國立政治大學資訊科學系

碩士學位論文

具常數密文之去中心化內積加密機制

Decentralized Inner Product Encryption with Constant-size Ciphertexts

指導教授：曾一凡　博士

研究生：高士傑　撰

中 華 民 國 110 年 8 月

DOI:10.6814/NCCU202101323

致謝

想當初，兩年前剛進入應用密碼實驗室，對於密碼學還只是一個懵懵懂懂的初學者，但有幸成爲我的老師曾一凡教授的第一個學生，從剛進來時，就親自教導我一些有關密碼學的相關知識，這讓我對於論文的閱讀上有很大的幫助，除此之外，由於我個人比較不善於口頭表達，所以老師也給了我很多書報討論時報告的準備方向以及報告方式，讓我在一次次的口頭報告時，有小幅的成長。另外，也要感謝左瑞麟教授及王紹睿教授，能在每次書報討論時，往往給予我一些很有建設性的建議或指導，讓我能夠思考一些從沒想過的問題。

再來，我得感謝學長、同學及學弟們在我的研究生活中都佔有了一席之地，大家不管在研究上互相交換意見，或是有時書報來不及準備完成，幫忙頂替，又或者是舉辦國際會議時，即使很累，大家還是很成功地共同合作努力，完成使命，這些都是我一輩子忘記不了的回憶，也期許各位將來能夠順利畢業並且有好的發展，往自己的目標前進。

最後，我得感謝我的家人，謝謝他們能把我培養到唸完研究所，這對於許多的家庭來說也許不是一件簡單的事，所以非常謝謝他們在我背後義無反顧地支持，讓我能夠專心地完成學業，期許不久的將來，我能成爲一位對社會有貢獻的人，不枉他們的培育之恩。

高士傑 謹誌

2021/06

DOI:10.6814/NCCU202101323

中文摘要

隨著近年來科技的興起，分散式的系統架構也越來越多人研究，例如:e 化政府系統。而去中心化架構是分散式系統的其中一種架構，也就是伺服器之間不需任何溝通就能達到分散式的效果，這種架構的優點在於當單點故障發生時，並不會使整個系統被其他攻擊者入侵，讓整個系統更具安全性。

爲了應用在這種去中心化架構上，因此，去中心化的加密機制已被深入地研究數年。不過，大部分都是對於去中心化屬性加密機制的研究，如:Lewko 和 Water [8] 在 2011 年所提出的去中心化屬性加密機制。然而，對於去中心化內積加密機制的相關研究並沒有很多，僅有 Michalevsky 和 Joye [10] 在 2018 年所發表的一篇而已。在此篇論文中，密文的長度會與權威機構的個數成正比的成長，這樣會增加系統儲存空間上的負擔。另外，由於此篇去中心化方式是每個權威機構負責將謂詞向量的每個分量產生解密金鑰的一部份，這也就意味著向量的長度必須與權威機構的個數相同，這對於實際的應用情境較不實用，因爲在內積加密的機制中，接收者的屬性會被一起編碼成一個向量，而不是一個屬性編碼成一個向量的分量。

爲了解決上述問題，我們基於 Attrapadung 等人 [1] 在 2010 年所發表的一個具常數密文的內積加密機制，提出了具常數密文大小的去中心化內積加密機制，此機制密文大小與權威機構個數和向量長度無關，除此之外，我們也實作了我們的機制與 Michalevsky 和 Joye 的機制，並對演算法做執行時間的比較，實驗結果顯示大部分的演算法，我們具有較佳的表現，最後，我們也提出相關的安全性證明，證明機制難以被破解。

關鍵字：去中心化內積加密、常數密文、雙線性配對

DOI:10.6814/NCCU202101323

Abstract

With the rise of technology in recent years, more people are studying distributed system architecture, such as e-government system. The decentralized architecture is one of the architectures of the distributed system, that is, the decentralization can be achieved without any communication between the servers. The advantage of this architecture is that when a single point of failure occurs, it does not cause the system invaded by other attackers, making the entire system more secure.

In order to apply to this decentralized system, therefore, the decentralized encryption has been intensively studied for several years. Nevertheless, most of them are researches on decentralized attribute-based encryption, such as the decentralized attribute-based encryption proposed by Lewko and Waters [8] in 2011. However, there is not much research on decentralized inner product encryption, only a work published by Michalevsky and Joye [10] in 2018. In their construction, the length of the ciphertext is proportional to the number of authorities, which will increase the burden on the system storage space. In addition, since the decentralization method in this work is that each authority is responsible for generating a part of private key for an element of the predicate vector. It means that the length of the vector must be the same as the number of authorities. This is impractical in reality. In the inner product encryption, the receiver's attributes will be encoded together into a vector, rather than an attribute encoded into an element of a vector.

In order to solve the above problems, based on the inner product encryption achieving constant-size ciphertexts published by Attrapadung [1] in 2010, we

DOI:10.6814/NCCU202101323

proposed a decentralized inner product encryption with constant-size ciphertexts. The length of ciphertext of our work is independent of the number of authorities and the length of the vector. Besides, we implement our scheme and the scheme of Michalevsky and Joye, and compare the execution time of the algorithms. The experiment result shows that the most of our algorithms have better performance. Finally, we also present related security proof, which proves that our work is difficult to break.

Keywords: Decentralized Inner Product Encryption, Constant-size Ciphertexts, Bilinear Pairing

DOI:10.6814/NCCU202101323

Contents

DOI:10.6814/NCCU202101323

DOI:10.6814/NCCU202101323

List of Tables

DOI:10.6814/NCCU202101323

List of Figures

DOI:10.6814/NCCU202101323

List of Theorems

DOI:10.6814/NCCU202101323

List of Definitions

DOI:10.6814/NCCU202101323

Chapter 1

Introduction

Identity-based encryption (IBE) was first introduced by Shamir [13] in 1985. It is a kind of encryption which allows the sender to use the recipient's identity (for example, a student ID number, an employee ID) to encrypt a message that he/she wants to transfer. The first IBE scheme is proposed by Boneh and Franklin [2] in 2001. However, a drawback of IBE is that the encrypted data can be only shared at coarse-grained control level. This is not impractical in real world, because the sender should know the particular recipient in advance. In the system, there may have a lot of recipients. If IBE is used, it may overburden the system, because all data needs to be encrypted by different recipients' identities, depending on the total number of recipients. Therefore, IBE is not adequate for actual case.

Thus, Sahai and Waters gave a fuzzy IBE scheme [12] in 2005, which is the concept of attribute-based encryption (ABE), the sender can use a set of attributes that the recipients who owns to encrypt a message. ABE is a fine-grained control system which will make the whole system more flexible, that is, the sender only needs to define the policy determining who is allowed to recover the encrypted data. Unlike IBE, the sender have to know the recipient's identity in every encryption.

Inner product encryption (IPE) is a kind of predicate encryption. IPE can be regarded as a variant of ABE, hence, it also provides fine-grained control over access to encrypted data. The first IPE scheme was first presented by Katz, Sahai and Waters [5] in 2008, which requires $\mathcal{O}(n)$ pairing computations for decryption with $\mathcal{O}(n)$ size of public key, master secret key, private key and ciphertext is constructed by composite order groups, where n is the length of vector. In an IPE scheme, each ciphertext associated with an attribute vector $\vec{Y}$ can be decrypted by a private

DOI:10.6814/NCCU202101323

key associated with a predicate vector $\vec{X}$ if and only if the predicate function $R(\vec{X}, \vec{Y}) = 1$, which represents that the inner product of $\vec{X}$ and $\vec{Y}$ is zero.

Until now, there are several IPE schemes have been published. In 2010, Lewko *et al.* [7] proposed a fully secure IPE scheme by dual pairing vector spaces. In the same year, Attrapadung and Libert [1] proposed two IPE schemes with constant-size ciphertexts. One of schemes is selectively secure, and the other is adaptively secure. Besides, pairing times for decryption is also constant, which significantly improves upon $\mathcal{O}(n)$. In 2011, Park [11] presents the first IPE scheme under the standard assumption. In 2015, Tan and Zhang [18] proposed an IPE scheme with multiplicative homomorphic property which supports multiparty cloud computation. In 2016, Kim *et al.* [6] presented an IPE scheme required only n exponentiation and three pairing computations for decryption. In 2019, Zhang *et al.* [17] proposed an IPE scheme which the time cost of encryption and key generation only needs $\mathcal{O}(n)$. However, compared with some previous works which the time cost of encryption and key generation needs $\mathcal{O}(n^2)$, the scheme is more efficient. In 2020, Soroush *et al.* [14] presented the first verifiable IPE scheme, which guarantees that corrupted encryptors and authorities, even when colluding together, are not able to generate ciphertext and private key. In the same year, Tseng *et al.* [15] introduced an efficient IPE scheme which the length of private key is independent of the length of the predicate vector and the decryption cost is only one pairing computation.

Traditional IPE scheme is centralized architecture which needs a trusted server to generate the private key for users. However, this means that the trusted server has to monitor all attributes, but it is not realistic. For example, a sender encrypts the personal data of patients with COVID-19 which can be decrypted by anyone who is a doctor of Wanfang Hospital or a major executives of Department of Health of Taipei City or a major executives of Ministry of Health and Welfare. In this case, the trusted server needs to maintain these three attributes of those who meet the access policy. In reality, these three attributes are maintained by three different authorities, namely, Wanfang Hospital, Taipei City Government, and Ministry of Health and Welfare, respectively. Besides, IPE is constructed under single authority framework, which will be harmful when a single point of failure occurs. That is, when the master secret key is exposed to an intentional attacker, the system will crush.

Decentralized IPE (DIPE) can solve the problem we mention above. DIPE uses multiple different authorities to encrypt a message. Each authority outputs a part of private key to

DOI:10.6814/NCCU202101323

user, and then the user will combine them to get the whole private key. Besides, there is no communication between authorities. In addition, DIPE system will not get into danger, when single point of failure occurs.

There have already been many studies on multi-authority ABE. In 2007, Chase [4] proposed an multi-authority ABE scheme which allows the sender to specify for each authority k a set of attributes monitored by that authority and a number d_k so that the message can be recovered only by a user who has at least d_k of the given attributes from each authority. However, this scheme still needs a central authority to know the state of each authorities. In 2011, Lewko and Waters [8] proposed a decentralized ABE scheme which does not require any central authority. In 2021, Zhang *et al.* [16] proposed a decentralized CP-ABE scheme with enhanced privacy and user collusion resilience.

In 2018, the first DIPE scheme was introduced by Michalevsky and Joye [10]. However, the large ciphertext length which is $\mathcal{O}(nk)$ group elements may cause the storage burden of the system, where n is the length of attribute/predicate vector and k is the parameter of k-linear assumption.

According to an analysis on collusion attacks of decentralized ABE, proposed by Meamari *et al.* [9] in 2020. Decentralized architecture addresses several issues related to centralized architecture. There are three type of collusion attacks, that is, collusion among data users, collusion among authorities, and collusion among authorities and data user. In our work, we can resist the first type of collusion, because each user has a different global identity (GID), which prevents users from combining their private keys to decrypt a ciphertext. For the second type of collusion, our work is able to resist collusion among at most $n-1$ authorities, where n is the total number of authorities. In our work, the third type of collusion is similar as the second one, the only difference is that there is a user in the collusion model. Thus, we can also resist collusion among at most $n-1$ authorities and a user.

In decentralized ABE scheme, the ciphertext attributes are a set of attributes can be encoded as access matrix (or more). If a user having certain attributes i belonging to an authority, then, the authority compute the corresponding partial key for user. Therefore, the scheme of Michalevsky and Joye [10] used the same concept on DIPE. In their scheme, they generate every partial keys for every element of a predicate vector by authorities managing the certain attributes. It means that each authority is responsible for each element of predicate vector. However, in IPE, a set of

DOI:10.6814/NCCU202101323

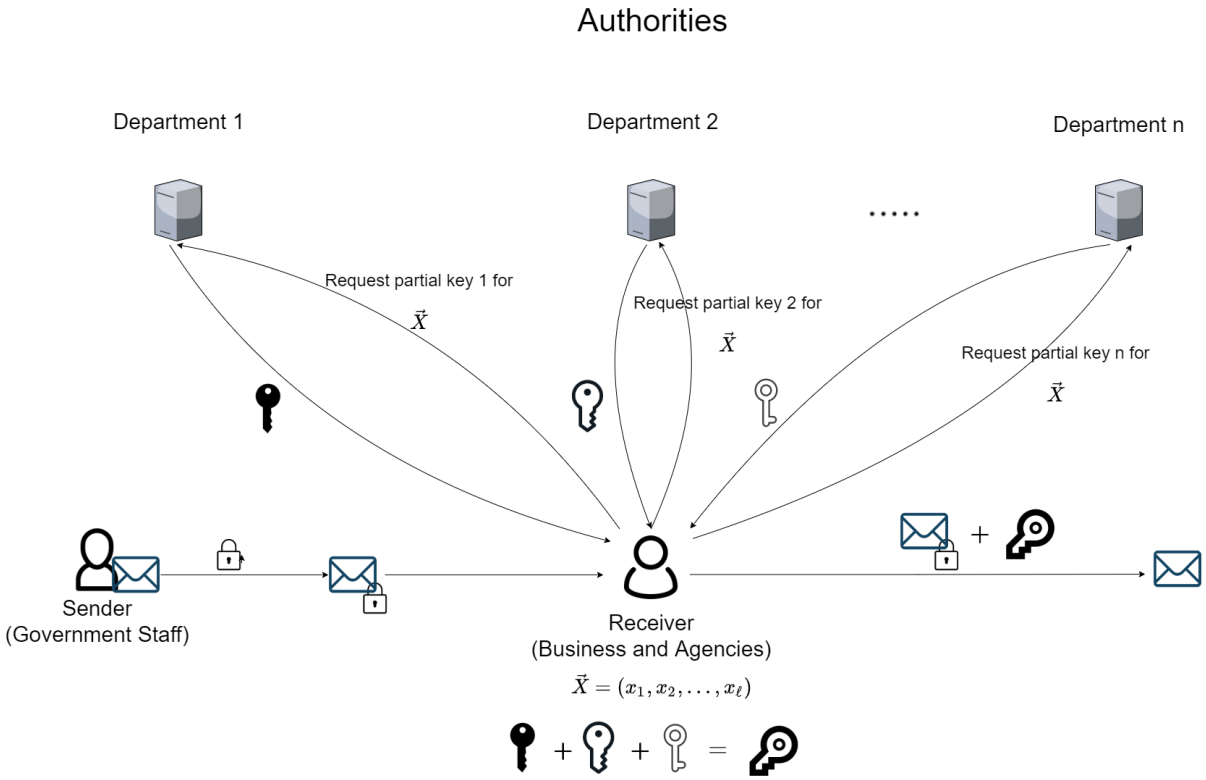


Figure 1: The scenario of DIPE on the e-government network

attributes may encoded as a vector that each element does not belonging to any certain authority. Thus, in our scheme, every authority outputs a partial key for a whole predicate vector of user. Nevertheless, it sacrifices some memory storage on storing private key.

Our proposed scheme can be applied to decentralized network. Therefore, it is more suitable for multi-authority architecture. For example, Figure 1 shows the scenario of our DIPE scheme on the e-government network. The data senders of the network are government staffs who can share the encrypted data to the network. Then, the data receivers of the network are some businesses and agencies they can access the encrypted data provided by government if they are valid users. Finally, the authorities of the system are the government departments which are responsible for validating e-government user's attributes. After the receiver receives all partial keys, he/she can perform the decryption procedure.

1.1 Contribution

Our goal is to minimize the length of ciphertext. In our paper, we propose a DIPE scheme with constant-size ciphertexts and we give a formal security proof which is selectively IND-CPA

DOI:10.6814/NCCU202101323

secure under q-DBDHE assumption. Besides, we implement our construction in Python with Charm-Crypto library.

1.2 Organization

In Chapter 2, we introduce the concept of predicate encryption. Then, we describe the formal definition for IPE scheme, DIPE scheme, complexity assumption used in our proof, and security model. In Chapter 3, we review a constant-size IPE scheme proposed by Attrapadung *et al.* in 2010 and a DIPE scheme proposed by Michalevsky *et al.* in 2018. In Chapter 4, we describe our proposed scheme in detail and show the correctness. In Chapter 5, we describe the security proof of our scheme. In Chapter 6, we give a comparison of our DIPE scheme and the first DIPE scheme. In Chapter 7, some implementation results of our work are shown, and the conclusion is presented.

DOI:10.6814/NCCU202101323

Chapter 2

Preliminaries

In this chapter, we introduce predicate encryption and review bilinear map and q-DBDHE assumption. Besides, we introduce the definition of inner product encryption and decentralized inner product encryption. Finally, we define the security model of our construction.

2.1 Notation

- Given a set S, "randomly choose an element x from the set S", is denoted as $x \xleftarrow{\$} S$.
- By the symbol "$\perp$", it means failed decryption that recover the certain message unsuccessfully.
- By "PPT" algorithm, it means "probabilistic polynomial time" algorithm that can run in polynomial-bounded time.

2.2 Bilinear Map

Assume $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups with prime order p and e is a map having following properties:

- **Bilinearity**: For $u, v \in \mathbb{G}$, and $a, b \in \mathbb{Z}_p$, the equation $e(u^a, v^b) = e(u, v)^{ab}$ holds.
- **Non-Degeneracy**: Assume g is the generator of $\mathbb{G}$, then, $e(g, g) \neq 1$.
- **Computability**: For $u, v \in \mathbb{G}$, there exists an efficient algorithm to compute $e(u, v)$.

DOI:10.6814/NCCU202101323

2.3 Predicate Encryption

Predicate encryption is a paradigm for public-key encryption that generalizes identity-based encryption, attribute-based encryption, inner product encryption, subset predicate encryption and more.

In predicate encryption, the private key SK_X is associated with a predicate attribute X and the ciphertext C_Y is associated with a ciphertext attribute Y. Besides, predicate encryption will emulate a predicate function $R : X \times Y \rightarrow \{0, 1\}$. The decryption procedure is successful if and only if $R(X, Y) = 1$, otherwise, decryption fails.

Different kind of encryption mechanisms have different conditions for successful decryption. For example, identity-based encryption evaluates an equality predicate. Attribute-based encryption emulates whose attributes meet the access policy. Subset predicate encryption determines whether predicate attribute is contained in ciphertext attribute.

2.4 Definition of Inner Product Encryption

Inner product encryption is abbreviated as IPE. It is a kind of predicate encryption. The predicate attribute X is a vector such as $(x_1, ..., x_l)$, and the attribute vector Y is also a vector such as $(y_1, ..., y_l)$. Predicate function evaluates the inner product of predicate vector and attribute vector. The function outputs 1 if and only if $\langle X, Y \rangle = 0$.

The formal definition of inner product encryption is shown as follow.

Definition 1 (Definition of inner product encryption)**.** *An inner product encryption scheme consists of four PPT algorithms* ***Setup***, ***KeyGen***, ***Encrypt***, ***Decrypt***, *respectively.*

- $Setup(1^\lambda)$. Taking as input a security parameter 1^λ, the algorithm outputs a public key PK and a master secret key MSK.
- $KeyGen(PK, MSK, \vec{X})$. Taking as inputs a public key PK, a master secret key MSK, and a predicate vector $\vec{X}$, the algorithm outputs a private key associated with $\vec{X}$.
- $Encrypt(PK, M, \vec{Y})$. Taking as inputs a public key PK, a message M and an attribute vector $\vec{Y}$, the algorithm outputs a ciphertext C associated with $\vec{Y}$.

DOI:10.6814/NCCU202101323

- $Decrypt(PK, sk_{\vec{X}}, C, \vec{X}, \vec{Y})$. Taking as inputs a public key PK, a private key $sk_{\vec{X}}$, a ciphertext C, a predicate vector $\vec{X}$, and an attribute vector $\vec{Y}$, the algorithm outputs a message M or $\perp$.

  For correctness, we need the PK and MSK that generate by $Setup(1^{\lambda})$, for any predicate vector $\vec{X}$, we have $sk_{\vec{X}}$ that generates by $KeyGen(PK, MSK, \vec{X})$, and attribute vector $\vec{Y}$:

  - If $\langle \vec{X}, \vec{Y} \rangle = 0$, then $Decrypt(PK, sk_{\vec{X}}, Encrypt(PK, M, \vec{Y})) = M$.
  - If $\langle \vec{X}, \vec{Y} \rangle \neq 0$, then $Decrypt(PK, sk_{\vec{X}}, Encrypt(PK, M, \vec{Y})) = \perp$.

2.5 Definition of Decentralized Inner Product Encryption

Decentralized inner product encryption is abbreviated as DIPE. The difference between DIPE and IPE is that the private key of DIPE is generated by multiple authorities, while the private key of IPE is generated by a centralized authority.

2.5.1 System Model

In our DIPE architecture, which consists of three entities, that is, Sender, Receiver, Authorities, respectively.

Sender is a participant of the system who transfer the encrypted data to receiver. The data is encrypted by an attribute vector before delivered to receiver.

Authorities are responsible for issuing the partial keys for receiver who makes a request to obtain partial keys. The authorities will issue partial keys by the predicate vector of receiver.

Receiver is a participant who wants to recover the encrypted data. After receiver receives all the partial keys from authorities, receiver will perform decryption procedure to recover the data.

2.5.2 Definition of DIPE

The formal definition of decentralized inner product encryption is shown as follow.

DOI:10.6814/NCCU202101323

Definition 2 (Definition of decentralized inner product encryption)**.** *A decentralized inner product encryption scheme consists of five PPT algorithms* ***Setup***, ***AuthSetup***, ***KeyGen***$_{A_i}$, ***Encrypt***, ***Decrypt***, *respectively.*

- $Setup(1^\lambda)$. A authority in the system or a third party will run the algorithm. Taking as input a security parameter 1^λ, the algorithm outputs a public parameter pp.
- $AuthSetup(pp, i)$. All authorities will run the algorithm. Taking as inputs a public parameter pp, and a number i, the algorithm outputs a master secret key MSK_i and a public key PK_i of each authority, where i is the index of authority.
- $KeyGen_{A_i}(pp, MSK_i, GID, \vec{X})$. All authorities will run the algorithm. Taking as inputs a public parameter pp, a master secret key MSK_i, a global identity GID and a predicate vector $\vec{X}$, the algorithm outputs a partial key of private key associated with $\vec{X}$ generated by i_{th} authority.
- $Encrypt(pp, \{PK_i\}_{i=1,\dots,n}, M, \vec{Y})$. A sender will run the algorithm. Taking as inputs a public parameter pp, all the public keys of each authority $\{PK_i\}_{i=1,\dots,n}$, a message M and an attribute vector $\vec{Y}$, the algorithm outputs a ciphertext C associated with $\vec{Y}$.
- $Decrypt(\{sk_i\}_{i=1,\dots,n}, C, \vec{Y})$. A receiver will run the algorithm. Taking as inputs all the partial key of private keys of each authority $\{sk_i\}_{i=1,\dots,n}$, a ciphertext C, and an attribute vector $\vec{Y}$, the algorithm outputs a message M or $\perp$.

  For correctness, we need a pp that generates by $Setup(1^\lambda)$. $\{PK_i\}_{i=1,\dots,n}$ and $\{MSK_i\}_{i=1,\dots,n}$ generated by $AuthSetup(pp, i)$. For any predicate vector $\vec{X}$, we have $\{sk_i\}_{i=1,\dots,n}$ that generates by $KeyGen_{A_i}(pp, MSK_i, GID, \vec{X})$, and attribute vector $\vec{Y}$:

  - If $\langle \vec{X}, \vec{Y} \rangle = 0$, then
    $Decrypt(\{sk_i\}_{i=1,\dots,n}, Encrypt(pp, \{PK_i\}_{i=1,\dots,n}, M, \vec{Y}), \vec{Y}) = M.$
  - If $\langle \vec{X}, \vec{Y} \rangle \neq 0$, then
    $Decrypt(\{sk_i\}_{i=1,\dots,n}, Encrypt(pp, \{PK_i\}_{i=1,\dots,n}, M, \vec{Y}), \vec{Y}) = \perp.$

DOI:10.6814/NCCU202101323

2.6 Complexity Assumption and Hard Problem

To prove security we use a generalization of bilinear Diffie-Hellman problem first proposed in [3].

Definition 3 (The q-Decisional Bilinear Diffie-Hellman Exponent Problem)**.** *Let $\mathbb{G}$ be a group. g is a generator of $\mathbb{G}$, and $\gamma, s \xleftarrow{\$} \mathbb{Z}_p$ are two integers. Given a tuple:*

$$(g, g^{\gamma}, g^{\gamma^2}, ..., g^{\gamma^{\ell}}, g^{\gamma^{\ell+2}}, ..., g^{\gamma^{2\ell}}, g^s, T)$$

and decides if $T = e(g,g)^{\gamma^{\ell+1}s}$ or $T \xleftarrow{\$} \mathbb{G}_T$ is a random element of $\mathbb{G}_T$.

2.7 Security Model

Our scheme is baesd on indistinguishability under selective chosen-plaintext attacks (sIND-CPA). "Indistinguishability" means that given a ciphertext, which is the encryption of two message chosen by an adversary, the adversary tries to tell which of the two messages is encrypted. Besides, "chosen-plaintext attacks" means that an adversary is allowed to obtain the ciphertext for the plaintext of its choice. Finally, "selective" means that an adversary chooses a target vector and submits to the challenger before Setup phase.

Definition 4 (sIND-CPA security model)**.** *Assume $\mathcal{A}$ is a probabilistic polynomial-time adversary in the game. $\mathcal{A}$ interacts with challenger $\mathcal{C}$ in the game. The scheme is selectively secure if PPT adversary has negligible advantage in the game.*

- ***Initialization***.

  $\mathcal{A}$ chooses an attribute vector $\vec{Y}^* = (y_1^*, y_2^*, ..., y_\ell^*)$ and sends $\vec{Y}^*$ to $\mathcal{C}$.

- ***Setup***.

  $\mathcal{C}$ runs the Setup algorithm to generate PK_i and MSK_i, where $1 \leq i \leq n$, is the index of authority. $\mathcal{C}$ sends $PK_1, ..., PK_n$ and $MSK_1, ..., MSK_{n-1}$ to $\mathcal{A}$.

- ***Phase1***.

  $\mathcal{A}$ can make polynomially times queries of the following oracle.

  - KeyExtract oracle: $\mathcal{A}$ sends a predicate vector $\vec{X}$ to $\mathcal{C}$, and $\mathcal{C}$ returns the private key of $\vec{X}$. There is a restriction, that is, $\langle \vec{X}, \vec{Y}^* \rangle \neq 0$.

DOI:10.6814/NCCU202101323

- ***Challenge***.

  $\mathcal{A}$ submits two distinct message M_0, M_1 to $\mathcal{C}$. $\mathcal{C}$ then randomly chooses $\beta \in \{0,1\}$ and generates ciphertexts $C^* = Encrypt(pp, \{PK_i\}_{i=1,\ldots,n}, M_\beta, \vec{Y}^*)$. Then, $\mathcal{C}$ sends C^* to $\mathcal{A}$.

- ***Phase2***.

  Same as Phase1.

- ***Guess***.

  $\mathcal{A}$ will output a bit $\beta' \in \{0,1\}$ and win the game if $\beta' = \beta$.

  The advantage of $\mathcal{A}$ winning the game is defined as

$$\mathbf{Adv}^{\text{sIND-CPA}}(\mathcal{A}) = \left| Pr[\beta' = \beta] - \frac{1}{2} \right|.$$

A DIPE scheme is sIND-CPA secure if for all PPT adversaries $\mathcal{A}$, $\mathbf{Adv}^{\text{sIND-CPA}}(\mathcal{A})$ is negligible.

DOI:10.6814/NCCU202101323

Chapter 3

Related Works

In this chapter, we first introduce the centralized IPE scheme proposed by Attrapadung *et al.* [1] in 2010. Our proposed DIPE scheme is based on their constant-size zero IPE scheme from spatial encryption. Second, we introduce the first DIPE scheme presented by Michalevsky *et al.* [10] in 2018. In their scheme, the security of their work is based on Symmetric External Diffie-Hellman (SXDH) assumption and K-Lin (KLIN) assumption. In addition, by using vector commitment, their work can protect the receiver privacy.

3.1 Attrapadung *et al.*'s Scheme

In 2010, Attrapadung and Libert proposed a new pairing-based inner product encryption construction providing constant-size ciphertexts, that is, the length of ciphertexts is independent from the length of attribute vector. Besides, the construction supports security against selective adaversary, which the adversary should provide the target vector before setup phase.

The Attrapadung *et al.*'s selectively secure scheme is shown as follow.

Setup$(1^\lambda, 1^\ell)$

The algorithm performs the following steps:

1. Randomly choose bilinear groups $\mathbb{G}, \mathbb{G}_T$ of prime order p with a generator $g \xleftarrow{\$} \mathbb{G}$.
2. Randomly choose $\alpha, \alpha_0, ..., \alpha_\ell \xleftarrow{\$} \mathbb{Z}_p$. Let $\vec{\alpha} = (\alpha_1, ..., \alpha_\ell)$.
3. Output the system public parameters $PK = (g, g^{\alpha_0}, g^{\vec{\alpha}}, Z = e(g,g)^\alpha)$.
4. Output the master secret key $MSK = g^\alpha$.

DOI:10.6814/NCCU202101323

KeyGen$(PK, MSK, \vec{X})$

The authority will perform the following steps to generate the private key for the receiver.

1. Return failure symbol $\perp$ if $x_1 = 0$.
2. Randomly choose $t \xleftarrow{\$} \mathbb{Z}_p$.
3. Output the private key $sk_{\vec{X}} = (D_0, D_1, K_2, ..., K_\ell)$ where

$$D_0 = g^t, \qquad D_1 = g^{\alpha+\alpha_0 t}, \qquad \{K_i = (g^{-\alpha_1 \frac{x_i}{x_1}} g^{\alpha_i})^t\}_{i=2,...,\ell}$$

Encrypt$(PK, M, \vec{Y})$

The sender will compute the ciphertexts by the following steps:

1. Choose $s \xleftarrow{\$} \mathbb{Z}_p$.
2. Output the ciphertexts as $C = (E_0, E_1, E_2)$, where

$$E_0 = M \cdot e(g,g)^{\alpha s}, \qquad E_1 = (g^{\alpha_0} g^{\langle \vec{\alpha}, \vec{Y} \rangle})^s, \qquad E_2 = g^s$$

Decrypt$(PK, sk_{\vec{X}}, C, \vec{X}, \vec{Y})$

To decrypt, the receiver uses the ciphertet C and private key $sk_{\vec{X}}$ as inputs to obtain the decryption result.

1. Compute the blinding factor as

$$\frac{e(D_1 \cdot K_2^{y_2} ... K_\ell^{y_\ell}, E_2)}{e(E_1, D_0)} = e(g,g)^{\alpha s}$$

2. $M = E_0 / e(g,g)^{\alpha s}$, if $\langle \vec{X}, \vec{Y} \rangle = 0$; Otherwise, return $\perp$.

3.2 Michalevsky *et al.*'s Scheme

In 2018, Michalevsky and Marc Joye proposed first DIPE scheme with a proof of being policy-hiding under k-linear assumption, which hides the encryption policy. In their DIPE scheme, each authority is responsible for a element of predicate vector which makes the total

DOI:10.6814/NCCU202101323

number of authorities and the length of vector be the same, and then generates a partial key of private key. Besides, they use a random oracle to generate masking value μ_i that depends on the combination of a public key of authority, a GID, and a predicate vector v. If it is a valid user, these masking values will allow the user to get the correct key. In addition, the length of the ciphertexts is $O(nk)$, where n is the total number of authorities, and k is the parameter of k-linear assumption. However, larger ciphertexts will burden the receiver on storing.

The Michalevsky *et al.*'s decentralized inner product encryption scheme is shown as follow.

Setup(λ)

The algorithm performs the following steps:

1. Input a security parameter λ. Then, generate $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. Let g_1 and g_2 be two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.
2. Randomly choose a matrix $A \in \mathbb{Z}_p^{(k+1)\times k}$ and a random matrix $U \in \mathbb{Z}_p^{(k+1)\times(k+1)}$.
3. Output the system public parameters $pp = (g_1, g_2, g_1^A, g_1^{U^T A})$.

AuthSetup(pp, i)

The algorithm inputs the public parameter pp and authority number i, and perfoms the following steps:

1. Randomly choose a random matrix $W_i \in \mathbb{Z}_p^{(k+1)\times(k+1)}$, a vector $\alpha_i \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, and a random $\sigma_i \in \mathbb{Z}_p$.
2. Output the master secret key $MSK_i = \{W_i, \alpha_i, \sigma_i\}$.
3. Output the public key $PK_i = \{g_1^{W_i^\intercal A}, e(g_1, g_2)^{\alpha_i^\intercal A}, y_i = g_2^{\sigma_i}\}$

Encrypt$_{pp}(\{PK_i\}_{i=1,...,n}, \vec{X}, M)$

The algorithm inputs the public keys $\{PK_i\}_{i=1,...,n}$, attribute vector $\vec{X}$ and a message M. The sender computes the ciphertexts by following steps.

1. Let $\vec{X} = (x_1, ..., x_n) \in \mathbb{Z}_p^n$. The algorithm chooses a random vector $s \in \mathbb{Z}_p^k$.
2. Output the corresponding ciphertext $C = \{C_0, \{C_i\}_{i=1,...,n}, C'\}$, which consists of the following components

DOI:10.6814/NCCU202101323

$C_0 = g_1^{As}$,
$C_i = g_1^{(x_i U^\intercal + W_i^\intercal) As}$,
$C' = M \cdot \prod_{i=1}^{n} e(g_1, g_2)^{\alpha_i^\intercal As} = M \cdot e(g_1, g_2)^{\alpha^\intercal As}$

where $\alpha = \sum_{i=1}^{n} \alpha_i$.

KeyGen$_{pp}(\{PK_i\}_{i=1,...,n}, MSK_i, GID, \vec{V})$

The i_{th} authority will execute the algorithm. The algorithm inputs the public keys $\{PK_i\}_{i=1,...,n}$, master secret key MSK_i, GID, and predicate vector $\vec{V}$. The algorithm computes the private key as follow.

1. Compute the masking value $\mu_i \in \mathbb{Z}_p^{k+1}$ by a random oracle $\mathcal{H} : \mathbb{G}_2 \times \{0,1\}^\lambda \times \mathbb{Z}_p^{k+1} \to \mathbb{Z}_p^{k+1}$. The masking term that depend on a combination of an element of $\mathbb{G}_2$, a GID, and the predicate vector $\vec{V}$.

$$\mu_i = \sum_{j=1}^{i-1} \mathcal{H}(y_j^{\sigma_i}, GID, \vec{V}) - \sum_{j=i+1}^{n} \mathcal{H}(y_j^{\sigma_i}, GID, \vec{V})$$

   We can note that $\sum_{i=1}^{n} \mu_i = 0$.

2. By using $H_1(GID, \vec{V}), ..., H_{k+1}(GID, \vec{V})$, we generate g_2^h, where $h \in \mathbb{Z}_p^{k+1}$. Besides, the exponent h is defined implicitly by the hash function. We denote

$$H(GID, \vec{V}) = \Big(H_1(GID, \vec{V}), ..., H_{k+1}(GID, \vec{V})\Big)^\intercal$$

3. Output the private key $sk_{i,GID,\vec{V}} = \{H(GID, \vec{V}), K_i\}$, where

$$K_i = g_2^{\alpha_i - v_i W_i h + \mu_i}$$

.

Decrypt$_{pp}(\{sk_{i,GID,\vec{V}}\}_{i=1,...,n}, C, \vec{V})$

To decrypt, the receiver inputs the private key $\{sk_{i,GID,\vec{V}}\}_{i=1,...,n}$, ciphertext C and predicate vector $\vec{V}$ to obtain the decryption result.

DOI:10.6814/NCCU202101323

1. Compute

$$e\left(C_0, \prod_{i=1}^{n} K_i\right) \cdot e\left(\prod_{i=1}^{n} C_i^{v_i}, H(GID, \vec{V})\right) = e(g_1, g_2)^{\alpha^\intercal As} \cdot e(g_1, g_2)^{\langle \vec{X}, \vec{V} \rangle h^\intercal U^\intercal AS}$$

and get the decryption result by

$$\frac{C'}{e(g_1, g_2)^{\alpha^\intercal As} \cdot e(g_1, g_2)^{\langle \vec{X}, \vec{V} \rangle h^\intercal U^\intercal AS}}$$

2. If $\langle \vec{X}, \vec{V} \rangle = 0$, we can obtain $e(g_1, g_2)^{\alpha^\intercal As}$ and can recover the message M. Otherwise, return $\perp$.

DOI:10.6814/NCCU202101323

Chapter 4

The Proposed Scheme

In this chapter, we will present a decentralized inner product encryption scheme with sIND-CPA security based on [1]. The notations used in the proposed scheme are defined in Table 1.

Table 1: Notations

Notation	Description
$\mathbb{G}$	a bilinear group with prime order p
$\mathbb{G}_T$	a bilinear group by pairing of the element of $\mathbb{G}$
e	a bilinear mapping; $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$
g	a generator of $\mathbb{G}$
n	total number of authorities
ℓ	the length of predicate/attribute vector
A_i	ith authority
pp	public parameter
PK_i	public key of authority i
MSK_i	master secret key of authority i
$\vec{X}$	a predicate vector
$\vec{Y}$	an attribute vector
GID	an identity of the user
M	a message

There are five PPT algorithms. The proposed decentralized selectively secure IPE scheme with constant-size ciphertexts is described as follow.

DOI:10.6814/NCCU202101323

Setup(1^λ)

The algorithm performs the following steps:

1. Randomly choose bilinear groups $\mathbb{G}, \mathbb{G}_T$ of prime order p with a generator $g \xleftarrow{\$} \mathbb{G}$.
2. Choose an one-way hash function, $H : \{0,1\}^* \times \mathbb{Z}_p^\ell \to \mathbb{G}$.
3. Output a public parameter $pp = \{g, H\}$.

AuthSetup(pp, i)

Each of authority in the system will perform the following steps to generate a public key and a master secret key.

1. Choose $\overline{\alpha}_i \xleftarrow{\$} \mathbb{Z}_p$, where $\alpha = \sum_{i=1}^{n} \overline{\alpha}_i$.
2. Choose $\alpha_{0,i} \xleftarrow{\$} \mathbb{Z}_p$, where $\alpha_0 = \sum_{i=1}^{n} \alpha_{0,i}$.
3. Choose $\alpha_{1,i}, \alpha_{2,i}, ..., \alpha_{\ell,i} \xleftarrow{\$} \mathbb{Z}_p$, where $\alpha_j = \sum_{i=1}^{n} \alpha_{j,i}$, and $j \in \{1, ..., \ell\}$.
4. Output a public key of authority i, $PK_i = \{g^{\alpha_{0,i}}, g^{\alpha_{1,i}}, ..., g^{\alpha_{\ell,i}}, Z_i = e(g,g)^{\overline{\alpha}_i}\}$.
5. Output a master secret key of authority i, $MSK_i = \{g^{\overline{\alpha}_i}, \alpha_{0,i}, \alpha_{1,i}, ..., \alpha_{\ell,i}\}$.

KeyGen$_{\mathbf{A_i}}(pp, MSK_i, GID, \vec{X})$

Each of authority in the system will perform the following steps to generate a part of private key for the receiver in the system.

1. Return failure symbol $\perp$ if $x_1 = 0$.
2. Output the private key $sk_i = \{D_0, D_{1,i}, \{K_{j,i}\}_{j=2,...,\ell}\}$ where

$$D_0 = H(GID, \vec{X})$$
$$D_{1,i} = g^{\overline{\alpha}_i} \cdot H(GID, \vec{X})^{\alpha_{0,i}}$$
$$\{K_{j,i} = H(GID, \vec{X})^{-\alpha_{1,i}\frac{x_j}{x_1}} \cdot H(GID, \vec{X})^{\alpha_{j,i}}\}_{j=2,...,\ell}$$

Encrypt$(pp, \{PK_i\}_{i=1,...,n}, M, \vec{Y})$

The sender will compute the ciphertext by the following steps:

1. Choose $s \xleftarrow{\$} \mathbb{Z}_p$.

DOI:10.6814/NCCU202101323

2. Output the ciphertexts as $C = \{E_0, E_1, E_2\}$, where

$$E_0 = M \cdot (\prod_{i=1}^{n} e(g,g)^{\overline{\alpha}_i})^s$$
$$E_1 = ((\prod_{i=1}^{n} g^{\alpha_{0,i}}) \cdot (\prod_{i=1}^{n} g^{\alpha_{1,i}})^{y_1} \cdot ... \cdot (\prod_{i=1}^{n} g^{\alpha_{\ell,i}})^{y_\ell})^s$$
$$E_2 = g^s$$

Decrypt$(\{sk_i\}_{i=1,...,n}, C, \vec{Y})$

To decrypt, the receiver use the ciphertet C and private key sk_i to recover the message M.

1. If $\langle \vec{X}, \vec{Y} \rangle = 0$, do the following computation; Otherwise, return $\perp$.

2. Compute the blinding factor as

$$\frac{e((\prod_{i=1}^{n} D_{1,i}) \cdot (\prod_{i=1}^{n} K_{2,i})^{y_2} ... (\prod_{i=1}^{n} K_{\ell,i})^{y_\ell}, E_2)}{e(E_1, D_0)} = e(g,g)^{\alpha s}$$

3. Compute $M = E_0 / e(g,g)^{\alpha s}$.

Correctness.

The correctness of decryption algorithm is described as follow.

Compute,

$$\begin{aligned} & \prod_{i=1}^{n} D_{1,i} \\ = \; & \prod_{i=1}^{n} g^{\overline{\alpha}_i + \alpha_{0,i} t} \\ = \; & g^{\sum_{i=1}^{n} \overline{\alpha}_i + \alpha_{0,i} t} \\ = \; & g^{\alpha + \alpha_0 t} \end{aligned}$$

$$\begin{aligned} & (\prod_{i=1}^{n} K_{2,i})^{y_2} ... (\prod_{i=1}^{n} K_{\ell,i})^{y_\ell} \\ = \; & (g^{-\alpha_1 \frac{x_2}{x_1}} g^{\alpha_2})^{t y_2} \cdot ... \cdot (g^{-\alpha_1 \frac{x_\ell}{x_1}} g^{\alpha_\ell})^{t y_\ell} \\ = \; & g^{[(-y_2) \cdot \frac{x_2}{x_1} + ... + (-y_\ell) \cdot \frac{x_\ell}{x_1}] \alpha_1 t} \cdot g^{(\alpha_2 y_2 + ... + \alpha_\ell y_\ell) t} \\ = \; & g^{\langle \vec{\alpha}, \vec{Y} \rangle t} \end{aligned}$$

DOI:10.6814/NCCU202101323

so,

$$
\begin{aligned}
& e((\textstyle\prod_{i=1}^{n} D_{1,i}) \cdot (\prod_{i=1}^{n} K_{2,i})^{y_2} ... (\prod_{i=1}^{n} K_{\ell,i})^{y_\ell}, E_2) \\
= \quad & e(g^{\alpha+\alpha_0 t} \cdot g^{\langle \vec{\alpha}, \vec{Y} \rangle t}, g^s) \\
= \quad & e(g,g)^{\alpha s + \alpha_0 st + \langle \vec{\alpha}, \vec{Y} \rangle st}
\end{aligned}
$$

and,

$$
\begin{aligned}
& e(E_1, D_0) \\
= \quad & e(((\textstyle\prod_{i=1}^{n} g^{\alpha_{0,i}}) g^{\langle \vec{\alpha}, \vec{Y} \rangle})^s, g^t) \\
= \quad & e((g^{\alpha_0} g^{\langle \vec{\alpha}, \vec{Y} \rangle})^s, g^t) \\
= \quad & e(g,g)^{\alpha_0 st + \langle \vec{\alpha}, \vec{Y} \rangle st}
\end{aligned}
$$

finally,

$$
\begin{aligned}
& \frac{e((\prod_{i=1}^{n} D_{1,i}) \cdot (\prod_{i=1}^{n} K_{2,i})^{y_2} ... (\prod_{i=1}^{n} K_{\ell,i})^{y_\ell}, E_2)}{e(E_1, D_0)} \\
= \quad & \frac{e(g,g)^{\alpha s + \alpha_0 st + \langle \vec{\alpha}, \vec{Y} \rangle st}}{e(g,g)^{\alpha_0 st + \langle \vec{\alpha}, \vec{Y} \rangle st}} \\
= \quad & e(g,g)^{\alpha s}
\end{aligned}
$$

$$E_0 / e(g,g)^{\alpha s} = M$$

Thus, the receiver can recover the message by computing the blinding factor $e(g,g)^{\alpha s}$, if $\langle \vec{X}, \vec{Y} \rangle = 0$.

DOI:10.6814/NCCU202101323

Chapter 5

Security Proof

In this chapter, we will introduce some security notion and security model for our DIPE scheme. Then, we will prove our scheme is selectively secure.

5.1 Security Notion

The goal of an adversary is ”Indistinguishability”, that is , given a ciphertext, which is the encryption of one of the two message chosen by the adversary, the adversary will try to tell which of the two messages encrypted. We also use a weaker notion called ”Selective security”, where the adversary is forced to submit a target vector before setting up public parameter. Besides, the attack model for encryption is ”Chosen-Plaintext Attacks” (CPA), that is, the adversary is allowed to obtain the ciphertext for the plaintext of its choosing. For a public key encryption, since the adversary is able to access the public key, the chosen-plaintext attacks can be easily achieved by the adversary.

5.2 Security Proof for Selectively Secure IPE Scheme

In this section, we will prove that our proposed selectively secure scheme is under sIND-CPA secure under q-DBDHE assumption. In our proof, we put a tuple of hard problem on the last authority's parameters to achieve the security of the construction.

Theorem 1. *The proposed DIPE scheme is sIND-CPA secure if the q-DBDHE assumption holds.*

DOI:10.6814/NCCU202101323

Proof. We use the concept of contradiction proof. Assume there is a polynomial-time adversary can wins the sIND-CPA game with non-negligible advantage. Then, we will also have a polynomial-time challenger $\mathcal{C}$ can solve the q-DBDHE assumption. First of all, $\mathcal{C}$ is given an instance of the q-DBDHE problem, that is, $\left\langle g, g^{\gamma}, g^{\gamma^2}, ..., g^{\gamma^{\ell}}, g^{\gamma^{\ell+2}}, ..., g^{\gamma^{2\ell}}, g^{s}, T \right\rangle$, whether T is $e(g,g)^{\gamma^{\ell+1}s}$, or T is a random element of $\mathbb{G}_T$. Then, $\mathcal{C}$ interacts with $\mathcal{A}$ in the game as follows:

Initialization.

$\mathcal{A}$ first sends the target vector $\vec{Y}^* = (y_1^*, y_2^*, ..., y_{\ell}^*)$ to $\mathcal{C}$.

Setup.

1. Set $\vec{h} = (g^{\alpha_{1,n}}, g^{\alpha_{2,n}}, ..., g^{\alpha_{\ell,n}}) = (g^{\gamma}, g^{\gamma^2}, ..., g^{\gamma^{\ell}})$, where $\vec{\alpha}_n = \langle \alpha_{1,n}, \alpha_{2,n}, ..., \alpha_{\ell,n} \rangle$.
2. Choose $\delta \xleftarrow{\$} \mathbb{Z}_p$.
3. Compute $Z_n = e(g,g)^{\overline{\alpha}_n} = e(g^{\gamma}, g^{\gamma^{\ell}})$ and $g^{\alpha_{0,n}} = \left((g^{\gamma})^{y_1^*} (g^{\gamma^2})^{y_2^*} ... (g^{\gamma^{\ell}})^{y_{\ell}^*} \right)^{-1} \cdot g^{\delta}$.
4. $\mathcal{C}$ chooses $\alpha_{0,1}, ..., \alpha_{0,n-1} \xleftarrow{\$} \mathbb{Z}_p$.
5. $\mathcal{C}$ chooses $\alpha_{j,1}, ..., \alpha_{j,n-1} \xleftarrow{\$} \mathbb{Z}_p$, where $j = 1, ..., \ell$.
6. $\mathcal{C}$ chooses $\overline{\alpha}_1, ..., \overline{\alpha}_{n-1} \xleftarrow{\$} \mathbb{Z}_p$.
7. Without loss of generality, assume $\mathcal{A}$ can obtain the first $n-1$ master secret keys MSK_i of authorities, where $i = 1, ..., n-1$. Thus,
   - $\mathcal{C}$ sends $\alpha_{0,1}, ..., \alpha_{0,n-1}$ to $\mathcal{A}$.
   - $\mathcal{C}$ sends $\alpha_{j,1}, ..., \alpha_{j,n-1}$ to $\mathcal{A}$, where $j = 1, ..., \ell$.
   - $\mathcal{C}$ sends $g^{\overline{\alpha}_1}, ..., g^{\overline{\alpha}_{n-1}}$ to $\mathcal{A}$.
8. Submit public keys $PK_i = \{g^{\alpha_{0,i}}, g^{\alpha_{1,i}}, ..., g^{\alpha_{\ell,i}}, Z_i = e(g,g)^{\overline{\alpha}_i}\}$, where $i = 1, ..., n$.

We can note that we implicitly set

$$\overline{\alpha}_n = \gamma^{\ell+1}, \qquad \alpha_{0,n} = -\langle \vec{\alpha}_n, \vec{Y}^* \rangle + \delta, \qquad \alpha_{j,n} = \gamma^j.$$

Phase1.

Assume $\mathcal{A}$ makes q queries in this phase. $\mathcal{C}$ maintains a hash list H-list to store the mapping result of $H(GID, \vec{X})$. Then, $\mathcal{A}$ can query private key for $\vec{X} = (x_1, x_2, ..., x_{\ell})$,

DOI:10.6814/NCCU202101323

where $\langle \vec{X}, \vec{Y^*} \rangle \neq 0$. Then, $\mathcal{C}$ will return the corresponding hash value and private key to $\mathcal{A}$.

1. Hash oracle:

   This oracle takes $\vec{X} \in \mathbb{Z}_p^\ell$ and $GID \in \{0,1\}^*$(global identity) as input, and outputs an element of $\mathbb{G}$. If there exists a record $(GID, \vec{X}, v_k)$ in H-list, return v_k, and computes D_0. Otherwise, do the following steps:

   - Choose $v_k \xleftarrow{\$} \mathbb{Z}_p^*$, and add to the H-list, where $1 \leq k \leq q$.
   - $\mathcal{C}$ implicitly sets

   $$t = \frac{x_1\gamma^\ell + x_2\gamma^{\ell-1} + ... + x_\ell\gamma}{\langle \vec{X}, \vec{Y^*} \rangle} + v_k$$

   - Compute $D_0 = H(GID, \vec{X}) = g^t$, it can be efficiently computed by the instance of q-DBDHE problem.

   $$D_0 = (g^\gamma)^{\frac{x_\ell}{\langle \vec{X}, \vec{Y^*} \rangle}} \cdot ... \cdot (g^{\gamma^{\ell-1}})^{\frac{x_2}{\langle \vec{X}, \vec{Y^*} \rangle}} \cdot (g^{\gamma^\ell})^{\frac{x_1}{\langle \vec{X}, \vec{Y^*} \rangle}} \cdot g^{v_k} = g^t$$

2. KeyExtract oracle:

   Upon receiving a vector $\vec{X} = (x_1, x_2, ..., x_\ell)$ from $\mathcal{A}$, then $\mathcal{C}$ performs as follows.

   - For $K_{j,n} = (g^{-\alpha_{1,n}\frac{x_j}{x_1}} g^{\alpha_{j,n}})^t$, $j = 2, ..., \ell$. Then, the exponent of $K_{j,n}$.

   $$\begin{aligned} &(-\alpha_{1,n}\frac{x_j}{x_1} + \alpha_{j,n})t \\ =&(-\gamma\frac{x_j}{x_1} + \gamma^j) \cdot (\frac{x_1\gamma^\ell + x_2\gamma^{\ell-1} + ... + x_\ell\gamma}{\langle \vec{X}, \vec{Y^*} \rangle} + v_k) \end{aligned}$$

   Consider the coefficient of $\gamma^{\ell+1}$, we compute that

   $$-\frac{x_j}{x_1} \cdot \frac{x_1}{\langle \vec{X}, \vec{Y^*} \rangle} + \frac{x_j}{\langle \vec{X}, \vec{Y^*} \rangle} = 0$$

   . So, $K_{j,n}$ can be efficiently computed by an instance of q-DBDHE problem.

DOI:10.6814/NCCU202101323

– For $D_{1,n} = g^{\overline{\alpha}_n + \alpha_{0,n} t}$. Then, the exponent of $D_{1,n}$.

$$
\begin{aligned}
&\overline{\alpha}_n + \alpha_{0,n} t \\
=&\gamma^{\ell+1} + (-\langle \vec{\alpha}_n, \vec{Y}^* \rangle + \delta) \cdot t \\
=&\gamma^{\ell+1} + (-\langle \vec{\alpha}_n, \vec{Y}^* \rangle) \cdot (\frac{x_1\gamma^\ell + x_2\gamma^{\ell-1} + ... + x_\ell\gamma}{\langle \vec{X}, \vec{Y}^* \rangle} + v_k) + \delta t \\
=&\gamma^{\ell+1} - \frac{(\alpha_1 y_1^* + \alpha_2 y_2^* + ... + \alpha_\ell y_\ell^*)(x_1\gamma^\ell + x_2\gamma^{\ell-1} + ... + x_\ell\gamma)}{\langle \vec{X}, \vec{Y}^* \rangle} - \langle \vec{\alpha}_n, \vec{Y}^* \rangle \cdot v_k + \delta t \\
=&\gamma^{\ell+1} - \frac{(\gamma y_1^* + \gamma^2 y_2^* + ... + \gamma^\ell y_\ell^*)(x_1\gamma^\ell + x_2\gamma^{\ell-1} + ... + x_\ell\gamma)}{\langle \vec{X}, \vec{Y}^* \rangle} - \langle \vec{\alpha}_n, \vec{Y}^* \rangle \cdot v_k + \delta t
\end{aligned}
$$

Consider the coefficient of $\gamma^{\ell+1}$, we compute that

$$1 - \frac{x_1 y_1^* + x_2 y_2^* + ... + x_\ell y_\ell^*}{\langle \vec{X}, \vec{Y}^* \rangle} = 1 - \frac{\langle \vec{X}, \vec{Y}^* \rangle}{\langle \vec{X}, \vec{Y}^* \rangle} = 0$$

So, $D_{1,n}$ also can be efficiently computed by an instance of q-DBDHE problem.

Challenge.

$\mathcal{A}$ submits two message M_0 and M_1. $\mathcal{C}$ computes the challenge ciphertext as follow.

1. Choose $\beta \xleftarrow{\$} \{0, 1\}$.
2. Set $E_2 = g^s$.
3. Compute $E_0 = M_\beta \cdot (\prod_{i=1}^{n-1} Z_i)^s \cdot T$.

$$
\begin{aligned}
E_0 \quad &= M_\beta \cdot Z_1^s \cdot ... \cdot Z_{n-1}^s \cdot T \\
&= M_\beta \cdot e(g^s, g^{\overline{\alpha}_1}) \cdot ... \cdot e(g^s, g^{\overline{\alpha}_{n-1}}) \cdot T
\end{aligned}
$$

DOI:10.6814/NCCU202101323

4. Compute $E_1 = \left((\prod_{i=1}^{n} g^{\alpha_{0,i}}) \cdot (\prod_{i=1}^{n} g^{\alpha_{1,i}})^{y_1^*} \cdot ... \cdot (\prod_{i=1}^{n} g^{\alpha_{\ell,i}})^{y_\ell^*}\right)^s$.

$$\begin{aligned} E_1 &= \left((g^{\alpha_{0,1}} \cdot ... \cdot g^{\alpha_{0,n-1}}) \cdot (g^{\alpha_{1,1}} \cdot ... \cdot g^{\alpha_{1,n-1}})^{y_1^*} \cdot ... \cdot (g^{\alpha_{\ell,1}} \cdot ... \cdot g^{\alpha_{\ell,n-1}})^{y_\ell^*}\right)^s \\ &\cdot \left(g^{\alpha_{0,n}} \cdot (g^{\alpha_{1,n}})^{y_1^*} \cdot ... \cdot (g^{\alpha_{\ell,n}})^{y_\ell^*}\right)^s \\ &= (g^s)^{\sum_{i=1}^{n-1} \alpha_{0,i} + y_1^* \sum_{i=1}^{n-1} \alpha_{1,i} + ... + y_\ell^* \sum_{i=1}^{n-1} \alpha_{\ell,i}} \cdot \left(g^{-\langle \vec{\alpha}_n, \vec{Y}^* \rangle + \delta} \cdot g^{\langle \vec{\alpha}_n, \vec{Y}^* \rangle}\right)^s \\ &= (g^s)^{\sum_{i=1}^{n-1} \alpha_{0,i} + y_1^* \sum_{i=1}^{n-1} \alpha_{1,i} + ... + y_\ell^* \sum_{i=1}^{n-1} \alpha_{\ell,i}} \cdot (g^s)^{\delta} \end{aligned}$$

5. Output $C^* = (E_0, E_1, E_2)$ to $\mathcal{A}$.

Phase2.

Same as Phase1.

Guess.

$\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$. $\mathcal{C}$ outputs 1 if $\beta' = \beta$; otherwise, $\mathcal{C}$ outputs 0.

We then analyze the correctness of the oracles. For the case of Hash oracle, we use an unique value for a corresponding pair $(GID, \vec{X})$, so we have that

$$H(GID, \vec{X}) = (g^{\gamma})^{\frac{x_\ell}{\langle \vec{X}, \vec{Y}^* \rangle}} \cdot ... \cdot (g^{\gamma^{\ell-1}})^{\frac{x_2}{\langle \vec{X}, \vec{Y}^* \rangle}} \cdot (g^{\gamma^{\ell}})^{\frac{x_1}{\langle \vec{X}, \vec{Y}^* \rangle}} \cdot g^{v_k},$$

and hence D_0 is a valid partial key for GID and $\vec{X}$.

For $K_{j,n}$ part of KeyExtract oracle, the coefficient of $\gamma^{\ell+1}$ is 0, so $K_{j,n}$ is a valid partial key for $\vec{X}$.

For $D_{1,n}$ part of KeyExtract oracle, the coefficient of $\gamma^{\ell+1}$ is also 0, so $D_{1,n}$ is also a valid partial key.

Since $\overline{\alpha}_1, \overline{\alpha}_2, ..., \overline{\alpha}_{n-1}$ are random elements of $\mathbb{Z}_p$, and $\alpha_{0,i}, \alpha_{1,i}, ..., \alpha_{\ell,i}$ are also random elements of $\mathbb{Z}_p$, where $i = 1, ..., n-1$, then

DOI:10.6814/NCCU202101323

$$
\begin{aligned}
E_0 &= M_\beta \cdot e(g^s, g^{\overline{\alpha}_1}) \cdot ... \cdot e(g^s, g^{\overline{\alpha}_{n-1}}) \cdot T, \\
E_1 &= (g^s)^{\sum_{i=1}^{n-1} \alpha_{0,i} + y_1^* \sum_{i=1}^{n-1} \alpha_{1,i} + ... + y_\ell^* \sum_{i=1}^{n-1} \alpha_{\ell,i}} \cdot (g^s)^{\delta}, \\
E_2 &= g^s,
\end{aligned}
$$

and hence $C^* = (E_0, E_1, E_2)$ is a valid ciphertext which can be computed by an q-DBDHE tuple.

Next, we will analyze the advantage for $\mathcal{C}$ in breaking the q-DBDHE assumption. Assume that the adversary wins the sIND-CPA game with advantage

$$
\mathbf{Adv}^{\mathsf{sIND\text{-}CPA}}(\mathcal{A}) = \left| Pr[\beta' = \beta] - \frac{1}{2} \right|,
$$

then the $\mathcal{C}$ outputs 1 with probability

$$
\mathbf{Adv}^{\mathsf{sIND\text{-}CPA}}(\mathcal{A}) + \frac{1}{2}.
$$

If T is a random element from $\mathbb{G}_T$, then the message M_β is completely hidden from the adversary's view, since E_0, E_1, E_2 are all independently random elements. Therefore, the advantage of the adversary is

$$
\mathbf{Adv}^{\mathsf{sIND\text{-}CPA}}(\mathcal{A}) = \left| Pr[\beta' = \beta] - \frac{1}{2} \right| = 0,
$$

and $\mathcal{C}$ outputs 1 with probability $\frac{1}{2}$. Finally, the advantage of $\mathcal{C}$ is

$$
\begin{aligned}
&\mathbf{Adv}^{\mathsf{q\text{-}DBDHE}}(\mathcal{C}) \\
=& \Big| Pr[\mathcal{C}(g, g^\gamma, g^{\gamma^2}, ..., g^{\gamma^\ell}, g^{\gamma^{\ell+2}}, ..., g^{\gamma^{2\ell}}, g^s, T = e(g,g)^{\gamma^{\ell+1} s}) = 1] \\
&- Pr[\mathcal{C}(g, g^\gamma, g^{\gamma^2}, ..., g^{\gamma^\ell}, g^{\gamma^{\ell+2}}, ..., g^{\gamma^{2\ell}}, g^s, T \xleftarrow{\$} \mathbb{G}_T) = 1] \Big| \\
=& \left| (\mathbf{Adv}^{\mathsf{sIND\text{-}CPA}}(\mathcal{A}) + \frac{1}{2}) - \frac{1}{2} \right| \\
=& \mathbf{Adv}^{\mathsf{sIND\text{-}CPA}}(\mathcal{A}).
\end{aligned}
$$

Therefore, if there is an adversary wins the sIND-CPA game with non-negligible advantage,

DOI:10.6814/NCCU202101323

then we can construct an algorithm $\mathcal{C}$ to break the q-DBDHE problem with non-negligible advantage in polynomial time.

DOI:10.6814/NCCU202101323

Chapter 6

Comparison

In this chapter, we compare our scheme with [1, 5, 6, 10] in time complexity, space complexity and some properties. Among these works, [1, 5, 6] are IPE schemes and [10] is DIPE scheme. Besides, we implement our scheme and the scheme of [10] in Python language. Thus, we will compare execution time of our algorithms with theirs.

6.1 Comparison

In Table 2, we show the encryption cost and decryption cost of each scheme. In encryption, an exponentiation computation cost is linear with the vector size which is better than others, except [1]. In addition, we only need ℓ times exponentiation computations plus two pairing computations in decryption, while, in [10], needs n times exponentiation computations plus $\mathcal{O}(k)$ pairing computations. Thus, both of the cost for encryption and decryption algorithm is more efficient makes the users of system more time-saving.

DOI:10.6814/NCCU202101323

Table 2: Comparison with the previous schemes in time complexity

	Encryption cost	Decryption Cost
[5]	$(4\ell+1)T_e$	$(2\ell+1)T_p$
[6]	$(2\ell+2)T_e$	$\ell T_e + 3T_p$
[1]	$(\ell+3)T_e$	$\ell T_e + 2T_p$
[10]	$[(2n+1)k^2+(2n+2)k]T_e$	$nT_e + (2k+2)T_p$
Ours	$(\ell+3)T_e$	$\ell T_e + 2T_p$

T_e: The cost of an exponentiation in multiplicative groups.
T_p: The cost of pairing computation.
n: The total number of authorities.
ℓ: The length of predicate/attribute vector.
k: The parameter of k-linear assumption. ($k \geq 2$)
$T_e \approx 1.6T_p$ (a symmetric curve with 512-bit base)
$T_p \approx 1.2T_e$ (a symmetric curve with 1024-bit base)

The length of ciphertext and private key are shown in Table 3. Due to decentralization, it is normal that the private key length of DIPE is larger than that of IPE. Besides, though, we can see that [10] needs about $\mathcal{O}(nk)$ elements in $\mathbb{G}$ for private key. Indeed, the value of k can be small in their work. However, in our work, the vector size could be large in reality. Therefore, our private key length is larger than others which may need more storage. Nevertheless, if the value of k is greater or equal than our vector size. Then, we only need less storage than [10] in storing private key.

In the comparison of ciphertext length, both our work and [1] have the least ciphertext length, and only need two elements in $\mathbb{G}$ plus an element in $\mathbb{G}_T$. It means that our ciphertext length is independent with vector size and the sum of authorities. It can reduce the burden of connection between sender and receiver for transmitting ciphertext. However, the ciphertext length of [10] dependent on n and k. To the best of our knowledge, our work is the first DIPE scheme achieving constant-size ciphertext.

DOI:10.6814/NCCU202101323

Table 3: Comparison with the previous schemes in space complexity

	Ciphertext length	Private key length
[5]	$(2\ell+1)\|\mathbb{G}\|$	$(2\ell+1)\|\mathbb{G}\|$
[6]	$(\ell+2)\|\mathbb{G}\|$	$3\|\mathbb{G}\|+\|\mathbb{Z}_p^\ell\|$
[1]	$2\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(\ell+1)\|\mathbb{G}\|$
[10]	$(nk+n+k+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$n(2k+2)\|\mathbb{G}\|$
Ours	$2\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$n(\ell+1)\|\mathbb{G}\|$

$|\mathbb{G}|$: The length of an element in $\mathbb{G}$.
$|\mathbb{G}_T|$: The length of an element in $\mathbb{G}_T$.
$|\mathbb{Z}_p^\ell|$: The length of an element in $\mathbb{Z}_p^\ell$.
n: The total number of authorities.
ℓ: The length of predicate/attribute vector.
k: The parameter of k-linear assumption. ($k \geq 2$)

In Table 4, only our work, as well as [10], has the decentralized network. In order to avoid collusion between users, a GID and a predicate (or an attribute) vector $\vec{X}$ mapping to a value by a random oracle. Therefore, security of ours and [10] are both based on random oracle model. As far as we know, there is no standard model for DIPE currently. Besides, although ours and [10] are both under CPA secure, the latter is an adaptive security model, which is more secure than our selective security model which has to know the target vector before setup phase.

Table 4: Property Comparison

	Decentralized	Confidentiality	Security Model	Group Order	Hard Problem
[5]	No	CPA	STD	Composite	subgroup decision assumption
[6]	No	CPA	STD	Prime	$\mathcal{P}$-DBDH
[1]	No	CPA	STD	Prime	q-DBDHE
[10]	Yes	CPA	ROM	Prime	k-Lin
Ours	Yes	CPA	ROM	Prime	q-DBDHE

CPA: Chosen-plaintext attack.
STD: Standard model.
ROM: Random oracle model.

DOI:10.6814/NCCU202101323

6.2 Experimental Result

In this section, we will show the experimental result of our construction and the construction of [10], and analyze the execution time of five algorithms.

Table 5: System configuration and elliptic curve

CPU	Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz
Memory	4GB
OS	Ubuntu-16.04 (64-bit)
Package	Python Charm-Crypto (v0.43) library
Pairing group	SS512

Table 5 shows the system configuration and the chosen pairing group of the experiment. Besides, we implement our construction by the Charm-Crypto library of Python language. In our implementation, the pairing group is a symmetric curve with a 512-bit based field. The experiment is executed on Intel(R) Core(TM) i7-10875H CPU at 3.60GHz processor, 4GB memory size and under Ubuntu-16.04 operating system.

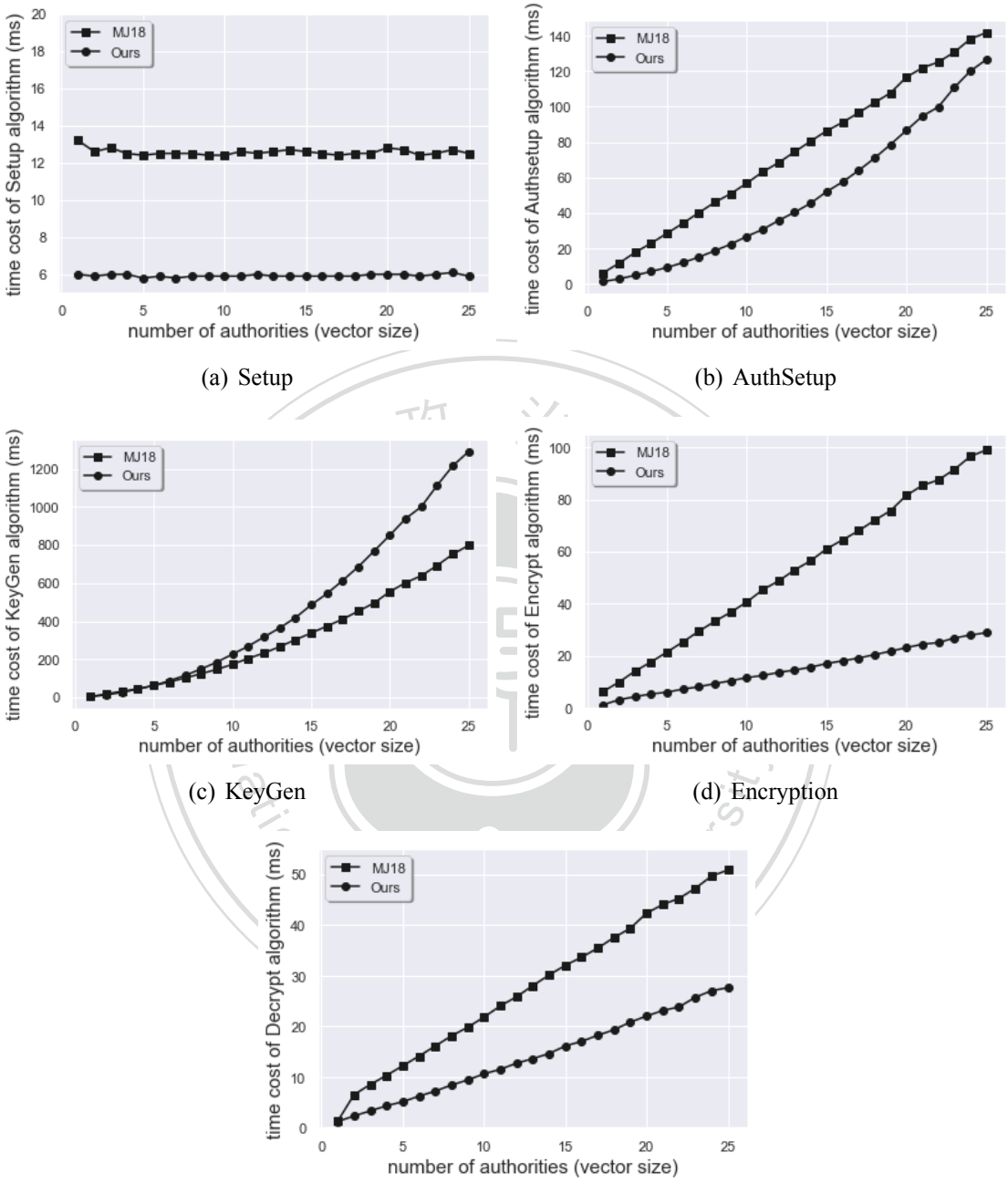
We analyze the time cost of each algorithm in our DIPE scheme below. In our experiment, the length of GID (global identity) is 10 bits. In [10], since each authority generates a partial key for a element of vector for users. Therefore, the length of vector size is same as the total number of authorities, ranging from 1 to 25. Besides, the value of k in [10] is set to 1. The value of each point on the figure is obtained by executing the algorithm 1000 times and getting the value of the average execution time.

The experiment for our DIPE scheme consists of the following five steps:

1. First, by running **Setup** algorithm, pp will be generated.
2. Second, taking pp, and an index of authority, then, a trusted party (may be one of authorities) runs **Authsetup** algorithm to generate PK_i and MSK_i.
3. Third, taking pp, $\{PK_i\}_{i=1,\ldots,n}$, attribute vector $\vec{Y}$ and message M as inputs, the sender runs **Encrypt** algorithm to get the ciphertext C.
4. Next, each authority in the system runs $\textbf{KeyGen}_{\mathbf{A_i}}$ algorithm. $\textbf{KeyGen}_{\mathbf{A_i}}$ algorithm generates a partial key sk_i by receiving pp, MSK_i, GID and predicate vector $\vec{X}$ as input.

DOI:10.6814/NCCU202101323

5. Finally, taking $\{sk_i\}_{i=1,\ldots,n}$, $\vec{Y}$ and C as inputs, the receiver executes **Decrypt** algorithm to get the decryption result.



(a) Setup (b) AuthSetup

(c) KeyGen (d) Encryption

(e) Decryption

Figure 2: The time cost of (a)Setup, (b)Authsetup, (c)KeyGen, (d)Encrypt, (e)Decrypt algorithm. (Pairing group:SS512, $|GID|$=10, # of authorities = $|\vec{X}|$ = $|\vec{Y}|$ = [1,...,25], k=1)

Figure 2 (b), shows that the time spent of [10] on **Authsetup** algorithm is more time-consuming than ours. Figure 2 (d),(e), we can note that **Encrypt** and **Decrypt** algorithms are

DOI:10.6814/NCCU202101323

both growing linearly in two schemes when the number of authorities increases. However, ours has better performance than theirs. Then, Figure 2 (c), exhibits that **KeyGen** is the most time-consuming algorithm due to the decentralized network. Nevertheless, we have relatively poor performance than [10]. Since our decentralization is different from [10], in our scheme, each authority generates a partial key for a whole vector instead of for a element of vector. Therefore, the execution time of **KeyGen** is longer. Finally, by Figure 2 (a), **Setup** algorithm only generates some generator of $\mathbb{G}$, some elements of $\mathbb{G}$ and hash function in both scheme, thus, execution time is independent with the total number of authorities and vector size. In addition, our scheme has one more advantage, that is, the length of predicate vector does not need to bind with the total number of authorities with same value.

DOI:10.6814/NCCU202101323

Chapter 7

Conclusion

As far as now, there is only one decentralized inner product encryption, Michalevsky *et al.* [10] first proposed in 2018. In their scheme, the length of ciphertexts are independent with the total number of authorities which may become the bottleneck of the system. Therefore, we would like to solve this problem.

Based on the construction of Attrapadung *et al.* [1], we present a decentralized inner product encryption which only needs constant-size ciphertexts. In addition, our scheme is proven to be selectively secure under q-DBDHE assumption. Besides, we implement our scheme and the scheme of [10] to analyze the execution time of these. Except for KeyGen algorithm, the remaining four algorithms (Setup, AuthSetup, Encrypt, Decrypt), our work has better performance.

DOI:10.6814/NCCU202101323

Bibliography

[1] Nuttapong Attrapadung and Benoît Libert. 2010. Functional Encryption for Inner Product: Achieving Constant-Size Ciphertexts with Adaptive Security or Support for Negation. In *Public Key Cryptography – PKC 2010*, Phong Q. Nguyen and David Pointcheval (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 384–402.

[2] Dan Boneh and Matt Franklin. 2001. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology — CRYPTO 2001*, Joe Kilian (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 213–229.

[3] Dan Boneh, Craig Gentry, and Brent Waters. 2005. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Advances in Cryptology – CRYPTO 2005*, Victor Shoup (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 258–275.

[4] Melissa Chase. 2007. Multi-authority Attribute Based Encryption. In *Theory of Cryptography*, Salil P. Vadhan (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 515–534.

[5] Jonathan Katz, Amit Sahai, and Brent Waters. 2008. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In *Advances in Cryptology – EUROCRYPT 2008*, Nigel Smart (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 146–162.

[6] Intae Kim, Seong Oun Hwang, Jong Hwan Park, and Chanil Park. 2016. An Efficient Predicate Encryption with Constant Pairing Computations and Minimum Costs. *IEEE Trans. Comput.* 65, 10 (2016), 2947–2958.

[7] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. 2010. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical)

DOI:10.6814/NCCU202101323

Inner Product Encryption. In *Advances in Cryptology – EUROCRYPT 2010*, Henri Gilbert (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 62–91.

[8] Allison Lewko and Brent Waters. 2011. Decentralizing Attribute-Based Encryption. In *Advances in Cryptology – EUROCRYPT 2011*, Kenneth G. Paterson (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 568–588.

[9] Ehsan Meamari, Hao Guo, Chien-Chung Shen, and Junbeom Hur. 2020. Collusion Attacks on Decentralized Attributed-Based Encryption: Analyses and a Solution. arXiv:2002.07811 [cs.CR]

[10] Yan Michalevsky and Marc Joye. 2018. *Decentralized Policy-Hiding ABE with Receiver Privacy: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II*. 548–567.

[11] Jong Hwan Park. 2011. Inner-product encryption under standard assumptions. *Designs, Codes and Cryptography* 58, 3 (2011), 235–257.

[12] Amit Sahai and Brent Waters. 2005. Fuzzy Identity-Based Encryption. In *Advances in Cryptology – EUROCRYPT 2005*, Ronald Cramer (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 457–473.

[13] Adi Shamir. 1985. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology*, George Robert Blakley and David Chaum (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 47–53.

[14] Najmeh Soroush, Vincenzo Iovino, Alfredo Rial, Peter B. Roenne, and Peter Y. A. Ryan. 2020. Verifiable Inner Product Encryption Scheme. In *Public-Key Cryptography – PKC 2020*, Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas (Eds.). Springer International Publishing, Cham, 65–94.

[15] Yi-Fan Tseng, Zi-Yuan Liu, and Raylin Tso. 2020. Practical Inner Product Encryption with Constant Private Key. *Applied Sciences* 10, 23 (2020).

[16] Leyou Zhang, Xuehuang Gao, Li Kang, Pengfei Liang, and Yi Mu. 2021. Distributed Ciphertext-Policy Attribute-Based Encryption With Enhanced Collusion Resilience and Privacy Preservation. *IEEE Systems Journal* (2021), 1–12.

DOI:10.6814/NCCU202101323

[17] Y. Zhang, Y. Li, and Y. Wang. 2019. Efficient inner product encryption for mobile clients with constrained computation capacity. *International Journal of Innovative Computing, Information and Control* 15 (02 2019), 209–226.

[18] Tan Zhenlin and Zhang Wei. 2015. A Predicate Encryption Scheme Supporting Multiparty Cloud Computation. In *2015 International Conference on Intelligent Networking and Collaborative Systems*. 252–256.

DOI:10.6814/NCCU202101323