國立臺灣師範大學理學院

資訊工程學系

碩士論文

指導教授: 紀博文 博士

可否認式配對加密系統

Deniable Matchmaking Encryption

研究生: 陳炫豪 撰

中華民國 110 年 9 月

致謝

感謝指導教授的指導與鞭策、感謝所有口試委員的建議與提問、感謝實驗室夥伴們的陪伴與討論、感謝系辦職員們的協助、也感謝家人的支持，很明顯的，我不是一個很會打致謝的人，但我心中確實充滿著對所有人的感謝。

摘要

我們提出了一個名為可否認式配對加密系統的新加密系統，這個加密系統是建立在配對加密系統的基礎上，並新增了可否認式加密系統的特性，使其在一些受到脅迫而必須公開密文或金鑰內容的場合，依舊能夠保護寄件者、收件者以及密文本身的安全性，這是一套能同時否認寄件者、收件者身分以及訊息內容的加密系統。

在可否認式配對加密系統中，寄件者以及收件者皆可以透過指定身分來保護密文，只有當雙方的身分皆符合加密時的指定身分時，密文才能被正確的解密，而對於任何非指定身分的一方，此密文皆不會洩漏任何資訊，並且，若今天發生了一些脅迫事件，迫使任一方或建立金鑰的公正第三方必須公開金鑰或公開原文時，因為此加密系統藉由變色龍雜湊函數的性質，可以產生出另一個不同寄件者、不同收件者以及不同原文的加密訊息，使得除了當事人雙方以外，其他人皆沒有辦法判定哪一組密文才是真正的密文，以此來保護當事人雙方的安全。

在理論方面，我們定義了可否認式配對加密系統的安全性，提供了完整的可否認式配對加密系統的架構，並證明了在配對加密系統是安全的情況下，我們的可否

認式配對加密系統是安全的。而在實作方面，我們計算了可否認式配對加密系統的計算時間需求與計算空間需求並與配對加密系統進行比較。

關鍵字：可否認式加密系統、配對加密系統、變色龍雜湊函數.

Abstract

We introduce a new encryption scheme, which is called Deniable Matchmaking Encryption (DME). This encryption scheme is based on Matchmaking Encryption (ME) and adds deniability on it. In some situations that users or the trusted third parties are coercered to reveal the plaintext or even the secret keys, this scheme can still protect the message and the identity of the sender and receiver. This is a Bi-identity-deniability encryption scheme.

In DME, The sender and the receiver can protect the message by specifying the other user's identity. Only when both the sender and the receivers' identities are matched, the ciphertext can be decrypted correctly, and for anyone does not match the identity acquirement, the ciphertext leaks no information.With the help of chameleon hash function, DME can

generate an indistinguishable fake ciphertext, which the sender identity, receiver identity and the message are all fake, to protect the true ciphertext. So that if any malicious adversary coercers the users or trusted third parties to reveal the ciphertext or the secret keys, the adversary can not distinguish whether the ciphertext is true.

On the theoretical side, we define the security of DME and provide a DME encryption scheme. We prove that if ME is secure, our DME is also secure. On the practical side, we compute the space cost and computation cost of DME and compare it to ME.

Keywords: Deniable Encryption, Matchmaking Encryption, Chameleon Hash Function.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Identity-based Encryption and Attribute-based Encryption

Before Identity-based Encryption(IBE) and Attribute-based Encryption(ABE) were published, the certificate and key management of cryptography systems were a little complicated. When Bob wants to communicate with Alice, he has to ask for Alice's public key from trusted third parties or herself. But with the usage of IBE and ABE, Alice's public key is no longer a random string, but a string which has some association with her identities or attributes. These encryption schemes are more powerful than other existing schemes. However, privacy becomes a hot topic due to the flourish of technology in the recent years. Since IBE and ABE does not fully protect the identities of the sender and the receiver, some cryptographers start to find a cryptosystem with the form of dead drop communication, which can be seen as a non-interactive secret handshake scheme. They want to find a encryption scheme with some properties: both the sender and the receiver

can decide that who can read the letters (or whose letters can be read); the scheme leaks no information to anyone who is not the sender or the receiver; the scheme prevents traffic analysis to protect the privacy of the sender and the receiver. Therefore Matchmaking encryption is published to overcome these obstacles.

1.2 Matchmaking Encryption

Matchmaking encryption[3](ME) is proposed by Giuseppe Ateniese et al. at 2019. The most eye-catching property of ME is that it can hide sender and receivers' identity at the same time. ME is the first encryption which can send the message without any leak of the participants' data. While using ME, the sender needs to hash the sender's identity and the receiver's identity with a random seed, and then combines both the hash identity and the message with exclusive or operation. As the sender, this encryption can promise that only the specific receiver can decrypt the message since other users can not find the receiver's identity from the ciphertext. Combining with the use of Mix network[8] or Onion routing[15], no one can find out who is the sender and receiver since every ciphertext sent to the public bulletin board is hashed and the hash function might not be reversed. To decrypt the message encrypted by ME, the receiver needs to know that who is trying to send messages to you. With the correct sender's identity and the receiver's identity itself, the receiver can figure out the same hash value and decrypt the ciphertext

with exclusive or operation.

1.3 Problems in Real Life

Although ME can technically achieve the dead drop communication without leaking any information about the participant, there are still some ”out-of-rules” problems which are able to break the security and privacy of ME. We all have heard some news that even in a well-developed democracy country, ruling party or government tried to trace the opposite party’s messages with the excuse of beating fake news. Some forums and bulletin boards were forced by the government to hand in specified users’ personal data not because these users were illegal, but they just said something which was unfavorable to the government. Another example of the problem is that on a public channel like blockchain, although the messages on the chain are always encrypted, the malicious adversary can still use some ” side-channel” ways to obtain private information. A coercer can force the user to reveal their status of digital wallet if he already knows that there is an digital wallet application in the user’s cellphone. Another story is that there are still some employees in telecom operators or insurance company who sells their customers personal data to scam groups in order to earn more money. Obviously, these problems can not be solved by traditional cryptographic ways since the public and secret keys may be forced to reveal, but there are some ways which can protect the users even if their keys leak.

1.4 Deniability in Matchmaking Encryption

To solve the problems above, we think that adding deniability into Matchmaking Encryption would be a good idea. Deniable Encryption are now widely-used when someone wants to save the message from being coercered. When facing a malicious coercer, handing in a forged message which cannot be distinguished from the true message is always a better way to protect users itself than just saying ”I forgot”. On the aspect of Matchmaking Encryption, having deniability on the sender’s identity, the receiver’s identity or even the message can apparently improve the security of the scheme and the safety of users. Imagine that when a malicious coercer forces the trusted third party decrypt all the ciphertexts and reveal all the keys it has published, every ciphertext can be decrypted into two messages which they have no relationship to each others. Since the coercer cannot distinguish that which decryption is true and which is fake, the message and the participants’ privacy can be safely protected. In our concept which is shown as figure 1.1, we want to use chameleon hash function to forge a string that the string can be hashed with the fake message and output the same hash value with the true message. Since both the messages can pass the hash value verification, there’s no other way can distinguish the true message from the fake message. All the ciphertext will be sent to a public bulletin board by some ways which can prevent traffic analysis (like Mixnet), and there will be some public channels like ads robots on the bulletin board in order to confuse the coercer.

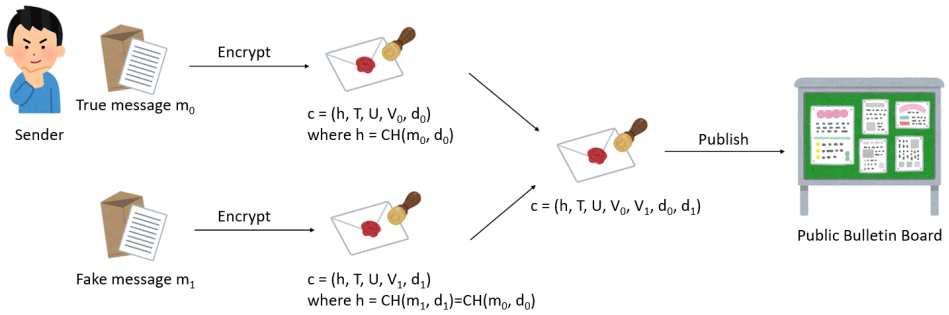

Figure 1.1: Deniable Matchmaking Encryption Concept

Therefore, with the protection from ME and chameleon hash, the sender can forge a message with a sender identity, receiver identity and message which are all fake. Moreover, as the receiver, it's not complicated to find the true message. All the receiver needs to do is decrypting both the messages, since the receiver can only decrypt the message which the identities are correct.

1.5 Our Contributions

We propose a scheme called Deniable Matchmaking Encryption(DME) to solve the problem that when the users' identity have been leaked, they can still use the mechanics provided by DME to protect themselves. We add a verification property into Identity-based Matchmaking Encryption(IB-ME) which is also proposed by Giuseppe Ateniese et al. at 2019. This verification is made of identity-based chameleon hash function[1], therefore the user can forge a fake message with the trapdoor of the hash function to avoid coercion and identity leaks. On the security aspect, we prove that DME can be reducted

to IB-ME, which means that DME is as safe as IB-ME. With DME, we believe that we have a safer way to communicate to a specific person without leaking any of our identity secret. Our DME scheme has several contributions as follow:

- Message deniability and Bi-identity-deniability (including sender identity and receiver identity). To the best of our knowledge, this is the first identity-based encryption scheme which has Bi-identity-deniability.

- No negotiation required with the cover identity. The cover identity including sender identity and receiver identity. We will discuss this at Chapter 4.

- The consistency of the fake proof. Since our fake proof is forged by hash function, it will not expire and doesn't need to update or refresh.

- Non-interactive fake proof. Unlike other deniable encryption, our scheme does not use "open in honest way" and "open in dishonest way" to generate fake proof, which means the sender does not need to be online when the ciphertext is opened.

Chapter 2 Related Work

2.1 Identity-based Encryption

Before Identity-based Encryption (IBE) had been published, users were bothered by asking for receiver's public key and recording the relationship between people and their public keys. In 1984, Shamir[28] asked for a public key encryption that users can encrypt the message with their own identities, not some random strings generated by the public key generator. After several proposals (e.g. [13] [29] [30] [22]) for IBE schemes, Boneh[6] proposed a well-known fully functional IBE scheme which has chosen ciphertext security in the random oracle model assuming an elliptic curve variant of the computational Diffie-Hellman problem. Boneh's IBE scheme is specified by four randomized algorithms:

- **Setup**$(1^{\lambda}) \rightarrow \{\mathbf{MK}, \mathbf{params}\}$: The setup algorithm takes input λ as the security parameter. It outputs the public parameters **params** and the master key **MK**. The master key will be known only to the Private Key Generator (PKG). We claim that every other algorithm takes **params** as input.

- **Extract**(**MK**, S) $\rightarrow$ **SK**: The extract algorithm takes input **MK** as the master key and an arbitrary identity $S \in \{0,1\}^*$, and returns a private key **SK**. S will be used as a public key, and **SK** is the corresponding private key.

- **Encrypt**(S, m) $\rightarrow c$: The encrypt algorithm takes input S as the identity and m as the message. It returns a ciphertext c.

- **Decrypt**(S, c, SK) $\rightarrow m$: The decrypt algorithm takes input S as the identity, c as the ciphertext and SK as the private key, and returns the message m.

Researches of IBE continued to be published. In 2005, Sahai[26] introduced a new type of IBE called Fuzzy Identity-Based Encryption, which could decrypt a ciphertext if user's identity and target identity were close to each other as measured by the "set overlap" distance metric. Waters[31] presented the first efficient IBE scheme that is fully secure without random oracles. In 2010, Fan et al.[14] published an anonymous multi-receiver IBE scheme which made it impossible for an attacker or any other message receiver to derive the identity of a message receiver such that the privacy of every receiver can be guaranteed. Li et al.[21] aimed at tackling the critical issue of identity revocation, and introduced outsourcing computation into IBE for the first time and propose a revocable IBE scheme in the server-aided setting.

2.2 Attribute-based Encryption

Attribute-based Encryption provides the users to encrypt the message and specify an associated access structure over attributes. A user will only be able to decrypt the ciphertext if that user's attributes fit the ciphertext's access structure. There are two major types of attribute-based encryption: Ciphertext-policy attribute based encryption(CP-ABE)[5] and Key-policy attribute based encryption(KP-ABE)[16]. The difference between these ABE is that CP-ABE puts the attribute access structure in the ciphertext, but KP-ABE puts the structure in the user's key.

An CP-ABE consists of four algorithms as below:

- **Setup**$(1^{\lambda}) \rightarrow \{$**PK**, **MK**$\}$: The setup algorithm takes input λ as the security parameter. It outputs the public key **PK** and the master key **MK**.
- **Encrypt**(**PK**, m, $\mathbb{A}) \rightarrow c$: The encrypt algorithm takes input **PK**, m as the message and $\mathbb{A}$ as the attribute access structure. It encrypts m and produces a ciphertext c, which can only be decrypted by the user who satisfies the access structure.
- **Key Generation**(**MK**, $S) \rightarrow$ **SK**: The key generation algorithm takes input **MK** as the master key and S as a set of attributes. It outputs the private key **SK**.

- **Decrypt**(**PK**, c, **SK**) $\rightarrow m$: The decrypt algorithmn takes input **PK** as the public key, c as the ciphertext and **SK** as the private key. If the set of attributes S in **SK** satisfies the access structure $\mathbb{A}$ in c, the algorithm will decrypt the ciphertext and return a message m.

Attributed-based Encryption has been developed well during these years: At 2006, Goyal et al. [16] published a new attributed-based encryption scheme that the attributes are placed in the ciphertext. Every user owned a key which had a policy on it, and if the policy on key fitted the attributes in the ciphertext, the ciphertext could be successfully decrypted. Next year, Bethencourt et al.[5] reversed the place of policy and attributes, which allowed the user to own a key with his or her attributes. Users no longer needed to obtain the key with the right policy to decrypt the ciphertext fitted their attributes every times, but they needed to refresh their attributes on key during a fixed period. Muller et al.[24] published a distributed attributed-based encryption at 2008 to solve the problem that a single trusted-third-party might be busy in delivering attributes and keys. It also helped the situation that there might be several management parties in a attributed system. At 2009, Attrapadung et al.[4] combined the advantages between CP-ABE and KP-ABE and proposed a broadcast attributed-based encryption. It can be used to construct ABE systems with direct revocation mechanism. Last but not least, Lewko et al.[20] decentralized the ABE system at 2011 that the trusted-third-party was no longer needed. Users in this

cryptosystem didn't need to wait the trusted-third-party to publish the key and attributes anymore.

2.3 Chameleon Hash Function

A hash function is a function that can be used to map data to fixed-size value, and usually it is collision-resistant, which means it is computationally infeasible to find two distinct values with the same hash value. But chameleon hash function[19] is an unique type of hash function that it can generate a trapdoor which can break the collision-resistance property. With the knowledge of the trapdoor, a user can easily find several data that have the same hash value. But without the trapdoor, this hash function is still collision-resistant. On input a message m and a random string r, chameleon hash function generates a hash value $CH(m, r)$ which satisfies the following properties:

- **Collision resistance**: There is no efficient algorithm that on input the public key can find pairs (m_0, r_0) and (m_1, r_1) where $m_0 \neq m_1$ such that $CH(m_0, r_0) = CH(m_1, r_1)$, except with negligible probability.
- **Trapdoor collisions**: There is an efficient algorithm that on input the secret key, any pair (m_0, r_0), and any additional message m_1, finds a value r_1 such that $CH(m_0, r_0) = CH(m_1, r_1)$.

- **Semantic Security**: All messages m induce the same probability distribution on $CH(m, r)$ for r chosen uniformly at random.

Chameleon hash function was first proposed by Krawcyzk et al.[19] at 1998, but it soon faced a key exposure problem. With the collection of the forged ciphertext, an adversary could easily forged a fake ciphertext that still had the same hash value. At 2004, Ateniese et al.[2] and Chen et al.[9] solved the key exposure problem with different ways. Ateniese built a exposure-free chameleon hash function on RSA and discrete logarithm problem, while Chen proposed a chameleon hashing scheme in the gap Diffie-Hellman group to solve the problem of key exposure. Furthermore, Chen et al.[10] introduced a double-trapdoor hash family based on the discrete logarithm assumption to solve the key exposure problem on on-line/off-line signature schemes. In the following years, chameleon hash function was brought in more and more applications and schemes. Choi et al.[11] proposes a handover authentication scheme using credentials at 2009. Next year, Mohassel et al.[23] transformed every chameleon hash function to a strongly unforgeable one-time signature scheme. In 2013, Guo et al.[17] proposed an elliptic curve based chameleon hash function in vehicular ad-hoc networks.

Identity-based chameleon hash function[1] is a non-repudiation, message hiding hash function with a trapdoor that the owner of hash function can forge several messages which all have the same hash value. The hash function is composed of the following algorithms:

- **Setup**$(1^{\lambda}) \rightarrow \{\mathbf{mpk}, \mathbf{msk}\}$: An efficient, probabilistic algorithm runs by a trusted party, takes input λ as the security parameter and generates public and secret key $(\mathbf{PK}, \mathbf{SK})$.

- **Extract**$(S, \mathbf{msk}) \rightarrow B$: An efficient, deterministic algorithm takes input S as the identity string and **msk** as the secret key and outputs the trapdoor parameter B.

- **Hash**$(S, m_0, r_0) \rightarrow h$: An efficient, probabilistic algorithm takes input S as the identity string, m_0 as the message and r_0 as a random string, outputs hash value h.

- **Forge**$(S, B, m_0, r_0, m_1) \rightarrow r_1$: An efficient algorithm takes input S as the identity string, B as the trapdoor parameter, m_0 as the message, r_0 as a random string and m_1 as the new message, outputs a random string r_1 associate to the input message m_1 such that the input message and the output random string can make collision with selected hash value h.

2.4 Deniable Encryption

Deniable encryption was first published by Canetti et al.[7] to solve the problem that the transmission of encrypted messages was intercepted by an adversary who could later ask the sender to reveal the secret. To protect the sender and the secret, deniable encryption can generate fake random choices that will make the ciphertext look like an encryption of

a different plaintext. As user's privacy being more and more popular these years, deniable encryption is also a good choice to protect ourselves. There are two types of deniability: multi-distributional deniability and full deniability. The difference between each other is that in multi-distributional scheme, users choose alternative algorithms of key generation and encryption and can claim that they are using which algorithms. In full deniability scheme, there is only one algorithm for key generation and encryption but users can decrypt their message to another message to claim their deniability. Although full deniability sounds better than multi-distributional since there are less algorithms in the scheme, most of the full deniability scheme are not as practical and efficient as multi-distributional. Hence, in this paper, we are using multi-distributional scheme to construct our deniable encryption. There are many ways to deny the random choices in the encryption scheme: In 1997, Canetti et al.[7] constructed a bitwise encryption as the start of the researches of deniable encryption; Klonowski et al.[18] implemented a practical ElGamal-based deniable encryption at 2008 to make the deniable scheme more efficient; In 2011, O'Neill et al.[25] used simulated encryption, which are introduced by Damgard[12] at 2000, to create a bi-deniable scheme that users can deny the sender and the receivers' secret at the same time; In 2014, Sahai[27] introduced a new technique called punctured programs to apply indistinguishability obfuscation towards cryptographic problems and solved the problem that deniable encryption might require some pre-planning by the party that must later issue a denial.

The basic scheme of Canetti's deniable encryption is as below, which is a full deniability scheme. The public encryption key **PK** is a method for generating uniformly at random a member of a translucent set $S_t \subset \{0,1\}^t$. The private decryption key is the corresponding secret **SK**.

- **Encrypt**(**PK**, m) $\rightarrow c$: An efficient algorithm takes input **PK** as the public key and m as the message, outputs the ciphertext c. To encrypt 1, send a random element of S_t. To encrypt 0, send a random element in $\{0,1\}^t$.

- **Decrypt**(**SK**, c) $\rightarrow m$: An efficient algorithm takes input **SK** as the secret key and c as the ciphertext, outputs the message m. If the ciphertext c is in S_t then output 1. Else output 0.

- **Open**(**SK**, c) $\rightarrow m$: An efficient algorithm takes input **SK** as the secret key and c as the ciphertext, outputs the message m which reveals the true random choices used.

- **DeniOpen**(**SK**, c) $\rightarrow m$: An efficient algorithm takes input **SK** as the secret key and c as the ciphertext, outputs the message m that if the encrypted bit is 1, then claim that c was chosen at random from $\{0,1\}^t$. If the encrypted bit is 0 then lying will be infeasible since the ciphertext c is in S_t only with negligible probability 2^{-k}.

If the scheme of deniable encryption can simultaneously deny the sender and the receiver, we call it "bi-deniable" encryption. A multi-distributional sender-, receiver-, or

bi-deniable encryption scheme is made up of the following algorithms:

- **Gen**(λ, r_R) $\rightarrow$ **PK**: An efficient algorithm takes input λ as the security parameter, outputs the public key **PK** and the randomness r_R is used as the associated secret decryption key.

- **Enc**(**PK**, m, r_S) $\rightarrow c$: An efficient algorithm takes input **PK** as the public key, m as the message and r_S as the randomness associated with the sender, outputs the ciphertext c.

- **Dec**(**PK**, r_R, c) $\rightarrow m$: An efficient algorithm takes input **PK** as the public key, r_R as the randomness associated secret decryption key and c as the ciphertext, outputs the message m.

- **DenGen**(λ) $\rightarrow$ (**PK**, **FK**): An efficient algorithm takes input λ as the security parameter, outputs the public key **PK** and the fake key **FK**.

- **SendFake**(**PK**, r_S, m', m) $\rightarrow r_S^*$: An efficient algorithm takes input **FK** as the fake key, r_S as the randomness associated with the sender, m' as the desired message for deniability and m as the message, outputs the fake randomness r_S^* for **Enc**.

- **RecFake**(**PK**, **FK**, c, $\hat{m}$) $\rightarrow r_R^*$: An efficient algorithm takes input **PK** as the public key, **FK** as the fake key, c as the ciphertext and $\hat{m}$ as the desired message, outputs the fake randomness r_R^* for **Dec**.

2.5 Matchmaking Encryption

Matchmaking Encryption was proposed by Ateniese et al.[3] at 2019. The main idea of the encryption scheme is that both sender and receiver can decide the attributes policy of the ciphertext. From the sender side, the sender can decide the policy of receiver's identities and encrypt the plaintext with the policy and the sender's identity. Then the ciphertext will be send to a public place (like a bulletin board), waiting the receiver whose attributes fit the policy to take and decrypt it. When the receiver wants to know whether someone sends the ciphertext to him or her, he or she needs to visit the public bulletin board and use his or her identity with the sender's identity to try to decrypt all the ciphertext. If one of the attributes does not match the ciphertext, the receiver will learn nothing about the ciphertext. This encryption scheme sounds more unconvenient than other existed encryption system since the receiver needs to look for and grab the ciphertext itself. But actually this is a very safe method to send message from one to the other since Matchmaking Encryption can fully hide both the sender and the receiver's identity. Without the accurate identities input, the adversary can learn nothing about the plaintext, the sender and the receiver. This method may not be a widely-used communication system, but it can be a most safe scheme to exchange message secretly.

Matchmaking Encryption(ME) is composed of the following algorithms:

- **Setup**$(1^\lambda) \to \{$**mpk**, **msk**$\}$: An efficient algorithm takes input λ as the security parameter, outputs the master public key **mpk** and master private key **msk**. All following algorithms are implicitly given **mpk** as input.

- **SKGen**(**msk**, σ) $\to$ **ek**$_\sigma$: An efficient algorithm takes input σ as the sender's identity, outputs the sender's encryption key **ek**$_\sigma$.

- **RKGen**(**msk**, ρ) $\to$ **dk**$_\rho$: An efficient algorithm takes input ρ as the receiver's identity, outputs the receiver's decryption key **dk**$_\rho$.

- **Enc**(**ek**$_\sigma$, ρ, m) $\to c$: An efficient algorithm takes input **ek**$_\sigma$ as the sender's encryption key, ρ as the receiver's identity and m as the message, outputs the ciphertext c.

- **Dec**(**dk**$_\rho$, σ, c) $\to m$ or $\perp$: An efficient algorithm takes input **dk**$_\rho$ as the receiver's decryption key, σ as the sender's identity and c as the ciphertexet, outputs the message m if the padding is valid. Otherwise, return $\perp$.

Chapter 3 Preliminaries

3.1 Notation

We use capital boldface words (such as **PK** and **Setup**) to denote algorithm and keys. Small letters (such as r) are used to denote concrete values, and blackboard bold letters (such as $\mathbb{G}$) are used to denote sets. When x is random chosen from $\mathbb{X}$, we write $x \xleftarrow{\$} \mathbb{X}$. An algorithm **A** is probabilistic polynomial-time (PPT) if **A** is randomized and for any input, the computation steps of **A** are a polynomial number in the input size. The function **neql(**λ**)** is called negligible in the security parameter λ if it vanishes faster than the inverse of any polynomial in λ.

3.2 Bilinear Mapping

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be the map function on $\mathbb{G}$. Let g be a generator of $\mathbb{G}$. We say that G is a bilinear map

group if G and e have the following properties:

- **Bilinearity**: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p, e(u^a, v^b) = e(u, v)^{ab}$
- **Non-degeneracy**: $e(g, g) \neq 1$.
- **Computability**: the group action in $\mathbb{G}$ and map function e can be computed efficiently.

Our Deniable Matchmaking Encryption can be reduce to the identity-based version of ME which is introduced by Ateniese et al.[3], and the scheme is provably secure under Computational Bilinear Diffie-Hellman (CBDH) assumption.

Definition 1 (CBDH assumption)**.** *Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups with prime order q and P be the generator of $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear mapping. The CBDH problem is hard if for every PPT algorithm A:*

$$\mathbb{P}[A(\mathbb{G}, \mathbb{G}_T, q, P, e, P^a, P^b, P^c) = e(P, P)^{abc}] \leq negl(\lambda)$$

where $P \xleftarrow{\$} \mathbb{G}^*$ and $a, b, c \xleftarrow{\$} \mathbb{Z}_q^*$.

3.3 Identity-based Chameleon Hash Function

Our Deniable Matchmaking Encryption can use any secure chameleon hash function to generate the fake ciphertext, but here we use the Identity-based Chameleon Hash Function proposed by Ateniese et al.[1] as an example. This scheme is semantic secure and resistant to forgery under active attacks, which are proved in the original paper. The Identity-based Chameleon Hash Function is composed by the steps below:

- **Setup**$(1^\lambda) \rightarrow \{$**mpk**, **msk**$\}$: With the security parameter 1^λ, the algorithm randomly chooses two prime numbers $x, y \in \{2^{(\lambda-1)}, ..., 2^\lambda - 1\}$. Let $n = xy$. The algorithm then generates a random prime integer v such that $v > 2^{2\lambda}$ and GCD$(v, (x-1)(y-1)) = 1$. Applying the extended Euclidean algorithm for GCD, the algorithm computes w, z such that $wv + z(x-1)(y-1) = 1$. The algorithm finally outputs the public key **mpk** $= (n, v)$ and the secret key **msk** $= (x, y, w)$.

- **Extract**$(S,$ **msk**$) \rightarrow B$: With C as the secure deterministic hash-and-encode scheme and S as the identity string associated to some party, the algorithm computes $J = C(S)$ and outputs $B = J^w$ mod n.

- **Hash**$(S, m_0, r_0) \rightarrow h$: With H as the secure hash function and r_0 as the random string, the algorithm hashes the input message m_0 and outputs $h = J^{H(m_0)} r_0^v$ mod n.

- **Forge**$(S, B, m_0, r_0, m_1) \rightarrow r_1$: With B as the trapdoor of the hash function, the algorithm outputs the hash string $r_1 = r_0 B^{H(m_0)-H(m_1)} \bmod n$ such that $Hash(S, m_0, r_0) = Hash(S, m_1, r_1)$.

Theorem 1. *The chameleon hashing scheme is resistant to forgery under active attacks, provided that the secure RSA signature scheme is similarly resistant.*

Theorem 2. *The chameleon hashing scheme is semantically secure if, for all identity strings S, and all pairs of messages m_0, m_1, the probability distributions of the random variables $Hash(S, m_0, r_0)$ and $Hash(S, m_1, r_1)$ are computationally indistinguishable.*

As the proof of Thm.1 and Thm.2, please refer to the original paper written by Ateniese et al.[1].

3.4 Matchmaking Encryption

Our Deniable Matchmaking Encryption is based on the identity-based ME (IB-ME) which is introduced by Ateniese et al.[3].The security of ME is based on CBDH problem and the proof is in the original paper. The scheme is composed by the following algorithms:

- **Setup** $(1^\lambda) \rightarrow \{\mathbf{mpk}, \mathbf{msk}\}$: With λ as the input security parameter, the algorithm outputs $\mathbf{mpk} = (e, \mathbb{G}, \mathbb{G}_T, q, P, P_0, H, H_1, H_2, \Phi)$ and $\mathbf{msk} = (r, s)$, where e :

$\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a symmetric pairing. P is a generator of $\mathbb{G}$. q is the order of $\mathbb{G}$ and $\mathbb{G}_T$ that depends on λ. $H, H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \mathbb{G}_T \to \{0,1\}^\ell$ are hash functions. $\Phi : \{0,1\}^n \to \{0,1\}^\ell$ is a padding function and $\Phi(m)$ is invertible. $r, s \in \mathbb{Z}_q$ and $P_0 = P^r$. We implicitly assume that all other algorithms take **mpk** as input.

- **SKGen** (**msk**, σ) $\to$ $\mathbf{ek}_\sigma$: With the input of **msk** and the sender identity σ, the algorithm outputs the sender key $\mathbf{ek}_\sigma = H_1(\sigma)^s$.

- **RKGen** (**msk**, ρ) $\to$ $\mathbf{dk}_\rho$: With the input of **msk** and the receiver identity ρ, the algorithm outputs the receiver key $\mathbf{dk}_\rho = (\mathbf{dk}_\rho^1, \mathbf{dk}_\rho^2, \mathbf{dk}_\rho^3) = (H(\rho)^r, H(\rho)^s, H(\rho))$.

- **Enc** ($\mathbf{ek}_\sigma$, ρ, m) $\to$ c : With the input of $\mathbf{ek}_\sigma$, the receiver identity ρ and the message m, the algorithm outputs the ciphertext $c = (T, U, V)$ where:

  - $T = P^t$ and $U = P^u$, where $u, t \xleftarrow{\$} \mathbb{Z}_q$.
  - $k_R = e(H(\rho), P_0^u)$ and $k_S = e(H(\rho), T \cdot \mathbf{ek}_\sigma)$.
  - $V = \Phi(m) \oplus H_2(k_R) \oplus H_2(k_S)$.

- **Dec** (c, $\mathbf{dk}_\rho$, σ) $\to$ m or $\perp$: With the input of the ciphertext c, the receiver key $\mathbf{dk}_\rho$ and the sender identity σ, the algorithm outputs the message m or $\perp$ by the following steps:

  - Parse C as (T, U, V).

– $k_R = e(\mathbf{dk}_\rho^1, U)$ and $k_S = e(\mathbf{dk}_\rho^2, H_1(\sigma)) \cdot e(\mathbf{dk}_\rho^3, T)$.

– $\Phi(m) = V \oplus H_2(k_R) \oplus H_2(k_S)$.

– If the padding is valid, return m. Otherwise, return $\perp$.

Theorem 3. *Let $\mathbb{G}$, $\mathbb{G}_T$ be two groups of prime order q, and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be an admissible bilinear map. If the CBDH problem is hard in $(\mathbb{G}, \mathbb{G}_T, e)$, then the IB-ME scheme is secure in the random oracle model.*

Ateniese et al.[3] proved that the IB-ME scheme is secure with two $IND - CPA^+$ games and a attack game which breaks CBDH problem with non-negligible advantages. As the detail proof of Thm.3, please refer to the original paper.

Chapter 4 Deniable Matchmaking Encryption

4.1 Our Idea

The main problem we want to solve in IB-ME is that there are still some ways to identify the sender's identity in real life. We can not promise that the trusted third party who runs the key generation algorithm will keep our key secretly if some more powerful party forces them reveal the keys they generate. Or maybe some routers are controlled by malicious hackers, so they do not need to break the ciphertext to find the sender's identity.

To solve the problem, we add the deniable property into IB-ME, so that when the ciphertext or user's identity is forced to reveal, the user has a chance to deny the relationship to the message. We add a verification hash value to the message, one use is to check the correctness of the decryption, the other is to make a place for fake message to be verified.

With the trapdoor of chameleon hash function, our DME can forge a fake message that it has the same hash value with the true message, which means no one can distinguish whether the message is true or not. So when a malicious coercerer forces the user reveal the plaintext, he can verify the fake message to deny the true message and no one can prove that he is using the fake message.

There are three deniable type of encryption: sender deniable, receiver deniable and sender-receiver deniable. Our DME can easily be receiver deniable and in some situation, DME can be sender-receiver deniable. On the aspect of receiver deniable, the sender is able to forge a fake message and input a fake receiver's identity easily. But on the aspect of sender-receiver deniable, since the sender's encryption key is private and is given from the trusted third party, there is no way to compute other sender's encryption key without master private key. Therefore, we have two ways for the sender to obtain other sender's encryption key: one is that the sender can have a collusion with other trusted sender, the other is that the trusted third party will provide a robot sender's encryption key in the public key.

Algorithms and the Executors	
Algorithms	Executors
Setup	Trusted third party
SKGen	Trusted third party
RKGen	Trusted third party
Enc/DeniEnc	Sender
Dec	Receiver

Table 4.1: DME's algorithms and the executors

4.2 Definition

There are four participants in our DME scheme: trusted third party, public bulletin board, sender and receiver. Trusted third party first generates the encryption environment and publishes keys for sender and receiver. Then, sender uses the encryption key to encrypt the message and sends the ciphertext to the public bulletin board. Finally, receiver uses the decryption key to decrypt the ciphertext on the public bulletin board and obtain the message from sender. Our DME scheme is composed of the following algorithms:

- **Setup**$(1^\lambda) \rightarrow \{\mathbf{mpk}, \mathbf{msk}\}$: Given the security parameter 1^λ, the algorithm outputs master public key **mpk** and master secret key **msk**. We implicitly assume that all other algorithms take **mpk** as input.

- **SKGen**(**msk**, σ) $\rightarrow$ $\mathbf{ek}_\sigma$: Given the master secret key **msk** and the user's identity σ, the algorithm outputs the user's sender key $\mathbf{ek}_\sigma$.

- **RKGen**(**msk**, ρ) $\rightarrow$ $\mathbf{dk}_\rho$: Given the master secret key **msk** and the user's identity

ρ, the algorithm outputs the user's receiver key $\mathbf{dk}_\rho = (\mathbf{dk}_\rho^1, \mathbf{dk}_\rho^2, \mathbf{dk}_\rho^3)$.

- **Enc**($\mathbf{ek}_\sigma$, ρ, m) $\rightarrow c$: Given the user's sender key $\mathbf{ek}_\sigma$, the identity of the receiver ρ and the message m, the algorithm computes chameleon hash value for the messages and outputs ciphertext c.

- **DeniEnc**($\mathbf{ek}_{\sigma_0}, \rho_0, m_0, \mathbf{ek}_{\sigma_1}, \rho_1, m_1$) $\rightarrow \hat{c}$: Given the fake sender's key $\mathbf{ek}_{\sigma_1}$, the fake receiver's identity ρ_1 and the fake message m_1, the algorithm computes chameleon hash value for the messages and outputs the deniable ciphertext $\hat{c}$.

- **Dec**(c^*, $\mathbf{dk}_\rho$, σ) $\rightarrow m$ or $\perp$: Given the ciphertext c^*, where $c^* = c$ or $\hat{c}$, the receiver key $\mathbf{dk}_\rho$ and the sender's identity σ, the algorithm outputs the message m if the padding is valid. Otherwise, return $\perp$.

As we have mentioned above, our DME can forge a fake ciphertext to accomplish deniability. When using DME, the sender can choose whether he or she wants to encrypt the message with deniable functionality. If the sender choose to encrypt with the deniable way, **DeniEnc** will be used to encrypt the message and also forge a fake ciphertext with the input fake identities and message. As the fake sender identities, the sender can have a collusion with another one to get his or her identity or just use a public robot identity. The sender can still use his or her own identity as the fake identity to send a fake message to another receiver. As the fake receiver identities and fake messages, the sender can put anyone as the receiver and write anything in the message. After input all the thing

needed, **DeniEnc** will generate two encrypted messages, a true message and a fake message, such that these two messages have same hash value. So even if there is a malicious adversary force you decrypt the message, or force the trusted third party to reveal the key, the adversary can not distinguish that which message is true. Moreover, to protect the deniable functionality, if the sender choose not to encrypt the message with **DeniEnc**, **Enc** algorithm will randomly generate a fake message that there is no one can distinguish that the message is generated by **Enc** or **DeniEnc**. On the decryption side, since every encrypted messages contains two messages, **Dec** algorithm will randomly choose a message to decrypt first. If the decryption fails, **Dec** will then choose the other one. To determine whether the decryption is successful or not, we add a invertible padding function $\Phi()$ on the message so that when the padding of the message is valid, we can know that this is the original message sent by the sender. Compare to ME, although the decryption time may cost double, but we think it is still a good trade off to get the deniability.

Our DME scheme satisfies the following properties:

- **Semantic Security**: All messages m induce the same probability distribution on the ciphertext c, for all σ and ρ chosen as the sender and the receiver.
- **Deniability**: There is no efficient algorithm that can distinguish the ciphertext encrypted by normal encryption from the ciphertext encrypted by deniable encryption.

After introducing the algorithms of DME, here we defines G_{DME} as the game model of DME, which is the chosen plaintext attack model between a adversary and a challenger.

- **Setup**: The challenger runs **Setup** and outputs **mpk** to the adversary.
- **Query**: With the assistance of O_1, O_2, which are oracles of sender and receiver identities, the adversary generates key queries to the challenger and obtains their key results.
- **Challenge**: The adversary chooses two messages m_0, m_1 and two pairs of sender and receivers' identities $\sigma_0, \sigma_1, \rho_0, \rho_1$, for the challenger. The challenger randomly chooses one bit $b \in \{0, 1\}$ and encrypts the message m_b via **Enc** as well as sends it to the adversary as the challenge ciphertext. These identity pairs should not be included in **Query**.
- **Guess**: The adversary returns guess result $b' \in \{0, 1\}$. If the challenger returns 1, which means $b' = b$, the adversary wins the game.

As a deniable encryption, we also define $G_{DeniEnc}$ as the Indistinguishability of **DeniEnc** game model, which is a multi-phased game between a adversary and a challenger.

- **Setup**: The challenger runs **Setup** and outputs **mpk** to the adversary.
- **Query**: With the assistance of O_1, O_2, which are oracles of sender and receiver

$(\mathbf{mpk}, \mathbf{msk}) \xleftarrow{\$} \mathbf{Setup}(1^\lambda)$
$(\mathbf{ek}_\sigma, \mathbf{dk}_\rho) \leftarrow \mathbf{Query}^{O_1,O_2}(\sigma, \rho)$
$(m_0, m_1, \sigma_0, \sigma_1, \rho_0, \rho_1, \text{aux}) \xleftarrow{\$} A_1^{O_1,O_2}(1^\lambda, \mathbf{mpk})$
$b \xleftarrow{\$} \{0, 1\}$
$\mathbf{ek}_{\sigma_b} \leftarrow \mathbf{SKGen}(\mathbf{msk}, \sigma_b)$
$c \leftarrow \mathbf{Enc}(\mathbf{ek}_{\sigma_b}, \rho_b, m_b)$
$b' \leftarrow A_2^{O_1,O_2}(1^\lambda, c, \text{aux})$
If $(b' = b)$ **return** 1
Else **return** 0

Table 4.2: Game Model G_{DME} of DME, O_1, O_2 are oracles of sender and receiver identity

identities, the adversary generates key queries to the challenger and obtains their key results.

- **Challenge**: The adversary chooses a message m and two pairs of sender and receivers' identities $\sigma_0, \sigma_1, \rho_0, \rho_1$, for the challenger. The challenger randomly chooses one bit $b \in \{0, 1\}$. If $b = 0$, the challenger runs $\mathbf{Enc}(\mathbf{ek}_{\sigma_b}, \rho_b, m)$. If $b = 1$, the challenger runs $\mathbf{DeniEnc}(\mathbf{ek}_{\sigma_b}, \rho_b, m, \mathbf{ek}_{\sigma_{1-b}}, \rho_{1-b}, m_\$)$ where $m_\$$ is a random fake message. Finally, the challenger sends the ciphertext c to the adversary as the challenge ciphertext. These identity pairs should not be included in **Query**.

- **Guess**: The adversary returns guess result $b' \in \{0, 1\}$. If the challenger returns 1, which means $b' = b$, the adversary wins the game.

The definitions below capture the security definitions of DME:

Definition 2 (Correctness of DME)**.** *A DME algorithm* (***Setup***, ***SKGen***, ***RKGen***, ***Enc***, ***DeniEnc***, ***Dec***)

$(\mathbf{mpk}, \mathbf{msk}) \xleftarrow{\$} \mathbf{Setup}(1^\lambda)$
$(\mathbf{ek}_\sigma, \mathbf{dk}_\rho) \leftarrow \mathbf{Query}^{O_1,O_2}(\sigma, \rho)$
$(m, \sigma_0, \sigma_1, \rho_0, \rho_1, \text{aux}) \xleftarrow{\$} A_1^{O_1,O_2}(1^\lambda, \mathbf{mpk})$
$b \xleftarrow{\$} \{0,1\}$
$\mathbf{ek}_{\sigma_b} \leftarrow \mathbf{SKGen}(\mathbf{msk}, \sigma_b)$
$\mathbf{ek}_{\sigma_{1-b}} \leftarrow \mathbf{SKGen}(\mathbf{msk}, \sigma_{1-b})$
If $b = 0$, $c \leftarrow \mathbf{Enc}(\mathbf{ek}_{\sigma_b}, \rho_b, m)$
If $b = 1$, $\hat{c} \leftarrow \mathbf{DeniEnc}(\mathbf{ek}_{\sigma_b}, \rho_b, m, \mathbf{ek}_{\sigma_{1-b}}, \rho_{1-b}, m_\$)$
$b' \leftarrow A_2^{O_1,O_2}(1^\lambda, c, \text{aux})$
If $(b' = b)$ **return** 1
Else **return** 0

Table 4.3: Game Model $G_{DeniEnc}$ of DME, O_1, O_2 are oracles of sender and receiver identity

is correct if $\forall \lambda \in \mathbb{N}, \forall (\boldsymbol{mpk}, \boldsymbol{msk})$ *outputs by* $\boldsymbol{Setup}(1^\lambda), \forall m \in \mathcal{M}, \forall \sigma, \rho \in \{0,1\}^*$*:*

$$\mathbb{P}[\boldsymbol{Dec}(c^*, \boldsymbol{dk}_\rho, \sigma) = m] \geq 1 - \boldsymbol{neql}(\lambda)$$

whenever σ*,* ρ *are all correct identity and* $c^* = c = \boldsymbol{Enc}(\boldsymbol{ek}_\sigma, \rho, m)$*.*

In the deniable case, if σ_1*,* ρ_1 *and* m_1 *are the fake identities and message encrypted by* ***DeniEnc****, which means* $c^* = \hat{c} = \boldsymbol{DeniEnc}(\boldsymbol{ek}_{\sigma_0}, \rho_0, m_0, \boldsymbol{ek}_{\sigma_1}, \rho_1, m_1)$*:*

$$\mathbb{P}[\boldsymbol{Dec}(c^*, \boldsymbol{dk}_{\rho_0}, \sigma_0) = m_0] \geq 1 - \boldsymbol{neql}(\lambda)$$

and

$$\mathbb{P}[\boldsymbol{Dec}(c^*, \boldsymbol{dk}_{\rho_1}, \sigma_1) = m_1] \geq 1 - \boldsymbol{neql}(\lambda)$$

Definition 3. *If all polynomial time adversaries* A *have at most negligible advantages in*

G_{DME}:

$$\mathbb{P}[G_{DME}(\lambda) = 1] - \frac{1}{2} \leq \textbf{\textit{neql}}(\lambda)$$

then the DME scheme is CPA secure.

Definition 4 (Indistinguishability of **DeniEnc**). *If all polynomial time adversaries A have at most negligible advantages in $G_{DeniEnc}$:*

$$\mathbb{P}[G_{DeniEnc}(\lambda) = 1] - \frac{1}{2} \leq \textbf{\textit{neql}}(\lambda)$$

*then the DME scheme has indistinguishability of **DeniEnc**.*

Definition 5 (Secure of DME). *We say that DME is secure if it satisfies definition 2, definition 3 and definition 4.*

4.3 Construction

The entire algorithms are as follow:

Setup(1^λ) $\rightarrow$ {**mpk**, **msk**}: Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a symmetric pairing, and P be a generator of $\mathbb{G}$. $\mathbb{G}$ and $\mathbb{G}_T$ has an order q that depends on λ. Let $H : \{0,1\}^* \rightarrow \mathbb{G}$, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$, $H_2 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_3 : \mathbb{G}_T \rightarrow \{0,1\}^\ell$ be four hash functions. $\Phi : \{0,1\}^n \rightarrow \{0,1\}^\ell$ is a polynomial-time computable padding function and $\Phi(m)$ is invertible. When input the security parameter 1^λ, the setup algorithm randomly chooses

$r, s \in \mathbb{Z}_q$ and sets $P_0 = P^r$. Then it randomly picks a non-identity string ξ and computes $\mathbf{ek}_\xi = H_1(\xi)^s$ as the robot encryption key. Finally, it outputs the master public key $\mathbf{mpk} = (e, \mathbb{G}, \mathbb{G}_T, q, P, P_0, H, H_1, H_2, H_3, \mathbf{ek}_\xi, \Phi)$ and the master secret key $\mathbf{msk} = (r, s)$. All other algorithms are implicitly given **mpk** as input.

SKGen(msk, σ) $\rightarrow$ $\mathbf{ek}_\sigma$: When input the master secret key **msk** and the user's identity σ, the algorithm outputs the user's sender key $\mathbf{ek}_\sigma = H_1(\sigma)^s$.

RKGen(msk, ρ) $\rightarrow$ $\mathbf{dk}_\rho$: When input the master secret key **msk** and the user's identity ρ, the algorithm outputs the user's receiver key $\mathbf{dk}_\rho = (\mathbf{dk}_\rho^1, \mathbf{dk}_\rho^2, \mathbf{dk}_\rho^3) = (H_2(\rho)^r, H_2(\rho)^s, H_2(\rho))$.

Enc(ek$_\sigma$, $\rho, m)) \rightarrow c$: When input the encryption key $\mathbf{ek}_\sigma$, the receiver's identity ρ and a message $m \in \{0,1\}^n$, the algorithm first randomly picks $u, t \in \mathbb{Z}_q$ and computes:

- $T = P^t$ and $U = P^u$.
- $k_R = e(H_2(\rho), P_0^u)$ and $k_S = e(H_2(\rho), T \cdot \mathbf{ek}_\sigma)$.
- $V_0 = \Phi(m) \oplus H_3(k_R) \oplus H_3(k_S)$.

Then, the algorithm chooses a chameleon hash function CH to compute the hash value of the message and try to forge a fake ciphertext. Let aux_{CH} be the public information of CH:

- Randomly choose $d_0, d_1 \in \{0,1\}^n$.

- Randomly choose $V_1 \in \{0,1\}^{\ell}$.
- Find h such that $h = CH(m, d_0)$.
- Output ciphertext $c = (h, T, U, V_0, V_1, d_0, d_1, aux_{CH})$. The order of (V_0, V_1) and (d_0, d_1) does not affect the security of the scheme.

DeniEnc($\mathbf{ek}_{\sigma_0}, \rho_0, m_0, \mathbf{ek}_{\sigma_1}, \rho_1, m_1$) $\rightarrow \hat{c}$: When input the sender key $\mathbf{ek}_{\sigma_0}$, the receiver's identity ρ_0, a message $m_0 \in \{0,1\}^n$ and the fake sender key $\mathbf{ek}_{\sigma_1}$, the fake receiver's identity ρ_1, a fake message $m_1 \in \{0,1\}^n$, the algorithm first randomly picks $u, t \in \mathbb{Z}_q$ and computes:

- $T = P^t$ and $U = P^u$.
- $k_{R_0} = e(H_2(\rho_0), P_0^u)$ and $k_{S_0} = e(H_2(\rho_0), T \cdot \mathbf{ek}_{\sigma_0})$.
- $V_0 = \Phi(m_0) \oplus H_3(k_{R_0}) \oplus H_3(k_{S_0})$.

Then, the algorithm chooses a chameleon hash function CH to compute the hash value of the message and try to forge a fake ciphertext. Let aux_{CH} be the public information of CH. To forge a deniable ciphertext which has the same hash value with the true ciphertext, the algorithm computes a fake ciphertext with prepared sender key $\mathbf{ek}_{\sigma_1}$ ($\mathbf{ek}_{\sigma_1}$ can be $\mathbf{ek}_{\sigma_0}$ itself, or someone's encryption key if there is a collusion, or just $\mathbf{ek}_{\xi}$ if there is no collusion), fake receiver identity ρ_1 and fake message m_1. The algorithm will do as follow:

- Randomly choose $d_0 \in \{0,1\}^n$.

- $k_{R_1} = e(H_2(\rho_1), P_0^u)$ and $k_{S_1} = e(H_2(\rho_1), T \cdot \mathbf{ek}_{\sigma_1})$.

- $V_1 = \Phi(m_1) \oplus H_3(k_{R_1}) \oplus H_3(k_{S_1})$.

- Find h and d_1 such that $h = CH(m_0, d_0) = CH(m_1, d_1)$.

- Output ciphertext $\hat{c} = (h, T, U, V_0, V_1, d_0, d_1, aux_{CH})$. The order of (V_0, V_1) and (d_0, d_1) does not affect the security of the scheme.

Dec(c^*, $\mathbf{dk}_\rho$, σ) $\rightarrow$ m or $\perp$: When input the ciphertext c^*, where $c^* = c$ or $\hat{c}$, a decryption key $\mathbf{dk}_\rho$ and the sender's identity σ, **Dec** computes as follow:

- Parse c^* as $(h, T, U, V_0, V_1, d_0, d_1, aux_{CH})$.

- Randomly choose between V_0 and V_1 to decrypt first. Suppose V_0 is chosen to decrypt first.

- $k_R = e(\mathbf{dk}_\rho^1, U)$ and $k_S = e(\mathbf{dk}_\rho^2, H_1(\sigma)) \cdot e(\mathbf{dk}_\rho^3, T)$.

- $\Phi(m) = V_0 \oplus H_3(k_R) \oplus H_3(k_S)$.

- If the padding is valid, return m. Otherwise, the algorithm chooses the other ciphertext V_1 and repeats the step of decryption.

- If both paddings are invalid, return $\perp$.

4.4 Correctness

As the algorithm above, we can see that if the sender and receiver are all correct, which means $\sigma = \sigma^*$ and $\rho = \rho^*$:

- $k_R = e(H_2(\rho), P_0^u) = e(H_2(\rho)^r, P^u) = e(H_2(\rho^*)^r, P^u) = e(\mathbf{dk}_{\rho^*}^1, U) = {k_R}^*$
- $k_S = e(H_2(\rho), T \cdot \mathbf{ek}_\sigma) = e(H_2(\rho), T \cdot H_1(\sigma)^s) = e(H_2(\rho), T) \cdot e(H_2(\rho)^s, H_1(\sigma)) =$ $e(H_2(\rho^*), T) \cdot e(H_2(\rho^*)^s, H_1(\sigma)) = e(\mathbf{dk}_{\rho^*}^3, T) \cdot e(\mathbf{dk}_{\rho^*}^2, H_1(\sigma)) = {k_S}^*$

Here we note that k_R is computed by the sender and ${k_R}^*$ is computed by the receiver, as so does k_S and ${k_S}^*$. We can see that the sender and the receiver can compute the same value of k_R and k_S, which also means that since $V = \Phi(m) \oplus H_3(k_R) \oplus H_3(k_S)$, the receiver can get $\Phi(m)$ by computing $\Phi(m) = V \oplus H_3(k_R) \oplus H_3(k_S)$.

If the sender and receiver's identity matches the fake ciphertext, which means $\sigma' = \sigma^*$ and $\rho' = \rho^*$, the fake ciphertext can also be decrypted successfully:

- $k'_R = e(H_2(\rho'), P_0^u) = e(H_2(\rho')^r, P^u) = e(H_2(\rho^*)^r, P^u) = e(\mathbf{dk}_{\rho^*}^1, U) = {k_R}^*$
- $k'_S = e(H_2(\rho'), T \cdot \mathbf{ek}_{\sigma'}) = e(H_2(\rho'), T \cdot H_1(\sigma')^s) = e(H_2(\rho'), T) \cdot e(H_2(\rho')^s, H_1(\sigma')) =$ $e(H_2(\rho^*), T) \cdot e(H_2(\rho^*)^s, H_1(\sigma)) = e(\mathbf{dk}_{\rho^*}^3, T) \cdot e(\mathbf{dk}_{\rho^*}^2, H_1(\sigma)) = {k_S}^*$

This situation might happen when a malicious adversary forces the trusted third party reveal all the keys it published and decrypt the ciphertexts by brute force attack, or a sender try to make some fake evidences to prevent the true message from a coercer.

Of course, there are some situation that the decryption will fail:

- $k_R = e(H_2(\rho), P_0^u) = e(H_2(\rho)^r, P^u) \neq e(\mathbf{dk}_{\rho^*}^1, U) = {k_R}^*$ if the receiver's identity is incorrect.
- $k_S = e(H_2(\rho), T \cdot \mathbf{ek}_\sigma) = e(H_2(\rho), T \cdot H_1(\sigma)^s) = e(H_2(\rho), T) \cdot e(H_2(\rho)^s, H_1(\sigma)) \neq e(\mathbf{dk}_{\rho^*}^3, T) \cdot e(\mathbf{dk}_{\rho^*}^2, H_1(\sigma^*)) = {k_S}'$ if one of the sender or receiver's identity is incorrect.

Chapter 5 Security Analysis

5.1 Semantic Security

Theorem 4. *Let A be an adversary that breaks G_{DME} with advantage ϵ. Then, there exist an algorithm A' with advantage ϵ_0 against ME, where $\epsilon_0 \approx \epsilon$.*

Proof: Assume that there is a PPT algorithm A which has non-negligible advantage against G_{DME}. We can build an adversary A', which is a PPT algorithm, to beat ME game model. First, the challenger starts the game by running the **Setup** algorithm of ME and gives the public parameters $(e, \mathbb{G}, \mathbb{G}_T, q, P, P_0, H, H_1, H_2, H_3, \Phi)$ to A'. A' interact with the challenger and adversary A in the following ways shown in table 5.1.

Here are the detail steps in table 5.1:

- **Setup**: The challenger runs **Setup** and outputs **mpk** to A', A' gives all the parameters to A.
- **Query**: With the assistance of O_1, O_2, which are oracles of sender and receiver

identities, A generates key queries to the challenger and obtains their key results.

- **Challenge**:
  - A chooses two messages m_0, m_1 and two pairs of sender and receivers' identities $\sigma_0, \sigma_1, \rho_0, \rho_1$ to A', and A' gives these parameters to the challenger.
  - The challenger randomly chooses one bit $b \in \{0,1\}$, encrypts the message m_b via **Enc** and sends it to A' as the challenge ciphertext.
  - A' runs the second part of **Enc** algorithm (here we denote it as **Enc***) to generates the fake part of c. That is, A' first parses $c = (T, U, V_0)$. Then A' randomly chooses $d_0 \in \{0,1\}^n$, $V_1 \in \{0,1\}^\ell$ and finds h and d_1 where $h = CH(m_0, d_0) = CH(m_1, d_1)$. Finally, A' sends $c' = (h, T, U, V_0, V_1, d_0, d_1, aux_{CH})$ to A.
- **Guess**: A returns guess result $b' \in \{0,1\}$ to A' and A' returns the same guess result to the challenger. Since A has non-negligible advantage against G_{DME},the advantage ϵ_0 of the guess result of A' has nearly the same advantage against ME.

5.2 Deniability

Theorem 5. *Let A be an adversary that breaks $G_{DeniEnc}$ with advantage ϵ. Then, there exist an algorithm A' with advantage ϵ_1 against ME, where $\epsilon_1 \approx \epsilon$.*

ME	A'	A
$(\mathbf{mpk}, \mathbf{msk}) \xleftarrow{\$} \mathbf{Setup}(1^\lambda)$		
		$(\mathbf{ek}_\sigma, \mathbf{dk}_\rho) \leftarrow \mathbf{Query}^{O_1,O_2}(\sigma, \rho)$
		$(m_0, m_1, \rho_0, \rho_1, \sigma_0, \sigma_1, aux) \xleftarrow{\$} A_1^{O_1,O_2}(1^\lambda, \mathbf{mpk})$
$b \xleftarrow{\$} \{0,1\}$		
$\mathbf{ek}_{\sigma_b} \leftarrow \mathbf{SKGen}(\mathbf{msk}, \sigma_b)$		
$\mathrm{c} \leftarrow \mathbf{Enc}(\mathbf{ek}_{\sigma_b}, \rho_b, m_b)$		
	$c' \leftarrow \mathbf{Enc}^*(c)$	
		$b' \leftarrow A_2^{O_1,O_2}(1^\lambda, c', \text{aux})$
	$b' \leftarrow A_2^{O_1,O_2}(1^\lambda, c', \text{aux})$	

Table 5.1: Attack Game of G_{DME}

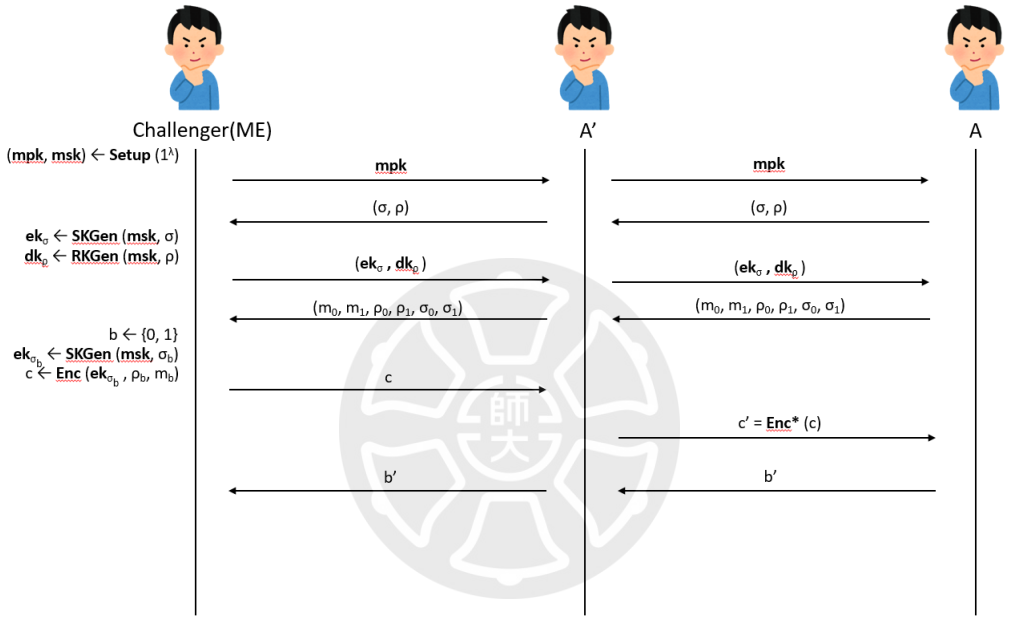

Figure 5.1: Game Process of G_{DME}

Proof: Assume that there is a PPT algorithm A which has non-negligible advantage against $G_{DeniEnc}$. We can build an adversary A', which is a PPT algorithm, to beat ME game model. First, the challenger starts the game by running the **Setup** algorithm of ME and gives the public parameters $(e, \mathbb{G}, \mathbb{G}_T, q, P, P_0, H, H_1, H_2, H_3, \Phi)$ to A'. A' interact with the challenger and adversary A in the following ways shown in table 5.2.

Here are the detail steps in table 5.2:

- **Setup**: The challenger runs **Setup** and outputs **mpk** to A', A' gives all the parameters to A.

- **Query**: With the assistance of O_1, O_2, which are oracles of sender and receiver identities, A generates key queries to the challenger and obtains their key results. A' also generates a key query of random identity σ^* to the challenger and obtains $\mathbf{ek}_{\sigma^*}$.

- **Challenge**:

  – A chooses two messages m_0, m_1 and two pairs of sender and receivers' identities $\sigma_0, \sigma_1, \rho_0, \rho_1$ to A', and A' gives these parameters to A'.

  – A' randomly generates a message m_2, ρ_2, σ_2 and sends $(m_1, m_2, \rho_1, \rho_2, \sigma_1, \sigma_2)$ to the challenger.

  – The challenger randomly chooses one bit $b \in \{0, 1\}$, encrypts the message $m_(b + 1)$ via **Enc** and sends it to A' as the challenge ciphertext.

  – A' runs **Enc**($\mathbf{ek}_{\sigma^*}, \rho_0, m_0$) to generate c'. Then A' randomly chooses $d_0 \in \{0, 1\}^n$ and finds h and d_1 where $h = CH(m_0, d_0) = CH(m_1, d_1)$. Finally, A' sends $c^* = (h, c, c', d_0, d_1, aux_{CH})$ to A.

- **Guess**: A returns guess result $b' \in \{0, 1\}$ to A' and A' returns the opposite of guess result to the challenger. If the challenger flips 0, $c^* = \mathbf{Deni}(\mathbf{ek}^*_\sigma, \rho_0, m_0, \mathbf{ek}_{\sigma_1}, \rho_1, m_1)$ since the identities and messages are all pre-selected. If the challenger flips 1,

ME	A'	A
$(\mathbf{mpk}, \mathbf{msk}) \xleftarrow{\$} \mathbf{Setup}(1^\lambda)$		
		$(\mathbf{ek}_\sigma, \mathbf{dk}_\rho) \leftarrow \mathbf{Query}^{O_1,O_2}(\sigma, \rho)$
	$(\mathbf{ek}_{\sigma^*}) \leftarrow \mathbf{Query}^{O_1,O_2}(\sigma^*)$	
		$(m_0, m_1, \rho_0, \rho_1, \sigma_0, \sigma_1, aux) \xleftarrow{\$} A_1^{O_1,O_2}(1^\lambda, \mathbf{mpk})$
	$(m_2, \sigma_2, \rho_2) \xleftarrow{\$} \{0,1\}^*$ $(m_1, m_2, \rho_1, \rho_2, \sigma_1, \sigma_2, aux)$	
$b \xleftarrow{\$} \{0,1\}$ $ek_{\sigma_{b+1}} \leftarrow \mathbf{SKGen}(\mathbf{msk}, \sigma_{b+1})$ $\mathbf{c} \leftarrow \mathbf{Enc}(ek_{\sigma_{b+1}}, \rho_{b+1}, m_{b+1})$		
	$c' \leftarrow \mathbf{Enc}(\mathbf{ek}_{\sigma^*}, \rho_0, m_0)$ $c^* = (h, c, c', d_0, d_1, aux_{CH})$	
		$b' \leftarrow A_2^{O_1,O_2}(1^\lambda, c^*, \text{aux})$
	$b' \leftarrow A_2^{O_1,O_2}(1^\lambda, c^*, \text{aux})$	

Table 5.2: Attack Game of $G_{DeniEnc}$

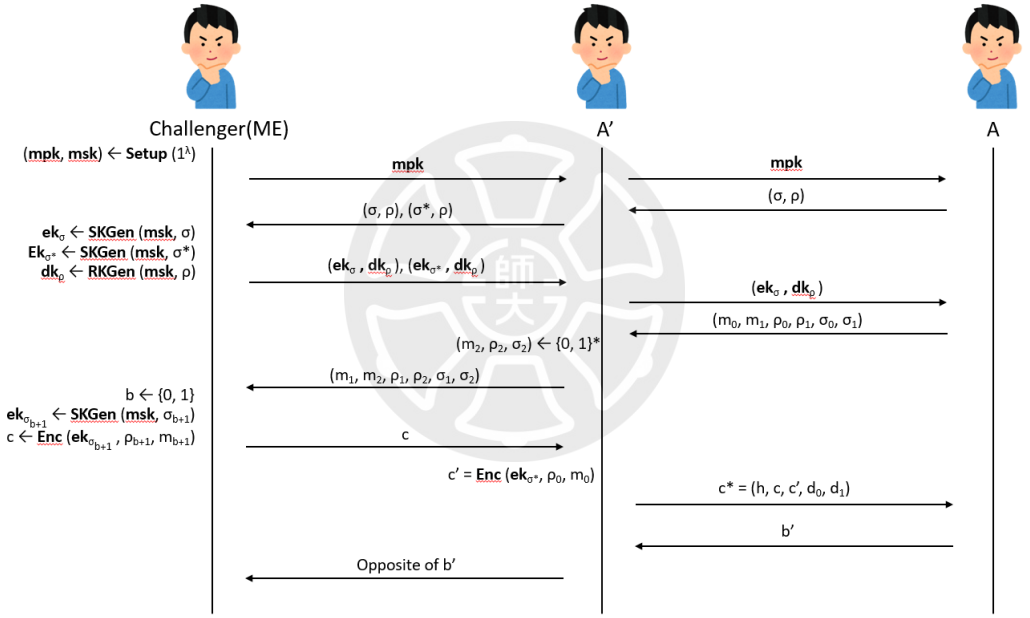

Figure 5.2: Game Process of $G_{DeniEnc}$

$c^* = \mathbf{Enc}(\mathbf{ek}_{\sigma^*}, \rho_0, m_0)$ since m_2 part is like the random string to A. Since A has non-negligible advantage against $G_{DeniEnc}$, the advantage ϵ_1 of the guess result of A' has nearly the same advantage against ME.

Chapter 6 Performance Estimation

In this section, we estimate DME's space cost and computation cost from ME. In table 6.1, we denote $|\mathbb{G}|$ as the length of the elements in $\mathbb{G}$, ℓ as the length of elements in $\mathbb{G}_T$ and $|\mathbb{G}_{CH}|$ as the length of the chameleon hash function's output. We can see that our space cost is more than ME's space cost due to our deniability. For the encryption key part, our DME have to keep $\mathbf{ek}_{\sigma_0}$ from normal encryption and $\mathbf{ek}_{\sigma_1}$ from deniable encryption. And for the ciphertext part, since we forge a fake ciphertext due to deniability, our space cost of ciphertext has a extra ℓ for the fake ciphertext and extra space for the hash value, also some random strings for the hash function. Although we spend almost double cost on space comparing with original ME, but the cost is affordable and the trade off for deniability is still worthy.

As the computation cost which is shown in table 6.2, we denote e_t as exponential computation, h as hash function computation, ch as chameleon hash function computation and e as bilinear mapping computation. Our DME focus at encryption scheme, so the computation time in **Setup**, **SKGen** and **RKGen** are equal to ME. In **Enc** and **DeniEnc**,

Space cost		
Element	Theoretical cost(ME)	Theoretical cost(DME)
Encryption key	$\|\mathbb{G}\|$	$2\|\mathbb{G}\|$
Decryption key	$3\|\mathbb{G}\|$	$3\|\mathbb{G}\|$
Message	n	n
Ciphertext	$2\|\mathbb{G}\| + \ell$	$\|\mathbb{G}_{CH}\| + 2\|\mathbb{G}\| + 2\ell + 2n$

Table 6.1: Space Cost of DME

Computation cost		
Element	Theoretical cost(ME)	Theoretical cost(DME)
Setup	e_t	e_t
SKGen	$h + e_t$	$h + e_t$
RKGen	$3h + 2e_t$	$3h + 2e_t$
Enc/DeniEnc	$2e + 2h + 3e_t$	$2(2e + 2h + 3e_t) + ch + 4e_t$
Dec	$3e + h$	$2(3e + h)$

Table 6.2: Computation Cost of DME

since our DME generate a fake ciphertext which has the same hash value with the true ciphertext, the computation time is twice as the ME's and has an additional computation time for forging the random string which is used in chameleon hash function. Finally, in **Dec**, the worst decryption case of DME is that the receiver picks the fake ciphertext to decrypt, so the worst computation cost is also twice as the ME's. Combining all the computation cost, we know that DME's computational time is nearly twice as the ME's, but if we consider with the deniability given by DME, the computation cost is also affordable and worthy.

Chapter 7 Conclusions

In this paper, we propose a new encryption called Deniable Matchmaking Encryption to solve the problem that Matchmaking Encryption might face in real life. We add a verification property that the users of DME can easily deny their association with the messages if they are coerced to reveal the plaintext or sender and receiver's identity. We also make a reduction from DME to ME to prove the security of DME. We believe that with our DME, the users can communicate to each others more safely. Our next step is to reduce the space and computation cost of DME to make it more light-weight. We are also trying to advance the identity-based scheme into the attribute-based scheme, which makes DME not only an one-to-one encryption scheme.

References

[1] G. Ateniese and B. de Medeiros. Identity-based chameleon hash and applications. In A. Juels, editor, Financial Cryptography, pages 164–180, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[2] G. Ateniese and B. de Medeiros. On the key exposure problem in chameleon hashes. In C. Blundo and S. Cimato, editors, Security in Communication Networks, pages 165–179, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[3] G. Ateniese, D. Francati, D. Nuñez, and D. Venturi. Match me if you can: Match-making encryption and its applications. In A. Boldyreva and D. Micciancio, editors, Advances in Cryptology – CRYPTO 2019, pages 701–731, Cham, 2019. Springer International Publishing.

[4] N. Attrapadung and H. Imai. Conjunctive broadcast and attribute-based encryption. In H. Shacham and B. Waters, editors, Pairing-Based Cryptography – Pairing 2009, pages 248–265, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[5] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In 2007 IEEE Symposium on Security and Privacy (SP '07), pages 321–334, 2007.

[6] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, Advances in Cryptology — CRYPTO 2001, pages 213–229, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[7] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. S. Kaliski, editor, Advances in Cryptology — CRYPTO '97, pages 90–104, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[8] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM, 24(2):84–90, Feb. 1981.

[9] X. Chen, F. Zhang, and K. Kim. Chameleon hashing without key exposure. In K. Zhang and Y. Zheng, editors, Information Security, pages 87–98, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[10] X. Chen, F. Zhang, W. Susilo, and Y. Mu. Efficient generic on-line/off-line signatures without key exposure. In J. Katz and M. Yung, editors, Applied Cryptography and Network Security, pages 18–30, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[11] J. Choi and S. Jung. A handover authentication using credentials based on chameleon hashing. IEEE Communications Letters, 14(1):54–56, 2010.

[12] I. Damgård and J. B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In M. Bellare, editor, Advances in Cryptology — CRYPTO 2000, pages 432–450, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[13] Y. Desmedt and J.-J. Quisquater. Public-key systems based on the difficulty of tampering (is there a difference between des and rsa?). In A. M. Odlyzko, editor, Advances in Cryptology — CRYPTO' 86, pages 111–117, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[14] C.-I. Fan, L.-Y. Huang, and P.-H. Ho. Anonymous multireceiver identity-based encryption. IEEE Transactions on Computers, 59(9):1239–1249, 2010.

[15] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding routing information. In R. Anderson, editor, Information Hiding, pages 137–150, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[16] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, page 89 – 98, New York, NY, USA, 2006. Association for Computing Machinery.

[17] S. Guo, D. Zeng, and Y. Xiang. Chameleon hashing for secure and privacy-preserving vehicular communications. IEEE Transactions on Parallel and Distributed Systems, 25(11):2794–2803, 2014.

[18] M. Klonowski, P. Kubiak, and M. Kutyłowski. Practical deniable encryption. In V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, and M. Bieliková, editors, SOFSEM 2008: Theory and Practice of Computer Science, pages 599–609, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[19] H. Krawczyk and T. Rabin. Chameleon hashing and signatures, 1998.

[20] A. Lewko and B. Waters. Decentralizing attribute-based encryption. In K. G. Paterson, editor, Advances in Cryptology – EUROCRYPT 2011, pages 568–588, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[21] J. Li, J. Li, X. Chen, C. Jia, and W. Lou. Identity-based encryption with outsourced revocation in cloud computing. IEEE Transactions on Computers, 64(2):425–437, 2015.

[22] U. M. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In D. W. Davies, editor, Advances in Cryptology — EUROCRYPT '91, pages 498–507, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[23] P. Mohassel. One-time signatures and chameleon hash functions. In A. Biryukov,

G. Gong, and D. R. Stinson, editors, Selected Areas in Cryptography, pages 302–319, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[24] S. Müller, S. Katzenbeisser, and C. Eckert. Distributed attribute-based encryption. In P. J. Lee and J. H. Cheon, editors, Information Security and Cryptology – ICISC 2008, pages 20–36, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[25] A. O'Neill, C. Peikert, and B. Waters. Bi-deniable public-key encryption. In P. Rogaway, editor, Advances in Cryptology – CRYPTO 2011, pages 525–542, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[26] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, pages 457–473, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[27] A. Sahai and B. Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, STOC '14, page 475–484, New York, NY, USA, 2014. Association for Computing Machinery.

[28] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, Advances in Cryptology, pages 47–53, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[29] H. Tanaka. A realization scheme for the identity-based cryptosystem. In C. Pomerance, editor, Advances in Cryptology — CRYPTO '87, pages 340–349, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[30] S. Tsujii and T. Itoh. An id-based cryptosystem based on the discrete logarithm problem. IEEE Journal on Selected Areas in Communications, 7(4):467–473, 1989.

[31] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, pages 114–127, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.