned to **SU**. On the other hand, if a service's quality on c_j is better than the expected best quality, the service's quality has already perfectly satisfied **SU**'s requirements, but better quality does not provide additional benefit for **SU**. Hence, all services' qualities are the same for **SU** if their qualities on c_j are larger than u_j when c_j is an utility QoS asepct or smaller than l_j when c_j is a cost QoS aspect, no matter how larger or smaller of their actual qualities.

If **SU** has no requirement on the expected least acceptable or best quality or does not know how to specify them, l_j and u_j may be left blank in req_j. In this case, l_j and/or u_j can be estimated as follows:

First, if the metric for c_j has upper bound and lower bound, l_j and u_j can be set to its metric's upper bound and lower bound respectively. For example, the QoS availability in Table 5.1 has an upper bound 100% and a lower bound 0%. Hence, for the QoS aspect availability, we can set $l_j = 0\%$ and $u_j = 100\%$ if their values are missed in req_j.

Table 5.2: The mapping between linguistic terms and importance weight values.

Linguistic term	**Weight value**
Don't care	0
Unimportant	0.25
Medium (default value)	0.5
Important	0.75
Very important	1

Second, if the c_j's metric has no upper bound and/or lower bound such as price in Table 5.1 which has a lower bound 0 but has no upper bound, the values of l_j and u_j are defined as the lowest and largest qualities of all available services returned by **SD**.

Importance Weight w_j

Because **SU** may have requirements on various QoS aspects, the importance weights allow **SU** to express his/her preferences on various QoS aspects. The w_j is a value within $[0,1]$. A larger w_j means that the corresponding QoS aspect is more important for **SU**, and services' qualities on this QoS aspect will have more impact on the service ranking.

When **SU** does not know how to specify the preferences on QoS aspects with numeric weights, **SU** can also specify preferences through linguistic terms. Linguistic terms are mapped to weight values as Table 5.2. If **SU** does not specify w_j in req_j, the default value 0.5 will be applied.

Confidence Value r_j

The confidence value r_j gives **SU** more flexibility to specify req_j, whose value is within $(0,1]$. A larger r_j means that **SU** has higher confidence in the specification of all the values of l_j, u_j and w_j, and a smaller r_j means lower confidence. If **SU** does not specify r_j's value in req_j, the default value 1 will be applied.

The value 1 means that **SU** is sure about his/her specifications of l_j, u_j and w_j, and then services' QoS will be evaluated exactly based on the specified values. A value within $(0,1)$ means that **SU** has some hesitation about the specifications, and hence the values of l_j, u_j and w_j will be adjusted accordingly during the evaluation of services' QoS.

- l_j is adjusted to $l_j r_j$, and u_j is adjusted to u_j / r_j. That is, the range of acceptable quality of c_j is extended from $[l_j, u_j]$ to $[l_j r_j, u_j / r_j]$
- For w_j, it is adjusted as $w'_i \rightarrow w_i r_i + 0.5(1 - r_i)$. When $r_j = 1$, there is no adjustment and $w'_i = w_i$. When r_j is close to 0 (i.e., **SU** has little confidence in the specifications), w'_j approaches to the default value 0.5.

Chapter 6

QOS MEASUREMENT

While services' qualities on some QoS aspects can be measured through observation, the qualities on some other QoS aspects are difficult to be quantitatively measured. In this chapter, we first discuss how to quantitatively measure services' security quality, which is one of the most important QoS aspect and is difficult to be measured. Then, we discuss how to develop quantitative metrics for those observable QoS aspects, such as delay and throughput. All metrics developed in this chapter are based on a set of parameters, which are adjustable by services and hence facilitate the tradeoff among various QoS aspects.

6.1 Security Metric

Most existing security metrics are qualitative [90, 89], which measure services' security as several discrete security levels, such as low, medium, and high. Security mechanisms in the higher level can provide better protection than those in the lower level. These security levels are usually manually defined by experts. For example, the National Security Agency suggests that 128-bit AES encryption can provide SECRECT level security, and 192-bit AES encryption can provide TOP SECRECT level security [107]. Another example of classifying services' security qualitatively is the online trusted third party authorizers. For example, TRUSTe (`http://www.truste.com/`), the world largest privacy seal program provider, has certificated thousands of businesses like Microsoft, IBM and eBay. These authorizers provide independent check of services' data management, security mechanisms, and reputations. They also check whether services satisfy standard regulations, such as HIPPA for Healthcare information, COPPA for information obtained from and/or about children, and GLBA for financial information. Based on the results of check, services' security can be classified to several levels. The major disadvantage of qualitative security metrics is that qualitative security metrics are too coarse for fine control of services' security. It cannot compare services within the same security level.

Compared with qualitative security metrics, quantitative security metrics measure services' security as numbers, and hence are much more accurate. A simple quantitative security metric can be defined as the number of vulnerabilities found by vulnerability scanners, the number of viruses detected by anti-virus software, or the number of intrusions detected by intrusion detection systems. However, these measurements only reflect the system's current status, but not the actual strength to resist attackers. A system with no intrusion detected only means that there is no attacker right now, but does not guarantee that the system will be secure under attacks.

There are some other qualitative security metrics which measure services' security by investigating the technical details of security mechanisms used by services [108, 88]. These security metrics measure services' security as the length of the key used by the services' security algorithms. For security algorithms, longer keys provide stronger security strength than shorter keys. Hence, the key length correctly represents the order relation of services' QoS on security. However, the key length cannot accurately represent the difference between services' QoS on security because there are other important factors besides key length affecting services' security, such as algorithm design, attacking approaches used by attackers, and attackers' computing power. For example, symmetric encryption algorithms and asymmetric encryption algorithms have different requirements on the key length. To achieve the same security strength of the symmetric encryption algorithm AES with a 256-bit key, the asymmetric encryption algorithm RSA needs a $15,360$-bit key, and the elliptic encryption needs a 571-bit key [109]. The security strength of security algorithms is not absolute, but relative to the capability of attackers. To achieve the same security strength, systems suffering from more powerful attackers have to use longer keys than systems with weak attackers. Furthermore, for selective encryption algorithms which encrypt partial information instead of all information for better performance [110, 111, 112, 113], the percentage of information encrypted also affects the security strength provided by the algorithms.

In this chapter, we will first use the security configuration vectors (SCV) to specify the parameters of security mechanisms, and then estimate possible attackers' computing power and their attacking technologies. The quantitative security metric developed in this chapter measures security as the probability of protecting a message from attackers' one attack attempt. Hence, a larger security value means that attackers need more effort to crack the protected message, and hence more secure.

Security Configuration Vector

The security configuration vector *SCV* is a set of security configuration parameters that determine services' security strength. If the service has multiple kinds of security functionalities, such as confidentiality and integrity, the service may have multiple *SCV* for each kind of security functionality correspondingly. *SCV* is defined as $\{F, A, l, v, p\}$.

- F is the security functionality supported by the services. It specifies the type of security protection the service has, such as confidentiality, integrity, and non-repudiation.
- A is the the algorithm used by the service to implement F. For confidentiality, A could be DES or AES. For integrity, A could be SHA-1 Hash or Digital Signature.
- l is the length of the key used by A. A longer key provides higher security.
- $v(A,l)$ is a function measuring the vulnerability of the algorithm A with l, which defines the probability of attackers cracking a message protected with A and keys with length l.
- p is the protection probability, which defines how much information of a message will be protected by A.

Vulnerability Function $\mathbf{v}(\mathbf{A},\mathbf{l})$

The vulnerability function $v(A,l)$ varies with the design and implementation of A and the evolution of attacking techniques. The vulnerability of AES is measured through the size of key space needed to be searched. For a brute-force attack, we have $v(AES,l) = 2^{-l}$. The most recent related-key analysis for AES [114] shows that the key space needed to be searched can be reduced to 2^{176} for 192-bits key length, and 2^{119} for 256-bits key length. Note that the complexity of AES with 256-bits key length is even smaller than AES with 192-bits key length because of the bad key schedule design for 256-bits key length. The AES with 128-bits key length is not affected by the related-key analysis, and all existing attacks that are better than the brute force attack are designed for reduced transformation rounds. For example, the complexity of the most efficient attack for AES with 128-bits key length is 2^{22} with 7 rounds, and 2^{44} with 8 rounds [114], while the standard implementation for AES with 128-bits key length requires 10 rounds. Hence, we can estimate $v(AES,l)$ as

$$v(AES,l) = \begin{cases} 2^{-128}, & l = 128 \\ 2^{-176}, & l = 192 \\ 2^{-119}, & l = 256 \end{cases}$$

For RSA and the general number field sieve attack (GNFS) [115], which is the known most efficient integer factoring algorithm for integers with more than 100 bits, the vulnerability function $v(RSA,l)$ can be estimated as

$$v(RSA,l) = \begin{cases} 0.93 \times 10^{-23}, & l = 768 \\ 0.77 \times 10^{-26}, & l = 1024 \\ 0.65 \times 10^{-33}, & l = 2048 \end{cases}$$

Protection Probability **p**

While cryptography attempts to protect every bit of information, sometimes such extensive protection is unnecessary and unaffordable. Encrypting a part of messages instead of the whole messages will dramatically reduce the overhead caused by security mechanisms and still well prevents attackers from learning too much information. Some selective encryption algorithms have been proposed for specific applications, like image selective encryptions [110, 111] and video selective encryptions [112, 113].

According to *SCV*, for each message, only p percentage of the information included in the message will be protected. Hence,

- With probability $1-p$, the sensitive information included in the message is not protected ,and attackers can get such information trivially with probability 1.
- With probability p, the sensitive information included in the message is protected. In this case, attachers can still get the sensitive information by trying to crack the security mechanism, which probability is measured by $v(A,l)$.

Attacking Power

Another consideration in the security metric is the attacker's computing power, which specifies how many times the attacker can attack the service within one second (i.e., the attacking speed) using given attacking approach. The attacker's computing power can be estimated based on the sensitivity of the protected information and the expected protection period. First, the computing power of potential attackers for military systems and commercial systems are different due to different sensitivity of the information. Second, if the security mechanism is handling information that should be protected for a very long period, the security mechanism needs to estimate the capability of attackers in the expected period.

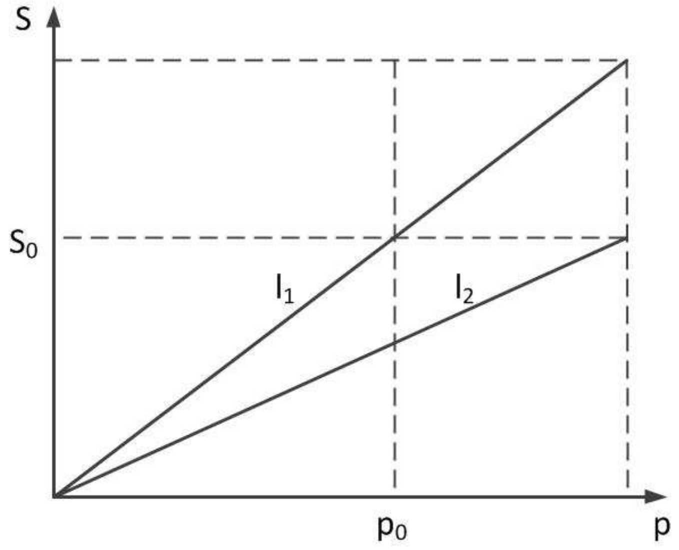

Figure 6.1: The relations between security S and protection percentage p for the same algorithm A with two different key lengths l_1 and l_2, where $l_1 > l_2$. S increases with the increasing of p, and increases faster with longer l.

Security Metric

Given the *SCV* and the attacker's computing power c, the probability of the attacker to successfully gain sensitive information from a protected message in one second is $c \cdot v(A,l)$. The overall security $S(SCV,c)$ is defined as follows:

$$S(SCV,c) = 1 - ((1-p) \times 1 + p \times cv(A,l)) = p(1 - cv(A,l)) \tag{6.1}$$

For the same security algorithm A with two different key lengths $l_1 > l_2$, Figure 6.1 shows the relation between the security metric (6.1) and the protection percentage p. As shown in Figure 6.1, the algorithm A can provide stronger security for the same p_0 with longer l_1, and can achieves the same security S_0 for longer l_1.

6.2 Quantitative Metrics for Observable QoS Aspects

A QoS aspect is observable if services' QoS on this QoS aspect can be measured by observing services' performance. For example, delay is an observable QoS aspect because it can be measured by recording requests' arriving times and sending times and computing their differences. On the other hand, security is unobservable because services cannot monitor the number of packages that have been cracked by the attackers, and then cannot compute the attackers' success probabilities. Although it is much easier for services to measure QoS through observation, developing quantitative metrics for observable QoS aspects similar to the security metric developed in Chapter 6 have several advantages. First, measuring QoS through observation needs to continuously monitor services, which will consume system resources. Second, measuring QoS through observation can only provide services' current QoS, but cannot predicate future QoS. Third, quantitative metrics with parameters enable services to control their QoS and adjust QoS by changing the values of parameters.

To develop quantitative metrics for observable QoS aspects, we first need to analyze which parameters/factors can affect services' QoS on these aspects, denoted as $P = \{p_1, p_2, \cdots, p_n\}$, and then find the relations between services' QoS and the parameter set P.

The Activity-State-QoS (ASQ) model developed in [15] can be used to develop such metrics. The ASQ model collects services' historic QoS data and the corresponding system resources and configurations. ASQ model uses Mann-Whitney test [116] to category all possible parameters and remove unrelated parameters. All remaining parameters belong to P. The relations between services' QoS and P are discovered with linear regression and Tukey's Honest Significant Difference test [117]. However, services still have no fine control on their QoS with metrics developed with ASQ model, because the parameter set P used by the ASQ model includes both parameters that services can control and available resources that services have no direct control. Better metrics should be only based on controllable parameters.

In this chapter, we use the secure VoIP service S as an example to show how to develop an quantitative metric for delay from a set of controllable parameters P. Unlike ASQ model which completely relies on historic data to determine P and discover relations, we first determine *P* and build the model to explain how *P* affects *S*'s delay through theoretical analysis, and develop the metric with *P* and a set of unknown variables. Then, we estimate these unknown variables' values by analyzing *S*'s historic data. The metric developed in this chapter is more accurate and controllable than the metrics based on observation and the metrics developed with ASQ model.

Delay Metric

The secure VoIP service *S* helps users to establish secure voice communication channels, which accepts voice data from the sources, encrypts the data, and then sends the encrypted voice data to the destinations. The user **SU** of *S* concerns three kinds of QoS: security, throughput and delay. The QoS on security is measured with *SCV* as the security metric (6.1), and the throughput is measured as the allowed number of incoming requests per second.

S's delay is the time between the source sending out voice data and the destination receiving the encrypted voice data, which consists of three parts. The first part is the delay generated by the network transmission, which is uncontrollable and determined by network environment, routing algorithms and many other factors. The second part is the time required by *S* to protect the voice communication. The third part is the waiting time of requests. When the throughput is large and *S* cannot process all requests on time, requests will be put in a waiting queue, which also increases the delay. With current fast network devices, the last two parts dominates the whole delay. Hence, we will ignore the first part, and measure *S*'s delay only with the time required for protection and the waiting time due to large throughput. We choose the parameter *P* as

$$P = \{SCV, t\} = \{F, A, l, v, p, t\} \tag{6.2}$$

where *SCV* is the security configuration vector defined in Section 6.1, *t* is the throughput.

Generally, increasing the parameter $t \in P$ for better throughput will increase the waiting time of requests that cannot be processed on time and hence increase the delay. Adjusting the parameters in $SCV \in P$ for better security will increase the protection time required by security enforcing mechanism and also increase the delay. There may be various metrics to define S's delay metric using P, but all these metrics should satisfy the following conditions. For simplicity without loosing generality, we assume that the size of **SU**'s requests are fixed, and each request message is divided to a set of packets with the same size.

- While all packets of requests need to wait in the waiting queue when S cannot handle requests on time, only partial packets need to be protected and hence suffer from the delay generated by the protection. According to p, the number of protected packet is $t \cdot p$.
- When there is no traffic, i.e. $t = 0$, the delay should be 0 for any SCV. Hence, we have $D(SCV, 0) = 0$.
- When t is small, S can complete the processing of one request before the next request coming in. That is, S has sufficient time to handle every request and no request needs to wait. In this case, D is mainly determined by the the time for protection, which is determined by SCV. Let $T_1(SCV)$ be the traffic threshold under which S can complete the processing of one request before the next request coming in. Then, when t is smaller than $T_1(SCV)$, i.e., $0 < t \leq T_1(SCV)$, we have $D(SCV, t) = D_1(SCV)$, where $D_1(SCV)$ is time of protecting one request. Because we assume that each request has the same size, $D_1(SCV)$ is a constant related to SCV.
- When t exceeds $T_1(SCV)$, S does not have sufficient system resources to protect a request on time and send the encrypted request out before the next request coming in. In this case, S will put requests in a waiting queue. Hence, when t keeps

increasing, more requests will be put in the queue, which leads to longer waiting time. That is, for the traffic t_1 and t_2 of two threads with $T_1(SCV) < t_1 < t_2$, we have $D(SCV,t_1) < D(SCV,t_2)$.

- The larger the t, the faster D increases as t increases. Hence, if $T_1(SCV) < t_1 < t_2$, $\partial D(SCV,t_2)/\partial t_2 > \partial D(SCV,t_1)/\partial t_1 > 0$.

- When t keeps increasing and approaches the maximum bandwidth capability, S will start to drop requests. Hence, there is an upper limit $T_2(SCV)$ for t , which then leads to an upper limit $D_2(SCV)$ for the delay D. For the VoIP service with time-out mechanisms, t may reach $T_2(SCV)$ and $D_2(SCV)$ before the system resources are exhausted.

- When t approaches $T_2(SCV)$, D approaches $D_2(SCV)$. Hence, the increasing speed of D will slow down when t approaches $T_2(SCV)$.

- When t is fixed, more efficient algorithm A, shorter key length l, or smaller percentage p for protecting the requests will lead to smaller D.

Based on the above observation, we obtain the following $D(SCV,t)$:

$$D(SCV,t) = \begin{cases} 0, & t = 0 \\ D_1(SCV), & 0 < t \leq T_1(SCV) \\ D_1(SCV) + a_1(1 - e^{-a_2(t - T_1(SCV))^{a_3}}), & t > T_1(SCV) \end{cases} \tag{6.3}$$

where $a_1, a_2, a_3, T_1(SCV)$ and $D_1(SCV)$ are five parameters related to SCV, but independent of t.

Metric (6.3) is a sigmoid function which is usually used to describe a progression starting from a small value and then accelerating and approaching an upper limit. For the metric (6.3), when t is less than $T_1(SCV)$, D is a constant $D_1(SCV)$. When the traffic exceeds T_1, D starts to accelerate and approaches the upper limit $D_1(SCV) + a_1$.

Parameter Estimation

S collects its delay with different P (6.2), and estimates the values of the parameters in the delay metric (6.3) through regression. The ideal case is that S can cover all possible values of P and the corresponding delay, which however is impossible when some variables in P are continuous. In the P for secure VoIP service S,

- F is discrete under the assumption that services only support a limited set of security functionalities.
- A is discrete under the assumption that services only support a limited set of algorithms.
- l is discrete. Although l can be any integer in theory, security algorithms usually have requirements on the lengths of keys. For example, AES can only supports keys with the length $\{128, 192, 256\}$. Hence, the value of l is also discrete.
- v is discrete. The vulnerability is associated with the algorithm A and the attacker's power. Because S cannot control the attacker's power, it is usually estimated with the worst case. Hence, the value of v is determined by A and l, both of which are discrete.
- p can be either discrete or continuously depending on S's implementation. For continuous p, it can be any value between 0% and 100%. For discrete p, S can set up several pre-defined p for users.
- t can be any positive integer and is continuous.

To construct different P for regression, S enumerates all possible combinations of discrete variables in P. For continuously variables in P, S selects a set of values for each variable with uniform intervals, and then enumerates all possible combinations. If there are too many combinations, S can randomly sample a subset of combinations.

We have implemented a set of security services supporting AES encryption algorithm for confidentiality. The security configuration vector for these security services is defined as $SCV_{enc} = \{Confidentiality, AES, k, v, p\}$, where k is the key length selected from $\{128, 192, 256\}$ bits, v is the vulnerability function for AES algorithm, and p is the protection percentage selected from $\{0, 25, 50, 75, 100\}$. That is, there are total 15 possible parameter configurations for SCV_{enc}.

Generally, a longer key length requires more operations in encryptions and a higher percentage requires the security service to encrypt more packets. Hence, the security service with a longer key length or a larger protection percentage in SCV_{enc} is expected to generate a longer delay on packets. To study the relation among SCV_{enc}, t and the delay D, we run the experiment by gradually increasing the throughput t from 10 to $5,000$ packets per second. Each packet has the same size as 1000 bits. We collect the average packet delay D under different throughput.

For each SCV_{enc}, we run the experiment for 10 seconds to find the average delay of packets as training data, which includes twelve curves for the relations between delay and throughput as shown in Figure 6.2. Figure 6.3 shows the delay metric D for $SCV_{enc} = \{Confidence, AES, 128, v, 100\}$. To make D clearer for small t, Figure 6.4 shows the same data of Figure 6.3, but with log x-axes. From Figures 6.3 and 6.4, it is clear that D increases with t as a sigmoid function like (6.3), which starts from a small value and then accelerates and approaches to an upper limit.

The parameter regression results for all SCV_{enc} are shown in Table 6.1, in which the last two columns are the coefficient of determination and the adjusted coefficient of determination [118], which are very close to 1. It indicates that our regression results match the training data very well.

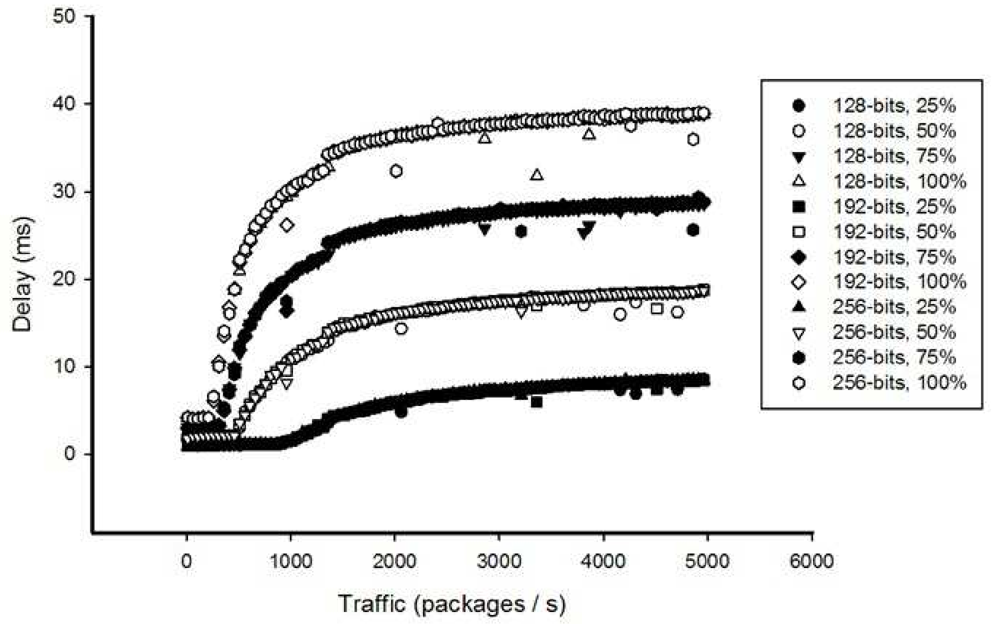

Figure 6.2: The relations between throughput and delay with different key lengths. Note that curves with the same protection percentage but different key lengths almost coincide together because the protection percentage dominates the security strength.

Specifically, the D for $SCV_{enc} = \{Confidence, AES, 128, v, 100\}$ shown in (6.3) becomes

$$D(SCV,t) = \begin{cases} 0, & t = 0 \\ 4.5187, & 0 < t \leq 283.6797 \\ 4.5187 + 34.4231(1 - e^{-0.0195((t-283.6797))^{0.6521}}), & t > 283.6797 \end{cases} \quad (6.4)$$

These experimental results show the following properties:

- The key length l has no significant effect on the relation between the delay and throughput. That is, the parameters $a_1, a_2, a_3, T_1(SCV)$ and $D_1(SCV)$ of the delay metric (6.3) are not significantly affected by l. In Figure 6.2, the twelve curves are grouped into four groups according to p. For each group, there are three data sets representing different l with same p, which coincide with each other. In Table 6.1, when p is the same, all parameters' values are very close for various key lengths.

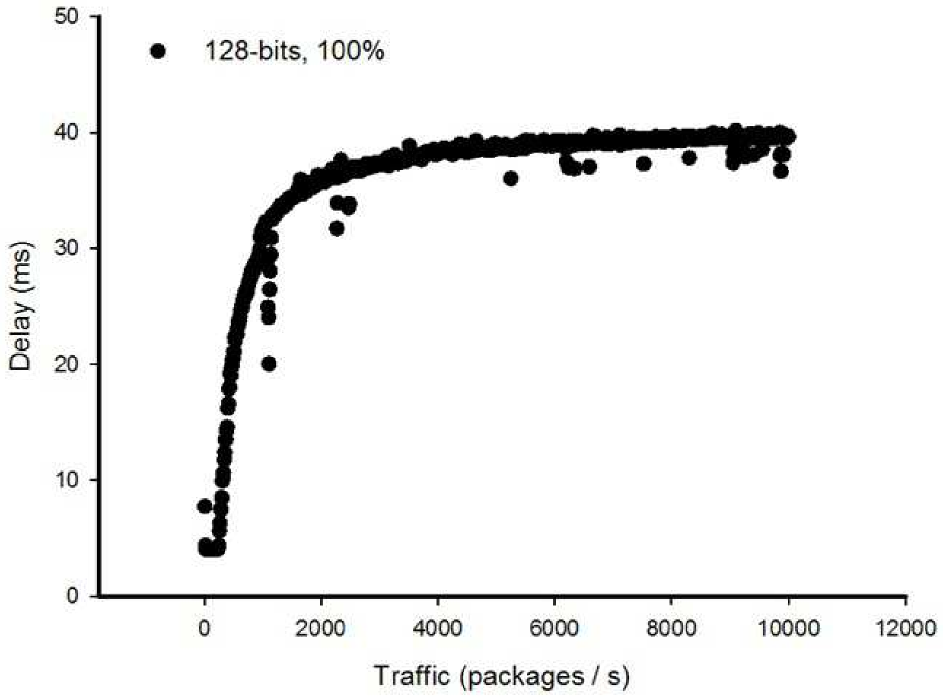

Figure 6.3: The delay metric for $SCV_{enc} = \{Confidentiality, AES, 128, v, 100\}$ with linear X-axes.

Table 6.1: Parameter Estimation for all SCV_{enc}

P	k_l	a_1	a_2	a_3	T_1	D_1	Rsqr	Adj Rsqr
25%	128	7.4962	0.0030	0.8364	995.5746	1.1166	0.9929	0.9926
	192	7.8844	0.0060	0.7293	1031.053	1.1237	0.9936	0.9933
	256	7.8718	0.0040	0.7872	992.1826	1.0953	0.9976	0.9975
50%	128	16.309	0.0040	0.8381	479.0611	2.0487	0.994	0.9938
	192	16.365	0.0036	0.8565	473.8439	2.0651	0.9974	0.9973
	256	16.4489	0.0031	0.8723	468.4268	2.0512	0.9962	0.996
75%	128	25.4971	0.0097	0.7377	345.332	3.0559	0.9962	0.9961
	192	25.7254	0.0093	0.7394	344.0336	3.0773	0.9964	0.9963
	256	25.5483	0.0083	0.76	341.5594	3.0899	0.9945	0.9943
100%	128	34.5291	0.0127	0.7141	252.7062	4.0311	0.9933	0.9931
	192	34.8059	0.0131	0.7072	253.4914	4.0651	0.9971	0.997
	256	34.4231	0.0195	0.6521	283.6797	4.5187	0.9952	0.995

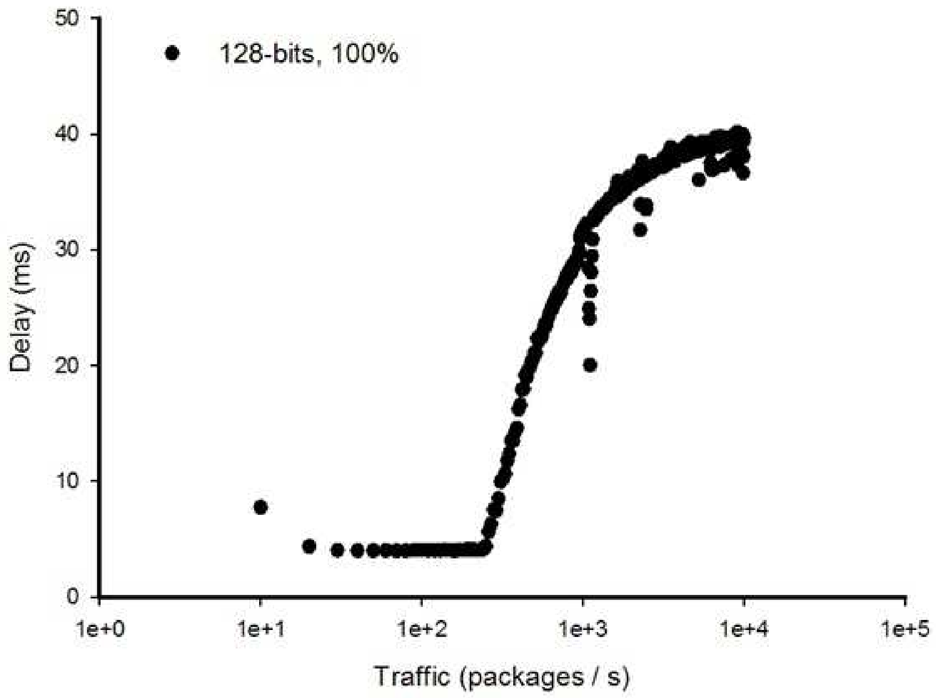

Figure 6.4: The delay metric for $SCV_{enc} = \{Confidentiality, AES, 128, v, 100\}$ with log X-axes.

- Because p determines the actual number of packets encrypted, with the same l, p dominates the relation between D and t. A larger p generates a longer D than a smaller p and makes the D to increase faster than smaller p. In Figure 6.2, the curve group with larger p is above that with smaller p, and is stepper. In Table 6.1, when p increases, the parameters a_1 and a_2 also increases, which speed up the increasing of the D.

- Because the time that the delay starting to increase exponentially is determined by the incoming throughput that needs to be encrypted, but not the overall throughput of the service, a larger p generates a larger incoming throughput that needs to be encrypted, when the throughput of the service is the same. In Figure 6.2, the curve group with a larger p starts to increase earlier than the curve group with a smaller p. In Table 6.1, the parameter T_1 for larger p is smaller.

Table 6.2: Selected experimental data for $SCV_{enc} = \{Confidence, AES, 128, v, 100\}$

Throughput	No. Packages	Observed Avg. Delay D (ms)	Predicated Avg. Delay D' (ms)	Deviation Error (%) $\lvert D - D' \rvert / D$
10	100	7.739	4.0311	47.91%
30	300	4.0237	4.0311	0.18%
50	500	3.995	4.0311	0.90%
80	800	3.9963	4.0311	0.87%
100	1000	4.0363	4.0311	0.13%
300	3000	8.4901	10.2899	21.20%
500	5000	21.072	20.5645	2.41%
800	8000	27.5223	27.6184	0.35%
1000	10000	31.6228	30.3422	4.05%
3000	30000	37.1973	37.6511	1.22%
5000	50000	39.0438	38.4003	1.65%
8000	80000	39.5463	38.5434	2.54%
10000	100000	39.6344	38.5559	2.72%

- Even when p is not 100%, all packets including the unprotected packets still consume some resources. In Figure 6.2, when the throughput is small and the D has not started to increase exponentially, the curve group with larger p is also above the curve group with smaller p. In Table 6.1, the parameter $D_1(SCV)$ for larger p is larger.

To verify the regressed metric, we ran the experiments again and collected the data for $SCV_{enc} = \{Confidence, AES, 128, v, 100\}$ with t from 10 to $10,000$ as testing data, and compare the observed average delay with the predicted average delay computed from . Some sampling data is listed in Table 6.2, where the experiment was run 10 seconds for each traffic level. The deviation error in Table 6.2 shows that the above regressed delay metric matches the real delay data very well. Because the security service needs to create the encryption key at the beginning of encryption, when we started the experiment from $t = 10$, the observed average delay includes both the delay caused by key generation and encryption which leads to a large deviation error. The large deviation error for $t = 300$ is due to the noises from system's unstable performance.

6.3 Tradeoff among Various QoS Aspects

One benefit of developing quantitative metrics for observable QoS aspects is that services can control their qualities on all QoS aspects simultaneously through an unique parameter set P. By using different values for P, services can provide various QoS for their users, similar to application level differentiated services [79, 80].

For the secure VoIP service S, the delay metric developed above enables S to control its qualities on security, throughput, and delay simultaneously through the same parameter set P (6.2). S can deploy several service instances with various tradeoff strategies and objective functions to provide differential services for its users.

Tradeoff Objective Function

The tradeoff objective function is the weighted sum of services' qualities on all QoS aspects. Because all QoS are measured through the unique parameter set P, the tradeoff objective function can be defined on P as

$$G(P) = w_1 q_1 + \cdots w_{n_1} q_{n_1} - w'_1 q'_1 - \cdots w'_{n_2} q'_{n_2} \tag{6.5}$$

where $w_1, \cdots, w_{n_1}$ are the weights for services' qualities on the utility QoS aspects, and $w'_1, \cdots, w'_{n_2}$ are the weights for services' qualities on the cost QoS aspects. A larger w_i means that services' qualities on c_i is more important. All weights are normalized to the range [0, 1] and satisfy $w_1 + \cdots + w_{n_1} + w'_1 + \cdots + w'_{n_2} = 1$. The tradeoff objective is to find the best values for P to maximize $G(P)$.

For the secure VoIP service S, the tradeoff objective function is defined as

$$G(SCV, t) = w_1 \cdot (p(1 - cv(A, l))) + w_2 \cdot t - w_3 \cdot D(SCV, t) \tag{6.6}$$

where $w_1 + w_2 + w_3 = 1$.

Minimum Requirement Validation

The QoS tradeoff among various QoS aspects enables services to provide differential services for their users. Although these differential services have different emphasis on QoS aspects, all of them should satisfy certain minimum requirements. Otherwise, the service is useless. For the secure VoIP service S and the objective functions (6.6), we denote its minimum requirements as $D(SCV,t) < D_0$, $t > T_0$, and $S(SCV,c) > S_0$.

Due to the minimum security requirement, we have

$$S(SCV,c) > S_0 \Rightarrow p > \frac{S_0}{1-cv(A,l)} \tag{6.7}$$

To satisfy the minimum throughput and delay requirements, we have

$$\begin{aligned} &D(SCV,t) < D_0 \\ \Rightarrow &\begin{cases} D_1(SCV) < D_0, & if\ T_0 \le T_1 \\ p < (\sqrt[a_3]{(-ln(1-\frac{D_0-D_1(SCV)}{a_1})/a_2+T_1})/T_0, & if\ T_0 > T_1 \end{cases} \end{aligned} \tag{6.8}$$

Note that (6.7) gives a lower bound for p, and (6.8) gives an upper bound for p. Hence, to check whether the minimum requirements can be satisfied, we only need to check whether the p between (6.7) and (6.8) exits, i.e., whether there exists an algorithm A and key length l satisfying

$$\frac{S_0}{1-cv(A,l)} \le \frac{\sqrt[a_3]{-ln(1-\frac{D_0-D_1(SCV)}{a_1})/a_2+T_1}}{T_0}, \quad if\ T_0 > T_1 \tag{6.9}$$

Delay Biased Objective Function

Given the throughput t, the delay biased objective function is a tradeoff objective function to minimize the delay without violating the minimum throughput requirement $t > T_0$ and the minimum security requirement $S(SCV,t) > S_0$. For $G(SCV,t)$ given in (6.6), the delay biased objective function sets $w_1 = w_2 = 0$ and $w_3 = 1$. In this case, to maximize $G(SCV,t)$ is equivalent to minimize $D(SCV,t)$. When the algorithm and the key length are fixed, we can compute the lower bound for p from the minimum security requirements according to (6.7).

Security Biased Tradeoff Function

The security biased tradeoff function is a tradeoff objective function to maximize the security without violating the minimum delay and throughput requirements. For $G(SCV,t)$ given in (6.6), the security biased tradeoff function sets $w_1 = 1$ and $w_2 = w_3 = 0$. In this case, to maximize $G(SCV,t)$ is equivalent to maximize S. When the algorithm and the key length are fixed, we can compute the upper bound for p from the minimum delay requirements according to (6.8).

Tradeoff Objective Function with Largest Satisfaction

Users' QoS requirements may be more complex than minimizing delay or maximizing security. For example, when the service S already provides good security, users may want to decrease delay rather than to improve security. On the other hand, when S's delay is very small, users may would like a better security more than smaller delay. In the Chapter 7, we will develop the concept of satisfaction score which measures how well a service satisfies users' QoS requirements. When the service's QoS on one QoS aspect has already satisfied the users' QoS requirements very well, further improvement on this QoS aspect will have little contribution in the improvement of the service's satisfaction score. Hence, to maximize service's satisfaction score, S needs to balance its QoS on various QoS aspects according to the users' QoS requirements. This property of the satisfaction score makes it a good candidate of the tradeoff objective function.

6.4 Optimizing Services' QoS with Adaptive Tradeoff

In this evaluation, we will evaluate our adaptive tradeoff and optimization approach with the VoIP application where the security service is required to provide protection for the voice or video data stream. First, we setup QoS requirements for VoIP applications as follows and summarized in Table 6.3:

- *Throughput*. The throughput requirement depends on the sampling rate, the voice compression algorithm, and the number of voice channels (i.e., the number of clients) that the security service would like to support. the International Telegraph Union (ITU) has proposed a set of standards for the VoIP applications, such as the G.711 for general telephone with bit rate 64 kbit/s and the G.729 for VoIP over low speed connection with bit rate 8 kbits/s (from `http://www.lammertbies.nl/comm/info/VoIP-overview.html`). If the security services would like to support 100 voice channels simultaneously, we set the least acceptable and the best QoS requirements for throughput as 800 kbps and 6400 kbps. The throughput requirement for VoIP applications is important but not critical, because the throughput requirement can be reduced with silence suppression. We set its importance as 0.8 with confidence 0.9.

- *Delay*. While the ITU G.114 recommendation states that the delays above 400 ms are unacceptable for VoIP applications, Qwest guarantees 50 ms delay in its service level agreement. Hence, we set the least acceptable and the best QoS requirements for delay as 400 ms and 50 ms. The delay requirement for VoIP applications is critical, and we set as 1 with confidence 1.

- *Security*. The security requirement depends on the sensitivity of the voice data. In this evaluation, we assume that the VoIP application is used for communicating sensitive but not classified information, and set the least acceptable and the best QoS requirements for security as 50% and 80%. The importance for security requirement is set as 0.6 with confidence 0.6.

Table 6.3: The QoS requirements for VoIP Applications

QoS aspect	**l**	**u**	**w**	**r**
Throughput	800 kpbs	6400 kpbs	0.8	0.9
Delay	50 ms	400 ms	1	1.0
Security	50%	80%	0.6	0.6

With the parameter estimated in Section 6.2, the adaptive tradeoff approach optimizes the security services' QoS, and finds the best values for $P = \{SCV, t\} = \{F, A, l, v, p, t\}$ as $\{Confidentiality, AES, 128, v, 100, 6000\}$. With this set of values, the security service's satisfaction score is 0.7336 with the QoS requirements listed in Table 6.3. The security service cannot provide perfect QoS because it cannot satisfy the expected best throughput.

If we change the delay requirement from $(50, 400, 1, 1.0)$ to $(5, 40, 1, 1.0)$, the best values for P changes to $\{Confidentiality, AES, 128, v, 75, 6000\}$ and the satisfaction score is reduced to 0.5185. This result shows that when the service user increases his/her delay requirement, the security service needs to sacrifice its QoS on security to maximize its satisfaction score.

Chapter 7

QOS-BASED SERVICE RANKING

This chapter presents a QoS-based service ranking approach for **SR** to match **SU**' QoS requirements with services' QoS and help **SU** to select the best service satisfying his/her QoS requirements.

When **SU** submits service requests to **SD**, **SU** selects a set of QoS aspects, such as throughput, delay, reliability, security, and price. For each QoS aspect, **SU** specifies his/her QoS requirement on the QoS aspect as in Section 5.2, including the lower bound and upper bound of the expected quality, the importance of this requirement, and the confidence in this requirement specification. With the QoS information from **QM**, **SR** first optimizes services' QoS by making tradeoff among various QoS aspects as discussed in Section 6.2, and then ranks services. While existing approaches rank only services' QoS from best to worst [91, 92] without the consideration of users' requirements and preferences, the service ranking approach presented in this chapter ranks services based on the degree of satisfaction of users' QoS requirements. Hence, when two services both perfectly satisfy users' QoS requirements, these two services have the same chance to be selected by the users even one service may have better QoS than the other. The ranking based on satisfactions rather than QoS helps to increase the availability of the services with the best QoS and improve the utilization of other services.

7.1 Satisfactory Score

The ranking of services is based on how well services' QoS satisfy users' QoS requirements. In this section, we will define the satisfaction score and discuss how to compute the satisfaction scores for services.

Let $C = \{c_1, c_2, \cdots, c_n\}$ be the QoS aspects selected by **SU**, and $\{S_1, S_2, \cdots, S_m\}$

be the list of services returned from **SD** that satisfy **SU**'s functionality requirements. The QoS of all services S_i is represented as the following matrix

$$QoS\begin{pmatrix} S_1 \\ \vdots \\ S_m \end{pmatrix} = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ \vdots & \vdots & \cdots & \vdots \\ q_{m1} & q_{m2} & \cdots & q_{mn} \end{pmatrix} \tag{7.1}$$

where each row $\{q_{i1}, q_{i2}, \cdots, q_{in}\}$ represents the service the QoS of $S_i, 1 \leq i \leq m$, QoS on all QoS aspects after QoS optimization, and each column $\{q_{1j}, q_{2j}, \cdots, q_{mj}\}$ represents all services' QoS on the aspect $c_j, 1 \leq j \leq n$. **SU** specifies his/her QoS requirements on each QoS aspect c_j as req_j as follows:

$$Req(c_1, \cdots, c_n) = (req_1, \cdots, req_n) \tag{7.2}$$

For each service S_i, a set of satisfactory scores sc_{ij} is computed for all QoS aspects c_j, which are numbers within $[0,1]$ measuring how well S_i's qualities on aspects C satisfies $Req(c_1, c_2, \cdots, c_n)$. When $sc_{ij} = 0$, S_i's QoS q_{ij} does not satisfy req_j at all; when $sc_{ij} = 1$, S_i's QoS q_{ij} satisfies req_j perfectly; when $0 < sc < 1$, S_i's QoS q_{ij} partially satisfies req_j. A larger sc_{ij} represents better satisfaction.

To compute the overall satisfaction scores of $\{S_1, S_2, \cdots, S_m\}$ on **SU**'s QoS requirements (7.2), all elements in the matrix (7.1) will first be normalized with normalization functions $Norm_1$ and $Norm_2$ defined in Section 7.2 to the range $[0,1]$, and then compared to (7.2) with the satisfaction score function $SSF^{w'}$ defined in Section 7.3 to compute the satisfaction scores for each S_i and each c_j as

$$SC\begin{pmatrix} S_1 \\ \vdots \\ S_m \end{pmatrix} = \begin{pmatrix} sc_{11} & sc_{12} & \cdots & sc_{1n} \\ \vdots & \vdots & \cdots & \vdots \\ sc_{m1} & sc_{m2} & \cdots & sc_{mn} \end{pmatrix} \tag{7.3}$$

Finally, each service S_i's overall satisfactory score sc_i is computed with the combination function CF defined in Section 7.4 to combine $\{sc_{i1}, \cdots, sc_{in}\}$ together, which is used by **SR** to rank services.

7.2 QoS Normalization

Due to the variety of QoS aspects' scales and value ranges, services' qualities on d-ifferent QoS aspects have to be normalized before aggregation. In this section, two normalization functions $Norm_1$ and $Norm_2$ are presented for services' qualities on u-tility QoS aspects and cost QoS aspects respectively, which normalize qualities on all aspects to the unique range $[0,1]$ to enable uniform satisfaction score function independent of QoS aspects' units. For a service S_i and a QoS aspect c_j, a larger q_{ij} means better quality if c_j is a utility QoS aspect, or worse quality if c_j is a cost QoS aspect. Hence, the QoS normalization function $Norm_1$ for utility QoS aspects should increase with q_{ij}, and the $Norm_2$ for cost QoS aspects should decrease with q_{ij}. $Norm_1$ and $Norm_2$ are defined as

$$Norm_1(q_{ij}) = \begin{cases} 0, & \text{if } q_{ij} < l_j r_j \\ \frac{q_{ij} - l_j r_j}{u_j/r_j - l_j r_j}, & \text{if } l_j r_j \leq q_{ij} < u_j/r_j \\ 1, & \text{if } q_{ij} > u_j/r_j \end{cases} \tag{7.4}$$

$$Norm_2(q_{ij}) = \begin{cases} 0, & \text{if } q_{ij} > u_j/r_j \\ \frac{u_j/r_j - q_{ij}}{u_j/r_j - l_j r_j}, & \text{if } l_j r_j \leq q_{ij} < u_j/r_j \\ 1, & \text{if } q_{ij} < l_j r_j \end{cases} \tag{7.5}$$

where $req_j = \{l_j, u_j, w_j, r_j\}$ is the requirement specified by **SU** on the QoS aspect c_j.

SU will not accept any service when $q_{ij} < l_j r_j$ if c_j is an utility QoS aspect, or $q_{ij} > u_j/r_j$ if c_j is a cost QoS aspect. Furthermore, the service has no additional advantage to satisfy users' QoS requirements if $q_{ij} > u_j/r_j$ for utility QoS aspect, or $q_{ij} < l_j r_j$ for cost QoS aspect. For example, text-based online chatting services only need small bandwidth, and hence one such service with 10 Mbps bandwidth and another service with 100 Mbps bandwidth are equivalent for users although the second service's bandwidth is much larger than the first one's bandwidth. While existing normalization functions like [91, 92] generate a larger number for the service with 100 Mbps bandwidth, the *Norm1* and *Norm2* will normalize both services' qualities to 1 which reflect the user's requirements more accurately.

The $Norm_1$ and $Norm_2$ also adjust the range $[l_j, u_j]$ to the range $[l_j r_j, u_j/r_j]$ according to the user's confidence in u_j and l_j. Noted that the range $[l_j r_j, u_j/r_j]$ is equal to the range $[l_j, u_j]$ when $r_j = 1$, and larger than $[l_j, u_j]$ when $r_j < 1$.

7.3 Satisfactory Score Function

The satisfaction score function $SSF^{w'}$ is used to computes the satisfactory scores sc_{ij} in (7.3) for each service S_i and each QoS aspect c_j, which are numbers within $[0,1]$.

Let n_{ij} be the normalized value of q_{ij}. $SSF^{w'}$ should satisfy the following requirements:

1. If the user does not care about services' qualities on c_j, the user will not specify any QoS requirement for c_j. Then, services' qualities on c_j will not be considered in the ranking of services.

2. If the user only have a threshold requirement about services' qualities on c_j (e.g., the use does not care about the price as long as it is within the budget), the user will use the same lower bound and upper bound in the QoS specification. That is, all services' qualities on c_j are normalized to 1 if their QoS satisfies the threshold or to 0 if their QoS does not satisfy the threshold. Hence, the $SSF^{w'}$ does not need to compute the satisfaction for the normalized QoS within $(0,1)$.

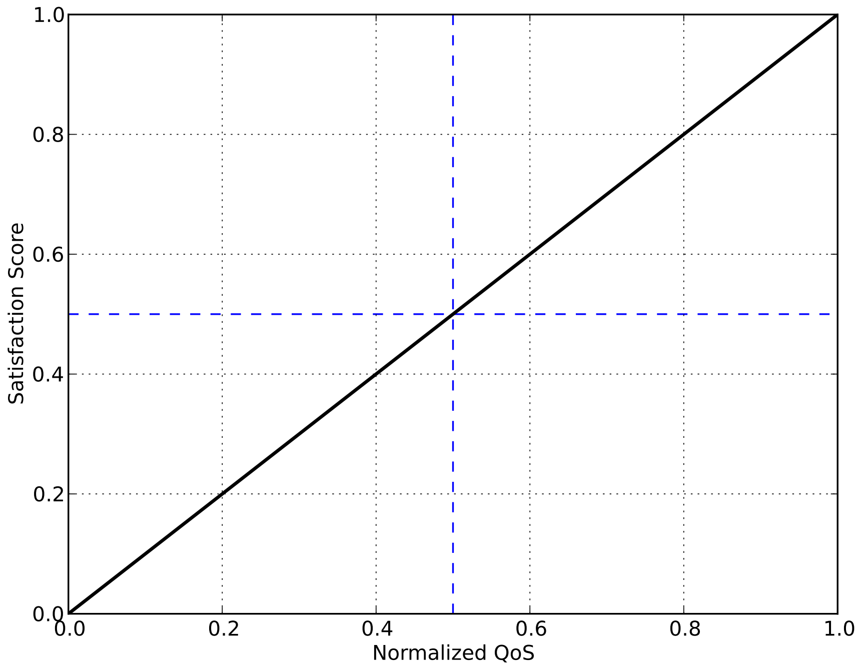

Figure 7.1: The relation between normalized QoS and satisfaction with a linear satisfaction score function.

3. If the user defines a range of acceptable QoS, the $SSF^{w'}$ should output 0 for all services' QoS below the range, and 1 for all services' QoS above the range. That is, $SSF^{w'}(0) = 0$ and $SSF^{w'}(1) = 1$.

4. If sc_{ij} is between 0 and 1, a larger sc_{ij} means that **SU** will be better satisfied with S_i's QoS, and S_i will be ranked higher in service selection.

A simple satisfaction function can just use the value of n_{ij} as sc_{ij} as shown in Figure 7.1, which is linear and satisfies all of the above three requirements. However, this simple function assumes that the user's satisfaction of a service is proportionate to the service's QoS which is not always true as shown in [94] and prospect theory in economics [119, 120].

Because the expected lower bound and upper bound represent the range of minimum acceptable QoS and best QoS. All QoS within this range is acceptable, but users will be more satisfied with better QoS. Using the normalized QoS as satisfaction score assumes that the satisfaction of services is only determined by the normalized QoS, and services' satisfaction will be doubled if their normalized QoS is doubled. However, the prospect theory suggests that the satisfaction should be based on the gains and losses relative to some reference point instead of absolutely determined services' normalized QoS.

If we define the reference point as $n_{ij} = 0.5$ for each QoS aspect c_j, users obtain gains if they choose a service which normalized QoS is larger than 0.5, and suffer losses if they choose a service which normalized QoS is smaller than 0.5. According to the prospect theory, the satisfaction function should be concave for gains but convex for losses. For example, the delay of VoIP services is usually expected to be within the range (50 ms, 400 ms) as discussed in Section 6.4. Hence, the reference point of delay in VoIP is 225 ms ($Norm_2(225) = 0.5$). It is easier for users to discriminate a longer delay from 150 ms to 200 ms, than to discriminate a longer delay from 50 ms to 100 ms because the pauses in speech are generally longer than 100 ms. Furthermore, It is also easier for users to discriminate a longer delay from 250 ms to 300ms, than to discriminate a longer delay from 350 ms to 400 ms because 350 ms has already longer than most pauses in speech.

Satisfaction is subjective and may be different for different users even for the exact same service. First, a user who usually talks fast may specify the range of expected delay in VoIP as (50 ms, 300 ms), while a user who usually talks slow may specify the range as (150 ms, 400 ms). Because the normalization functions presented in Section 7.2 normalize services' QoS with users' QoS requirements. The actual delay of a normalized delay 0.5 will be different for different users. Second, even with the same expected range, the relation between the satisfaction and the normalized QoS also depends on users' subjective feelings. For example, if a user is more sensitive

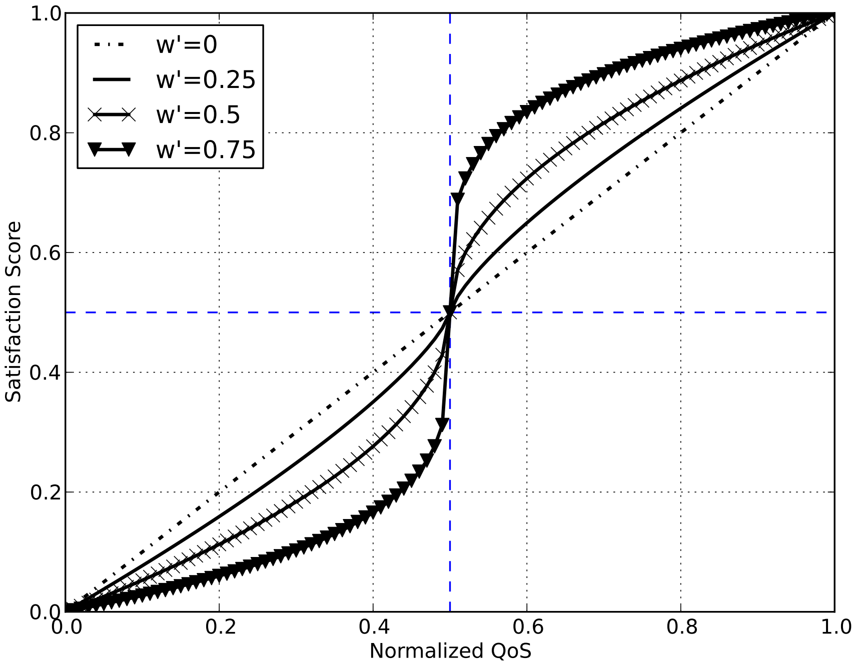

Figure 7.2: The effect of parameter w' on the function $SSF^{w'}$.

about the speed of speaking, the VoIP service's satisfaction will change faster with the same change in delay. That is, the satisfaction function becomes more steep around the reference point.

Based on the prospect theory and above discussion, the satisfaction score function $SSF^{w'}$ is defined as

$$sc_{ij} = SSF^{w'}(n_{ij}) = \begin{cases} 0.5(2n_{ij}-1)^{1-w'}+0.5, & \text{if } n_{ij} > 0.5 \\ -0.5(-2n_{ij}+1)^{1-w'}+0.5, & \text{if } n_{ij} \leq 0.5 \end{cases} \tag{7.6}$$

where w' is the adjusted importance weight of the QoS aspect c_j as shown in Section 5.2. For any value of the parameter w', $SSF^{w'}(0) = 0$, $SSF^{w'}(1) = 1$,and $SSF^{w'}(n_{ij})$ increases with the increasing of $n_{ij} \in [0,1]$. Because larger n_{ij} indicates better QoS for both utility QoS and cost QoS, $SSF^{w'}(n_{ij})$ always outputs a larger satisfactory score for better QoS.

The satisfaction function (7.6) has the following properties:

- When $w' \to 0$, $SSF^{w'}$ is becoming a linear function with n_{ij} as SSF^0 shown in Figure 7.2.
- $SSF^{w'}$ uses 0.5 as the reference point. In Figure 7.2, $SSF^{w'}(0.5) = 0.5$ for any w'.
- With the increasing of n_{ij}, the increasing speed of $SSF^{w'}$ accelerates at the beginning until $n_{ij} = 0.5$ where $SSF^{w'}$ reaches the maximum increasing speed. After $n_{ij} = 0.5$, the increasing speed of $SSF^{w'}$ starts to decrease. Finally, $SSF^{w'}$ reaches 1 at $n_{ij} = 1$. This trend is in accordance with the prospect theory [119, 120].

7.4 Combination Function

When the satisfaction scores for QoS aspects in (7.3) have been computed by the function $SSF^{w'}$, the combination function CF presented in this section computes overall satisfaction scores for services by combining their satisfaction scores on various QoS aspects together with weights.

The combination function CF is defined as

$$CF(S_i) = \frac{\sum_{1 \le j \le n} sc_{ij} w'_j}{\sum_{q \le j \le n} w'_j} \tag{7.7}$$

where sc_{ij} is the S_i's satisfaction score on c_j computed with (7.6), and $w'_i \to w_i r_i + 0.5(1 - r_i)$ is the adjusted importance weight w_j of c_j discussed in Section 5.2.

7.5 Case Study

This section gives an example to illustrate the QoS-based service ranking mechanism. In this example, we assume that a developer of a voice communication system wants to improve the system by providing secure peer-to-peer voice communication. Because the voice communication system is developed as a SBS, the developer can select an existing encryption service to use in the system. The encryption service should accept a data stream and output an encrypted data stream. Furthermore, the developer has QoS requirements on the following four QoS aspects:

- *Throughput*. The encryption service can support at least 1000 packages per second. Each package has 1000 bits. That is, the throughput of the service should be at least 1Mbps.
- *Delay*. The encryption delay of each package should be less than 10 microseconds.
- *Security*. The encryption service should provide approximate security protection for voice communication of sensitive but not classified information. With the security metric developed in Chapter 6, the encryption service should prevent attackers from cracking messages with probability at least 60%. if the encryption service can prevent attackers from cracking messages with probability 80%, it provides sufficient security protection for the voice communication, and no need to select an encryption service stronger than 80%.
- *Price*. The price of the service should be less than 1 dollar per day.

First, the developer specifies the QoS requirements as shown in Table 7.1, and sends it to **SR**. Some parameters are blank because the developer does not know how to specify them. Throughput and security are utility QoS aspects. Delay and price are cost QoS aspects.

Suppose **SD** finds three encryption services S_1, S_2 and S_3. Their QoS on the required four QoS aspects are shown in Table 7.2. To rank these three services, **SR** computes satisfaction scores for S_1, S_2 and S_3 as follows

1. **SR** sets values for parameters that are blank in Table 7.1. Because there is no natural upper bound for the throughput, the parameter u for throughput will be set to the largest throughput of all encryption services. The parameter w for price will be set to the default value 0.5. Furthermore, all linguistic terms will be mapped to values with Table 5.2, and all w will be adjusted with r as discussed in Section 5.2. The adjusted QoS requirements are shown in Table 7.3.

2. **SR** normalizes throughput and security with $Norm_1$ (7.4), and delay and price with $Norm_2$ (7.5). The normalized QoSs are shown in Table 7.4.

3. **SR** computes the satisfaction scores for each QoS aspect with $SSF^{w'}$ (7.6) and combines them together with (7.7). All satisfaction scores are shown in Table 7.5.

Hence, **SR** ranks services as $S_2 > S_3 > S_1$, and returns the ranked services to **SU**.

Table 7.1: User-Specified QoS Requirements on Usable Encryption Services

QoS aspect	**l**	**u**	**w**	**r**
Throughput	1Mbps		0.7	0.9
Delay	0	10 ms	Important	0.8
Security	60%	80%	Very important	1
Price	0	1 dollar/day		1

Table 7.2: The QoS of Encryption Services Returned By **SD**

	Throughput	**Delay**	**Security**	**Price**
S_1	10 Mbps	10 ms	0.7	1 dollar/day
S_2	1.5 Mbps	5 ms	0.8	0.5 dollar/day
S_3	5 Mbps	1 ms	1	2 dollar/day

Table 7.3: Adjusted User-Specified QoS requirements on Usable Encryption Services

QoS aspect	**l**	**u**	**w**	**r**
Throughput	1Mbps	10 Mbps	0.68	0.9
Delay	0	10 ms	0.74	0.8
Security	60%	80%	1	1
Price	0	1 dollar/day	0.5	1

Table 7.4: Normalized QoS of Encryption Services Returned By **SD**

	Norm$_1$ (Throughput)	**Norm$_2$ (Delay)**	**Norm$_1$ (Security)**	**Norm$_2$ (Price)**
S_1	0.89	0.2	0.5	0
S_2	0.06	0.6	1	0.5
S_3	0.4	0.92	1	0

Table 7.5: Satisfaction Scores of Encryption Services Returned By **SD**

	S_1	S_2	S_3
Satisfaction Score on Throughput	0.96	0.02	0.20
Satisfaction Score on Delay	0.06	0.83	0.98
Satisfaction Score on Security	0	1	1
Satisfaction Score on Price	0	0.5	0
Overall Satisfaction Score	0.239	0.643	0.637

Chapter 8

CONCLUSION AND FUTURE RESEARCH

The goal of this dissertation is to facilitate the development of SBS systems by helping system developers find appropriate services and evaluate whether these services can satisfy the SBS systems' QoS requirements. On this dissertation, an approach to discovering services in untrusted service directory without revealing the privacy of service providers and users is proposed. The services returned by this approach are ranked based on how well their QoS satisfy the service users' QoS requirements after the QoS optimization through adaptive tradeoff among various QoS aspects. In this chapter, we summarize our research and discuss some future directions.

8.1 Research Summary

Specifically, our approach includes the following three major parts:

(1) Privacy preserving and controlled access service discovery. All current efficient service discovery approaches require centralized service directories to organize all available services and help service users to match their service requests with services. However, if the centralized service directories are not trusted, they are too powerful and may invade the privacy of service providers and users without appropriate protection. To restrict these service directories' capability but still enable them to organize and match services, we have developed a privacy preserving and controlled access service discovery and matching approach. With this approach, all service registrations from service providers and service requests from service users are protected from the service directory through encryption. The service directory cannot learn any information from encrypted service registrations and requests besides the matching results.

(2) Quantitative QoS metrics and tradeoff among various QoS aspects. The developers of SBS systems have requirements on both services' functionalities and QoS. To support the matching between services' QoS and QoS requirements, we have conducted research on developing quantitative metrics for QoS aspects, and the adaptive tradeoff among various QoS asepcts:

- *Quantitative QoS metrics.* We have developed a security metric to measure services' security strength as the probability to withstand attackers. Besides the key length which is the only factor used by existing quantitative security metrics, our security metric also incorporates the vulnerability of algorithm design and the attackers' power. To get better flexibility in QoS tradeoff, we also allow partial encryption by including the protection percentage as one parameter of the security metric. For other QoS aspects that are usually measured through observation such as delay and throughput, we have proposed an approach to develop quantitative metrics for these QoS aspects also through parameters.
- *Adaptive tradeoff among various QoS aspects.* Because the metrics on all QoS aspects are quantitative and are based on the same set of parameters that can be controlled by the services, services can optimize their QoS according to the users' QoS requirements by making tradeoff among various QoS aspects. We have presented an adaptive tradeoff approach and given some tradeoff strategies.

(3) QoS-based service ranking. When there are multiple services providing equivalent functionalities, QoS-based service ranking is to rank services' QoS and help users to make selection among these services. Compared with existing QoS-based service ranking algorithms which rank services based on their QoS, we have defined a satisfaction score function to compute how well a service's QoS satisfy the user's QoS requirements, and presented a QoS-based service ranking algorithm to rank services based on their satisfaction scores. The satisfaction scores allow users to select the best suitable services and can avoid to select overqualified services.

8.2 Future Research

In order to further improve our approach, the following research should be conducted:

- *Developing more efficient privacy preserving service discovery and matching.*

  Our current approach heavily relies on the additively and multiplicatively homomorphic properties of the BGN encryption [98]. Because BGN encryption requires to compute the pairing on a composite order group which is time consuming, our approach cannot support large keyword set or access key set if the service directory need to return the matching results within reasonable responsible time. The progress in the research of efficient homomorphic encryption algorithms may lead to more efficient design of privacy preserving service discovery and matching protocols.

- *Developing more flexible access control approach to services.*

  Currently, our approach only supports simple key-based access control to services. A user can access the service if and only if he/she possesses one of the access key associated with the service, and there is no hierarchy structure within all access keys. Recently, attribute-based encryption [121] has been proposed to enable any user to decrypt the encryption as long as the user has some required attributes [122]. All attributes can be organized as a hierarchy tree. The problem of incorporating the attribute-based encryption algorithms is that our approach can only hide the existence of services based on one access key and cannot support the decryption policies used in the attribute-based encryption algorithms.

- *Developing more quantitative metrics to cover more QoS aspects.*

Users may have requirements on a lot of QoS aspects, and some of them are very difficulty to be quantitatively measured. In this dissertation, we have presented the quantitative metric for security, which is one of the most important QoS aspects that cannot be easily measured quantitatively. However, other QoS aspects, such as services' reputations and business values, have not been thoroughly studied.

- *Supporting more flexible and user-friendly QoS requirement specification.*

  Although our approach has greatly improved the flexibility of the QoS requirement specification by allowing users to define their expected best and least QoS, and made it much easier for users to specify their QoS requirements by allowing users to specify their preferences on QoS aspects through linguistic terms, specify their confidence on their specification, and leave the requirement parameters blank if they do not know how to specify them. However, current supporting on linguistic terms are primitive, more sophisticated approach on processing linguistic terms is desirable to further help unexperience users to specify their QoS requirements.

- *Developing more satisfaction score functions.*

  Satisfaction scores represent how well a service's QoS satisfy a user's QoS requirements, which is mutable and related to users' personal perspectives. Hence, the satisfaction is subjective and challenging to be modeled. Our proposed satisfaction score function try to model the changing of the satisfaction with the service's QoS through the importance of each QoS aspect. It is interesting to develop more functions to model satisfactions in different ways.

REFERENCES

[1] H. Kreger and J. Estefan, "Navigating the SOA Open Standards Landscape Around Architecture," *The Open Group*, 2009.

[2] N. M. Josuttis, *SOA in Practice: The Art of Distributed System Design*, O'Reilly Media, 1 edition, 2007.

[3] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services - Concepts, Architectures and Applications*, Springer, 2004.

[4] W3C, "Web Services Description Language (WSDL) Version 1.1," *W3 standards, available at `http://www.w3.org/TR/wsdl`*, 2001.

[5] W3C, "Simple Object Access Protocol (SOAP) Version 1.2," *W3 standards, available at `http://www.w3.org/TR/soap/`*, pp. 17–24, 2003.

[6] OASIS, "OASIS Universal Description Discovery and Integration (UDDI) Version 3," *OASIS standards, available at `http://uddi.org/pubs/uddi_v3.htm`*, 2004.

[7] J. Beatty, G. Kakivaya, D. Kemp, and et. al., "Web Services Dynamic Discovery (WS-Discovery)," *available at `http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf`*, 2005.

[8] OASIS, "Web Services Dynamic Discovery (WS-Discovery) Version 1.1," *OASIS standards, available at `http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html`*, 2009.

[9] OASIS, "Web Services Security," *OASIS standards, available at `http://www.oasis-open.org/committees/wss/`*, 2006.

[10] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz, "An Architecture for a Secure Service Discovery Service.," in *Proc. of 5th Ann. ACM/IEEE Int'l Conf. on Mobile Computing and Networking (MobiCom)*, 1999, pp. 24–35.

[11] "UPnP Security Ceremonies V1.0," *available at `http://www.upnp.org/download/standardizeddcps/UPnPSecurityCeremonies_1_0secure.pdf`*, 2003.

[12] J. Osullivan, "What's in Service? Towards Accurate Description of Non-Functional Service Properties," *Distributed and Parallel Database*, vol. 12, pp. 117–133, 2002.

[13] N. Thio and S. Karunasekera, "Automatic Measurement of a QoS Metric for Web Service Recommendation," in *Proc. of Australian conf. on Software Engineering*, 2005, pp. 202–211.

[14] L. Zeng, H. Lei, and H. Chang, "Monitoring the QoS for Web Services," in *Proc. of Int'l Conf. on Service-Oriented Computing (ICSOC)*, 2007, pp. 132–144.

[15] S. S. Yau, N. Ye, H. Sarjoughian, D. Huang, A. Roontiva, M. Baydogan, and M. Muqsith, "Toward Development of Adaptive Service-Based Software Systems," *IEEE Tran. on Services Computing*, vol. 2, no. 3, pp. 247–260, 2009.

[16] A. Ambrogio, "A Model-driven WSDL Extension for Describing the QoS ofWeb Services," in *Proc. of IEEE Int'l Conf. on Web Services*, 2006, pp. 789–796.

[17] K. Jacek, V. Tomas, B. Carine, and F. Joel, "SAWSDL: Semantic Annotations for WSDL and XML Schema," *IEEE Internet Computing*, vol. 11, pp. 60–67, 2007.

[18] J. Farrell and H. Lausen, "Semantic Annotations for WSDL and XML Schema," *W3C Recommendation, available at `http://www.w3.org/TR/sawsdl/`*, 2007.

[19] V. Agarwal and P. Jalote, "From Specification to Adaptation: An Integrated QoS-driven Approach for Dynamic Adaptation of Web Service Compositions," in *Proc. of IEEE Int'l Conf. on Web Services*, 2010, pp. 275–282.

[20] A. ShaikhAli, O. Rana, R. Al-Ali, and D. Walker, "UDDIe: an extended registry for Web services," in *Proc. of Applications and the Internet Workshops*, 2003, pp. 85 – 89.

[21] Z. Xu, P. Martin, W. Powley, and F. Zulkernine, "Reputation-Enhanced QoS-based Web Services Discovery," in *Proc. of IEEE Int'l Conf. Web Services*, 2007, pp. 249 –256.

[22] C. Lo, D. Cheng, P. Lin, and K. Chao, "A Study on Representation of QoS in UDDI for Web Services Composition," in *Proc. of Int'l Conf. on Complex, Intelligent and Software Intensive Systems (CISIS)*, 2008, pp. 423–428.

[23] E. Kim and Y. Lee, "Quality Model for Web Services v2.0," *OASIS Committee Draft, available at `http://www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc`*, 2005.

[24] D. Martin, M. Burstein, J. Hobbs, and et. al., “OWL-S: Semantic Markup for Web Services,” *W3C Member Submission, available at http://www.w3.org/Submission/OWL-S/*, 2004.

[25] J. Bruijin, C. Bussler, J. Domingue, and et. al., “Web Service Modeling Ontology (WSMO),” *WSMO Final Draft, available at http://www.wsmo.org/TR/d2/v1.3/*, 2006.

[26] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, “A QoS-aware Selection Model for Semantic Web Services,” in *Proc. of Int'l Conf. on Service-Oriented Computing*, 2006, pp. 309–401.

[27] A. Andrieux, K. Czajkowski, A. Dan, and et.al., “Web Service Agreement Specification (WS-Agreement),” *Open Grid Forum Proposed Recommendation, available at http://www.ogf.org/documents/GFD.107.pdf*, 2007.

[28] M. P. Papazoglou, “Service-oriented computing: concepts, characteristics and directions,” in *Proc. of 4th Int'l Conf. on Web Info. Systems Engineering*, 2003, pp. 3–12.

[29] “IBM Web Services Architecture Overview,” *available at http://www-128.ibm.com/developerworks/library/w-ovr/*.

[30] R. Laddaga, P. Robertson, and H. Shrobe, “Introduction to Self-Adaptive Software: Applications,” in *Proc. of 2nd Int'l Workshop on Self-Adaptive Software, Lecture Note in Computer Science, vol. 2614*, 2003, pp. 1–5.

[31] W3C, “Web Services Policy,” *W3C standards, available at http://www.w3.org/Submission/WS-Policy/*, 2006.

[32] M. Mactaggart, “Enabling XML Security: An Introduction to XML Encryption and XML Signature,” *IBM Developer Works*, vol. 9, 2001.

[33] D. Jordan and J. Evdemon, “Web Services Business Process Execution language (BPEL) Version 2.0,” *OASIS Standard, available at http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf*, 2007.

[34] B. Benatallah, M.-C. Fauvet M. Dumas, and F. Rabhi, *Towards Patterns of Web Services Composition*, pp. 265–296, Springer-Verlag, 2003.

[35] H. Hsu, “Special Issue on Workflow and Extended Transaction Systems,” *Bulletin of the IEEE Technical Committee on Data Engineering*, vol. 16, no. 2, 1993.

[36] D. Georgakopoulos, H. Schuster, A. Chichocki, and D. Baker, "Managing Process and Service Fusion in Virtual Enterprises," *J. Info. Systems*, vol. 24, no. 6, pp. 429–456, 1999.

[37] E. Sirin, J. A. Hendler, and B. Parsia, "Semi-Automatic Composition of Web Services Using Semantic Descriptions," in *Proc. of Web Services: Modeling, Architecture and Infrastructure (WSMAI) Workshop in conjunction with the 5th Intl Conf. on Enterprise Info. Systems (ICEIS)*, 2003, pp. 17–24.

[38] M. P. Wil van der Aalst, "Patterns and XPDL: A Critical Evaluation of the XML Process Definition Langauge," *Eindhoven University of Technology, available at `http://is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports/2003/BPM-03-09.pdf`*, 2003.

[39] S. R. Ponnekanti and A. Fox, "SWORD: A developer toolkit for Web Service Composition," in *Proc. of 11th Conf. on World Wide Web*, 2002.

[40] R. Jinghai and S. Xiaomeng, "A Survey of Automated Web Service Composition Methods," in *Proc. of 1st Intl Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, 2004, pp. 43–54.

[41] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz, "An Architecture for a Secure Service Discovery Service.," in *Proc. of 5th Ann. ACM/IEEE Int'l Conf. on Mobile Computing and Networking (MobiCom)*, 1999, pp. 24–35.

[42] B. Carminati, E. Ferrari, and P. C. K. Huang, "Web Service Composition: A Security Perspective," in *Proc. of 2005 Intl Workshop on Challenges in Web Info. Retrieval and Integration (WIRI)*, 2005, pp. 248–253.

[43] F. Zhu, M. W. Mutka, and L. M. Ni, "Service Discovery in Pervasive Computing Environments," *IEEE Pervasive Computing*, vol. 4, no. 4, pp. 81–90, 2005.

[44] F. Zhu, M. W. Mutka, and L. M. Ni, "A Private, Secure, and User-Centric Information Exposure Model for Service Discovery Protocols.," *IEEE Trans. on Mob. Comput.*, vol. 5, no. 4, pp. 418–429, 2006.

[45] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private Information Retrieval," in *Proc. of 36th Ann. Symp. on Foundations of Computer Science (FOCS)*, 1995, pp. 41–50.

[46] E. Kushilevitz and R. Ostrovsky, "Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval," in *Proc. of 38th Ann. Symp. on Foundations of Computer Science (FOCS)*, 1997, pp. 364–373.

[47] A. Beimel, Y. Ishai, E. Kushilevitz, and J. Raymond, "Breaking the O(n1/(2k-1)) Barrier for Information-Theoretic Private Information Retrieval," in *Proc. of 43rd Symp. on Foundations of Computer Science (FOCS)*. 2002, pp. 261–270, IEEE Computer Society.

[48] Y. Chang, "Single Database Private Information Retrieval with Logarithmic Communication," in *Proc. of 9th Australasian Conf. on Information Security and Privacy (ACISP)*, 2004, pp. 50–61.

[49] C. Gentry and Z. Ramzan, "Single-Database Private Information Retrieval with Constant Communication Rate," in *Proc. of 32nd Inte'l Colloquium on Automata, Languages and Programming (ICALP)*, 2005, pp. 803–815.

[50] H. Lipmaa, "An Oblivious Transfer Protocol with Log-Squared Communication," in *Proc. of 8th Inte'l Conf. on Information Security (ISC)*. 2005, vol. 3650 of *Lecture Notes in Computer Science*, pp. 314–328, Springer.

[51] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, "Protecting Data Privacy in Private Information Retrieval Schemes," *J. Comput. Syst. Sci.*, vol. 60, no. 3, pp. 592–629, 2000.

[52] T. Itoh, "On Lower Bounds for the Communication Complexity of Private Information Retrieval," *IEICE Tran. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 84, no. 1, pp. 157–164, 2001.

[53] S. Goldwasser and S. Micali, "Probabilistic Encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.

[54] B. Chor, N. Gilboa, and M.Naor, "Private Information Retrieval by Keywords," *Unpublished manuscript available at http://www.cs.technion.ac.il/~gilboa*, 1998.

[55] D. X. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data.," in *Proc. of IEEE Symp. on Security and Privacy (S&P)*, 2000, pp. 44–55.

[56] Y. Xie, M. K. Reiter, and D. O'Hallaron, "Protecting Privacy in Key-Value Search Systems," in *Proc. of 22nd Ann. Computer Security Applications Conf. (ACSAC)*, Washington, DC, USA, 2006, pp. 493–504, IEEE Computer Society.

[57] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proc. of ACM Conf. on Computer and Communications Security (CCS)*. 2006, pp. 79–88, ACM.

[58] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an Encrypted and Searchable Audit Log," in *Proc. of the Network and Distributed System Security Symp. (NDSS)*. 2004, The Internet Society.

[59] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," in *Advances in Cryptology - EUROCRYPT*, 2004, pp. 506–522.

[60] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions.," in *Advances in Cryptology - CRYPTO*, 2005, pp. 205–222.

[61] J. Baek, R. Safavi-Naini, and W. Susilo, "On the Integration of Public Key Data Encryption and Public Key Encryption with Keyword Search.," in *Proc. of 9th Information Security Conf. (ISC)*, 2006, pp. 217–232.

[62] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption.," in *Advances in Cryptology - EUROCRYPT*, 2005, pp. 457–473.

[63] E. Shi, J. Bethencourt, H. T.-H. Chan, D. X. Song, and A. Perrig, "Multi-Dimensional Range Query over Encrypted Data," in *Proc. of IEEE Symp. on Security and Privacy (S&P)*, 2007, pp. 350–364.

[64] D. Boneh and B. Waters, "Proc. of Conjunctive, Subset, and Range Queries on Encrypted Data.," in *Theory of Cryptography (TCC)*, 2007, pp. 535–554.

[65] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes.," in *Advances in Cryptology - EUROCRYPT*, 1999, pp. 223–238.

[66] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection.," in *Advances in Cryptology - EUROCRYPT*, 2004, pp. 1–19.

[67] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword Search and Oblivious Pseudorandom Functions.," in *Theory of Cryptography (TCC)*, 2005, pp. 303–324.

[68] L. Kissner and D. X. Song, "Privacy-Preserving Set Operations.," in *Advances in Cryptology - CRYPTO*, 2005, pp. 241–257.

[69] J. Bethencourt, D. X. Song, and B. Waters, "New Constructions and Practical Applications for Private Stream Searching (Extended Abstract)," in *Proc. of IEEE Symp. on Security and Privacy (S&P)*, 2006, pp. 132–139.

[70] M. Bartoletti, P. Degano, and G. L. Ferrari, "Enforcing Secure Service Composition," in *Proc. of 18th IEEE Computer Security Foundations Workshop (CSFW)*, 2005, pp. 211–223.

[71] X. Gu, K. Nahrstedt, and B. Yu, "SpiderNet: An Integrated Peer-to-Peer Service Composition Network," in *Proc. of 13th IEEE Intl Symp. on High Performance Distributed Computing*, 2004, pp. 110–119.

[72] L. W. Li, L. C. Chun, C. K. Ming, and Y. Muhammad, "Fuzzy Consensus on QoS in Web Services Discovery," in *Proc. of Int'l Conf. on Advanced Information Networking and Applications (AINA)*, 2006, pp. 791–798.

[73] S. Nepal, W. Sherchan, J. Hunklinger, and A. Bouguettaya, "A Fuzzy Trust Management Framework for Service Web," in *Proc. of IEEE Int'l Conf. on Web Services (ICWS)*, 2010, pp. 321–328.

[74] Z. Malik and A. Bouguettaya, "Rater Credibility Assessment in Web Services Interactions," *World Wide Web*, vol. 12, pp. 3–25, 2009.

[75] H. T. Nguyen, W. Zhao, and J. Yang, "A Trust and Reputation Model Based on Bayesian Network for Web Service," in *Proc. of IEEE Int'l Conf. on Web Services (ICWS)*, 2010, pp. 251–258.

[76] S. S. Yau, J. Huang, and Y. Yin, "Improving the Trustworthiness of Service QoS Information in Service-based Systems," in *Proc. of 7th Autonomic and Trusted Computing (ATC)*, 2010, pp. 208–218.

[77] M. Godse, U. Bellur, and R. Sonar, "Automating QoS Based Service Selection," in *Proc. of IEEE Int'l Conf. on Web Services (ICWS)*, 2010, pp. 534–541.

[78] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, "Monitoring, Prediction and Prevention of SLA Violations in Composite Services," in *Proc. of IEEE Int'l Conf. on Web Service (ICWS)*, 2010, pp. 369–376.

[79] L. Eggert and J. Heidemann, "Application-Level Differentiated Services for Web Services," *J. World-Wide Web*, vol. 2, no. 3, pp. 133–142, 1999.

[80] G. Rao and B. Ramamurthy, "DiffServer: Application Level Differentiated Services for Web Servers," in *Proc. of IEEE Int'l Conf. on Communication*, 2001, pp. 1633–1637.

[81] T. F. Abdelzaher, J. A. Stankovic, R. Zhang C. Lu, and Y. Lu, "Feedback Performance Control in Software Services," *IEEE Control Systems Magazine*, vol. 23, no. 3, pp. 74–90, 2003.

[82] C. Lu, Y. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son, "Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers," *IEEE Trans. on Parallel and Distributed Systems*, vol. 17, no. 9, pp. 1014–1027, 2006.

[83] S. Godik and T. Moses, "Extensible Access-Control Markup Language (XACM-L) v1.0. OASIS Standard," *available at http://www.oasis-open.org/*, 2003.

[84] P. Hallam-Baker and S. Mysore, "XML Key Management Specification (XKM-S 2.0)," *W3C Recommendation, available at http://www.w3.org/TR/xkms2/*, 2005.

[85] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, , P. Hallam-Baker, and et al, "Web Services Policy Framework (WS-Policy)," *BEA Systems Inc., Int'l Business Machines Corporation, Microsoft Corporation, Inc., SAP AG, Sonic Software, and VeriSign Inc*, 2004.

[86] A. Pashalidis and C. J. Mitchell, "A Taxonomy of Single Sign-On Systems," in *Proc. of Info. Security and Privacy*, 2003, pp. 249–264.

[87] W. Yurcik, C. Woolam, G. Hellings, L. Khan, and B. Thuraisingham, "SCRUB-tcpdump: A Multi-level Packet Anonymizer Demonstrating Privacy/Analysis Tradeoffs," in *Proc. of 3rd Security and Privacy in Commu. Networks and the Workshops (SecureComm)*, 2007, pp. 49–56.

[88] K. Kang and S. Son, “Systematic Security and Timeless Tradeoffs in Real-Time Embedded Systems,” in *Proc. of 12th IEEE Int’l Conf. on Embedded and Real-Time Computing Systems and Application*, 2006, pp. 183–189.

[89] E. Spyropoulou, T. Levin, and C. Irvine, “Calculating Costs for Quality of Security Service,” in *Proc. of 16th Ann. Conf. Computer Security Applications*, 2000, pp. 334–343.

[90] S. H. Son, R. Zimmerman, and J. Hansson, “An Adaptable Security Manager for Real-Time Transactions,” in *Proc. of 12th Euromicro Conf. on Real-Time Systems*, 2000, pp. 63–70.

[91] Y. Liu, A. H. Ngu, and L. Zeng, “QoS Computation and Policing in Dynamic Web Service Selection,” in *Proc. of Int’l World Wide Web Conf. (WWW)*, 2004, pp. 66–73.

[92] L. Taher, H. E. Khatib, and R. Basha, “A Framework and QoS Matchmaking Algorithm for Dynamic Web Services Selection,” in *Proc. of 2nd Int’l Conf. on Innovations in Information Technology (IIT)*, 2005, pp. 117–133.

[93] M. Comuzzi and B. Pernici, “A Framework for QoS-Based Web Service Contracting,” *ACM Tran. on The Web*, vol. 3, no. 3, 2009.

[94] A. Srivastava and P. G. Sorenson, “Service Selection Based on Customer Rating of Quality of Service Attributes,” in *Proc. of IEEE Int’l Conf. on Web Services (ICWS)*, 2010, pp. 1–8.

[95] E. A. Masri and Q. H. Mahmoud, “QoS-based Discovery and Ranking of Web Services,” in *Proc. of Int’l Conf. on Computer Communications and Networks (ICCCN)*, 2007, pp. 529–534.

[96] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*, Cambridge University Press, 2009.

[97] A. Joux, “A One Round Protocol for Tripartite Diffie-Hellman.,” *J. Cryptology*, vol. 17, no. 4, pp. 263–276, 2004.

[98] D. Boneh, E. Goh, and K. Nissim, “Evaluating 2-DNF Formulas on Ciphertexts.,” in *Theory of Cryptography (TCC)*, 2005, pp. 325–341.

[99] C. Gentry, *A Fully Homomorphic Encryption Scheme*, Ph.D. thesis, Stanford University, 2009.

[100] A. Yamamura and T. Saito, "Private Information Retrieval Based on the Subgroup Membership Problem.," in *Information Security and Privacy, 6th Australasian Conference (ACISP)*, 2001, pp. 206–220.

[101] A. Kiayias and A. Mitrofanova, "Testing Disjointness of Private Datasets.," in *Proc. of Int'l Conf. on Financial Cryptography (FC)*, 2005, pp. 109–124.

[102] M. Naor and B. Pinkas, "Oblivious Transfer with Adaptive Queries.," in *Advances in Cryptology - CRYPTO*, 1999, pp. 573–590.

[103] M. Naor and B. Pinkas, "Oblivious Transfer and Polynomial Evaluation.," in *Proc. of the Annu. ACM Symp. on Theory of Computing (STOC)*, 1999, pp. 245–254.

[104] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara, "Semantic Matching of Web Services Capabilities.," in *Proc. of 1st Int'l Semantic Web Conf. (ISWC)*, 2002, pp. 333–347.

[105] OASIS, "UDDI Core v2 and v2/v3 Utility Classification Schemes, Taxonomies, Identifier Systems, and Relationships," *available at http://www.uddi.org/taxonomies/UDDI_Taxonomy_tModels.htm*, 2004.

[106] Y. Chen, T. Farley, and N. Ye, "QoS Requirements of Network Applications on the Internet," *Info., Knowledge, Systems Management*, vol. 4, no. 1, pp. 57–76, 2004.

[107] L Hathaway, "National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information," *National Security Agency*, 2003.

[108] D. Lie and M. Satyanarayanan, "Quantifying the Strength of Security Systems," in *Proc. of 2nd USENIX Workshop on Hot Topics in Security*, 2007, pp. 1–6.

[109] A Lenstra and E Verheul, "Selecting Cryptographic Key Sizes," *J. of Cryptology*, vol. 14, pp. 255–293, 2001.

[110] H. Cheng and X. Li, "Partial Encryption of Compressed Images and Video," *IEEE Trans. on Signal Processing*, vol. 48, no. 8, pp. 2439–2451, 2000.

[111] M. V. Droogenbroeck and R. Benedett, "Techniques for a Selective Encryption of Uncompressed and Compressed Images," in *Proc. of Adv. Concepts for Intelligent Vision Systems (ACIVS)*, 2002, pp. 9–11.

[112] C. Shi, S. Y. Wang, and B. Bhargava, "MPEG Video Encryption in Real-Time Using Secret key Cryptography," in *Proc. of Int'l Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, 1999.

[113] W. Zeng, , and S. Lei, "Efficient Frequency Domain Selective Scrambling of Digital Video," *IEEE Trans. on Multimedia*, vol. 5, no. 1, pp. 118–129, 2003.

[114] A. Biryukov and D. Khovratovich, "A New Security Analysis of AES-128," *available at `http://rump2009.cr.yp.to/`*, 2009.

[115] C. Pomerance, "A Tale of Two Sieves," *Notices of the AMS*, vol. 43, no. 12, pp. 1473–1485, 1996.

[116] H. B. Mann and D. R. Whitney, "On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other," *Annuals of Math. Statistics*, pp. 50–60, 1947.

[117] D. C. Montgomery, G. C. Runger, and N. F. Hubele, *Engineering Statistic*, Wiley, four edition, 2006.

[118] R. Steel and J. Torrie, *Principles and Procedures of Statistics*, McGraw-Hill, 1960.

[119] Daniel Kahneman and Amos Tversky, "Prospect Theory: An Analysis of Decision under Risk," *Econometrica*, vol. 47, no. 2, pp. 263–292, 1979.

[120] Amos Tversky and Daniel Kahneman, "Advances in prospect theory: Cumulative representation of uncertainty," *Journal of Risk and Uncertainty*, vol. 5, pp. 297–323, 1992, 10.1007/BF00122574.

[121] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine Grained Access Control of Encrypted Data," 2006, p. 89C98.

[122] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of 13th ACM Conf. on Computer and Communications Security (CCS)*, 2006, pp. 89–98.