天主教輔仁大學圖書資訊學系碩士班

指導老師：李正吉 博士

雲端儲存服務之資訊安全研究

The Study on Information Security for Cloud Storage Service

研究生：邱詩婷 撰

中國民國 103 年 12 月

私立輔仁大學圖書資訊學系碩士班
論文口試委員會審定書

邱詩婷 先生之碩士學位論文

雲端儲存服務之資訊安全研究
The Study on Information Security for Cloud Storage Service

經本委員會審議合格，特此證明。

論文口試委員	指導老師
謝建成（召集人）	李正吉
李俊達	

系主任
陳舜德

中華民國 103 年 12 月

誌謝 (ACKNOWLEDGEMENTS)

首先必須感謝我的指導教授李正吉老師在這兩年多在研究方面的指導，不僅僅是給予我研究的方向，也指導我論文撰寫以及做研究的技巧。謝謝我的兩位口試教授：謝建成老師以及李俊達老師，兩位老師給予的意見讓我的論文更加完整。

謝謝彥銘和哲維兩位學長在研究上給予我的幫忙，學長們不厭其煩的解決我的疑問，也讓我從他們身上學到許多研究的技巧。在這兩年多也很感謝圖資所 401 級的同學們，彼此的互相鼓勵也增添了些許動力。謝謝 LE504 的夥伴們，謝謝姿穎、彥宏這段時間和我彼此監督進度，也謝謝心怡學妹給予的關心。除了感謝輔大圖資所的同學們之外，亦要感謝陪伴在我身邊的朋友們，謝謝鄭婷總是在我英文文法或是單字有問題的時候給予我指導，沒有這些同學以及朋友的陪伴，這段日子必定會枯燥乏味又艱辛，論文也不會如期完成。

最後要感謝一直在我背後支持我的父母，沒有他們的支持、支援與肯定，我是沒有辦法如期完成這論文以及學業。

謝謝大家

中文摘要

隨著網際網路日漸發展與成長，許多的應用相應而生，而近幾年受到矚目的便是雲端運算。由於他強大的架構方式以及運算能力，使得各種應用相應而生，雲端儲存服務便為其中一項。回歸雲端運算的本質，其依舊需要倚靠網路進行資料的傳輸，而在這傳輸的過程中並非完全安全，例如在傳輸的過程中，非法使用者攔截重要的資訊、偽造合法使用者的身分進行通訊或是偽造密文等。因此，如何確保使用者在使用雲端儲存服務時能夠保有隱私並正確的儲存及接受資訊是個重要的議題。在近幾年研究中，許多學者研究如何讓使用者更方便的使用雲端儲存環境並且讓資料更安全的被傳輸以及儲存，例如：基於 ID(identity)的金鑰管理方案、階層式基於屬性的加密方案、隱私維護的稱謂語加密方案、隱私維護的關鍵字搜尋方案等。這些方法提供使用者安全及便利的雲端儲存環境，不但保障了資訊與使用者的隱私外，也減輕了使用者的負擔。

在本論文中，我們將分析近幾年應用於雲端儲存服務的研究，包含了關鍵字搜尋、代理重新加密及以屬性為基礎的資料加密等，並基於雙線性映射函數的數學原理上提出新的方法。根據安全性與特性的分析，我們的方法比過去的研究更為安全且更能應用於實際環境中。

關鍵字：雲端運算、雲端儲存、關鍵字搜尋、雙線性映射函數、時效性

ABSTRACT

With the progression and growth of Internet, many applications have been developing based on Internet. The most popular application is cloud computing. Because of its strong architectural and high computing performance, many applications have been developing and cloud storage service is one of them. Back the essence of cloud computing, it still to rely on the Internet to transfer the data. It is not secure during the procedure of transfer data, for example: the attacker will intercept the important information during the procedure, forge identity of legitimate users to join communication or forge the ciphertext. Therefore, how to ensure the privacy of the user and access the data correctly in cloud storage service is an important issue. In recent years, many researchers focus on how to use the cloud storage service more secure and convenient for the data and the user, for example, identity-based key management, hierarchical attribute-based encryption scheme, and controllable privacy preserving search based on symmetric predicate encryption etc. These schemes provide the secure and convenient cloud storage environment for the user, it not only guarantee the privacy of data and user, but also reduce the burden for user.

In this study, we will analyze the researches of recent years which applied to cloud storage services, and these researches contain the keyword search scheme, proxy re-encryption scheme and attribute-based encryption. Based on bilinear pairing, we present some new schemes. According to the analyses of security and properties, our method is more secure than previous researches and more flexible in a practical environment.

Keywords: Cloud computing, Cloud storage, Keyword search, Bilinear pairing, Time-bound

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

1.1 Research Motivation

Technology grows so as to satisfy our needs, and our everyday behavior patterns are shaped by the growth of technology. This applies to people's habit of data storage. Not long ago, people were still keeping their data in flash drives and portable hard drives, enjoying the convenience those "advanced" handy little devices brought rather than storing everything in the computer itself. Nowadays, with the rapid development of cloud computing technologies, many yet more advanced applications have begun to surface, cloud storage service among the rest. According to Wikipedia, cloud storage can be defined as a model of data storage where the digital data is stored in logical pools, the physical storage can usually span multiple servers, and the physical environment is typically owned and managed by a hosting company. The cloud storage service provider has the responsibility of keeping the data available and accessible to the client at any time and securing the data stored in the cloud against any form of attack. So, speaking of data storage, instead of flash drives and portable hard drives, more and more people now will think of cloud storage services such as Dropbox, SkyDrive, and MEGA. With everything saved in the cloud, people can readily access their data anytime and anywhere using any device as long as that device gets online. However, as far as data security is concerned, Edward Snowden pointed out that popular consumer Internet services like Dropbox are "hostile to privacy". As more and more sensitive data are trusted to cloud storage, information security becomes a bigger and bigger issue. Or, to put it another way, whichever cloud storage service provider can offer a better solution to the data security problem will surely have a huge advantage over others because

cloud storage users will not trust their sensitive data to a service provider otherwise.

Indeed, cloud data security, privacy, and confidentiality protection has become a major focus of research [1]. In 2010, Wang et al. [66] and Yu et al. [76] exploited the concept of attribute-based encryption and proposed a high performance fine access control scheme with collusion resistance and a fine-grained, scalable data access control scheme with data confidentiality for cloud computing, respectively. Then in 2011 Huang et al. proposed an efficient identity-based key management scheme for configurable hierarchical cloud environment that offers high performance at low communication costs on encryption [32].

On the other hand, with tons and tons of data stored in the cloud, how the user can have easy access to some specific data desired is also a major concern. In the past, there used to be two ways to retrieve the desired data from the cloud [23]. The first and most straightforward way was for the user to download everything stored in the cloud and then decrypt all the data and then search the whole thing for the part or parts of data desired. The second way was for the user to send a secret key to the cloud server, who then could use that secret key to decrypt and find the desired data for the user. Unfortunately, just as it sounds, the first way is a lot of trouble for the user. As for the second, it is nothing better because serious security problems can arise given that there is always a possibility that the cloud server, now holding the user's secret key, is a malicious server ready to do something evil.

Keyword search is a solution to the above problems. The concept of keyword search through encrypted data was proposed by Song et al. in 2000 [59]. In a keyword search scheme, people can use a keyword to search through encrypted data and find the part or parts of data previously encrypted by using that keyword. This way, no information will leak out during the keyword search process, and the downloading and

decryption will only involve the part or parts the user wishes to access. So far, quite a lot of research endeavors have been devoted to the development of cloud storage keyword search technologies [14, 26, 28, 33, 49, 54]. In 2004, Boneh et al. [7] proposed a scheme called public key encryption with keyword search (PEKS). Then, in 2008, Baek et al. extended Boneh et al.'s PEKS into a secure channel-free public key encryption scheme with keyword search (SCF-PEKS) [2] where the secure channel between the server and the user is removed to reduce the cost. After these studies, researchers have been working on different keyword search mechanisms. For example, in 2009, Liu et al. [43] offered an efficient privacy-preserving keyword search scheme (EPPKS) that can be viewed as an improved version of PEKS. Then, in 2011, Li et al. proposed another type of keyword search called fuzzy keyword search [41]. In 2012, Liu et al. [44] improved Liu et al.'s EPPKS [43] into a secure, privacy-preserving keyword search (SPKS) scheme. In the meanwhile, Zhao et al. [77] also proposed a new efficient trapdoor-indistinguishable public key encryption scheme with keyword search which does not require a secure channel between the receiver and the server, where the trapdoor is updated and kept fresh for every session. For cases where the data owner wishes to put some limits to the user's time of data access, issuing a time-bound key to the user is a good way. In 2014, Liu et al. combined the concept of attribute-based encryption and time-based key to create a time-based proxy re-encryption scheme for data sharing in cloud environments [45]. In this paper, we shall propose three new schemes to satisfy all the requirements raised up in the scenarios mentioned above with security issues such as data confidentiality, privacy, integrity, and authority well taken care of.

1.2 Research Subject

This study did not only focus on data confidentiality, privacy protection, and data integrity maintenance in cloud storage environments but also aimed to make the cloud data access process more user-friendly. The first scheme we shall propose in this thesis is a secure trapdoor-indistinguishable public encryption scheme with keyword search. The trapdoor-indistinguishability property means if the user sends the trapdoor of a certain keyword to the CSP multiple times, the trapdoor is updated and thus kept fresh every time when the user sends the requirement. In 2012, Zhao et al. proposed a new efficient trapdoor-indistinguishable public key encryption scheme with keyword search that does not require a secure channel between the receiver and the server. Although Zhao et al.'s scheme can satisfy such security requirements as user authentication and authorized identity protection, it fails to keep the CSP from storing fake ciphertexts. In other words, if the CSP didn't verify the identity of the data owner and thus stored a fake ciphertext, it cannot later search for the data the user wants.

The second scheme is a searchable hierarchical conditional proxy re-encryption scheme. In Weng et al.'s conditional proxy re-encryption scheme [72], the data owner can assign which ciphertext satisfies a certain keyword condition set, and the semi-trusted proxy server can do re-encryption. It is true that Weng et al.'s ideas are very helpful in handling the huge amounts of data in cloud environments; however, in reality, their scheme fails in both encrypted data searching and conditional proxy re-encryption. Inspired by Fang et al. [24], we shall propose a searchable hierarchical conditional proxy re-encryption scheme we have created that combines keyword search and conditional proxy re-encryption.

The third scheme we shall propose in this thesis is a time-bound key-aggregate encryption scheme. Handling huge loads of data that are subject to change at any time,

cloud storage services are facing the challenge of properly dealing with the problem of user legality management while making sure that the services provided are conveniently user-friendly. Chu et al. [19] proposed a scheme called Key-Aggregate Cryptosystem (KAC). Distinct form typical attribute-based encryption schemes, in Chu et al.'s scheme, the user can use one aggregate key to decrypt data of all the attributes specified. This is a very convenient design for the user. Besides that, there are cases where the data owner does not want the data stored in the cloud to be open for access all the time, and this is when the concept of time control comes in. Inspired by Chu et al.'s scheme, we have created a new scheme that combines the concept of key-aggregate cryptosystem and the use of time-bound key. Our third new scheme is not only extremely user-friendly but also guarantees data security.

1.3 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we will introduce Zhao et al.'s trapdoor-indistinguishable public key encryption scheme with keyword search, followed by our improved version of the scheme. Then, in Chapter 3, we will present our new scheme that combines keyword search and conditional proxy re-encryption for cloud storage. In Chapter 4, we will detail our time-bound key-aggregate encryption scheme for cloud storage service. Finally, the conclusion will be in Chapter 5.

Chapter 2 Public Key Encryption Scheme with Keyword Search

Cloud storage allows users to easily access their data in cloud anytime and anywhere by using any device that can get online, such as a wireless PDA, a smartphone, or a notebook computer. Nevertheless, how can we make sure that this simple access to cloud storage comes at a satisfactory security level? Keyword search with data encryption seems to be a good answer. In 2012, Zhao et al. proposed a trapdoor-indistinguishable public key encryption scheme with keyword search to be applied to the field of cloud storage service. However, we found a weakness in Zhao et al.'s scheme. In this paper, we shall point out the weakness and offer an improved version of trapdoor-indistinguishable public key encryption with keyword search for cloud environments. In our improved scheme, we make the keyword trapdoor indistinguishable while protecting the PEKS ciphertext against forgery attacks. Compared with other PEKS schemes, our new design is not only more efficient but gives better performance in terms of correctness and security.

2.1 Preliminaries

Cloud computing refers to both the applications delivered as services over the Internet and the hardware as well as systems software in the data centers that provide those services [68]. Cloud storage is one of the most popular applications served by the cloud. Nowadays, more and more people and businesses keep their data in the cloud. Thanks to the cloud storage service, with a tiny, lightweight device such as a wireless PDA, smartphone or notebook in their hands, users can readily access their data anytime

and anywhere. As cloud storage technologies advance, the security of the data stored in cloud environments becomes a more and more important issue. To keep any malicious party from accessing and making use of the data stored in the cloud, data owners often need to encrypt the data before uploading them to the cloud server. In that case, when a legal user wishes to access the data stored in the cloud, he/she will have to download the data as a whole instead of picking out and downloading only the relevant part or parts. For example, let's suppose both Alice and Bob are legal users of some specific data. Alice stored the data in the cloud, and Bob wants to access some information about "computer". Bob has no choice but to download all the data stored in the cloud before he can sort out the parts of the data that are actually related to "computer". The downloading of the whole pack of data can be a real waste of time and resources especially when the data stored in the cloud are in very large quantities while only very small portions of them need to be accessed. To retrieve only the part or parts of the data that the user really needs, keyword search seems to be a good solution.

However, if the uploaded data in the cloud has been encrypted by the data owner, then how can we make keyword search work? In 2000, Song et al. [59] proposed a secure keyword search scheme using a symmetric cipher. In 2004, in their well-celebrated article entitled "Public Key Encryption with Keyword Search", Boneh et al. [6] went a step further and offered a scheme later often referred to as PEKS. Boneh et al.'s PEKS scheme has a secure channel between the cloud server and the user. In 2008, in order to reduce the cost, Baek et al. extended Boneh et al.'s PEKS scheme into a secure-channel-free public key encryption scheme with keyword search (SCF-PEKS) [2]. However, in 2009, Rhee et al. pointed out that Baek et al.'s SCF-PEKS was vulnerable to the keyword guessing attack [51], and so they proposed the concept of trapdoor indistinguishability [52]. On the other hand, Liu et al. proposed an efficient

privacy-preserving keyword search scheme (EPPKS) [43] which improved the performance of PEKS. Meanwhile, in 2010, Li et al. [41] proposed a fuzzy keyword search scheme based on keyword similarity semantics capable of responding with the closest possible matching files. In 2012, Liu et al. [44] improved their earlier work EPPKS [43] and proposed a secure and privacy-preserving keyword search (SPKS) scheme. Besides, Zhao et al. [77] also proposed a trapdoor-indistinguishable public key encryption scheme with keyword search that does not require a secure channel between the receiver and the server. In addition to the researches mentioned above, quite a number of studies can be found in the literature concerned that focus on the quest for PEKS and keyword search with high efficiency and security [14, 26, 28, 31, 33, 39, 49, 54].

Although PEKS schemes do enable users to get to the data they wish to access, how to make that happen in cloud environments with privacy fully protected is an important research issue. In 2013, Hsu et al. [31] made a list of some security requirements to be met in cloud computing environments as follows:

1. **User authentication**

   The CSP (Cloud Service Provider) needs to confirm that the trapdoor of the keyword is sent from the authorized user and no one can discover the authorized user's real identity except for the CSP.

2. **Authentication of data owner**

   When the CSP receives the ciphertext from the data owner, in order to avoid having fake ciphertext stored, the CSP needs to authenticate that the ciphertext is sent from the real data owner.

3. **Protection of authorized identity**

   In case an attacker has the trapdoor ciphertext intercepted on the way from the data owner to the CSP, the attacker cannot derive the user's identity from the intercepted trapdoor ciphertext.

4. **Trapdoor indistinguishability**

   Due to the fact that the trapdoor ciphertext is sent via a public channel, an attacker may intercept the trapdoor ciphertext and try to figure out the real keyword. Trapdoor indistinguishability is the kind of protection that ensures no malicious attacker can obtain the information hidden in the trapdoor ciphertext by analyzing the trapdoor ciphertext.

5. **Resistance to keyword-guessing attack**

   The trapdoor is frequently updated, and that is why it is said to be indistinguishable. With the trapdoor collected, an attacker still cannot offline/online guess the real keyword from the trapdoor.

The PEKS schemes currently available can indeed provide user authentication and identity protection. However, there is not a mechanism to keep the CSP from storing fake ciphertext. Figure 1 shows a scenario where the data owner intends to send the data's ciphertext and PEKS ciphertext to the CSP, but both pieces of ciphertext get intercepted by an attacker. The attacker then sends some fake ciphertext to the CSP. When receiving the fake ciphertext, without verifying the validity of the data owner, the CSP stores them as always so that the data can be searched and retrieved by users. Later on, when a legal user needs to access some data which can be directed to by a certain keyword, he/she creates a trapdoor for that keyword and sends it to the CSP. Since the CSP stored the wrong ciphertext, the server fails to retrieve the correct data. Finally, the user cannot get the due ciphertext to decrypt.

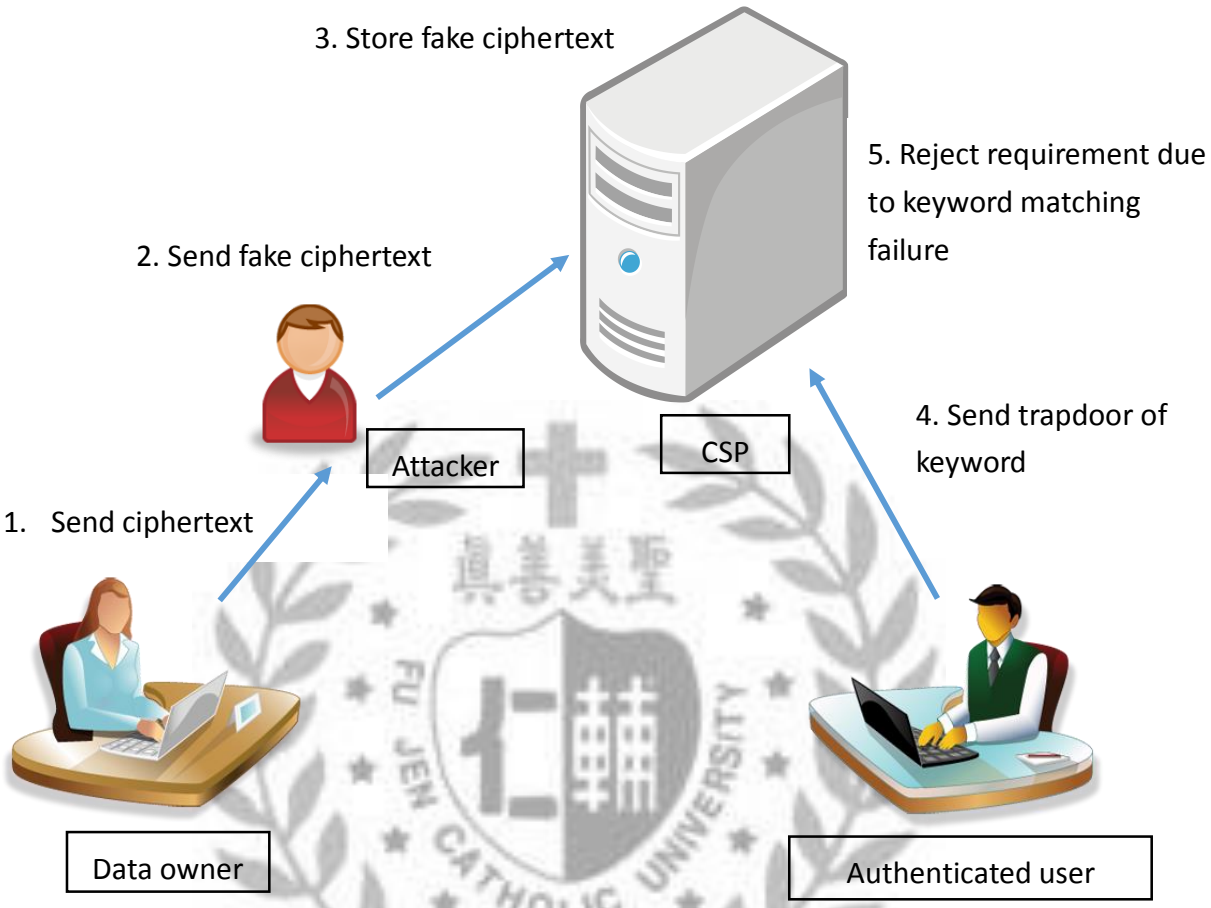

Figure 1 PEKS without data owner authentication

To mend this flaw, in this Section, we propose a secure trapdoor-indistinguishable public key encryption scheme with keyword search for cloud storage that satisfies the following requirements:

1. There is no need for a secure channel between the cloud user and the cloud service provider (CSP). In other words, the trapdoor can be sent via a public channel.
2. The trapdoor is indistinguishable. Even though an attacker can intercept the trapdoor, he/she still has no way to derive the real keyword by analyzing the trapdoor.

3. The CSP can search through the ciphertext for keywords. The CSP can check whether or not the data contains certain keywords specified by the user without knowing the keywords and the content of the data.
4. The CSP can verify whether the PEKS ciphertext is sent from the data owner and thereby avoid the forgery attack.

2.2 Related Works

In this section, we will quickly introduce the bilinear pairing technique [7] as well as some complexity assumptions and review the trapdoor-indistinguishable public key encryption scheme with keyword search (TI-PEKS) by Zhao et al. [77].

2.2.1 Bilinear Pairing

Let $\mathbb{G}_1$ be a cyclic additive group with prime order q and $\mathbb{G}_2$ be a cyclic multiplicative group with prime order q, and suppose P is the generator of group $\mathbb{G}_1$. With $x, y \in \mathbb{Z}_q$ and bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, there are some properties as follows:

- Bilinearity: For all $x, y \in \mathbb{Z}_q$ and $R, Q \in \mathbb{G}_1$, $e(xR, yQ) = e(R, Q)^{xy}$.
- Computability: For any $R, Q \in \mathbb{G}_1$, there exists an efficient algorithm to compute $e(R, Q) \in \mathbb{G}_2$.
- Non-degeneration: $e(R, Q) \neq 1$.

2.2.2 Complexity Assumptions

Some complex problems can be created out of $\mathbb{G}_1$ as follows:

- Discrete Logarithm Problem (DLP)

  Given two elements R and Q in $\mathbb{G}_1$, it is difficult to find $n \in \mathbb{Z}_q$ such that $R = nQ$ if n exists.

- Computation Diffie-Hellman Problem (CDHP)

  Given R, xR, yR for $x, y \in \mathbb{Z}_q$, it is difficult to compute xyR.

- Bilinear Diffie-Hellman Problem (BDHP)

  Given R, R^x, R^y, R^z for $x, y, z \in \mathbb{Z}_q$, it is difficult to compute $e(R, R)^{xyz} \in \mathbb{G}_2$.

2.2.3 Trapdoor-indistinguishable Public Key Encryption with Keyword Search

In this subsection, we will review Zhao et al.'s trapdoor-indistinguishable public key encryption scheme with keyword search. In Zhao et al.'s TI-PEKS, there are three parties involved, namely the sender, the server, and the receiver. The scheme contains six algorithms as follows:

- $KeyGen_{Param}(k)$: A common parameter generation algorithm. With a security parameter $k \in \mathbb{N}$ entered, the algorithm outputs the system's common parameters cp.

- $KeyGen_{Server}(cp)$: The public/private key generation algorithm for the server. It takes in the common parameters cp and outputs the public key pk_S and the private key sk_S for the server.

- $KeyGen_{Receiver}(cp)$: The public/private key generation algorithm for the receiver. With the common parameters cp taken in, the algorithm outputs the public key pk_R and the private key sk_R for the receiver.

- $PEKS(cp, pk_S, pk_R, w)$: The generation algorithm of the ciphertext's PEKS R. The data owner inputs the system's common parameters cp, server's public key pk_S, receiver's public key pk_R, as well as the keyword w, and then the algorithm outputs the ciphertext's PEKS R that is searchable.

- $Trapdoor(cp, sk_R, w)$: The trapdoor generation algorithm. The receiver inputs the system's common parameters cp, his/her private key sk_R, as well as the keyword w, and then the algorithm generates the trapdoor TW of the keyword w.

- $Test(cp, TW, sk_S, R)$: The keyword test algorithm. Input the system's common parameters cp, the server's public key sk_S, the ciphertext's PEKS R and the trapdoor TW of the keyword w, and the algorithm will return "correct" if $w' = w$ and "incorrect" otherwise.

2.3 New Scheme

In this section, we shall first illustrate the architecture of our propoesd TI-PEKS scheme and then give the details of each step.

2.3.1 Process

In our improved scheme, there are 8 steps to take, namely system parameter generation, key generation for cloud service provider (CSP), key generation for user, key generation for data owner, PEKS ciphertext generation, ciphertext verification, keyword trapdoor generation, and search. Three participants are involved, including the data owner, who generates the data's ciphertext and PEKS ciphertext and sends them to the CSP; the CSP, who provides the storage, stores the data and searches the data for the specific parts that the user requests; and the user, who wishes to retrieve certain parts of the data that contain a specific keyword and therefore sends the keyword's trapdoor to the CSP. Figure 2 is the flowchart of our scheme with the purpose each step serves specified:

- $KeyGen_{Param}$: In this step, some security parameters will be input to the system, and the system will output the common parameters.

- $KeyGen_{CSP}$: With the public parameter taken in as input, the system outputs the CSP's public key and private key.

- $KeyGen_{User}$: With the public parameter and the user's identity entered as input, the system outputs the user's public key and private key.

- $KeyGen_{Data\ owner}$: Taking in the public parameter and the data owner's identity as input, the system outputs the data owner's public key and private key.

- $PEKS$: With the data encrypted, the data owner uses the common parameters and the user's public key to generate the keyword w's PEKS ciphertext. In addition, the data owner uses his/her private key to generate the verification message and sends the data's ciphertext, PEKS ciphertext and verification message to the CSP.

- $Verify$: Upon receiving the encrypted data, the CSP uses the data owner's public key to verify whether the ciphertexts were actually sent by the data owner. If yes, the CSP stores the data; otherwise, the ciphertexts are rejected.

- $Trapdoor$: When the user wants to retrieve some parts of the data that contain a certain keyword, he/she uses his/her private key and the CSP's public key to generate the keyword's trapdoor and sends it to the CSP.

- $Test$: Upon retrieving the trapdoor, the CSP uses his/her private key and the user's public key to check whether the trapdoor is equal to the PEKS ciphertext sent from the data owner. If positive, the CSP sends the ciphertext to the user; otherwise, the CSP denies the request.

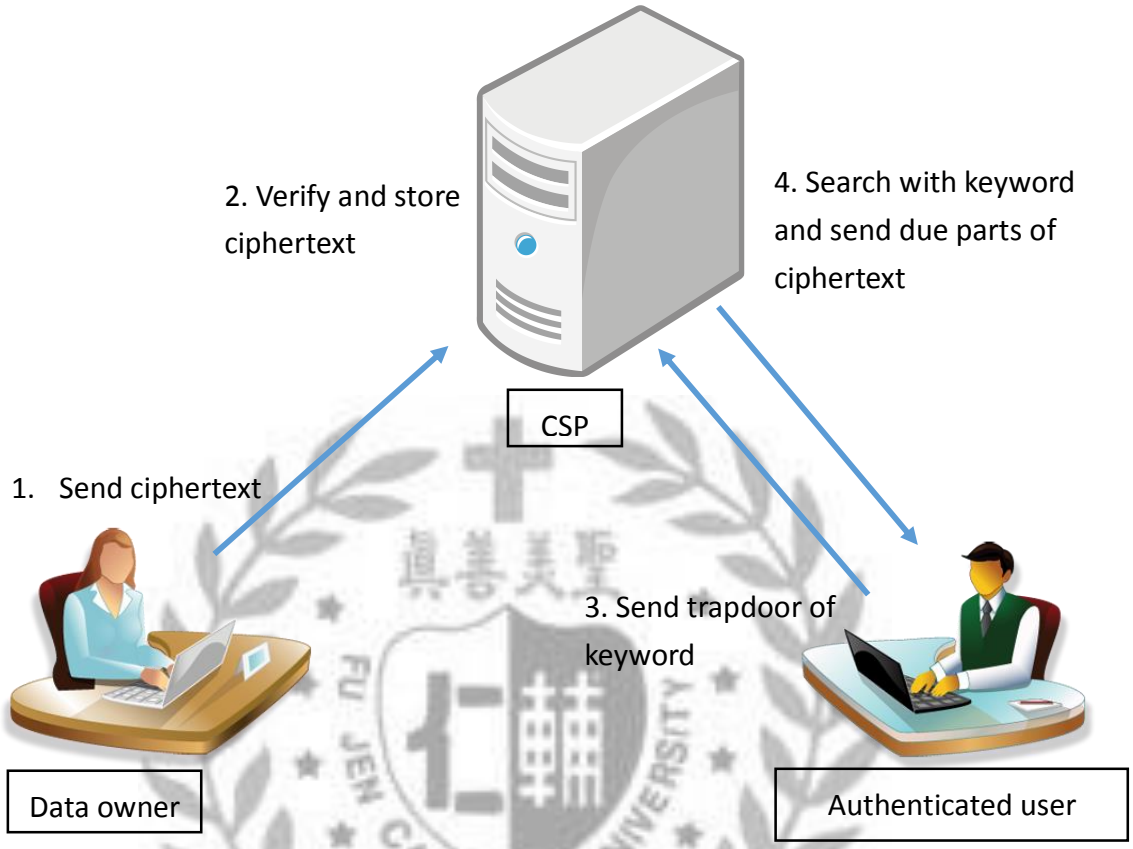

Figure 2 The proposed trapdoor-indistinguishable PEKS

2.3.2 The Proposed Scheme

First of all, Table 1 lists the notations that will be used throughout our scheme. Then, each step that is to be taken in the scheme will be detailed.

Table 1 Notations used in the proposed trapdoor-indistinguishable PEKS

Notations	Descriptions
k	Security parameter, $k \in \mathbb{N}$
ID_O	Identity of data owner
ID_U	Identity of user
pk_s, sk_s	Public key and private key of CSP
pk_O, sk_O	Public key and private key of data owner
pk_U, sk_U	Public key and private key of CSP
w	Keyword
$\oplus$	XOR operation

- $KeyGen_{Param}$: With a security parameter $k \in \mathbb{N}$ input, the system generates a group $\mathbb{G}_1$ of prime order $q \geq 2^k$, a random generator P of $\mathbb{G}_1$, and a bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Three hash functions are produced, namely $H_0: \{0,1\}^* \to Z$, $H_1: \{0,1\}^* \to \mathbb{G}_1$ and $H_2: \mathbb{G}_2 \to \{0,1\}^k$. In addition, d_w denotes a description of the keyword space, and the common parameters are $cp = (q, \mathbb{G}_1, \mathbb{G}_2, e, P, H_0, H_1, H_2, d_w)$.

- $KeyGen_{Server}$: Input the common parameters cp, choose a random number $x \in \mathbb{Z}_q^*$ and $Q \in \mathbb{G}_1^*$, and compute $X = xP$. Output the server's public key $pk_s = (cp, Q, X)$ and private key $sk_s = x$.

- $KeyGen_{User}$: The CSP inputs the common parameters cp and the user's identity ID_U. Then the CSP computes $Y = H_0(ID_U)P$ and $y = xY$ and sends the public key $pk_U = (cp, Y)$ and private key $sk_U = y$ to the user.

- $KeyGen_{Data\ owner}$: The CSP inputs the common parameters cp and the data owner's identity ID_O. Then the CSP computes $A = H_0(ID_O)P$ and $a = xA$ and sends the public key $pk_O = (cp, A)$ and private key $sk_O = a$ to the data owner.

- $PEKS$: The data owner inputs $cp, pk_S, pk_U, sk_O, w,$ and A and chooses a random number $r \in \mathbb{Z}_q^*$. Then the data owner computes R as PEKS ciphertext, where $R = (U, V, t, V_O)$, $U = rP$, $V = rA$, $t = e(H_1(w), U)e(rQ, X)$, $V_O = H_1(t) \oplus H_1(\alpha)$ and $\alpha = e(Q, ra)$. The data owner sends R and the data's ciphertext to the CSP.

- $Verify$: Upon receiving the data, the CSP inputs $pk_S, pk_O, V_O,$ and V and computes $V_O' = H_1(t) \oplus H_1(e(xQ, V))$. The CSP checks whether V_O' is equal to V_O or not. If yes, the CSP stores the received data; otherwise, the CSP rejects the ciphertext.

- $Trapdoor$: The user inputs cp, sk_R, w and Y and chooses a random number $\tilde{a} \in \{0,1\}^*$. Then the user computes $T_{w1} = [y^{-1}H_1(w) + H_1(\tilde{a})] \oplus [H_1(e(Q, \tilde{a}y))]$ and $T_{w2} = yH_1(\tilde{a}) \in \mathbb{G}_1$ and returns TW and $\tilde{a}\,Y$, where $TW = (T_{w1}, T_{w2})$, as a trapdoor for the keyword w.

- $Test$: The CSP inputs $cp, TW, sk_S, R,$ and $\tilde{a}\,Y$ and computes $Tw = T_{w1} \oplus H_1(e(xQ, \tilde{a}Y))$, $S = e(T_{w2}, U)$, $t' = e(xQ, U)^{-1}$ and $T = tt' = e(H_1(w), U)$. If $H_2(e(T_w, V)) = H_2(T \cdot S)$, it returns "Correct"; otherwise, it returns "Incorrect".

2.4 Security and Performance Analysis

In this section, we shall show how our improved PEKS scheme compares with Boneh et al.'s [6], Beak et al.'s [2], Liu et al.'s [44], Rhee et al.'s [52], and Zhao et al.'s [77] in terms of security and performance. Then there will be a BAN logic [11, 72] correctness verification of the proposed scheme, followed by a security analysis.

2.4.1 Comparison

To begin with, let's evaluate the security of the proposed scheme by comparing it with a number of related schemes. Table 2 shows the comparison results, where abbreviations User Auth, Owner Auth, AuthID Pro, Trap Ind and KW Gue are used to represent user authentication, data owner authentication, authorized identity protection, trapdoor indistinguishability and resistance to keyword-guessing attack, respectively. As Table 2 reveals, the proposed scheme does reach a higher security level and is therefore more user-friendly.

Table 2 Security comparison among related schemes

	Boneh et al.'s	Beak et al.'s	Liu et al.'s	Rhee et al.'s	Zhao et al.'s	Our scheme
User Auth	○	○	○	○	○	○
Owner Auth	×	×	×	×	×	○
AuthID Pro	○	○	○	○	○	○
Trap Ind	×	×	×	○	○	○
KW Gue	×	×	×	○	○	○

User Auth : user authentication
Owner Auth : data owner authentication
AuthID Pro : authorized identity protection
Trap Ind: trapdoor indistinguishability
KW Gue : resistance to keyword-guessing attack

Since PEKS ciphertext generation, data owner verification, trapdoor generation, and keyword test are the four major parts of a secure PEKS scheme and should be performed in each session, we only took the computation costs of these four steps into consideration when comparing our improved scheme with the others in terms of performance. Table 3 shows the comparison results, where simplified expressions such as PEKS, Verification, Trapdoor, and Test are used to represent PEKS ciphertext generation, data owner verification, trapdoor generation, and keyword test, respectively. In addition, P denotes a map-to-point hash function operation, E denotes a pairing operation, and M denotes a multiplication operation. As Table 3 reveals, Liu et al.'s PEKS scheme is the most efficient of them all. However, Liu et al.'s scheme, as well as Boneh et al.'s and Beak et al.'s, does not satisfy the trapdoor indistinguishability requirement. On the other hand, although our improved scheme requires more computation in PEKS and Test, in Trapdoor it costs less than Zhao et al.'s scheme. Considering the fact that our improved scheme offers an obviously higher level of security with data owner authentication, trapdoor indistinguishability and resistance to keyword-guessing attack all covered, we find the slight extra computation in PEKS and Test pays off well.

Table 3 Performance comparison among related schemes

	Boneh et al.'s	Beak et al.'s	Liu et al.'s	Rhee et al.'s	Zhao et al.'s	Our scheme
PEKS/SCF-PEKS	$1P + 1E$	$1P + 2E + 1M$	$1P + 1E$	$1P + 1E$	$1P + 2E + 3M$	$3P + 3E + 3M$
Verification	×	×	×	×	×	$2P + 1E$
Trapdoor	$1P$	$1P + 1M$	$1P$	$2P$	$4P + 1E + 3M$	$4P + 1E + 2M$
Test	$1P + 1E$	$1M + 1E$	$1E$	$1P + 1E$	$1P + 4E + 2M$	$1P + 4E + 2M$

P denotes a map-to-point hash function operation.

E denotes a pairing operation.

M denotes a multiplication operation.

2.4.2 Correctness Analysis

The BAN logic is a well-accepted method to analyze the correctness of cryptographic protocols. In this subsection, we will have some notations, goals and assumptions defined and then use the BAN logic [11, 72] to verify the correctness of our scheme.

◆ Notations

Let's take a quick look at the syntax and notations of the BAN logic. First, we have A and B that denote two specific participators, X stands for a formula (statement), and $\overset{K_A}{\mapsto} A$, $\overset{K_B}{\mapsto} B$, K_A^{-1} and K_B^{-1} are A's and B's public key and secret key, respectively. There are some rules as follows [11, 72]:

1. $A|\equiv X$ means A believes that formula X is ture.

2. $A|\equiv B$ means A believes B's action.
3. $A|\Rightarrow X$ means A has complete control over formula X.
4. $A \lhd X$ means A holds or sees formula X.
5. $\#(X)$ means formula X is fresh or has not been used before.
6. $\overset{K_A}{\mapsto} A$ means K is the public key for A and K_A^{-1} is the private key for A.
7. $\frac{Rule\ 1}{Rule\ 2}$ means $Rule\ 2$ can be derived from $Rule\ 1$.

◆ Goals

With three roles involved, namely the data owner ($Owner$), the cloud service provider (CSP) and the user ($User$), in our scheme, there are two goals to be achieved: in the data owner verification process, CSP is to believe that $Owner$ has the private key to create the PEKS ciphertext; in the keyword search process, CSP is to believe that $User$ has the private key to create the trapdoor of the keyword. These two goals of our scheme can be rephrased in the language of the BAN logic as follows:

$G1.\ CSP|\equiv Owner \lhd K_{owner}^{-1}$

$G2.\ CSP|\equiv User \lhd K_{User}^{-1}$

◆ Assumptions

To analyze the correctness of our scheme, there are some assumptions as follows:

$A1.\ CSP|\equiv \overset{K_{owner}}{\mapsto} Owner$

$A2.\ CSP|\equiv \overset{K_{User}}{\mapsto} User$

$A3.\ Owner|\equiv \overset{K_{CSP}}{\mapsto} CSP$

$A4.\ Owner|\equiv \overset{K_{User}}{\mapsto} User$

$A5.\ User|\equiv \overset{K_{CSP}}{\mapsto} CSP$

$A6.\ CSP|\Rightarrow K_{CSP}^{-1}$

$A7.\ CSP|\Rightarrow K_{owner}^{-1}$

$A8.\ CSP|\Rightarrow K_{User}^{-1}$

◆ Verification of The Data Owner

With the goals and assumptions confirmed, now we can analyze the correctness of our data owner verification process with the BAN logic. The details are as follows:

Message 1: $Owner \longrightarrow CSP: R = (U, V, t, V_O, rA)$

$V1.\ CSP \lhd R$

$V2.\ \frac{CSP \lhd R, CSP \lhd K_{CSP}^{-1}, CSP \lhd rA}{CSP \lhd V_O'}$

$V3.\ \frac{CSP| \Rightarrow K_{CSP}^{-1}, CSP \lhd V_O'}{CSP| \equiv V_O}$

$V4.\ \frac{CSP| \equiv V_O}{CSP| \equiv Owner \lhd K_{owner}^{-1}}$

Finally, we can infer from formula $V4$ that our scheme does achieve the goal we set up. In the end, CSP does believe that $Owner$ has the private key to create the PEKS ciphertext.

◆ Verification of The User

Now we analyze the correctness of our user verification process with the BAN logic as follows:

Message 1: $User \longrightarrow CSP: TW = (T_{w1}, T_{w2})$ and $\tilde{a}\ Y$.

$V1.\ CSP \lhd TW, \tilde{a}\ H_1(ID_U)$

$V2.\ \frac{CSP \lhd T_{W1}, CSP \lhd K_{CSP}^{-1}, CSP \lhd \tilde{a}\ Y}{CSP \lhd T_W}$

$V3.\ \frac{CSP \lhd (T_W, T_{W1}, T_{W2}, U, t)}{CSP \lhd (S, t', T)}$

$V4.\ \frac{CSP| \equiv (T_W, V, T), CSP| \equiv K_{CSP}^{-1}}{CSP| \equiv S}$

$V5.\ \frac{CSP| \equiv S}{CSP| \equiv T_{w2}}$

$V6.\ \frac{CSP|\equiv T_{w2}}{CSP|\equiv User \lhd K_{User}^{-1}}$

Finally, we can infer from formula $V6$ that our scheme does achieve the goal we set up. In the end, CSP does believe that $User$ holds the private key to create the trapdoor of the keyword.

2.4.3 Security Analysis

In this subsection, we shall analyze the proposed scheme to see if it satisfies the following security requirements:

(1) Only the CSP can use the keyword created by the data owner to do keyword search.

If an attacker captures the PEKS ciphertext $R = (U, V, t, V_O, rH_1(ID))$ through the communication channel between the data owner and the CSP and captures the trapdoor of keyword $TW = (T_{w1}, T_{w2})$ through the communication channel between the user and the CSP, he/she still cannot compute $H_1(e(xQ, \tilde{a}H_1(ID_U)P))$, $H_1(e(Q, \tilde{a}y)$ and $t' = e(xQ, U)^{-1}$ from the captured (*R*, *TW*) because to do that is as difficult as to solve the BDH problem. In other words, only the CSP, who owns the private key, can determine whether the trapdoor of the keyword is truly sent from the user by confirming it against what the data owner set up.

(2) The trapdoor of the keyword is indistinguishable.

In our scheme, since the random string $\tilde{a}$ chosen by the user differs from session to session, a keyword cannot generate the same trapdoor a second time. In other words, the trapdoor of the same keyword will be changed in every session. This way, even if an attacker captures the trapdoor in a given session,

the captured trapdoor still cannot be used to come by the keyword in any following session.

(3) The CSP can determine that the PEKS ciphertext is sent by an authorized data owner.

Only an authorized data owner have A and a, both generated by the CSP. The authorized data owner can use A and a to generate the PEKS ciphertext and authentication information. Upon receiving the message, the CSP can utilize its private key to determine whether the PEKS ciphertext is truly sent by the authorized data owner.

(4) Only the CSP can verify the user's identity.

Even if an attacker captures the data delivered through the communication channel between the user and the CSP, there is still no way the attacker can analyze the information and get to know the user's identity. Only the CSP can verify the user's identity by testing the received data against the values the CSP holds.

Chapter 3 Hierarchical Conditional Proxy Re-encryption Scheme

As cloud technologies thrive, researches in the field of cloud storage have switched their focus from encryption-decryption techniques that help data owners protect their privacy and data confidentiality to the application of searching techniques on encrypted data while maintaining high level security and privacy of outsource data. To begin with, Song et al. offered some practical techniques for searches on encrypted data. After that, Weng et al. presented their conditional proxy re-encryption scheme where the data owner can decide which ciphertext satisfies a certain keyword condition set and then can have the retrieved data re-encrypted by the semi-trusted proxy server. The basic concepts of the above schemes are indeed quite innovative and do lead the way towards the solutions to the major practical cloud storage application problems; however, of all the researches that follow, none has had both searching on encrypted data and conditional proxy re-encryption combined. In this paper, we propose a new scheme for cloud storage services that integrates keyword search with conditional proxy re-encryption. This say, with a newly added keyword or new proxy, the cloud service provider is able to generate a hierarchical key. As far as data security is concerned, our scheme provides proven data owner authentication, re-delegation, and chosen-ciphertext security. The superior performance of the proposed scheme has been established by comparing it with related works, and our security analysis as well as BAN logic correctness check also offered solid proof that the new scheme is both practical and robust.

3.1 Preliminaries

Nowadays, due to the amazing mobility and convenience the thriving Internet and related wireless technologies have brought, more and more people have fallen into the habit of keeping their data in cloud storage instead using traditional portable storage devices such as USB flash drives. As people get more and more dependent on cloud storage services, cloud servers have to handle larger and larger quantities of data, sensitive information included. In other words, how to provide satisfactory mobility and convenience without sacrificing data security and confidentiality in cloud environment is the main concern. Currently, when a data owner wants to store some sensitive data in cloud storage, he/she needs to encrypt the data before uploading them to the cloud storage so as to maintain data secrecy. After uploading the data to cloud storage, he/she can then access them wherever Internet connection is available; in other words, he/she can either access the data at home or office where cabled connection is ready, or he/she can use a mobile device such as a smart phone or tab with Wi-Fi when he/she is out somewhere. Of course there can also be cases where a person (the data owner) has the data uploaded to the cloud storage and then another person (the authorized data user) accesses the data stored. However, oftentimes a data owner can have tons and tons of data uploaded to cloud storage. How can he/she access a certain part or certain parts of the data stored in cloud, then? In the past, there were two ways to get the job done [23]:

1. The user downloads all his/her data from cloud. Since the data are in encrypted form, after the downloading, the user must decrypt all the data. Now the data are in plaintext format, and the user can finally search through them and pick

out the part or parts he/she desires. Just as it appears, this whole process is a lot of trouble for the user.

2. The user sends his/her secret key to the cloud server. With the user's secret key, the cloud server decrypts all the data uploaded by the user and finds the part or parts of the data that the user desires. In this design, the user has no choice but to totally trust the cloud server, which can be a serious security problem if the cloud server has malicious purposes.

To deal with the above problems, Song et al. [59] were the first to raise the concept of searching on encrypted data and named it the method of keyword search. In their method, the data owner can encrypt the data with some keywords, and the user can later access a certain part of the encrypted data that contains a specified keyword without having to download all the encrypted data, decrypt them all, and then do the searching. This way, the user can easily retrieve the part of the data that is needed without leaking any information. Here is a scenario to illustrate the concept of keyword search on encrypted data: Suppose Alice wants to store some data in cloud storage. She generates the ciphertext of the data. To make the data easy to access, Alice also sets the keyword "October" for the data. After generating the ciphertext of the keyword "October", Alice sends all the encrypted data to cloud storage. Later, when Bob, an authorized user, wants to retrieve the data that contains the keyword "October", he first generates the trapdoor of the keyword "October" and then sends this trapdoor to the cloud server as an access request. Upon receiving the request, the cloud server searches through the encrypted data and finds the data that contains the keyword "October" without decrypting the ciphertext. After that, the cloud server returns the corresponding ciphertext to Bob.

However, in real-world practice, there are always risks when the cloud user has to fully trust the cloud service provider. In other words, there is no way the data owner

should hand his/her private key over to the server. To solve this problem, Blaze et al. [5] presented the concept of proxy re-encryption which allows the delegated semi-trusted server to re-encrypt the ciphertext by using a re-encryption key without learning any information about the plaintext. There is a scenario to illustrate the concept of proxy re-encryption: Alice uses her public key to encrypt the data and uploads the encrypted data to the server. Alice has some data for Bob, but she does not want Bob to have her private key. Without Alice's private key, Bob cannot decrypt the data. In order for Bob to be able to decrypt the ciphertext by using his own private key, Alice exploits her pubic key and Bob's public key to generate a new key for the server called a re-encryption key. With this key, the server can re-encrypt the ciphertext without getting the plaintext. Then Bob can use his private key to decrypt the ciphertext without getting Alice's private key.

Later in 2009, the notion of conditional proxy re-encryption was brought up by Weng et al. [71]. As the name suggests, by applying conditional proxy re-encryption, the data owner is enabled to decide which ciphertext satisfies a certain keyword condition set that can be re-encrypted by the proxy. Then, in 2012, Fang et al. took a step further and proposed a hierarchical conditional proxy re-encryption scheme [24]. Inspired by Fang et al., in this paper, we shall propose a searchable hierarchical conditional proxy re-encryption scheme we have designed for cloud storage. As the name reveals, the aim of our new scheme is to combine keyword search and conditional proxy re-encryption. Our scheme has the following properties:

1. Searching data without decrypting the ciphertext

   The CSP (Cloud Server Provider) does not need to decrypt the ciphertext; all the CSP does with the data in cloud storage is search on the encrypted data with a keyword in encrypted format to find the data the user needs.

2. User authentication

   The CSP can confirm the user's real identity with the trapdoor sent from the user.

3. Data owner authentication

   The CSP can utilize the ciphertext uploaded by the data owner and some public parameters to verify the legality of the data owner's identity and the ciphertext.

4. Re-delegation

   The CSP can utilize its re-encryption key to derive the sub-re-encryption key for the newly added keyword or for their children.

5. Chosen-ciphertext security

   Our scheme is based on Fang et al.'s design [24]; by the same token, our scheme provides the same level of chosen-ciphertext security on the first and the second ciphertext.

3.2 Related Works

In this section, some related works dealing with keyword search on encryption data as well as some proxy re-encryption and conditional proxy re-encryption schemes will be quickly reviewed.

3.2.1 Keyword Search on Encrypted Data

To make searching on encrypted data possible, Song et al. [59] first proposed a secure keyword search scheme in 2000. After that, many researchers have focused on how to design secure, efficient schemes for searches on encrypted data [2, 6, 9, 13, 26, 31, 36, 39, 41, 43, 44, 49, 51, 58, 73, 77]. In 2004, Boneh et al. [6] proposed the idea of public key encryption with keyword search (PEKS), which allows the server to

search through the stored data for the parts that contain certain keywords without decrypting the ciphertext. Golle et al. [26] proposed a conjunctive keyword search mechanism that allows the user to search with a conjunction of multiple keywords. Later, Park et al. [49] proposed an efficient public encryption scheme with conjunctive keyword search. On the other hand, to avoid the use of pairing operations, in 2006, Khader [36] proposed a public key encryption scheme with keyword search based on K-Resilient IBE. In 2008, Baek et al. [2] extended the PEKS into a secure channel free public key encryption scheme with keyword search (SCF-PEKS), which does not include any secure channel between the user and the server. Then, in 2009, Liu et al. [43] proposed an efficient privacy preserving keyword search (EPPKS) scheme to improve the performance of PEKS, while Rhee et al. [51] brought up the concept of trapdoor indistinguishability and proposed a new scheme to mend the weakness they found in Baek et al.'s SCF-PEKS. In 2012, Liu et al. [44] improved Liu et al.'s EPPKS and proposed a new keyword search scheme called Secure and Privacy-preserving Keyword Search (SPKS) that can do searches on encrypted data with the server in charge of the re-encryption of the ciphertext.

3.2.2 Proxy Re-encryption

A proxy re-encryption (PRE) scheme allows the delegated semi-trusted server to re-encrypt the ciphertext by using its re-encryption key without learning any information about the plaintext. The concept of proxy re-encryption was proposed by Blaze et al. [5] in 1998. Later on, the pairing operation was commonly used in schemes of this kind [1, 12, 20, 27, 42, 70]. In 2007, Ateniese et al. proposed an identity-based proxy re-encryption scheme where the ciphertext can be transformed from one identity to another [1]. In addition, Chu and Tzeng [20] also proposed an identity-based proxy

re-encryption scheme without random oracles. Finally, due to the fact that the pairing operation consumes too much communication resources, in recent years, some PRE schemes have been proposed to avoid the use of the pairing operation [18, 22, 47, 53].

3.2.3 Conditional Proxy Re-encryption

Firstly, type-based proxy re-encryption (TB-PRE) is a design where the data owner can categorize his/her ciphertext into different subsets and then delegate the decryption right of each subset to a specific delegator. In 2008, Tang [61] first proposed the construction of TB-PRE, providing fine-grained delegation and enabling the semi-trusted server to re-encrypt ciphertext of a specific type by using a re-encryption key. Since then, quite a big portion of research endeavors in the field of study have been dedicated to the development of TB-PRE schemes [21, 24, 25, 56, 63, 71, 72]. Among the schemes, Seo et al.'s TB-PRE scheme offered proven security against the standard-model chosen ciphertext attack and achieved proxy invisibility [56]. Since by definition TB-PRE means that the data owner can categorize the ciphertext into different subsets, TB-PRE is also referred to as conditional proxy re-encryption (C-PRE), where a condition is equivalent to a type [56]. Weng et al. [71] presented a kind of conditional proxy re-encryption where the data owner can assign some specific ciphertext to match a certain keyword condition set that can be re-encrypted by the semi-trusted proxy server. Later, Weng et al. [72] pointed out that Weng et al.'s scheme [721] had failed to achieve chosen ciphertext attack security (CCA-security), and so they proposed a new C-PRE scheme to fix that problem. In addition, Fang et al. [25] also proposed an anonymous conditional proxy re-encryption scheme without random oracle. Chu et al. [21] presented a conditional proxy broadcast re-encryption scheme where the proxy can delegate decryption rights to a set of users at a time. In 2010, Vivek et al. [63] improved

the performance of Weng et al.'s [72] C-PRE scheme and proposed a more efficient construction for C-PRE. In 2012, Fang et al. proposed a hierarchical conditional proxy re-encryption (HC-PRE) scheme that enhanced the concept of C-PRE by allowing more general re-encryption key delegation patterns [24].

To this day, no scheme proposed has had both ideas of searching on encrypted data and conditional proxy re-encryption combined. Inspired by Fang et al. [24], in this paper, we propose a new scheme that puts together keyword search and conditional proxy re-encryption.

3.3 Review Weng et al.'s Scheme

In this section, we shall review bilinear pairing [7], give some complexity assumptions in our scheme, and then introduce the idea of hierarchical conditional proxy re-encryption [72].

3.3.1 Bilinear Pairing

Let $\mathbb{G}_1$and $\mathbb{G}_2$ be two cyclic group with prime order p, and g is the generator of group $\mathbb{G}_1$. Suppose we have $a, b \in \mathbb{Z}_q$ and a bilinear map e: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Then there are some notable properties as follows [7]:

- Bilinearity for all $a, b \in \mathbb{Z}_q$and$P, Q \in \mathbb{G}_1$,$e(aP, bQ) = e(P, Q)^{ab}$.
- Computability. There in always an efficient polynomial time algorithm to compute $e(P, Q) \in \mathbb{G}_2$, for any $P, Q \in \mathbb{G}_1$.
- Non-degeneration. There is always such a pair of P and $Q \in \mathbb{G}_1$ that satisfies $e(P, Q) \neq 1$.

3.3.2 Hierarchical Conditional Proxy Re-encryption

Here is the hierarchical conditional proxy re-encryption design proposed by Weng et al. in 2009 [72]. In their scheme, there are eight algorithms: setup, key generation, re-encryption key generation, level 2 encryption, level 1 encryption, re-encryption, level 2 decryption, and level 1 decryption. Figure 3 gives a rough idea of how the system works, and the algorithms are as follows:

- Setup: The setup algorithm is executed by a trusted party with the input being the security parameter 1^K and the output the global parameters GP.

- KeyGen: The key generation algorithm produces the public key pk_i and secret key sk_i for the user i.

- RKeyGen: The re-encryption key generation algorithm takes the secret key sk_i, the conditional keyword w, and the other public key pk_j as input and then outputs the re-encryption key $rk_{i\to j}^{w}$.

- Enc2: Level 2 encryption algorithm intakes the public key pk, the plaintext $m \in \mathcal{M}$ and the conditional keyword w and then outputs level 2 ciphertext CT. Here $\mathcal{M}$ is the message space.

- Enc1: Level 1 encryption algorithm takes the public key pk and the plaintext $m \in \mathcal{M}$ as input and then outputs level 1 ciphertext CT. Notice that this ciphertext cannot be encrypted by any other user.

- ReEnc: The re-encryption algorithm intakes the second ciphertext CT and the re-encryption key $rk_{i\rightarrow j}^{w}$.

- Dec2: Level 2 decryption algorithm takes the second ciphertext CT and the secret key sk as input and then outputs the message m.

- Dec1: Level 1 decryption algorithm intakes the first ciphertext CT and the secret key sk and then outputs the message m.

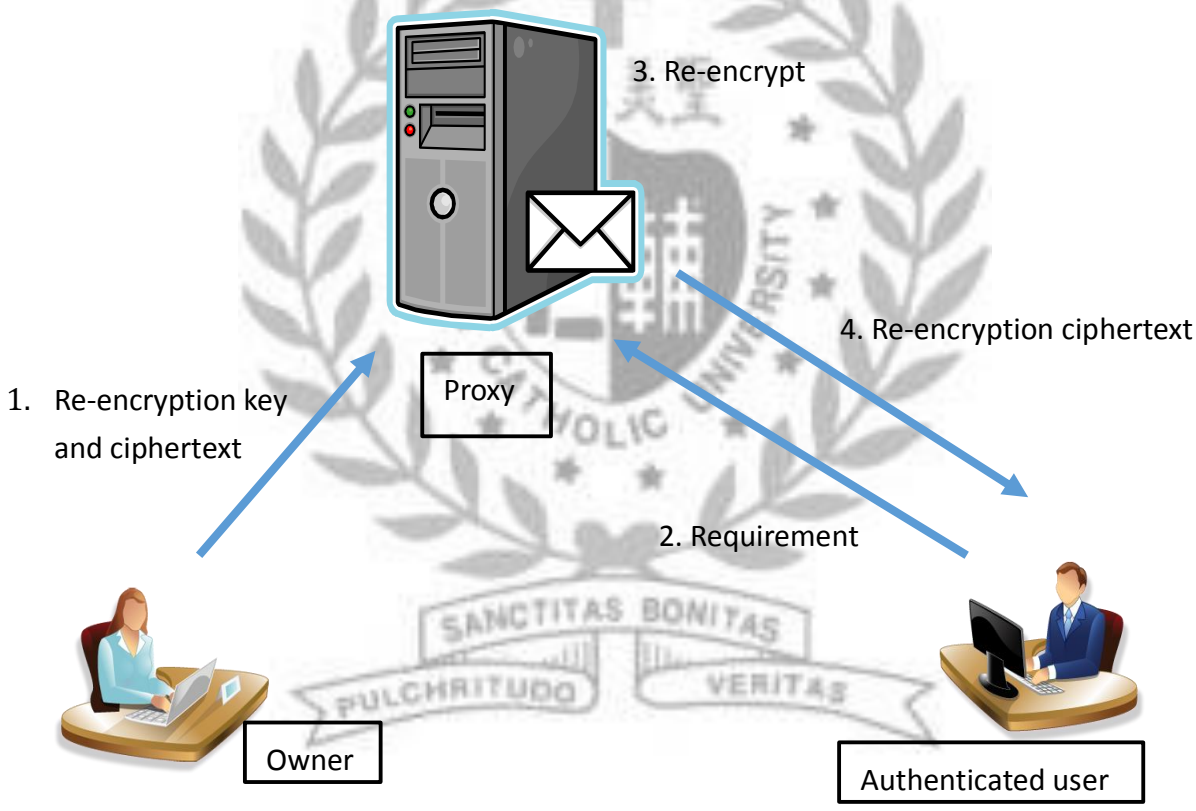

Figure 3 Hierarchical conditional proxy re-encryption

3.4 New Scheme

In this section, we shall present our searchable hierarchical conditional proxy re-encryption scheme. We will first illustrate the framework of our scheme and then give detailed descriptions to all the phases of our scheme.

3.4.1 Framework

In this subsection, we shall first introduce the participants in our scheme and then the phases. There are four kinds of participants in our scheme: the trusted third party (TTP), the cloud service provider (CSP), the data owner, and the users. The role each participant plays is shown as follows.

1. Trusted third party (TTP): The trusted third party is responsible for generating the public key and the secret key for the user and the data owner and also generating the re-encryption key for the cloud server provider.

2. Cloud service provider (CSP): The function of CSP is to accept and store the ciphertext sent by the data owner. Upon receiving the retrieval request from the user, CSP searches through the stored data and finds what the user wants. Besides that, CSP is able to re-encrypt the ciphertext and uses a re-encryption key to generate a hierarchical key for a newly added keyword.

3. Data owner: The data owner generates ciphertext on two different levels. One does not contain the keyword vector, while the other contains the keyword vector set by the data owner.

4. Users: When a user wants to retrieve some data that contains a certain keyword, the user needs to generate the trapdoor of the keyword and then send it to CSP as a request. Then, when the user receives the re-encrypted ciphertext that CSP returns, he/she can use his/her secret key to decrypt it.

There are 11 phases in our scheme: setup, key generation, re-encryption key generation, level 1 encryption, level 2 encryption, verification, trapdoor generation, keyword searching, re-encryption, level 1 decryption, and level 2 decryption. The flowchart of our scheme is shown in Figure 4, and the function of each phase is as follows:

- Setup: In this phase, the security parameter λ is the input, the bilinear map is set, and then the system public parameters are outputted.

- KeyGen: In this phase, the system public parameters are inputted, and the public key and the secret key for the data owner and the user are outputted.

- Re-keyGen: In this phase, the inputs are the user's secret key, the data owner's secret key and a conditional keyword vector, and then the output is the re-encryption key for CSP. When a new keyword is added to the conditional keyword vector, CSP can use the current re-encryption key to generate a new re-encryption key. This is called hierarchical key derivation.

- Enc1: In order to have the message encrypted, the data owner inputs the message along with his/her public key and then gets the first level ciphertext for CSP.

- Enc2: To encrypt the message with a conditional keyword vector, the data owner inputs the message along with his/her public key and a conditional keyword vector. The output is the second level ciphertext for CSP.

- Verify: Upon receiving the ciphertext, CSP determines whether the ciphertext is truly sent by the data owner and has not been tampered by a malicious attacker.

- Trapdoor: In order to retrieve the data which contains a certain keyword, the user generates the trapdoor of the keyword vector and then sends it to CSP.

- Search: To search for the data the user requests, CSP inputs the ciphertext, the user's public key and the trapdoor.

- ReEnc: When CSP finds the data that the user requests, CSP uses the re-encryption key to encrypt the ciphertext.

- Dec1: The user inputs his/her secret key and the first level ciphertext to decrypt the ciphertext.

- Dec2: The user inputs his/her secret key and the second level ciphertext to decrypt the ciphertext.

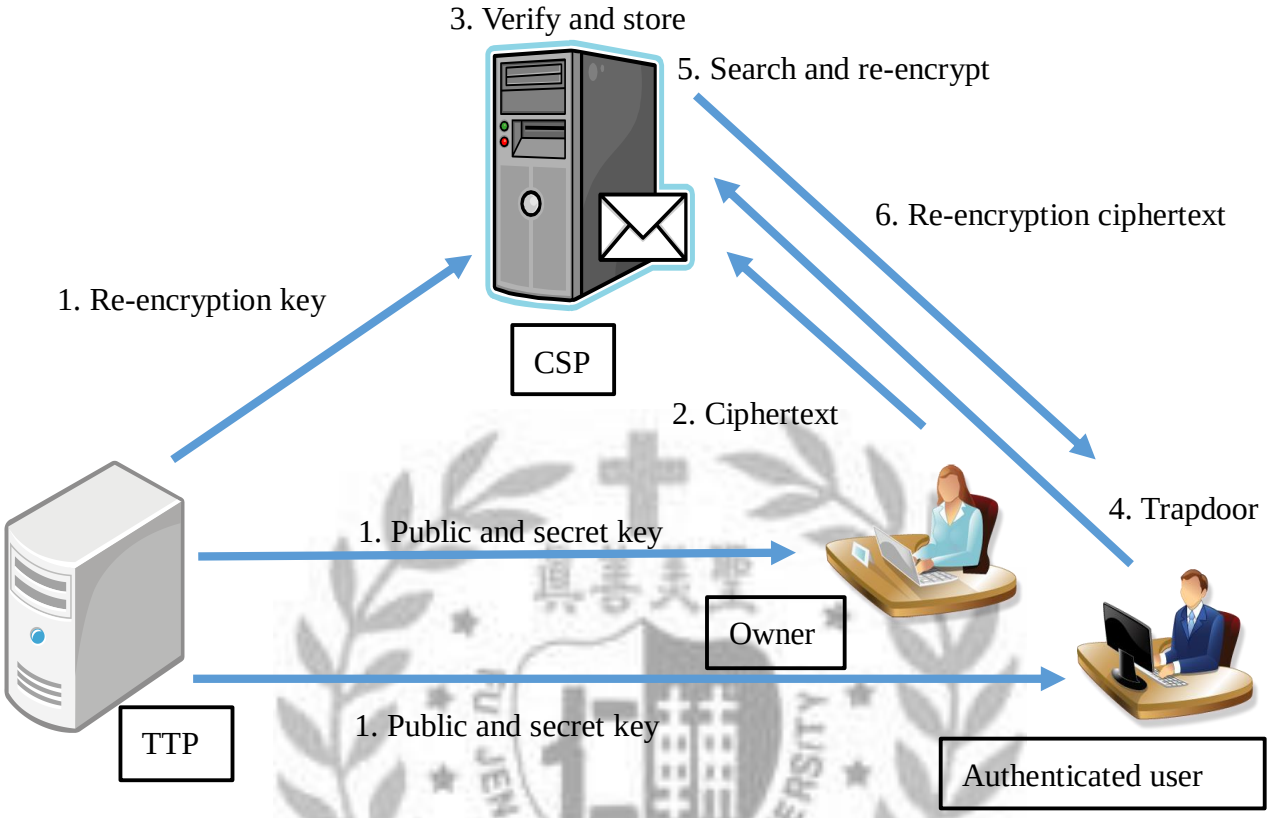

Figure 4 Searchable hierarchical conditional proxy re-encryption

3.4.2 Searchable Hierarchical Conditional Proxy Re-encryption

In this subsection, we look into the details of the phases in our scheme. Table 4 lists the notations used in our scheme.

Table 4 Notations used in searchable hierarchical conditional PRE

Notations	Descriptions
p	A prime order
g	A generator of G_1
G_1, G_2	Multiplicative cyclic groups of prime order p
e	Bilinear map $e: G_1 \times G_1 \rightarrow G_2$
L	The maximum length of keyword vector
m	The message, $m \in \mathcal{M}$
$\oplus$	XOR operation

- Setup: With a security parameter λ inputted, set (p, g, G_1, G_2, e) as bilinear map parameters. Then, $\mathcal{M} = \{0,1\}^{k_1}$ is set as the message space, and there are four one-way hash functions $H_1: \{0,1\}^* \rightarrow Z_p^*, H_2: G_2 \rightarrow \{0,1\}^{k_1}, H_3: \{0,1\}^* \rightarrow G_1^*$, and $H_4: \{0,1\}^* \rightarrow Z_p^*$. Let the conditional keyword vector be $W = (w_1, w_2, \dots, w_k) \in \{0,1\}^*$, where k is the length of W. Generate the random numbers $g_1, g_2, h_1, h_2, \dots, h_L \in G_1$. The system public parameters are $(p, g, G_1, G_2, e, g_1, g_2, h_1, \dots, h_L, k_1, L, H_1, H_2, H_3, H_4)$.

- KeyGen: Generate a random number $x_i \in Z_p^*$ for user i and then compute $X_i = g^{x_i}$. Set the public key as $pk_i = X_i$ and secret key as $sk_i = x_i$ for user i.

- Re-keyGen: Given the data owner's secret key sk_i, the conditional keyword vector $W = (w_1, w_2, \dots, w_k)$, and the user's secret key sk_j, select a random

number $r \in Z_p^*$ and compute $a_0 = g_2^{x_i - x_j}\left(\prod_{l=1}^{k} h_l^{H_4(pk_i, w_l)} g_1\right)_{l\in\{k+1,\ldots,L\}}^{r}$, $a_2 = g^r$, and $b = (b_l = h_l^r)_{l\in\{k+1,\ldots,L\}}$. The re-encryption key for CSP is $rk_{i,W,j} = (a_0, a_1, b)$. When CSP needs to generate a new re-encryption key for a new keyword vector $W = (w_1, w_2, \ldots, w_k, w_{k+1})$, CSP picks a random number $t \in Z_p^*$ and then computes $a_0{}' = a_0 b_{k+1}^{H_4(pk_i, w_{k+1})}\left(\prod_{l=1}^{k+1} h_l^{H_4(pk_i, w_l)} g_1\right)_{l\in\{k+2,\ldots,L\}}^{t}$, $a_1{}' = a_1 g^t$, and $b' = (b_l = h_l^t)_{l\in\{k+2,\ldots,L\}}$. The hierarchical re-encryption key is $rk_{i,W_{k+1},j} = (a_0{}', a_1{}', b')$, which is properly distributed to W_{k+1} for $r' = r + t$.

- Enc1: Data owner chooses a random number $R \in G_2^*$ and then computes $s = H_1(m, R), B = g^s, D = e(X_i, g_2)^s R$, and $E = m \oplus H_2(R)$. The first level ciphertext is $CT_i = (B, D, E)$.

- Enc2: To encrypt the message with the conditional keyword vector $W = (w_1, w_2, \ldots, w_k)$, data owner chooses $R \in G_2^*$ and then computes $s = H_1(m, R), B = g^s, C = \left(\prod_{l=1}^{k} h_l^{H_4(pk_i, w_l)} g_1\right)^s, D = e(X_i, g_2)^s R, E = m \oplus H_2(R)$, and $F = H_3(B, C, D, E)^s$. The second level ciphertext is $CT_i = (B, C, D, E, F)$.

- Verify: After receiving the ciphertext, CSP checks out $e\left(\prod_{l=1}^{k} h_l^{H_4(pk_i, w_l)} g_1, B\right) =? e(C, g)$ and $e(H_3(B, C, D, E), B) =? e(F, g)$. If both check out, CSP accepts and stores the ciphertext.

- Trapdoor: When the user wants to retrieve a part of the stored data that contains the conditional keyword vector $W = (w_1, w_2, \ldots, w_k)$, he/she

computes the trapdoor of the conditional keyword vector as $T_{w_j} = \left(\prod_{l=1}^{k} h_l^{H_4(X_i, w_l)} g_1\right)^{x_j}$ and then sends it to CSP.

- Test: When receiving the trapdoor from the user, CSP tests to see whether $e(B, T_{w_j})$ is equal to $e(pk_j, C)$ or not. If the result is positive, CSP re-encrypts the ciphertext and then sends it to the user.

- ReEnc: After finding the data that the user requests, CSP re-encrypts the ciphertext by computing $D' = \frac{e(a_1, C)}{e(a_0, B)} \cdot D$. The re-encrypted ciphertext, namely $CT_j = (B, D', E)$, is then sent to the user.

- Dec1: To decrypt the re-encrypted first level ciphertext $CT_j = (B, D', E)$, the user uses his/her secret key sk_j and computes $R = \frac{D'}{e(B, g_2)^{x_j}}, m = E \oplus H_2(R)$, and $s = H_1(m, R)$. After computing R, m and s, the user checks $B =? g^s$. If it checks out, then the message m is returned.

- Dec2: To decrypt the re-encrypted second level ciphertext $CT_j = (B, C, D', E, F)$ containing the conditional keyword vector, the user uses his/her secret key sk_j and computes $= \frac{D'}{e(B, g_2)^{x_j}}, m = E \oplus H_2(R)$, and $s = H_1(m, R)$. After computing R, m and s, the user checks $B =? g^s$, $C = ? \left(\prod_{l=1}^{k} h_l^{H_4(pk_i, w_l)} g_1\right)^s$, and $F =? H_3(B, C, D', E)^s$. If all check out, then the message m is returned.

3.5 Security and Function Analysis

In this section, we shall first show how our new scheme compares with Zhao et al.'s [77], Liu et al.'s [44], Fang et al.'s [24], and Seo et al.'s scheme [56] in terms of function as well as performance. Then, we will analyze the security of our scheme and confirm the correctness with a BAN logic [11, 73] check.

3.5.1 Comparisons

In this subsection, we compare the functions and performance of our scheme with those of Zhao et al.'s, Liu et al.'s, Fang et al.'s, and Seo et al.'s scheme. Of all the schemes compared, Zhao et al.'s, and Liu et al.'s focus on secure keyword search , while Fang et al.'s, and Seo et al.'s focus on conditional proxy re-encryption.

3.5.1.1 Function Comparison

Before looking into the comparison results, let's define some abbreviations we use. Expressions such as AuthID Pro, User Auth, Owner Auth, Searching, and P-Re are used to indicate authorized identity protection, user authentication, data owner authentication, search on encrypted data, and proxy re-encryption, respectively. The comparison results are given in Table 5. As the table reveals, Zhao et al.'s, and Liu et al.'s both fall short of offering data owner authentication, which means vulnerability to the modification attack where the attacker sends fake ciphertext to CSP and the user never receives the data he/she requests. On the other hand, although Fang et al.'s and Seo et al.'s are under the protection of data owner authentication, they are both incapable of supporting searches on encrypted data. In contrast, our scheme offers both data owner authentication but also searching on encrypted data.

Table 5 Function comparison of our scheme and other schemes

	AuthID Pro	User Auth	Owner Auth	Searching	P-Re
Zhao et al.'s	v	v	x	v	x
Liu et al.'s	v	v	x	v	v
Fang et al.'s	v	v	v	x	v
Seo et al.'s	v	v	v	x	v
Our scheme	v	v	v	v	v

AuthID Pro : authorized identity protection
User Auth : user authentication
Owner Auth : data owner authentication
Searching : search on encrypted data
P-Re : proxy re-encryption

3.5.1.2 Performance Comparison

For the performance comparison, we use Encrypt, Trapdoor, Verification, Test, and Re-encryption as abbreviations for conditional encryption, trapdoor generation, verification of data owner, keyword test, and proxy re-encryption, respectively. Note that conditional encryption includes conditional encryption, type-based encryption, and keyword encryption. In addition, we define P as a map-to-point hash function operation, E as a pairing operation, and M as a multiplication operation in G_1. The performance comparison results are given in Table 6.

Table 6 Performance comparison of our scheme and other schemes

	Encrypt	Trapdoor	Verification	Test	Re-encryption
Zhao et al.'s	$1P + 2E + 3M$	$4P + 1E + 3M$	$-$	$1P + 4E + 2M$	$-$
Liu et al.'s	$1P + 1E$	$1P$	$-$	$1E$	$1P + 2E + 2M$
Fang et al.'s	$3P + 1E + 3M$	$-$	$2E + 1M$	$-$	$2E$
Seo et al.'s	$1E + 4M$	$-$	$1E + 2M$	$-$	$1M$
Our scheme	$3P + 1E + 3M$	$0P + 0E + 0M$	$2E + 1M$	$1E$	$2E$

P denotes a map-to-point hash function operation.

E denotes a pairing operation.

M denotes a multiplication operation in G_1.

3.5.2 Security Analysis

In this subsection, we analyze the security of our scheme.

1. CSP can verify the data owner's identity.

   To determine the legitimacy of the data owner, CSP utilizes the ciphertext B, C, D, E, F, data owner's public key, and the keyword vector to verify the data owner's identity. Because the data owner uses the public key to generate the ciphertext, CSP can confirm the data owner's identity by checking out the ciphertext.

2. CSP can verify that the sender of the ciphertext is an authorized data owner.

   To avoid mistakenly accepting tampered ciphertext from a malicious attacker, CSP must check the integrity of the ciphertext. When CSP verifies the data owner's identity, the ciphertext is examined at the same time. If any part of the ciphertext is tampered, it cannot pass the verification.

3. CSP can verify the user's identity.

   Upon receiving the trapdoor of the keyword vector from a user as a searching request, CSP must check the user's identity to make sure he/she is properly authorized. CSP utilizes the ciphertext B, C, D, E, F, the data owner's public key, the user's pubic key and the keyword vector to verify the user's identity. Only a legitimate user owns the secret key that can be used to generate the trapdoor. In fact, CSP can verify the user's identity and search for the data the user requests as the same time.

4. The user can verify whether the ciphertext is tampered.

   Upon receiving the re-encrypted ciphertext, the user verifies the integrity of the re-encrypted ciphertext to determine whether it has been tampered by a malicious attacker. The user exploits his/her secret key to decrypt the re-encrypted ciphertext. After decrypting the re-encrypted ciphertext, the user exploits the re-encrypted ciphertext and the plaintext to check the integrity of the ciphertext. Only CSP has the re-encryption key and thus can have the ciphertext re-encrypted, and only the legitimate user can exploit his/her secret key to recover the integral plaintext.

5. Our scheme can achieve chosen-ciphertext security.

   Based on Fang et al.'s design [15], our scheme inherits the chosen-ciphertext security on the first and the second ciphertext.

3.5.3 Correctness Analysis

In this subsection, we use the BAN logic [11, 73] to check the correctness of the data owner verification, user verification, and ciphertext verification of our scheme. The BAN logic is a well-accepted method to analyze the correctness of cryptographic protocols. Before applying the BAN logic, let's define some notations, goals and assumptions as follows.

- **Notations**

Here we deal with the syntax and notations of the BAN logic. Assume that A and B are some specific participators, and X is the formula (statement). The basic rules of language are as follows [11, 73]:

8. $A|\equiv X$ means A believes that formula X is ture.
9. $A|\equiv B$ means A believes B's action.
10. $A|\Longrightarrow X$ means A has complete control over formula X.
11. $A \lhd X$ means A holds or sees formula X.
12. $\#(X)$ means formula X is fresh and has not been used before.
13. $\overset{K_A}{\longmapsto} A$ means K_A is the public key for A and K_A^{-1} is the private key for A.
14. $\frac{Rule\ 1}{Rule\ 2}$ means $Rule\ 2$ is derived from $Rule\ 1$.

- **Goals**

The roles and the goals in our scheme are as follows. First, there are four roles in our scheme: the trusted third party (TTP), the data owner ($Owner$), the cloud service provider (CSP), and the user ($User$). Then, there are three goals to be achieved. In the

BAN logic language, the three goals are:

$G1.CSP|\equiv Owner \lhd K_{owner}^{-1}$

$G2.CSP|\equiv User \lhd K_{User}^{-1}$

$G3.User|\equiv CSP \lhd rk$

$G1$ means in verification phase CSP needs to make sure that the sender of the ciphertext is $Owner$ and that the ciphertext has not been tampered by an attacker. So CSP must believe that $Owner$ holds his/her private key so that he/she can create the ciphertext. $G2$ means in the test phase CSP needs to verify $User$'s identity to determine that the trapdoor is permissible by believing that $User$ holds his/her private key so that he/she can create the trapdoor. $G3$ means $User$ needs to determine that the re-encrypted ciphertext has not been tampered by an attacker; in other words, $User$ needs to believe that CSP holds the re-encryption key rk to generate the re-encrypted ciphertext.

◆ Assumptions

With the goals set, now let's state our assumptions as follows:

$A1.CSP|\equiv \overset{K_{owner}}{\mapsto} Owner$

$A2.User|\equiv \overset{K_{Owner}}{\mapsto} Owner$

$A3.CSP|\equiv \overset{K_{User}}{\mapsto} User$

$A4.Owner|\Rightarrow K_{owner}^{-1}$

$A5.User|\Rightarrow K_{User}^{-1}$

$A6.CSP|\Rightarrow rk$

$A7.CSP|\Rightarrow W$

◆ Verification of The Data Owner

The data owner verification process in the verification phase is checked with the BAN logic as follows:

Message 1: $Owner \longrightarrow CSP: CT_i = (B, C, D, E, F)$

$V1.\ CSP \lhd B, C, D, E, F$

$V2.\frac{CSP \lhd w_l, CSP \lhd B}{CSP \lhd C}$

$V3.\frac{CSP \lhd C, CSP \lhd D, CSP \lhd E}{CSP \lhd F}$

$V4.\frac{CSP \mid \equiv F}{CSP \mid \equiv (B,D,E)}$

$V5.\frac{CSP \mid \equiv D, CSP \mid \equiv \overset{K_{owner}}{\longmapsto} Owner}{CSP \mid \equiv Owner \lhd K_{owner}^{-1}}$

When CSP receives the ciphertext from $Owner$, CSP can exploit the information to determine the correctness. From formula $V5$, we can infer that our scheme does achieve the goal we set. By formula $V5$, CSP believes that $Owner$ holds the private key to create the ciphertext.

◆ Verification of The User

The correctness of user verification in the test phase is verified with the BAN logic as follows:

Message 1: $User \longrightarrow CSP: T_{w_j}$

$V1.\ CSP \lhd T_{w_j}$

$V2.\frac{CSP \mid \equiv W, CSP \mid \equiv (B,C), CSP \mid \equiv \overset{K_{User}}{\longmapsto} User}{CSP \mid \equiv T_{w_j}}$

$V3.\frac{CSP|\equiv T_{w_j}}{CSP|\equiv User \lhd K_{User}^{-1}}$

When CSP receives the trapdoor, CSP can exploit the ciphertext sent from $Owner$ and $User$'s public key to determine the correctness. Formula $V3$, we can infer that our scheme achieves the goal we set for $Test$ phase. By formula $V3$, CSP believes that $User$ holds the private key to create the trapdoor.

◆ Verification of The Ciphertext

In this subsection, we examine the correctness of the re-encrypted ciphertext verification process in the decryption phase (including $Dec1$ and $Dec2$) with the BAN logic. The details are as follows:

For $Dec1$:

Message 1: $CSP \longrightarrow User: CT_j = (B, D', E)$

$V1.\ User \lhd B, D', E$

$V2.\frac{User \lhd (B,D'), User \lhd K_{User}^{-1}}{User \lhd R}$

$V3.\frac{User \lhd E, User \lhd R}{User \lhd m}$

$V4.\frac{User \lhd (m,R)}{User \lhd s}$

$V5.\frac{User|\equiv (s,B)}{User|\equiv (R,m)}$

$V6.\frac{User|\equiv R}{User|\equiv D'}$

$V7.\frac{User|\equiv D'}{User|\equiv CSP \lhd rk}$

When $User$ receives the ciphertext, he/she exploits all information contained in it to determine that the re-encrypted ciphertext is truly sent by CSP and has not been

tampered by an attacker. By formula $V7$, $User$ believes CSP holds the re-encryption key that can be used to re-encrypt the ciphertext, and therefore we can infer that our scheme achieves the goal we set for phase $Dec1$.

For $Dec2$:

Message 1: $CSP \longrightarrow User: CT_j = (B, C, D', E, F)$

$V1.\ User \lhd B, C, D', E, F$

$V2.\frac{User \lhd (B,D'), User \lhd K_{User}^{-1}}{User \lhd R}$

$V3.\frac{User \lhd E, User \lhd R}{User \lhd m}$

$V4.\frac{User \lhd (m,R)}{User \lhd s}$

$V5.\frac{User|\equiv (s,B)}{User|\equiv (C,R,m)}$

$V6.\frac{User|\equiv (R,m)}{User|\equiv (D',E)}$

$V7.\frac{User|\equiv D'}{User|\equiv F}$

$V8.\frac{User|\equiv F}{User|\equiv CSP \lhd rk}$

When $User$ receives the ciphertext that contains the keyword, he/she exploits all information contained in it to determine whether the re-encrypted ciphertext sent from CSP has been tampered by an attacker. By formula $V8$, $User$ believes that CSP holds the re-encryption key for the re-encryption of the ciphertext. Therefore, we can infer that our scheme achieves the goal we set for phase $Dec2$.

Chapter 4 Key-aggregate Encryption

Handling huge loads of data that are subject to change within every second, cloud storage services are facing the challenge of properly dealing with the problem of user legality management while making sure that the services are conveniently user-friendly. Ideally, the concept of attribute-based encryption (ABE) should be applied, meaning that data should be able to be encrypted using some specific attributes before it is uploaded to cloud, so that fine access control is possible. However, in a traditional attribute-based encryption scheme, the user typically needs to have different attribute-based keys for the decryption of various pieces of data downloaded, which really is a lot of trouble. To solve this problem, the idea of key-aggregate cryptosystem (KAC) has been brought up. With KAC, the user gets to use one single aggregate key to decrypt data that match all the attributes specified by the user. In addition, in some cases of cloud data usage we as users might not exactly want to share our cloud data with others 24 hours a day and for as long as it gets. Therefore, in this paper, we shall propose a time-bound key-aggregate encryption scheme for cloud storage, together with the results of some comparisons as well as correctness and security analyses we have made to prove the superiority of our new scheme over related works. Not only will our new scheme take the burden of maintaining the attribute-based keys off the user, but it will also provide satisfactory confidentiality and security for cloud data in a more efficient way.

4.1 Preliminaries

Thanks to the immense advancement of recent cloud computing technologies, quite a number of new ways to deal with vast amounts of data have been created and brought into our everyday lives, cloud storage service among the rest. The appearance of cloud storage service has swiftly changed people's common practice of bringing USB flash drives or portable hard drives or other devices around into embracing cloud storage space providers such as Dropbox, SkyDrive, and MEGA. Cloud storage service surprises every beginner with the amazing convenience and freedom of easily accessing their data wherever there is access to the Internet. Now clients of cloud storage services include not only individual people but also businesses or other kinds of organizations, and what is stored in cloud can range from public, totally non-sensitive data to highly confidential information. Therefore, thorough protection of the data trusted in cloud against any possible malicious access is crucial to the success of a cloud storage service provider.

As cloud storage service gains its popularity, people's choice of place to keep their data switches from devices right at hand to some far-away storage space you do not even know where it is, like Dropbox, Box.com, SugarSync, etc. To start using these cloud storage services, we typically provide an account and password pair. Once logged in, we are ready to upload our data. However, if our data is uploaded in the form of plaintext, then anyone at the server end will have an easy chance to obtain our data and make whatever malicious use of it they wish to. In order to prevent this from happening, before uploading, we can have our data encrypted and thus keep it incomprehensible to all the staff at the cloud end. Related issues may include privacy protection, confidentiality, etc. [10, 30, 37, 38, 50, 67].

With tons and tons of data stored in cloud, another major issue is how to make sure

that the cloud storage service user can have easy, fast access to the part or parts of data in need without having to download a whole warehouse of data and sorting everything out. To achieve fine access control, some exploit the keyword search process, and others use attribute-based encryption. Sahai and Waters were the first to propose the concept of attribute-based encryption [55]. With attribute-based encryption, the plaintext can be encrypted if it contains some specific attribute. Following this route, Wang et al. proposed a hierarchical attribute-based encryption scheme for cloud storage environments to make fine access control possible [66]. Unfortunately, traditional attribute-based encryption schemes may not be exactly user friendly and bring the right convenience. An example is like this: Suppose Alice encrypted some data using English, Chinese, mathematics, science, society, and computer as attributes respectively and then uploaded the data to cloud. One day, Bob wants to access the data with attributes English, Chinese, mathematics, and computer. First, he sends the access requirement to Alice. Since to each attribute there is a corresponding key, Alice then needs to respond to Bob with the decryption keys for attributes English, Chinese, mathematics, and computer respectively. This may not exactly be what Bob has in mind because now he has four different keys to manage. This scenario is illustrated in Figure 5. Just imagine if some data stored in cloud came through 1,000 different attributes. In that case, a user who wants to access data through 500 different attributes will have to ask the data owner for 500 separate keys, each corresponding to one attribute. Keeping those 500 keys is of course a lot of trouble. In order to solve this problem, Chu et al. proposed a new scheme called Key-Aggregate Cryptosystem (KAC) [19]. In Chu et al.'s scheme, Alice does not need to return four different keys to Bob; instead, only one aggregate key is generated for the collective attribute of English, Chinese, mathematics, and computer. With this aggregate key, Bob can decrypt the parts of data he wishes to get. This

scenario is illustrated in Figure 6.

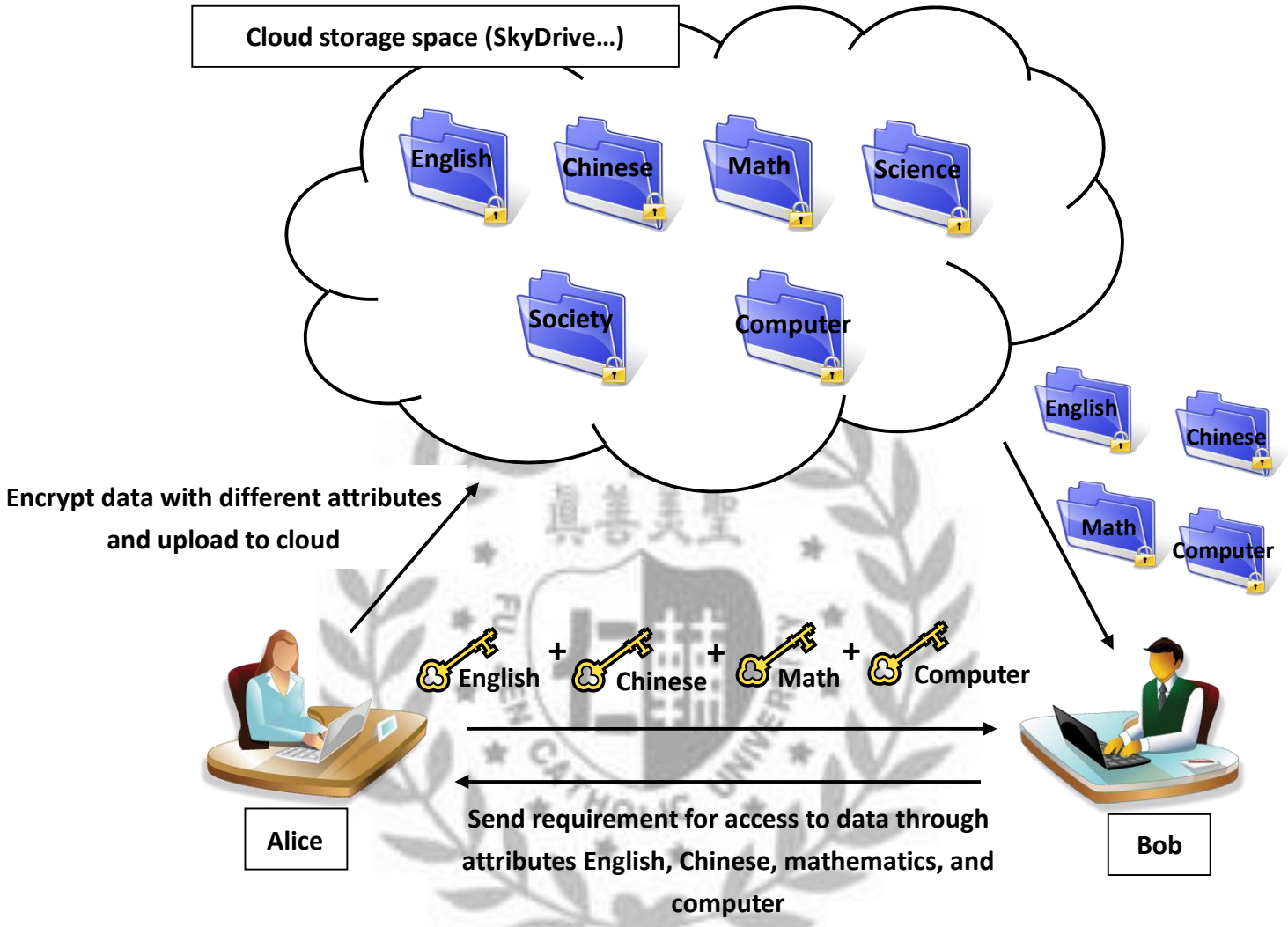

Figure 5 How traditional attribute-based encryption works

In addition, there are also times when Alice thinks opening all her data to Bob at all times may not be a good idea. That is when a time-bound aggregate key comes in. Since we have found no time-based aggregate key encryption mechanism proposed among the many previous studies related, in this paper we shall offer the very first time-bound key-aggregate encryption scheme for cloud storage. Since by nature the cloud server is obviously not to be fully trusted, in our scheme, not only do we combine time-bound key assignment together with key-aggregate encryption, but we also introduce the concept of proxy re-encryption. As a result, our new scheme offers better data

security and is much more user-friendly than its predecessors. To be more specific, the properties that our new scheme features are as follows:

1. Our scheme takes the heavy burden of managing decryption keys off the user:
   In a traditional attribute-based encryption scheme, the user needs to keep different attribute-based decryption keys for the downloading of data previously uploaded by using different attributes. This task of key management can be a heavy load on the user. Our scheme is capable of relieving the user of such a burden.
2. No tamper-resistant device is required:
   In order to resist collusion attacks, many time-bound key assignment schemes have to exploit the tamper-resistant devices. In our scheme, we exploit some public parameters instead.
3. The confidentiality of the data is guaranteed.
   To prevent a malicious cloud server from tampering the ciphertext, we empower the cloud server to re-encrypt the time-bound ciphertext for the corresponding set of attributes. Only the qualified user can use his/her key to decrypt the re-encrypted ciphertext.
4. Our time-bound key offers more flexibility:
   The data owner may or may not want to share the data with a user at all times. To offer this flexibility of time-bound access control, our time-bound key design enables the data owner to set a key for the user that gives the user permission to access the data desired within a certain period of time.

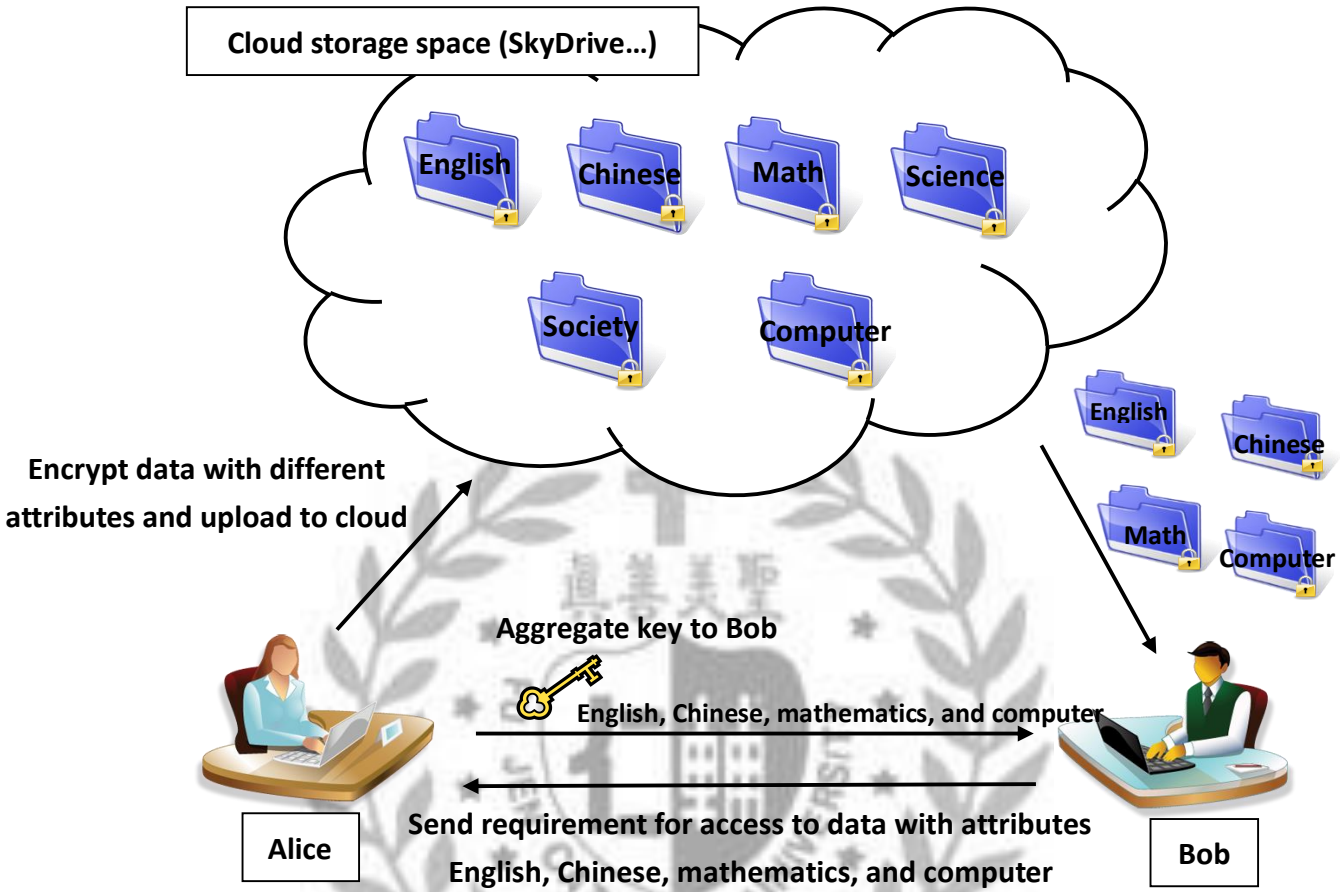

Figure 6 Key-aggregate encryption

4.2 Related Works

4.2.1 Attribute-based Encryption

Distinct from identity-based encryption (IBE), there is a new encryption type called attribute-based encryption (ABE). With attribute-based encryption, the plaintext can be encrypted if it contains some specific attribute. Sahai and Waters were the first to introduce the concept of attribute-based encryption to the world [55]. Sahai and Waters proposed their fuzzy identity-based encryption scheme in 2005. In 2006, Goyal proposed a new type attribute-based encryption named Key-Policy Attribute-Based Encryption (KP-ABE). In KP-ABE, each private key is associated with an access

structure that specifies which type of ciphertext the key can decrypt [29]. Then, in the next year, Bethencount et al. proposed another new attribute-based encryption scheme by the name of Ciphertext-Policy Attribute-Based Encryption (CP-ABE) which allows the user to associate the access structure with specific attributes [4]. Then, in 2008, Muller et al. extended the CP-ABE into their Distributed Attribute-Based Encryption (DABE), which supports an adjustable, unlimited number of attribute authorities and allows new users and authorities to join in dynamically at any time [48].

4.2.2 Time-bound Key Assignment

In some cases we want the user to have the freedom of accessing the data at any time, but in other cases we want to put some limit to it. When the access time is to be limited, setting a time-bound key for the user is a good idea. In 2002, Tzeng [62] proposed a time-bound key assignment scheme, where the user can access some certain data within a certain period of time specified by the time-bound key. To be more specific, in Tzeng's cheme, there is a class time-bound key $K_{i,t}$ at time t for class C_i. However, later in 2003, Yi and Ye pointed out that Tzeng's scheme was vulnerable to collusion attacks [75]. In 2004, Chien proposed a new time-bound hierarchical key management scheme based on a low-cost tamper-resistant device [17]. Chien's scheme uses the hash function instead of public key cryptography and thus reduces the computation cost effectively. Unfortunately, in 2005, Yi found that Chien's scheme was vulnerable to collusion attacks [74]. In order to provide protection against collusion attacks, in 2008, Bertino et al. proposed a new efficient time-bound hierarchical key management scheme that makes use of tamper-resistant devices [3]. Then, in 2009, Sun et al. offered proof that Bertino et al.'s scheme is indeed robust against collusion attacks [60]. More recently in 2012, Shen et al. proposed a time-bound hierarchical access

control and key management scheme for the multicast system that protects the confidential multicast data [58]. In the meantime, Chen et al. proposed an efficient time-bound hierarchical key management scheme that can do without a tamper-resistant device [15].

4.2.3 The Encryption Mechanism for Cloud Storage

In 2010, Wang et al. proposed a hierarchical attribute-based encryption scheme for fine-grained access control in cloud storage. Their scheme combines a hierarchical attribute-based encryption system and a ciphertext-policy attribute-based encryption system, resulting in high performance, fine-grained access control, and collusion attack resistance [66]. Yu et al. exploited ciphertext-policy attribute-based encryption and added in proxy re-encryption to construct the first scheme that simultaneously achieves fine access control, scalability, and data confidentiality in cloud computing [76]. In 2011, Huang et al. proposed an efficient identity-based key management mechanism for configurable hierarchical cloud environments that gives better performance at lower communication cost on encryption [32]. Due to the heavy loads of data kept in cloud storage, Liu et al. pointed out some problems with Liu et al.'s efficient privacy preserving keyword search scheme [43] and proposed a new secure and privacy preserving keyword search scheme for cloud storage services in 2012 [44]. In the same year, Koo et al. exploited ABE and proposed a new searchable encryption scheme which provides efficient data retrieval for cloud storage [37]. In 2013, Fan and Hiang proposed a variant of symmetric predicate encryption for cloud environment that provides controllable privacy preserving search functionalities [23]. Chen et al. proposed a new scheme to support data dynamics for remote data possession checking in cloud environment by exploiting the Merkle hash tree [16]. Liu et al. proposed a new

secure data sharing scheme by the name of Mona, where the dynamic group design in cloud environment is made possible by leveraging the group signature and employing some dynamic broadcast encryption techniques [46]. To take care of in intra-domain and inter-domain query requirements, Han et al. proposed an identity-based proxy re-encryption scheme suitable for cloud computing applications [23]. Then Wang et al. proposed a privacy-preserving public auditing scheme for a secure cloud storage system where the third-party auditor (TPA) would not learn anything about the data contents stored in cloud during the auditing process [64]. In 2014, in order to make fine access control possible over searchable encrypted data, Li et al. proposed a new scheme for hybrid clouds that offers practical keyword search where a private cloud is introduced as an access interface between the user and the public cloud [40]. Meanwhile, Liu et al. combined the concepts of attribute-based encryption and time-based access control to build a time-based proxy re-encryption scheme for data sharing in cloud environment that achieves scalable user revocation and fine access control [45].

4.3 Time-bound Key-aggregate Encryption

Before illustrating our scheme, we will go over some basic principles of bilinear pairing [7, 34, 35] and give some complexity assumptions. Then we will get into the details of our time-bound key-aggregate encryption scheme.

4.3.1 Bilinear Pairing

Let $\mathbb{G}_1$ be a cyclic additive group with prime order q, and let $\mathbb{G}_2$ be a cyclic multiplicative group with prime order q, and p is the generator of group $\mathbb{G}_1$. With $x, y \in \mathbb{Z}_q$, we have the bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ that satisfies the following requirements:

- Bilinearity: For all $x, y \in \mathbb{Z}_q$ and $P, Q \in \mathbb{G}_1$, $e(P^x, Q^y) = e(P, Q)^{xy} = e(P^y, Q^x)$.

  Sometimes the property of bilinearity can be alternatively expressed as $e(xP, yQ) = e(P, Q)^{xy} = e(yP, xQ)$.

- Computability: For any $P, Q \in \mathbb{G}_1$, tere always exists an efficient algorithm to compute $e(P, Q) \in \mathbb{G}_2$.

- Non-degeneration: $e(P, Q) \neq 1$.

4.3.2 Complexity Assumption

The security of our new scheme is based on the following complexity assumptions:

- Discrete Logarithm Problem (DLP)

  Given two elements P and Q in $\mathbb{G}_1$, it is extremely difficult to find $n \in \mathbb{Z}_q$ such that $P = nQ$ if n exists.

- Computation Diffie-Hellman Problem (CDHP)

  Given P, xP, yP for $x, y \in \mathbb{Z}_q$, it is extremely difficult to compute xyP.

- Bilinear Diffie-Hellman Problem (BDHP)

  Given P, P^x, P^y, P^z for $x, y, z \in \mathbb{Z}_q$, it is extremely difficult to compute $e(P, P)^{xyz} \in \mathbb{G}_2$.

4.3.3 The Proposed Scheme

In this subsection, we will propose our time-bound key-aggregate encryption scheme inspired by Boneh et al. [8] and Chu et al. [19]. First, we will illustrate the

architecture of our scheme, and then we will give the details of each phase in our scheme.

To begin with, the participants in our scheme include the cloud storage provider (CSP), the data owner, and the user. The three parties behave as follows:

- Cloud storage provider (CSP): CSP needs to store the ciphertext and accept requirements sent from the user. CSP also has the ability of re-encrypting the time-bound ciphertext.
- Data owner: The data owner needs to encrypt the ciphertext and set a corresponding class to each piece of it. The data owner also generates a time-bound aggregate key for the user.
- User: The user needs to send a requirement to the data owner in order to get his/her key, and then send another requirement to CSP to get the re-encrypted ciphertext. The user then uses the time-bound aggregate key given by the data owner to decrypt the ciphertext received.

The notations we will use throughout the presentation of our new scheme are listed in Table 7.

Table 7 Notations in time-bound key-aggregate encryption

Notations	Descriptions
p	A prime order
G, G_T	Bilinear groups of prime order p
$\hat{e}$	Bilinear map $\hat{e}: G \times G \rightarrow G_T$
g	A generator of G
α, β, a, b	The secret random numbers, $\alpha, \beta, a, b \in Z_p$
n	The maximum number of ciphertext classes
e_k	A secret value
z	The time line, $z < p$
T	The maximum continuous subscription time
t	Current time
t_1	The initiate time of register time
t_2	The termination time of register time
x	The past time
y	The remaining time

We set λ as the total of the valid time for the user. This way, when the user subscribes for a time period $[t_1, t_2]$, the variables T, λ, x, y, t, t_1, and t_2 satisfy the following description:

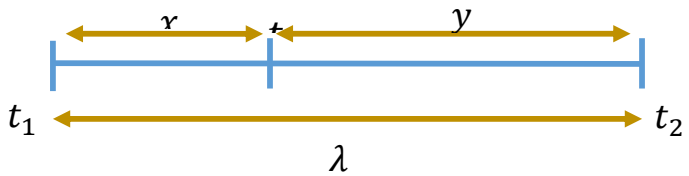

Figure 7 The relationship of T, λ, x, y, t, t_1, and t_2

$$t_1 + x = t = t_2 - y,\ x + y = \lambda,\ t_2 - t_1 = \lambda \leq T$$

There are six algorithms in our scheme as follows:

- SystemSetup: Set $g_i = g^{\alpha^i} \in G$ for $i = 1, \dots, n, n+2, \dots, 2n$. Then compute T sets of public parameters $B = \{B_1, B_2, \dots, B_T\}$, where each set B_k includes $k+1$ keys. These $k+1$ keys as a whole are called $D_{k,u} = \alpha + a^u b^{k-u}$, $\forall u \in [0,k]$, $\forall k \in [0,t]$, where u and k are the past time and the total time, respectively. Now we have the system parameters $param = (B, g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, e_k g, g^{\alpha})$.

  Notice that the relationship of B, B_k and $D_{k,u}$ can be described as follows:

  Assume $T = 4$, there exist $B = \{B_1, B_2, B_3, B_4\}$, and each B_k includes $k+1$ keys:

  $B_1 = \{D_{1,0}, D_{1,1}\}$
  $= \{\alpha + a^0 b^1, \alpha + a^1 b^0\}$

  $B_2 = \{D_{2,0}, D_{2,1}, D_{2,2}\}$
  $= \{\alpha + a^0 b^2, \alpha + a^1 b^1, \alpha + a^0 b^2\}$

  $B_3 = \{D_{3,0}, D_{3,1}, D_{3,2}, D_{3,3}\}$
  $= \{\alpha + a^0 b^3, \alpha + a^1 b^2, \alpha + a^2 b^1, \alpha + a^3 b^0\}$

  $B_4 = \{D_{4,0}, D_{4,1}, D_{4,2}, D_{4,3}, D_{4,4}\}$
  $= \{\alpha + a^0 b^4, \alpha + a^1 b^3, \alpha + a^2 b^2, \alpha + a^3 b^1, \alpha + a^4 b^0\}$

- KeyGen: Pick $\gamma \in_R Z_p$, then compute the public key $pk = v = g^{\gamma}$ and master-secret key $mk = \gamma$.
- Encrypt: The data owner encrypts the message and sets a corresponding class to each ciphertext. For a message $m_i \in G_T$ and an index $i \in \{1, \dots, n\}$,

randomly choose $\beta \in Z_p$, and compute the ciphertext $C_i = (c_1, c_2, c_3, c_4) = (g^{\alpha\beta}, g^{\beta}, (vg_i)^{\beta}, m_i \cdot \hat{e}(g_1, g_n)^{\beta})$.

- Extract: Upon receiving the requirement from the user, the data owner generates a time-bound aggregate key for the user. For the set S of indices j 's the aggregate key can be computed by $K_S = e_k g^{\gamma} a^{t_1} b^{z-t_2} \cdot \prod_{j \in S} g^{\gamma}_{n+1-j}$. After computing K_S, the data owner sends it back to the user. With this key, the user can decrypt the ciphertext desired.
- Re-encryption: Upon receiving the requirement from the user, CSP generates a new time-bound ciphertext for the user. CSP needs to re-encrypt the stored ciphertext. CSP computes $c_4' = c_4 / \hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, c_2\right)^{a^{t} b^{z-t} e_k} \cdot \hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, c_1\right)^{a^{t_1} b^{z-t_2} e_k}$ then returns $C_i' = (c_1, c_2, c_3, c_4')$ to the user as re-encrypted ciphertext.
- Decrypt: If the user decrypts the ciphertext in valid time, the user can use K_S to decrypt the re-encrypted ciphertext by computing $m_i = c_4' \cdot \hat{e}\left(K_S \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2\right)^{D_{k,u}} \cdot \hat{e}(g_{n+1}, c_2)^{D_{k,u}} / \hat{e}\left(\prod_{j \in S} g_{n+1-j}, c_3\right)^{D_{k,u}} \cdot \hat{e}(g_{n+1}, c_2)$. Before computing m_i, the user needs to find the corresponding B_k in B to obtain $D_{k,u} = \alpha + a^{u} b^{k-u}$ in order to decrypt the ciphertext.

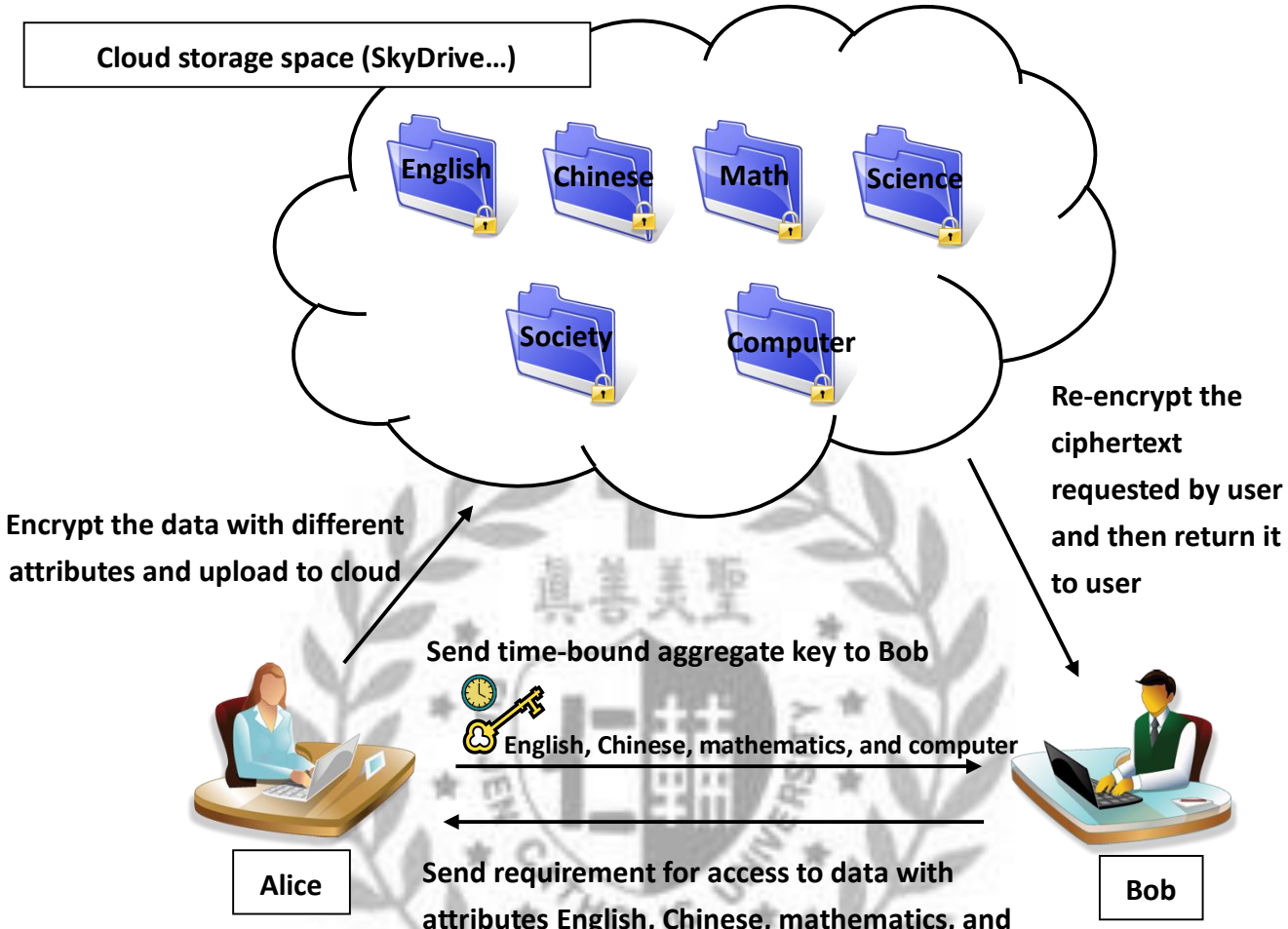

Figure 8 Time-bound key-aggregate encryption scheme

4.4 Security and Performance Analysis

In this section, let's confirm the correctness of our scheme, examine how it compares with related schemes in terms of functions, and then check the security against some possible attacks. For the correctness analysis, we especially checked the correctness of the decryption algorithm, and the BAN logic was also employed to check the whole scheme.

4.4.1 Correctness Analysis

4.4.1.1 Correctness of Decryption Algorithm

In the decryption algorithm, the user can use his/her time-bound key to recover m_i. The correctness of this algorithm can be confirmed as follows:

$$c_4' \cdot \frac{\hat{e}\left(K_S \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i},\ c_2\right)^{D_{k,u}} \cdot \hat{e}(g_{n+1}, c_2)^{D_{k,u}}}{\hat{e}\left(\prod_{j\in S} g_{n+1-j},\ c_3\right)^{D_{k,u}} \cdot \hat{e}\left(g_{n+1},\ c_2\right)}$$

$$= c_4' \cdot \frac{\hat{e}\left(\left(e_k g^{\gamma} a^{t_1} b^{z-t_2} \cdot \prod_{j\in S} g_{n+1-j}^{\gamma}\right) \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i},\ g^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(g_{n+1}, g^{\beta}\right)^{D_{k,u}}}{\hat{e}\left(\prod_{j\in S} g_{n+1-j},\ (v g_i)^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(g_{n+1},\ g^{\beta}\right)}$$

$$= c_4' \cdot \frac{\hat{e}\left(\left(e_k g^{\gamma} a^{t_1} b^{z-t_2}\right) \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}, g^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(\prod_{j\in S} g_{n+1-j}^{\gamma} \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}, g^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(g_{n+1}, g^{\beta}\right)^{D_{k,u}}}{\hat{e}\left(\prod_{j\in S} g_{n+1-j}, (v g_i)^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(g_{n+1}, g^{\beta}\right)}$$

$$= c_4' \cdot \frac{\hat{e}\left(\prod_{j\in S} g_{n+1-j}^{\gamma}, g^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(\prod_{j\in S, j\neq i} g_{n+1-j+i}, g^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(\left(e_k g^{\gamma} a^{t_1} b^{z-t_2}\right) \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}, g^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(g_{n+1}, g^{\beta}\right)^{D_{k,u}}}{\hat{e}\left(\prod_{j\in S} g_{n+1-j}, (g^{\gamma})^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(\prod_{j\in S} g_{n+1-j}, g_i^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(g_{n+1}, g^{\beta}\right)}$$

$$= c_4' \cdot \frac{\left(\frac{\hat{e}\left(\prod_{j\in S} g_{n+1-j+i}, g^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(\left(e_k g^{\gamma} a^{t_1} b^{z-t_2}\right) \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}, g^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(g_{n+1}, g^{\beta}\right)^{D_{k,u}}}{\hat{e}\left(g_{n+1}, g^{\beta}\right)^{D_{k,u}}}\right)}{\hat{e}\left(\prod_{j\in S} g_{n+1-j+i}, g^{\beta}\right)^{D_{k,u}} \cdot \hat{e}\left(g_{n+1}, g^{\beta}\right)}$$

$$= m_i \cdot \frac{\hat{e}(g_1, g_n)^{\beta}}{\hat{e}\left(g \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}^{\gamma}, g^{\beta}\right)^{a^t b^{z-t} e_k} \cdot \hat{e}\left(g \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}^{\gamma}, g^{\alpha\beta}\right)^{a^{t_1} b^{z-t_2} e_k}} \cdot \frac{\hat{e}\left(\left(e_k g^{\gamma} a^{t_1} b^{z-t_2}\right) \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i},\ g^{\beta}\right)^{D_{k,u}}}{\hat{e}(g_{n+1}, g^{\beta})}$$

$$= m_i \cdot \frac{\hat{e}\left((e_k g^{\gamma} a^{t_1} b^{z-t_2}) \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i},\ g^{\beta(\alpha + a^u b^{k-u})}\right)}{\hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\beta}\right)^{a^t b^{z-t} e_k} \cdot \hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\alpha\beta}\right)^{a^{t_1} b^{z-t_2} e_k}}$$

$$= m_i \cdot \frac{\hat{e}\left((e_k g^{\gamma} a^{t_1} b^{z-t_2}) \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^{\alpha\beta}\right) \cdot \hat{e}\left((e_k g^{\gamma} a^{t_1} b^{z-t_2}) \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^{\beta a^u b^{k-u}}\right)}{\hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\beta}\right)^{a^t b^{z-t} e_k} \cdot \hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\alpha\beta}\right)^{a^{t_1} b^{z-t_2} e_k}}$$

$$= m_i \cdot \frac{\hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\alpha\beta}\right)^{e_k a^{t_1} b^{z-t_2}} \cdot \hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\beta}\right)^{(e_k a^{t_1} b^{z-t_2})(a^u b^{k-u})}}{\hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\beta}\right)^{a^t b^{z-t} e_k} \cdot \hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\alpha\beta}\right)^{a^{t_1} b^{z-t_2} e_k}}$$

$$= m_i \cdot \frac{\hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\alpha\beta}\right)^{e_k a^{t_1} b^{z-t_2}} \cdot \hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\beta}\right)^{a^t b^{z-t} e_k}}{\hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\beta}\right)^{a^t b^{z-t} e_k} \cdot \hat{e}\left(g \cdot \prod_{j \in S, j \neq i} g^{\gamma}_{n+1-j+i}, g^{\alpha\beta}\right)^{a^{t_1} b^{z-t_2} e_k}}$$

$$= m_i$$

Note that according to the relationship of T, λ, x, y, t, t_1, and t_2, we can get $t = t_2 - y$. Consequently, we can derive $a^{(t_1+u)} b^{(z-t_2+k-u)} = a^t b^{(z-t_2+y)} = a^t b^{z-t}$.

4.4.1.2 BAN Logic Check

The BAN logic [11, 72] is a well-accepted method to analyze the correctness of cryptographic protocols. To apply the BAN logic, we have to define some notations, goals and assumptions of our scheme.

- **Notations**

Here are the syntax and notations of the BAN logic. A and B are the specific participators, and X is the formula (statement). There are some rules as follows [11, 72]:

15. $A|\equiv X$ means A believes the formula X is ture.

16. $A|\equiv B$ means A believes B's action.

17. $A|\Longrightarrow X$ means A has complete control over the formula X.

18. $A \lhd X$ means A holds or sees the formula X.

19. $\#(X)$ means the formula X is fresh.

20. $\overset{K_A}{\longmapsto} A$ means K is the public key for A and K_A^{-1} is the private key for A.

21. $\frac{Rule\ 1}{Rule\ 2}$ means $Rule\ 2$ can be derived from $Rule\ 1$.

◆ Goals

There are three roles interacting in our scheme, namely the data owner ($Owner$), the cloud service provider (CSP) and the user ($User$). In the language of the BAN logic, our scheme is to achieve the two goals as follows:

$G1.\ User|\equiv K_{User}^{-1}$

$G2.\ User|\equiv m_i$

Because the user needs to use his/her secret key to decrypt C_i' to recover m_i, in $G1$ the user should believe that the decryption key is truly sent from the data owner. Then, as $G2$ indicates, the user should believe that the m_i that he/she decrypts by using his/her key is true.

◆ Assumptions

With the goals set, the assumptions used to analyze our scheme can be stated as follows:

$A1.\ Owner|\Longrightarrow K_{Owner}^{-1}$

$A2.\ User|\equiv Owner \lhd K_{Owner}^{-1}$

$A3.\ User \lhd K_{User}^{-1}$

$A4.\ User|\equiv B_i$

Since the set B is protected by the secret values $\alpha, a,$ and b, the user should believe that B cannot be tampered by an attacker.

◆ Correctness of Scheme

Now we are ready to use the BAN logic to confirm the correctness of our scheme:

Message 1: $CSP \longrightarrow User: C_i' = (c_1,\ c_2,\ c_3, c_4')$

$V1.\ User \lhd C_i'$

$V2.\ \dfrac{User \lhd C_i', K_{User}^{-1}, User|\equiv B_i}{User \lhd m_i}$

$V3.\ \dfrac{User|\equiv Owner \lhd K_{Owner}^{-1}, User \lhd m_i}{User|\equiv K_{User}^{-1}}$

$V4.\ \dfrac{User|\equiv K_{User}^{-1}}{User|\equiv m_i}$

When CSP sends $C_i' = (c_1,\ c_2,\ c_3, c_4')$ to the user, the user can hold $C_i' = (c_1,\ c_2,\ c_3, c_4')$. In $V2$, there are T sets in B, and not only does the user believe B_i but he/she also holds C_i' and K_s. So, he/she can recover m_i by exploiting $B, C_i',$ and K_s. Since K_s is generated by using the data owner's master-secret key, the user can use K_s to recover m_i and therefore can believe that K_s is efficacious. Then, since m_i is obtained through the decryption process using the key K_s, the user believes that m_i is what he/she wishes to obtain. By formulas $V3$ and $V4$, the user believes K_s and m_i, and so the goals of our scheme are achieved.

4.4.2 Comparisons

In this subsection, we shall compare our new scheme with Han et al.'s [30], Koo et al.'s [37], Li et al.'s [40], Liu et al.'s [45], and Chu et al.'s [19] scheme. In our table of comparison results, namely Table 2, the five terms AttEn, Time-bound, KeyAgg, Re-encrypt, and Confidentiality are used to indicate attribute-based encryption, time-bound key assignment, key-aggregate encryption, proxy re-encryption, and data confidentiality, respectively.

Data confidentiality is an important requirement any encryption scheme applied in any field should satisfy. In Table 8 we gladly see that all of the schemes satisfy this requirement. As for attribute-based encryption, it helps in data categorization and is the key to fine access control. Unfortunately, since Han et al.'s scheme is based on identity-based encryption, it cannot provide fine access control. Among the schemes compared, Liu et al.'s scheme, with attribute-based encryption and time-bound key assignment combined, satisfies four of the five requirements. However, due to the lack of key aggregation, in Liu et al.'s scheme, the user needs more than one key for different attributes. As Table 8 reveals, our scheme is the only scheme to satisfy all five requirements.

Table 8 Comparison results among related works

	AttEn	Time-bound	KeyAgg	Re-encrypt	Confidentiality
Han et al.	×	×	×	○	○
Koo et al.	○	×	×	×	○
Li et al.	○	×	×	×	○
Liu et al.	○	○	×	○	○
Chu et al.	○	×	○	×	○
Our scheme	○	○	○	○	○

AttEn : attribute-based encryption
Time-bound : time-bound key assignment
KeyAgg : key-aggregate encryption
Re-encrypt : proxy re-encryption
Confidentiality : data confidentiality

4.4.3 Security Analysis

In general, there are two ways to analyze the security of a scheme: formal analysis and heuristic analysis. In this study, we followed the route of heuristic analysis.

1. System security is ensured by protected α.

   In our scheme, α plays an important role. If α leaked out, the system would be exposed to danger. To keep an attacker from obtaining α through analyzing the public parameter $D_{k,u}$, we make $D_{k,u} = \alpha + a^u b^{k-u}$ so any attempt of exploiting the Euclidean algorithm will be in vain.

2. Proxy re-encryption ensures data confidentiality.

   CSP has no way to obtain m_i by analyzing the ciphertext. CSP needs to use a time-based parameter ($a^t b^{z-t}$) to re-encrypt C_i before sending it to the user.

3. Only the legitimate user can decrypt the re-encrypted ciphertext.

   Since CSP has had the ciphertext re-encrypted by using the time-based parameter ($a^t b^{z-t}$), only the legitimate user with the right aggregate key can decrypt the ciphertext, and this can only be done within the time interval $[t_1, t_2]$ because the aggregate key expires after the time limit.

4. CDHP and BDHP offer protection against collusion attacks.

   If some dishonest users hold the same attribute but different prescription of time-bound aggregate key or the same prescription but different attribute of time-bound aggregate key, they do not have the ability to exploit each key to obtain the data owner's master secret key $mk = \gamma$ and time parameter $a^{t_1} b^{z-t_2} e_k$ because of the protection of CDHP and BDHP. Therefore, no user can decrypt more than the data they are entitled to.

5. The user can verify whether or not the ciphertext has been tampered.

   The moment the user successfully recovers the ciphertext with his/her time-bound aggregate key, the time-bound ciphertext proves to be the real thing. This is because only the real CSP has the ability to re-encrypt the ciphertext within the time limit.

6. The time parameter offers protection against the ciphertext-only attack
   Since each time-bound ciphertext has a unique time parameter $a^{t_1}b^{z-t_2}e_k$, there is no way an attacker can analyze the ciphertext to get the key or the plaintext.

Chapter 5 Conclusions

In this study, we proposed three scheme for cloud storage service. In Chapter 2, Since Boneh et al. offered their concept of public key encryption with keyword search (PEKS), many researchers have extended it to various PEKS schemes such as the secure channel-free public key encryption scheme with keyword search (SCF-PEKS), the efficient privacy-preserving keyword search scheme (EPPKS), the trapdoor-indistinguishable public key encryption scheme with keyword search (TI-PEKS) and so on. We have proposed a secure trapdoor-indistinguishable public key encryption scheme with keyword search. Using a public channel, the proposed scheme is capable of keeping the CSP from being tricked by an attacker sending in fake ciphertext.

In Chapter 3, we have presented a searchable hierarchical conditional proxy re-encryption scheme for cloud storage services. Not only does our new scheme support hierarchical proxy re-encryption but it also allows CSP to do keyword searching on the encrypted data. If a new keyword is added, our scheme can exploit the current re-encryption key to generate a new re-encryption key for the newly added keyword. The correctness of our new scheme has been proven by a BAN logic examination. Compared with similar schemes, our scheme shows superiority in terms of function, performance, and security. So far, quite a number of new schemes including ours can support the generation of new re-encryption keys for when new keywords are added. In the future, we hope to develop a new re-encryption key that can handle keyword reduction.

In Chapter 4, we have proposed the first time-bound key-aggregate encryption scheme for cloud storage. With our scheme, the data owner can finely adjust the user's range and time of data access. In addition, our scheme is very user-friendly because the

user only has to keep an aggregate key. Our correctness check, feature comparison, and security analysis have also shown the superiority of our scheme over related works.

References

[1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, pp. 29-44, 2005.

[2] J. Baek, R. Safiavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," *Proceedings of the 8th International Conference on Computational Science and Its Applications (ICCSA'08)*, vol. 5072, pp. 1249-1259, 2008.

[3] E. Bertino, S. Ning, and S. S. Wagstaff, "An efficient time-bound hierarchical key management scheme for secure broadcasting," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 2, pp. 65-70, 2008.

[4] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-policy attribute-based encryption," *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 321‑334, 2007.

[5] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," *Proceedings of EUROCRYPT 1998, Lecture Notes in Computer Science*, vol. 1403, pp. 127-144, 1998.

[6] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," *Proceedings of 2004 International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04)*, vol. 3027, pp. 506-522, 2004.

[7] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *Cryptology — CRYPTO 2001, Lecture Notes in Computer Science*, vol. 2139, pp. 213-229, 2001.

[8] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with

short ciphertexts and private keys," *Lecture Notes in Computer Science*, vol. 3621, pp. 258-275, 2005.

[9] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," *Proceedings of TCC 2007, Lecture notes in computer science*, vol. 4392, pp. 535-554, 2007.

[10] S. Bugiel, S. Nűurnberger, A. Sadeghi, and T. Schneider, "Twin clouds: an architecture for secure cloud computing (extended abstract)," *Proceedings of the Workshop on Cryptography and Security in Clouds (WCSC 2011)*, pp. 1-11, 2011.

[11] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions Computer Systems*, vol. 8, no. 1, pp. 18-36, 1990.

[12] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ACM New York, NY, USA, pp. 185-194, 2007.

[13] Y. C Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," *Proceedings of ACSN 2005, Lecture notes in computer science*, vol. 3531, pp. 442-455, 2005.

[14] Z. Chen, C. Wu, D. Wang, and S. Li, "Conjunctive keywords searchable encryption with efficient pairing, constant ciphertext and short trapdoor," *Lecture Notes in Computer Science*, vol. 7299, pp. 176-189, 2012.

[15] C. M. Chen, T. Y. Wu, B. Z. He, and H. M. Sun, "An efficient time-bound hierarchical key management scheme without tamper-resistant devices," *Proceedings of the IEEE International Conference on Computing, Measurement, Control and Sensor Network (CMCSN)*, pp. 285-288, 2012.

[16] L. Chen, S. Zhou, X. Huang, and L. Xu, "Data dynamics for remote data possession checking in cloud storage," *Computers and Electrical Engineering*, vol. 29, no. 7,

pp. 2413-242, 2013.

[17] H. Y. Chien, "Efficient time-bound hierarchical key assignment scheme," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 10, pp. 1301-1304, 2004.

[18] S. Chow, J. Weng, Y. Yang, and R. Deng, "Efficient unidirectional proxy re-encryption," *Proceedings of AFRICACRYPT 2010, Lecture Notes in Computer Science*, vol. 6055, pp. 316-332, 2010.

[19] C. K. Chu, Sherman S.M. Chow, W. G. Tzeng, J. Zhou, and Robert H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 468-477, 2014.

[20] C. Chu and W. Tzeng, "Identity-based proxy re-encryption without random oracles," *Proceedings of ISC 2007, Lecture Notes in Computer Science*, vol. 4779, pp. 189-202, 2007.

[21] C. Chu, J. Weng, S. Chow, J. Zhou, and R. Deng, "Conditional proxy broadcast reencryption," *Proceedings of ACISP 2009, Lecture Notes in Computer Science*, vol. 5594, pp. 327-342, 2009.

[22] R. Deng, J. Weng, S. Liu, and K. Chen, "Chosen-cipertext secure proxy re-encryption without pairings," *Proceedings of CANS 2008, Lecture Notes in Computer Science*, vol. 5339, pp. 1-17, 2008.

[23] C. I. Fan and S. Y. Huang, "Controllable privacy preserving search based on symmetric predicate encryption in cloud storage," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1716-1724, 2013.

[24] L. Fang, W. Susilo, C. Ge, and J. Wang, "Hierarchical conditional proxy re-encryption," *Computer Standards & Interfaces*, vol. 34, no. 4, pp. 380-389, 2012.

[25] L. Fang, W. Susilo, and J. Wang, "Anonymous conditional proxy re-encryption

without random oracle," *Lecture Notes in Computer Science*, vol. 5848, Springer, pp. 47-60, 2009.

[26] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," *Lecture Notes in Computer Science*, vol. 3089, pp. 31-45, 2004.

[27] M. Green and G. Ateniese, "Identity-based proxy re-encryption," *Proceedings of ACNS 2007, Lecture Notes in Computer Science*, vol. 4521, pp. 288-306, 2007.

[28] C. Gu and Y. Zhu, "New efficient searchable encryption schemes from bilinear pairings," *International Journal of Network Security*, vol. 10, no. 1, pp. 25-31, 2010.

[29] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," *Proceedings of 13th ACM Conf. Computer and Comm. Security*, pp. 89-98, 2006.

[30] J. G. Han, W. Susilo, and Y. Mua, "Identity-based data storage in cloud computing," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 673-681, 2013.

[31] S. T. Hsu, C. C. Yang, and M. S. Hwang, "A study of public key encryption with keyword search," *International Journal of Network Security*, vol. 15, no. 2, pp. 71-79, 2013.

[32] J. Y. Huang, I-En Liao, and C. K. Chiang, "Efficient identity-based key management for configurable hierarchical cloud computing environment," *Proceedings of the IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 833-887, 2011.

[33] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee, "Constructing PEKS schemes secure against keyword guessing attacks is possible?," *Computer Communications*, vol. 32, no. 2, pp. 394-396, 2009.

[34] A. Joux, "A one round protocol for tripartite Diffie-Hellman," *Journal of Cryptology*, vol. 17, no. 4, pp. 263-276, 2004.

[35] A. Joux and K. Nguyen, "Separating decision Diffie–Hellman from computational Diffie–Hellman in cryptographic groups," *Journal of Cryptology*, vol. 16, no. 4, pp.239-247, 2003.

[36] D. Khader, "Public key encryption with keyword search based on k-resilient IBE," in *Computational Science and Its Application-ICCSA 2006*, vol. 3982 of *Lecture Notes in Computer Science*, pp. 298-308, 2006.

[37] D. Y. Koo, J. B. Hur, and H. S. Yoon, "Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage," *Computers & Electrical Engineering*, vol. 39, no. 1, pp. 34-46, 2013.

[38] C. C. Lee, P. S. Chung, and M. S. Hwang, "A survey on attribute-based encryption schemes of access control in cloud environments," *International Journal of Network Security*, vol. 15, no. 4, pp. 231-240, 2013.

[39] C. C. Lee, S. T. Hsu, and M. S. Hwang, "A study of conjunctive keyword searchable schemes," *International Journal of Network Security*, vol. 15, no. 5, pp. 311-320, 2013.

[40] J. W. Li, J. Li, X. F. Chen, Z. L. Liu, and C. F. Jia, "Privacy-preserving public auditing for secure cloud storage," *Future Generation Computer Systems*, vol. 30, no. 11, pp. 98-106, 2014.

[41] J. Li, Q. Wang, C. Wang, K. R. N. Cao, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," *Proceedings of the 29th conference on Information communications (INFOCOM'10)*, San Diego, California, USA. IEEE, pp. 1-5, 2010.

[42] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy reencryption," *Proceedings of PKC 2008, Lecture Notes in Computer Science*, vol. 4939, pp. 360-379, 2008.

[43] Q. Liu, G. Wang, and J. Wu, "An efficient privacy preserving keyword search scheme in cloud computing," *Proceedings of the 2009 International Conference on Computational Science and Engineering*, pp.715-720, 2009.

[44] Q. Liu, G. Wang, and J. Wu, "Secure and privacy preserving keyword searching for cloud storage services," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 927-933, 2012.

[45] Q. Liu, G. J. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information Sciences*, vol. 258, no. 24, pp. 355-370, 2014.

[46] X. F. Liu, Y. Q. Zhang, B. Y. Wang, and J. B. Yan, "Mona: secure multi-owner data sharing for dynamic groups in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1182-1191, 2013.

[47] T. Matsuda, R. Nishimaki, and K. Tanaka, "CCA proxy re-encryption without bilinear maps in the standard model," *Proceedings of PKC 2010, Lecture Notes in Computer Science*, vol. 6056, pp. 261-278, 2010.

[48] S. Muller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," *Proceedings of the International Conference on Information Security and Cryptology*, pp. 20-36, 2008.

[49] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," *Lecture Notes in Computer Science*, vol. 3325, pp. 73-86, 2005.

[50] Y. Peng, W. Zhao, F. Xie, Z. H. Dai, Y. Gao, and D. Q. Chen, "Secure cloud storage based on cryptographic techniques," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, no. 2, pp. 182-189, 2012.

[51] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Improved searchable public key encryption with designated tester," *ASIACCS '09 Proceedings of the 4th*

International Symposium on Information, Computer, and Communications security, pp. 376-379, 2009.

[52] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *The Journal of Systems and Software*, vol. 83, no. 5, pp. 763-771, 2010.

[53] H. S. Rhee, W. Susilo, and H. J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electronics Express*, vol. 6, no. 5, pp. 237-243, 2009.

[54] E. K. Ryu and T. Takagi, "Efficient conjunctive keyword-searchable encryption," in *Advanced Information Networking and Application Workshops, 2007, AINAW '07. 21st International Conference on*, vol. 1, pp. 409-414, 2007.

[55] A. Sahai and B. Waters, "Fuzzy identity-based encryption," *Lecture Notes in Computer Science*, vol. 3494, pp. 457-473, 2005.

[56] J. W. Seo, D. H. Yumb, and P. J. Lee, "Proxy-invisible CCA-secure type-based proxy re-encryption without random oracles," *Theoretical Computer Science*, vol. 491, pp. 83-93, 2013.

[57] V.R.L. Shen, W. C. Huang, and T. L. Chen, "A time- bound hierarchical access control for multicast systems," *Proceedings of the 2012 International Conference on Machine Learning and Cybernetics*, pp.543-548, 2012.

[58] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," *Proceedings of IEEE symposium on security and privacy*, pp. 350-364, 2007.

[59] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pp.44-55, 2000.

[60] H. M. Sun, K. H. Wang, and C. M. Chen, "On the security of an efficient time-bound hierarchical key management scheme," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 2, pp. 159-160, 2009.

[61] Q. Tang, "Type-based proxy re-encryption and its construction," *Proceedings of INDOCRYPT 2008, Lecture Notes in Computer Science*, vol. 5365, pp. 130-144, 2008.

[62] W. G. Tzeng, "A time-bound cryptographic key assignment scheme for access control in a hierarchy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 1, pp. 182-188, 2002.

[63] S. S. Vivek, S. S. D. Selvi, V. Radhakishan, and C. P. Rangan," Conditional proxy re-encryption—a more efficient construction," *Proceedings of CNSA 2011, Communications in Computer and Information Science*, vol. 196, pp. 502-512, 2011.

[64] C. Wang, Sherman S.M. Chow, Q. Wang, K. Ren, and W. J. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362-375, 2013.

[65] G. Wang, Q. Liu, and J. Wu, "Achieving fine-grained access control for secure data sharing on cloud servers," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 12, pp. 1443-1464, 2011.

[66] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp. 735-737, 2010.

[67] Q. J. Wang, Q. Liu, J. Wu, and M. Y. Guo, "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," *Computers & Security*, vol. 30, no. 5, pp. 320-331, 2011.

[68] A. Weiss, "Computing in the clouds," *Networker*, vol. 11, no. 4, pp. 16-25, 2007.

[39] J. Weng, M. Chen, Y. Yang, R. Deng, K. Chen, and F. Bao, "CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles," *SCIENCE CHINA Information Sciences*, vol. 53, no.3, pp. 593-606, 2010.

[70] J. Weng, R. H. Deng, C. Chu, X. Ding, and J. Lai, "Conditional proxy re-encryption secure against chosen-ciphertext attack," *Proceedings of the 4th International Symposium on ACM Symposium on Information, Computer and Communications Security (ASIACCS 2009)*, pp. 322-332, 2009.

[71] J. Weng, Y. Yang, Q. Tang, R. H. Deng, and F. Bao, "Efficient conditional proxy reencryption with chosen-ciphertext security," *Proceedings of the 12th International Conference on Information Security (ISC 2009)*, pp. 151-166, 2009.

[72] Jan Wessels, "Application of BAN-logic," *CMG FINANCE B.V.*, 19 April 2001.

[73] H. M. Yang, C. X. Xu, and H. T. Zhao, "An efficient public key encryption with keyword scheme not using pairing," in *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp. 900-904, 2011.

[74] X. Yi, "Security of Chien's efficient time-bound hierarchical key assignment scheme," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 9, pp. 1298-1299, 2005.

[75] X. Yi and Y. Ye, "Security of Tzeng's time-bound key assignment scheme for access control in a hierarchy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 1054-1055, 2003.

[76] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," *Proceedings of the IEEE INFOCOM*, pp. 1-9, 2010.

[77] Y. Zhao, X. Chen, H. Ma, Q. Tang, and H. Zhu, “A new trapdoor-indistinguishable public key encryption with keyword search,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 3, no. 1/2, pp. 72-81, 2012.