Almost-tight Identity Based Encryption against Selective Opening Attack

Junqing Gong† Xiaolei Dong‡ Zhenfu Cao‡ Jie Chen§

October 10, 2015

Abstract

The paper presented an identity based encryption (IBE) under selective opening attack (SOA) whose security is almost-tightly related to a set of computational assumptions. Our result is a combination of Bellare, Waters, and Yilek’s method [TCC, 2011] for constructing (not tightly) SOA secure IBE and Hofheinz, Koch, and Striecks’ technique [PKC, 2015] on building almost-tightly secure IBE in the multi-ciphertext setting. In particular, we first tuned Bellare et al.’s generic construction for SOA secure IBE to show that a one-bit IBE achieving ciphertext indistinguishability under chosen plaintext attack in the multi-ciphertext setting (with one-sided publicly openability) tightly implies a multi-bit IBE secure under selective opening attack. Next, we almost-tightly reduced such a one-bit IBE to static assumptions in the composite-order bilinear groups employing the technique of Hofheinz et al. This yielded the first SOA secure IBE with almost-tight reduction.

Contents

1 Introduction ................................................................. 2
  1.1 Background and Problem ........................................... 2
  1.2 Our Technique ......................................................... 2
  1.3 Related Work .......................................................... 5
  1.4 Roadmap ................................................................. 5

2 Preliminaries ............................................................... 5
  2.1 Notations ................................................................. 5
  2.2 Code-Based Games ...................................................... 6
  2.3 Identity Based Encryptions .......................................... 6
  2.4 Security against Selective Opening Attacks ...................... 7
  2.5 Indistinguishability with One-sided Publicly Openability ....... 7

3 Tightly Reducing SOA Security to IND-CPA with \( \delta \)-1SPO .... 8
  3.1 Construction ............................................................ 8
  3.2 Security Analysis ....................................................... 9

4 Tightly Reducing IND-CPA IBE with \( \delta \)-1SPO to Static Assumptions 11
  4.1 Composite-order Bilinear Groups ................................ 11
  4.2 Construction ........................................................... 13
  4.3 Security Analysis ....................................................... 13

5 Conclusion .................................................................. 22

*Department of Computer Science and Engineering, Shanghai Jiao Tong University. Email: gongjunqing@126.com
†Shanghai Key Lab for Trustworthy Computing, East China Normal University. Email: dongxiaolei@sei.ecnu.edu.cn
‡Shanghai Key Lab for Trustworthy Computing, East China Normal University. Email: zfcao@sei.ecnu.edu.cn
§Shanghai Key Laboratory of Multidimensional Information Processing and Shanghai Key Lab of Trustworthy Computing, East China Normal University. Email: S080001@e.ntu.edu.sg
1 Introduction

1.1 Background and Problem

Boneh and Franklin [BF01] formalized the notion of ciphertext indistinguishability under chosen ciphertext attack (IND-CCA) for identity based encryptions (IBE) and proposed the first practical solution in the random oracle model. Since then IND-CCA security and its weakened version, ciphertext indistinguishability under chosen plaintext attack (IND-CPA), have been accepted as standard security definitions for IBE. However, stronger security guarantee is required in some application scenarios. In 2011, Bellare, Waters, and Yilek proposed the selective opening security (SOA) for IBE, which may act as the basis for discussing adaptively secure multi-party computation protocol [BWY11].

Different from IND-CPA, the SOA security formalized in [BWY11] considers a communication system with multiple senders and multiple receivers. Besides eavesdropping all ciphertexts from communication channel, the adversary can also corrupt a subset of senders, extracting their plaintexts as well as random coins they used when generating corresponding ciphertexts. The SOA security ensures that ciphertexts sent by uncorrupted senders should not reveal any useful information on their corresponding plaintexts.

To construct an IBE achieving SOA security described above, Bellare et al. [BWY11] introduced a new primitive, IND-CPA one-bit IBE with one-sided publicly openability (1SPO), which is analogous to a weaker form of deniable public key encryption (PKE) [CDNO97]. Informally, the 1SPO property here requires that one can publicly recover the random coins used to encrypt message 1. They showed that a one-bit IBE with such a security guarantee can be generically transformed to a multi-bit IBE with SOA security in a quite straight way and provided two concrete constructions for such type of IBE based on Boyen and Waters’s anonymous IBE [BW06] and De Caro, Iovino and Persiano’s anonymous IBE [DCIP10], which is further based on Lewko and Waters’ IBE [IW10] employing the recently developed dual system technique [Wat09]. This results in two SOA secure IBE schemes based on decisional linear assumption and general subgroup decision assumption, respectively.

However both resulting constructions are not tight, the security loss is $O(\ell)$ where $k$ is the number of senders and $\ell$ is the length of each message in binary form. Namely the advantage of breaking the SOA security of these scheme is bounded by the advantage of solving some computational assumption times a factor $O(\ell)$. This means that, in order to reach certain bit security, we have to use larger security parameters to compensate the security loss, which often leads to large group size and inefficient group operation. Therefore a tightly secure construction is desirable from both theoretical and practical points of view.

Our paper is devoted to develop an IBE scheme reaching SOA security in a tighter fashion. In particular, we give a SOA secure IBE almost-tightly reduced to several static assumptions using composite order bilinear groups. The “almost-tight” means the security loss is proportional to the security parameter $\lambda$ and independent of $k$ and $\ell$. In general, we consider $\lambda$ as a number far smaller than $k\ell$.

1.2 Our Technique

Our work is motivated by the work of Hofheinz, Koch, and Striecks on almost-tight IBE in the multi-instance, multi-ciphertext setting [HKS15]. Roughly speaking, the so-called “ciphertext indistinguishability in the multi-instance, multi-ciphertext setting” ensures the confidentiality of multiple ciphertexts simultaneously. In the paper, we only consider a special case, i.e., the “single-instance, multi-ciphertext” setting (IND-mCPA). On the other hand, in the terms of Hofheinz et al., the one-bit IBE used in Bellare et al.’s generic construction of SOA secure IBE is IND-CPA in the single instance, single ciphertext setting. Therefore a straight observation shows that, if we replace the one-bit IBE here with an IND-mCPA one-bit IBE also with 1SPO, this generic construction would become constantly tight. This result is quite obvious and is easy to demonstrate, we briefly present it in Section 3 to make the paper self-contained.

Having a tight generic transformation from IND-mCPA one-bit IBE to SOA secure multi-bit IBE, the remaining work is to build such a one-bit IBE whose security is almost-tightly related to some computational assumptions. Our solution is a combination of Bellare, Waters, and Yilek’s second construction [BWY11] and Hofheinz, Koch, and Striecks’ technique [HKS15]. Before explaining our solution, we first review these two basic work.

Dual System Technique. Both work follows the dual system technique invented by Waters [Wat09]. For an IBE scheme employing dual system technique, we define two forms for secret keys and ciphertexts, normal and semi-functional, which should be indistinguishable from each other. The normal keys and ciphertexts are
used in the real system. We will say they are in the normal space. The semi-functional keys and ciphertexts will only be used in security proof and are always defined as normal keys and ciphertexts mixed with some additional components. We will call these components semi-functional components and say they are in the semi-functional space. Relying on certain algebraic feature, we can ensure the independence of normal space and semi-functional space in some sense, which allows us to make some changes (say, increasing entropy and breaking some algebraic structure) in the semi-functional space but avoid negative impact on the normal space, i.e., the real system.

Bellare, Waters, and Yilek’s Method. Bellare et al.’s idea [BWY11] is originated from the work on deniable encryption [CDNO97]. From a high level, they built a one-bit IBE with both IND-CPA and one-sided invertible sampleability (1SIS). An encryption of message 0 has some specific algebraic structure which is detectable for secret keys (of course, related to the same identity) but is pseudo-random from view of outsiders (adversaries); an encryption of message 1 is truly random and invertible samplable using just master public key.

Assume a composite order bilinear groups \( (\mathbb{G}, \mathbb{G}_T, e, N = p_1p_2p_3p_4) \), we let \( \mathbb{G}_N \) be the subgroup of order \( N' | N \). We note that one can decompose \( \mathbb{G} \) as \( \mathbb{G}_{p_1} \times \mathbb{G}_{p_2} \times \mathbb{G}_{p_3} \times \mathbb{G}_{p_4} \) and pairing operation on two elements from two subgroups of co-prime orders equals \( 1_{\mathbb{G}_T} \). Their second construction works as follows. An encryption of 0 for identity \( \mathbb{ID} \in \mathbb{Z}_N \) is in the form of 

\[
(U_{14}^{\mathbb{ID}}X_{14})^{s_1}g_4, \quad W_{14}^{s_1}g_4
\]

where \( U_{14}, X_{14}, W_{14} \in \mathbb{G}_{p_1}, g_4 \in \mathbb{G}_{p_4} \) are parts of the master public key and \( s_1, t_4, l_4 \in \mathbb{Z}_N \) are random coins for encryption. Since both encryption and master public key are independent in subgroup \( \mathbb{G}_{p_4} \), the algebraic structure is hidden for the outsider, which can be proved following [DCIP10, LW10]. However the structure can be detected using the secret key in the form of

\[
g_1'g_3', \quad (U_{14}^{\mathbb{ID}}X_1)^r g_3
\]

because they let \( U_1, X_1, g_1 \in \mathbb{G}_{p_1} \) share the same \( \mathbb{G}_{p_1} \)-component with \( U_{14}, X_{14}, W_{14} \), respectively, and subgroup \( \mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3} \) and \( \mathbb{G}_{p_4} \) are pairwise orthogonal under pairing operations. Here the subgroup \( \mathbb{G}_{p_4} \) acts as the normal space while the subgroup \( \mathbb{G}_{p_2} \) is the semi-functional space. The remaining two subgroups \( \mathbb{G}_{p_2} \) and \( \mathbb{G}_{p_4} \) are used to additionally randomize keys and ciphertexts, respectively.

Hofheinz, Koch, and Striecks’s Construction and Proof Technique. Their construction inherited the main algebraic structure of Chen and Wee’s almost-tightly secure IBE in the single-ciphertext setting [CW13] which employed a variant form of Waters Hash [Wat05]. It also works on a bilinear group of composite order \( N = p_1p_2p_3p_4 \) as above but supports the identity space defined as the set of all \( n \)-bit binary string. An encryption of message \( m \) for identity \( \mathbb{ID} \in \{0, 1\}^n \) is in the form of

\[
g_1', \quad g_3^{\sum_{i=0}^{n-1} w[2^{i-m(i)}]}, \quad e(g_1, h)^{m \cdot R_4}
\]

where \( g_1 \in \mathbb{G}_{p_1}, g_3^w \in \mathbb{G}_{p_1}^2 \) and \( e(g_1, h)^m \in \mathbb{G}_T \) are parts of the master public key, and \( s \) is the random coin for encryption, while secret keys are in the form of

\[
h^a \cdot h_{123}^{\sum_{i=0}^{n-1} w[2^{i-m(i)}]} \cdot R_4, \quad h_{123}' \cdot R_4'
\]

where \( a \in \mathbb{Z}_N, h, h_{123} \in \mathbb{G}_{p_1}, h_{123}^w \in \mathbb{G}_{p_2}, h_{123} \in \mathbb{G}_{p_3} \) are given in the master secret keys, and \( r \in \mathbb{Z}_N, R_4, R_4' \in \mathbb{G}_{p_4} \) are random coins. Here subgroup \( \mathbb{G}_{p_1} \) again acts as the normal space, the semi-functional space consists of subgroup \( \mathbb{G}_{p_2} \) and \( \mathbb{G}_{p_3} \). The last subgroup \( \mathbb{G}_{p_4} \) is now used to randomize keys.

We now briefly review the proof procedure. The proof begins with introducing \( \mathbb{G}_{p_2} \)-component into ciphertexts and conceptually inserting an random function \( R_0 \) mapping identity space to subgroup \( \mathbb{G}_{p_2} \) whose output is truly random but independent of the input as follows.

\[
\text{ct:} \quad (g_1, g_2)^{r_1}, \quad (g_1, g_2)^{\sum_{i=0}^{n-1} w[2^{i-m(i)}]} \cdot e((g_1, g_2)^t, h^{\cdot R_0(\mathbb{ID})}) \cdot M
\]

\[
\text{sk:} \quad h^a \cdot R_0(\mathbb{ID}) \cdot h_{123}^{\sum_{i=0}^{n-1} w[2^{i-m(i)}]} \cdot R_4, \quad h_{123}' \cdot R_4'
\]

The next step is to gradually increase the dependence of \( R_0 \)'s output on its input, from 0 bit to \( n \) bits. Here we need \( n \) computationally arguments and arise \( \Theta(n) \) security loss but unrelated to the number of secret keys or
particular, the first step of the proof now results in the following ciphertexts and secret keys

\[
\begin{align*}
\text{CT} : & \quad (g_1 g_2)^y, \quad (g_1 g_2)^{\sum w[2i-\alpha(i)]} s, \quad e((g_1 g_2)^y, h^a \left[ R_{5}(\text{id}) \right]) \cdot m \\
\text{SK} : & \quad h^a \cdot \left[ R_{5}(\text{id}) \right] \cdot h_{123}^{\sum w[2i-\alpha(i)]} r \cdot R_4, \quad h_{123}^r \cdot R_4'
\end{align*}
\]

where now \( R_n \)'s output depends on all bits of identity \text{id}. Combining the feature of random function and the entropy of \( s \), the random coin for encryption, we can readily show the pseudo-randomness of term \( e((g_1 g_2)^y, h^a \cdot R_{5}(\text{id})) \) in all ciphertexts simultaneously in a tight fashion.

Our Attempts and Solution. At first glance, we can apply Bellare et al.'s idea directly to Hofheinz, Koch, and Striecks' construction. We may remove the master secret key \( h^a \) and drop payload mask \( e(g_1, h)^a \), put the structure into a bilinear group of order \( N = p_1 p_2 p_3 p_4 p_5 \), and use the extra subgroup \( \mathbb{G}_{p_5} \) to hide the structure of ciphertexts as \( [DCI10, BWY11] \). The result is as follows. An encryption of message \( 0 \) for identity \( \text{id} \in [0, 1]^n \) looks like

\[
g_1^s g_5^t, \quad g_1^{\sum w[2i-\alpha(i)]} s t'_i \cdot g_5^r
\]

where \( g_5^r, g_5^t \in \mathbb{G}_{p_5} \) as \( g_1^s \) and \( g_5^t \), while \( g_1 \in \mathbb{G}_{p_1} \) and \( g_1^w \in \mathbb{G}_{2^n} \) as secret keys for \( \text{id} \) before they are published with \( g_5 \) as the master public key. On the other hand, secret keys for \( \text{id} \) should look like

\[
\begin{align*}
&h_{123}^{\sum w[2i-\alpha(i)]} r \cdot R_4, \quad h_{123}^r \cdot R_4'
\end{align*}
\]

It is easy to verify the correctness of this construction. However the proof technique for proving IND-mCPA is not applicable now. We remark that it is the master secret \( \alpha \) that allows us to introduce the random function into the system after introducing \( \mathbb{G}_{p_5} \)-components into all ciphertexts.

A natural idea to overcome this problem may be to take \( h^a \) back in secret keys and put an additional term \( e(g_1, h)^a \) in ciphertexts. However we still don’t know how to prove the resulting scheme to be IND-mCPA yet even though Hofheinz et al.’s proof technique now works. To see that we remind the reader of the fact that our technical goal is to show all components in ciphertexts of message \( 0 \) are pseudo-random. The proof technique here finally allows us to argue the pseudo-randomness of \( e(g_1, h)^a \), but not \( \sum g_1^{\sum w[2i-\alpha(i)]} s t'_i \cdot g_5^r \), since no additional entropy has been introduced in it during the proof. It is worth noting that Attrapadung et al. [AHY15] showed how to tightly obtain anonymity from a similar structure (using prime-order bilinear groups), i.e., the IND-mCPA in our setting. However they required the key generation procedure to be somewhat deterministic. Namely they use the same random coins for generating secret keys for the same identity. In our paper, we continue to work with the classical definition where each secret key is created from fresh random coins.

Our solution is to turn to the original version of Waters Hash [Wat05] where the encoding of identity \text{id} is not linear but affine. In particular, we define an encryption for message \( 0 \) as

\[
g_1^s g_5^t, \quad g_1^{(u+\sum w[2i-\alpha(i)])} s t'_i \cdot g_5^r
\]

where the newly introduced \( g_1^u \in \mathbb{G}_{p_1} \) is also published as a part of master public key after hidden by a random element of \( \mathbb{G}_{p_1} \); secret keys are defined as

\[
\begin{align*}
h_{123}^{(u+\sum w[2i-\alpha(i)])} r \cdot R_4, \quad h_{123}^r \cdot R_4'
\end{align*}
\]

where \( u \) is also introduced to maintain correctness. At this time, we can insert a random function relying on the entropy of \( u \in \mathbb{Z}_N \) and continue the proof procedure, which circumvents the issues in our first attempt. In particular, the first step of the proof now results in the following ciphertexts and secret keys

\[
\begin{align*}
\text{CT} : & \quad (g_1 g_2)^y g_5^s, \quad (g_1 g_2)^{\sum w[2i-\alpha(i)]} s t'_i \cdot g_5^r \cdot g_2^{R_{5}(\text{id})} \\
\text{SK} : & \quad h_{123}^{R_{5}(\text{id}) r} \cdot h_{123}^{(u+\sum w[2i-\alpha(i)])} r \cdot R_4, \quad h_{123}^r \cdot R_4'
\end{align*}
\]

where we define the random function mapping from identity space to \( \mathbb{Z}_N \) instead of \( \mathbb{G}_{p_2 \cdot p_5} \). We note that, different from Hofheinz et al.’s proof, the random function in our proof is associated with random coin \( r \) in secret keys instead of standing alone, and is associated with random coin \( s \) directly in ciphertexts instead of
connected through bilinear map \( e \). Finally, we will reach the following configuration using Hofheinz et al.’s technique with tiny technical tuning due to our different definition of random function \( R_0 \) and \( R_n \).

\[
\text{CT} : \quad (g_1 g_2)^{R_{f(s)}} \cdot (g_1 g_2)^{(e + \sum w(2i-m[i])) s} g_3^{R_{f(s)}} \cdot g_2^{R_{f(s)}} \\
\text{SK} : \quad h_{23}^{R_{f(s)}} \cdot h_{123}^{(e + \sum w(2i-m[i])) r} \cdot R_4 \cdot h_{123}^{R_{f(s)}} \cdot R_4'
\]

At this moment, we see that the pseudo-random terms \( g_2^{R_{f(s)}} \) and \( h_{23}^{R_{f(s)}} \) are bound with \( g_1^{(e + \sum w(2i-m[i])) s} \) and \( h_{123}^{(e + \sum w(2i-m[i])) r} \) respectively, which facilitates the proof of pseudo-randomness of the entire ciphertexts and circumvents the problem in our second attempt. We show the formal description of this construction and its security analysis in Section 4.

1.3 Related Work

**Identity Based Encryption.** The notion of identity based encryption was introduced by Shamir [Sha85] to alleviate the cost of key management in traditional PKI framework. The first practical solution with formal security analysis for this promising primitive was found by Boneh and Franklin [BF01] in 2001 using bilinear groups. A bunch of constructions are proposed in the next several years, including classical Boneh-Boyen [BB04b, BB04a], Waters [Wat05], and Gentry [Gen06] construction.

A recent breakthrough in this field is the introduction of dual system technique by Waters [Wat09], which is now widely used for building various types of adaptively secure functional encryptions [OT10, LOS10, OT11, LW11, CLL13, Lew12, LW12, OT12b, OT12a, RCS12, JR13]. There also appears general frameworks investigating the dual system technique, especially its application in functional encryption [Att14, BKP14, Wee14, CGW15].

In 2013, Chen and Wee [CW13] established the first IBE scheme with almost-tight reduction to static assumptions in the standard model, combining the dual system technique with Naor-Reingold technique for pseudo-random functions [NR04]. The method was extended to the “multi-instance, multi-ciphertext” setting by Hofheinz, Koch, and Striecks [HKS15]. Two very recent work [AHY15, GCD’15] gave the prime-order realization of Hofheinz et al.’s composite-order construction. Besides that the work by Attrapadung et al. [AHY15] also proposed a framework for building almost-tight IBE from broadcast encodings [Att14], which leads to a lot of new constructions with diverse features.

**Selective Opening Security.** The study of selective opening security started from the field of public key encryption. Since the work of Bellare, Hofheinz, and Yilek [BHY09], the community obtained several solutions [FHKW10, HLOV11, Hof12, HLQ13, HJKS15, LP15] covering various settings such as chosen plaintext attack (SO-CPA) and chosen ciphertext attack (SO-CCA). In 2011, Bellare, Hofheinz, and Yilek brought SOA security into the field of IBE [BWY11] by considering SO-CPA security. The blank of SO-CCA security was recently filled by Lai et al. [LDL+14] relying on several newly introduced primitives. Our paper only considers SO-CPA following [BWY11]. Very recently, He et al. [HLL’15] gave an IBE scheme achieving indistinguishability-based SOA security under chosen-plaintext attack in the selective identity model, which is weaker than the security model used in [BWY11, LDL+14] and ours as well.

1.4 Roadmap

The remainder of our paper is organized in three parts. We first define some notations and review several concepts and security models in Section 2. Section 3 shows a generic construction from IND-mCPA one-bit IBE to SOA-secure multi-bit IBE with tight reduction. Such a one-bit IBE is presented in Section 4 followed by a series of static constructions in composite-order bilinear groups. Finally, we conclude the paper in Section 5.

2 Preliminaries

2.1 Notations

We employ \( x := f(y) \) to denote the process of assigning to \( x \) the result of \( f(y) \) for some formula \( f(\cdot) \) and some value \( y \). For any positive number \( n \), we define \([n] := \{1, \ldots, n\} \). For any list or vector \( \mathbf{w} \), we
let \( w[i] \) denote the \( i \)th entry of \( w \). Similarly, for any binary string \( i.d \in \{0,1\}^* \), we use \( i.d[i] \) to indicate the \( i \)th bit of \( i.d \). The notation \( y \leftarrow \text{Alg}(x_1, \ldots, x_n; r_1, \ldots, r_m) \) or \( \text{Alg}(x_1, \ldots, x_n; r_1, \ldots, r_m) \to y \) refer to the process of running algorithm \( \text{Alg} \) with inputs \( x_1, \ldots, x_n \) and random coins \( r_1, \ldots, r_m \), then assigning the result to variable \( y \). The random coins may be omitted for brevity. We may also use a more compact form \( y \leftarrow \text{Alg}(x; r) \) where \( x := (x_1, \ldots, x_n) \) and \( r := (r_1, \ldots, r_m) \). For any fixed input \( x \), the set \( \{\text{Alg}(x)\} \) is defined as \( \{y : \exists \text{ s.t. } \text{Alg}(x; r) = y\} \).

Given a finite cyclic group \( G \), we use \( X \leftarrow G \) to denote the process of sampling a random element in \( G \). In particular, the notation is for the so-called “lazy sampling”. Assuming \( g \in G \) is a generator of group \( G \) of order \( N \), we sample \( x \) from \( G \) by randomly sampling \( x \) from \( \mathbb{Z}_N \) and set \( x := g^x \). As \([\text{BWY11}]\), we consider sampling random element from \( \mathbb{Z}_N \) (for any \( N \in \mathbb{Z}_{>0} \)) as the unique random source in our system, and denote this atomic process by \( x \leftarrow \mathbb{Z}_N \). For any element \( g \in G \) and list or vector \( w = (w_1, \ldots, w_n) \in \mathbb{Z}_N^n \), we define \( g^w := (g^{w_1}, \ldots, g^{w_n}) \in G^n \). Given \( g = (g_1, \ldots, g_n) \) and \( h = (h_1, \ldots, h_n) \), two vectors over \( G \), we define \( g \cdot h := (g_1 \cdot h_1, \ldots, g_n \cdot h_n) \in G^n \).

### 2.2 Code-Based Games

As \([\text{BWY11}]\), we employ code based games \([\text{BR06}]\) for our security definitions and proofs. A code based game is defined by an \textbf{Initialize} procedure and a \textbf{Finalize} procedure plus a series of procedures, which will be used to answer adversary \( \mathcal{A} \)'s queries and depends on the security notion we concern. The game begins with running \textbf{Initialize} procedure and transmitting the result to adversary \( \mathcal{A} \). During the game, \( \mathcal{A} \) is allowed to make various types of queries in any order. In general, \( \mathcal{A} \) is capable of making polynomial-many queries. Finally, \( \mathcal{A} \) is expected to return an output before it halts. The output of the game is then obtained by invoking \textbf{Finalize} procedure on \( \mathcal{A} \)'s output. As usual, we let \( \text{Game}_\mathcal{A}(\lambda) = y \) denote the event that the output of executing Game with adversary \( \mathcal{A} \) on security parameter \( 1^\lambda \) is \( y \).

### 2.3 Identity Based Encryptions

**Algorithms.** An identity-based encryption scheme with identity space \( \text{IdSp} \) and message space \( \text{MsgSp} \) consists of four (probabilistic) polynomial time algorithms defined as follows:

- \( \text{Setup}(1^\lambda) \to (\text{MPK}, \text{MSK}) \). The setup algorithm takes as input a security parameter \( 1^\lambda \), and outputs a master public key \( \text{MPK} \) and the corresponding master secret key \( \text{MSK} \).
- \( \text{KeyGen}(\text{MPK}, \text{MSK}, \text{ID}) \to \text{SK} \). The key generation algorithm takes as input a master public key \( \text{MPK} \), a master secret key \( \text{MSK} \) and an identity \( \text{ID} \in \text{IdSp} \), and outputs a secret key \( \text{SK} \).
- \( \text{Enc}(\text{MPK}, \text{ID}, M) \to \text{CT} \). The encryption algorithm takes as input a master public key \( \text{MPK} \), an identity \( \text{ID} \in \text{IdSp} \) and a message \( M \in \text{MsgSp} \), outputs a ciphertext \( \text{CT} \).
- \( \text{Dec}(\text{MPK}, \text{SK}, \text{CT}) \to M \). The decryption algorithm takes as input a master public key \( \text{MPK} \), a secret key \( \text{SK} \) and a ciphertext \( \text{CT} \), outputs a message \( M \) or a failure symbol \( \bot \).

**Correctness.** The correctness (with negligible failure probability) requires that, for all security parameter \( \lambda \), for all \((\text{MPK}, \text{MSK}) \in [\text{Setup}(1^\lambda)]\), all \( \text{ID} \in \text{IdSp} \), and all \( M \in \text{MsgSp} \), it holds that

\[
\Pr\left[\text{Dec}(\text{MPK}, \text{KeyGen}(\text{MPK}, \text{MSK}, \text{ID}), \text{Enc}(\text{MPK}, \text{ID}, M)) = M\right] \geq 1 - \epsilon,
\]

where \( \epsilon \) is negligible in \( \lambda \) and the probability space is defined by random coins consumed by algorithm KeyGen and Enc.

**More Notations.** We now define two sets for an IBE scheme. We let \( \text{Coins}(\text{MPK}, M) \) be the set of random coins used to encrypt message \( M \) for all \((\text{MPK}, \text{MSK}) \in [\text{Setup}(1^\lambda)]\). We also use \( \text{Coins}(\text{MPK}, \text{ID}, \text{CT}, M) \) to denote \( \{r : \text{CT} = \text{Enc}(\text{MPK}, \text{ID}, M; r)\} \), the set of all random coins which makes algorithm Enc to produce \( \text{CT} \) as the ciphertext for message \( M \) under \( \text{MPK} \) and \( \text{ID} \).
2.4 Security against Selective Opening Attacks

Bellare et al. [BWY11] formally defined selective opening security in the setting of IBE based on the work of [BHY09]. This subsection review their definition through two games: SOAREAL and SOASIM, defined in Figure 1 and Figure 2, respectively. In the figures, \(\mathcal{M}\) is called \((k, \ell)\)-message sampler which takes \(a \in \{0, 1\}^n\) as input and returns \(m\) such that \(|m| = k\) and \(m[i] \in \{0, 1\}^\ell\) for all \(i \in [k]\), and \(\mathcal{R}\) is any randomized algorithm with binary output. In both games, an adversary must make one \(\text{NewMesg}\) query before one \(\text{Corrupt}\) query. Besides, game SOAREAL additionally allows the adversary to adaptively make polynomially-many \(\text{Extract}\) queries.

The advantage function of adversary \(A\) in game \(\text{INDmCPA}\) is defined as:

\[
\text{Adv}^\text{SOA}_{\mathcal{M}, \mathcal{R}, \mathcal{S}}(\lambda) := \left| \Pr \left[ \text{SOAREAL}_{\mathcal{M}, \mathcal{R}, \mathcal{S}}(\lambda) = 1 \right] - \Pr \left[ \text{SOASIM}_{\mathcal{M}, \mathcal{R}, \mathcal{S}}(\lambda) = 1 \right] \right|.
\]

An IBE scheme is SOA secure if and only if, for any message sampler \(\mathcal{M}\), any binary relation \(\mathcal{R}\), any probabilistic polynomial time SOA-adversary \(\mathcal{A}\), there exists an SOA-simulator \(\mathcal{S}\) such that the advantage function \(\text{Adv}^\text{SOA}_{\mathcal{M}, \mathcal{R}, \mathcal{S}}(\lambda)\) is negligible in \(\lambda\).

2.5 Indistinguishability with One-sided Publicly Openability

The ciphertext indistinguishability under chosen plaintext attack [BF01] (IND-CPA) is one of well-known security definitions for IBE. In the model, given master public key, an adversary is able to obtain polynomially-many secret keys adaptively and a single challenge ciphertext for one of challenge messages, and is asked to guess a secret bit (which indicates which challenge message is used to generate challenge ciphertext). We consider IND-CPA in the multi-ciphertext setting (IND-mCPA) where the adversary now has access to polynomially-many challenge ciphertexts, which is recently proposed and investigated in more general multi-instance, multi-ciphertext setting [HKS15].

More formally, we review the notion of IND-mCPA [HKS15] through game INDmCPA shown in Figure 3. The advantage function of adversary \(\mathcal{A}\) in game INDmCPA is defined as:

\[
\text{Adv}^\text{INDmCPA}_{\mathcal{A}}(\lambda) := \left| \Pr \left[ \text{INDmCPA}_{\mathcal{A}}(\lambda) = 1 \right] - 1/2 \right|.
\]
An IBE scheme is said to be IND-mCPA if and only if the advantage function $\text{Adv}^{\text{IND-mCPA}}(\lambda)$ is negligible in $\lambda$ for any probabilistic polynomial time adversary $\mathcal{A}$.

However the IND-mCPA alone is not sufficient for realizing SOA security. We require an additional property, called One-Sided Publicly Openability (1SPO), proposed by Bellare et al. [BWY11] based on the concept of deniable PKE [CDN97]. Roughly speaking, 1SPO for an IBE with $\text{MsgSp} = \{0, 1\}$ means that one can publicly open a ciphertext for message 1 by presenting its random coins. More formally, we review the following algorithm.

- OpenToOne$(\text{MPK}, \text{ID}, \text{CT}) \rightarrow r$. The setup algorithm takes as input a master public key MPK, an identity ID $\in \text{IdSp}$ and a ciphertext CT of 1, outputs a random coin $r$ such that $\text{CT} = \text{Enc}(\text{MPK}, \text{ID}, 1; r)$.

We allow algorithm OpenToOne to output failure symbol $\perp$ with probability $\delta$ and require that, for all $\text{MPK} \in \text{Setup}(1^2)$, all $\text{ID} \in \text{IdSp}$, all $\text{CT} \in \{\text{Enc}(\text{MPK}, \text{ID}, 1)\}$, and all $\hat{r} \in \text{Coins}(\text{MPK}, \text{ID}, \text{CT}, 1)$,

$$\Pr [r \leftarrow \text{OpenToOne}(\text{MPK}, \text{ID}, \text{CT}) : r = \hat{r} \neq \perp] = \frac{1}{|\text{Coins}(\text{MPK}, \text{ID}, \text{CT}, 1)|}$$

The algorithm is called $\delta$-1SP opener and an IBE with a $\delta$-1SP opener is called $\delta$-one-sided publicly openable ($\delta$-1SPO).

## 3 Tightly Reducing SOA Security to IND-CPA with $\delta$-1SPO

Bellare et al. [BWY11] presented a trivial construction for $\ell$-bit IBE $\text{IBE}^\ell$ from one-bit IBE IBE, and proved that IBE$^\ell$ achieves SOA security if the underlying IBE is IND-CPA with $\delta$-1SPO. Their reduction has polynomial security loss. We review this trivial construction (with different identity space) and prove that IBE$^\ell$ achieves SOA security if the underlying IBE is IND-mCPA with $\delta$-1SPO. We emphasize that our reduction only has constant security loss.

### 3.1 Construction

Assume an one-bit IBE $\text{IBE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ with $\text{IdSp} = \{0, 1\}^n$ and $\text{MsgSp} = \{0, 1\}$. The $\ell$-bit IBE $\text{IBE}^\ell = (\text{Setup}^\ell, \text{KeyGen}^\ell, \text{Enc}^\ell, \text{Dec}^\ell)$ with $\text{IdSp} = \{0, 1\}^n$ (identical to IBE) and $\text{MsgSp} = \{0, 1\}^\ell$ is defined as in Figure 4.

It is quite clear that the correctness of the resulting IBE follows from the underlying one. More concretely, if IBE is correct with failure probability $\epsilon$, $\text{IBE}^\ell$ is correct with failure probability $\ell \cdot \epsilon$.
3.2 Security Analysis

We want to prove the following theorem.

**Theorem 1** For any message sampler $\mathcal{M}$, any binary relation $\mathcal{R}$, any probabilistic polynomial time SOA-adversary $\mathcal{A}$ making at most $Q_\mathcal{A}$ Extract queries, there exists a probabilistic polynomial time SOA-simulator $\mathcal{S}$ and adversary $\mathcal{B}$ such that

$$\text{Adv}^{\text{SOA}}_{\mathcal{A},\mathcal{S},\mathcal{M},\mathcal{B}}(\lambda) \leq 2 \cdot \text{Adv}^{\text{IND}_{\text{mCPA}}}_{\mathcal{B}}(\lambda) + k\ell \cdot \delta.$$

The proof begins with constructing a SOA-simulator $\mathcal{S}$ in SOASIM game, which takes any probabilistic polynomial time SOA-adversary $\mathcal{A}$ as oracle. Our proof continues employing Bellare et al.’s construction and we recall it in Figure 5. The SOA-simulator $\mathcal{S}$ prepares a fresh master public/secret key pair $(\text{MPK}, \text{MSK})$ and initializes a SOA-adversary $\mathcal{A}$ using MPK. To obtain $\mathcal{A}$’s output, $\mathcal{S}$ answers all queries made by it, i.e., simulating SOAREAL game for $\mathcal{A}$. The Extract queries are directly answered using MSK. For the NewMesg query, SOA-simulator $\mathcal{S}$ returns $k$ encryptions of message $0^\ell$. In the figure, we use $\text{NewMesg}_{\mathcal{A}}$ to denote the NewMesg oracle for $\mathcal{A}$ in SOASIM game, which returns nothing to $\mathcal{S}$. For the Corrupt query, $\mathcal{S}$ opens corrupted messages correctly relying on the power of OpenToOne algorithm. In particular, if the corrupted message bit is $0$, the random coin used when answering NewMesg query could be returned directly; if the bit is $1$ instead, the random coin must be resampled so as to explain the corresponding ciphertext $\mathcal{S}$ has given to $\mathcal{A}$, which is originally an encryption of $0$, to be an encryption of $1$.

![Figure 5: SOA-simulator $\mathcal{S}$](image)

One may see that the simulation of $\mathcal{S}$ is different from the specification of SOAREAL. Therefore the next step of the proof is to show that the simulation described in Figure 5 and the real SOAREAL are computationally indistinguishable from the viewpoint of $\mathcal{A}$. In detail, we employ hybrid argument using the game sequence shown in Figure 6.

We have the following lemmas immediately.

**Lemma 1** For any message sampler $\mathcal{M}$, any binary relation $\mathcal{R}$, any adversary $\mathcal{A}$,

$$|\Pr[\text{Game}_{0,\mathcal{A},\mathcal{M},\mathcal{R}}(\lambda) = 1] - \Pr[\text{Game}_{1,\mathcal{A},\mathcal{M},\mathcal{R}}(\lambda) = 1]| \leq k\ell \cdot \delta.$$

**Lemma 2** For any message sampler $\mathcal{M}$, any binary relation $\mathcal{R}$, any adversary $\mathcal{A}$,

$$\Pr[\text{Game}_{2,\mathcal{A},\mathcal{M},\mathcal{R}}(\lambda) = 1] = \Pr[\text{Game}_{3,\mathcal{A},\mathcal{M},\mathcal{R}}(\lambda) = 1].$$

**Lemma 3** For any message sampler $\mathcal{M}$, any binary relation $\mathcal{R}$, any adversary $\mathcal{A}$,

$$\Pr[\text{Game}_{3,\mathcal{A},\mathcal{M},\mathcal{R}}(\lambda) = 1] = \Pr[\text{SOASIM}_{\mathcal{S},\mathcal{A},\mathcal{M},\mathcal{B}}(\lambda) = 1].$$

The first lemma follows from the fact that the algorithm OpenToOne is not perfect, its failure probability is $\delta$, and there are at most $k\ell$ applications of OpenToOne. For the second lemma, since we can answer NewMesg queries without knowing anything about $\mathcal{m}$, it is safe to defer the sampling of $\mathcal{m}$. The last lemma follows the observation that the SOA-simulator described in Figure 5 is able to simulate Game and $\mathcal{S}$’s output always equals $\mathcal{A}$’s.

To finish the proof of the theorem, we must fill the gap between Game$_1$ and Game$_2$. Especially, we prove the following lemma.

<table>
<thead>
<tr>
<th>main()</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\text{MPK}, \text{MSK}) \leftarrow \text{Setup}^\ell(1^\lambda)$</td>
</tr>
<tr>
<td>return $\mathcal{A}(\text{MPK})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NewMesg(id, $\alpha$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $i \in [k]$ do</td>
</tr>
<tr>
<td>$r[i] \leftarrow \text{Coins}(\text{MPK}, 0^\ell)$</td>
</tr>
<tr>
<td>$ct[i] \leftarrow \text{Enc}(\text{MPK}, \text{id}[i], 0^\ell; r[i])$</td>
</tr>
<tr>
<td>return $ct$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corrupt(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m[I] \leftarrow \text{Corrupt}_{\mathcal{A}}(I)$</td>
</tr>
<tr>
<td>for $i \in I$ and $j \in [\ell]$ do</td>
</tr>
<tr>
<td>if $m[i][j] = 1$ then</td>
</tr>
<tr>
<td>$r[i][j] \leftarrow \text{OpenToOne}(\text{MPK}, \text{id}[i], ct[i][j])$</td>
</tr>
<tr>
<td>return $(r[i], m[I])$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extract(id)</th>
</tr>
</thead>
<tbody>
<tr>
<td>return $\text{KeyGen}^\ell(\text{MPK}, \text{MSK}, \text{id})$</td>
</tr>
</tbody>
</table>

Figure 5: SOA-simulator $\mathcal{S}$. 

9
Figure 6: Game Sequence for SOA security

Lemma 4 (Game₁ ≈ Game₀) For any message sampler \( \mathcal{M} \), any binary relation \( \mathcal{R} \), any probabilistic polynomial time adversary \( \mathcal{A} \) making at most \( Q_K \) key extraction queries, there exists an adversary \( \mathcal{B} \) such that

\[
\Pr[\text{Game}_1,\mathcal{A},\mathcal{M},\mathcal{R},\mathcal{B}(\lambda) = 1] - \Pr[\text{Game}_2,\mathcal{A},\mathcal{M},\mathcal{R},\mathcal{B}(\lambda) = 1] \leq 2 \cdot \text{Adv}^{\text{INDmCPA}}_{\mathcal{B}}(\lambda).
\]

and \( \text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + (k\ell + Q_K) \cdot \text{poly}(\lambda) \) where \( \text{poly} \) is independent of \( \mathcal{A} \).

Proof. Given \( \mathcal{M} \) and oracle access to \( \text{Extract}_{\mathcal{A}} \) and \( \text{Challenge} \), algorithm \( \mathcal{B} \) proceeds as follows:

Initialize Return \( \mathcal{M} \).  
Extract(\( \text{id} \)) Return \( \text{Extract}_{\mathcal{A}}(\text{id}) \).

\text{NewMesg}(\text{id}, \alpha) \quad \text{Return } \bot \text{ when } \text{id} \cap \text{ExID} \neq \emptyset. \text{ Update ChlD := ChlD } \cup \text{id}. \text{ Sample } \text{m} \leftarrow \mathcal{M}(\alpha). \text{ For } i \in [k] \text{ and } j \in [\ell], \text{ if } \text{m}[i][j] = 1, \text{ set}

\[ \text{ct}[i][j] \leftarrow \text{Challenge}(\text{id}[i], 0, 1), \]

otherwise, sample

\[ \text{r}[i][j] \leftarrow \text{Coins}(\mathcal{M} \mathcal{P} \mathcal{K}, 0) \text{ and } \text{ct}[i][j] \leftarrow \text{Enc}(\mathcal{M} \mathcal{P} \mathcal{K}[i][i], \text{r}[i][j]|j). \]

Return ct.

\text{Corrupt}(I) \quad \text{For } i \in I \text{ and } j \in [\ell], \text{ if } \text{m}[i][j] = 1, \text{ set}

\[ \text{r}[i][j] \leftarrow \text{OpenToOne}(\mathcal{M} \mathcal{P} \mathcal{K}[i], \text{ct}[i][j]|j). \]

Return \( \text{r}[i], \text{m}[i] \).

When \( \mathcal{A} \) halts with output \( \text{out} \), algorithm \( \mathcal{B} \) outputs \( \mathcal{R}(\text{m}, \text{ChlD}, I, \text{out}) \).

Observe that if \( \beta = 0 \), the outputs of \text{Challenge} are encryptions of 0 and the simulation is identical to \text{Game}_2. \text{On the other hand, if } \beta = 1, \text{ the outputs of } \text{Challenge} \text{ are encryptions of 1} \text{ and the simulation is identical to } \text{Game}_1. \text{ Therefore we can conclude that}

\[ \Pr[\text{Game}_1,\mathcal{A},\mathcal{M},\mathcal{R},\mathcal{B}(\lambda) = 1] - \Pr[\text{Game}_2,\mathcal{A},\mathcal{M},\mathcal{R},\mathcal{B}(\lambda) = 1] \leq 2 \cdot \text{Adv}^{\text{INDmCPA}}_{\mathcal{B}}(\lambda). \]

\[ \square \]
4 Tightly Reducing IND-CPA IBE with $\delta$-1SPO to Static Assumptions

4.1 Composite-order Bilinear Groups

We assume a group generator $\text{GrpGen}$ for composite-order bilinear groups, which takes as input a security parameter $\lambda$, outputs $(N, G, G_T, e, p_1, \ldots, p_5)$ where $N = p_1 \cdots p_5$ and all $p_i$ are prime numbers in $[2^\lambda - 1, 2^\lambda - 1]$, both $G$ and $G_T$ are cyclic groups of order $N$, and $e : G \times G \to G_T$ is an admissible bilinear map. We let $G$ and $G_T$ also contain their generators, denoted by $g \in G$ and $g_T \in G_T$, respectively. We define public group description $\mathcal{G} = (N, G, G_T, e)$ and take factoring of $N$, i.e., $p_1, \ldots, p_5$, as secret information. It is the secret information that allow us to derive the generator of each subgroup from the generator of the entire group, i.e., $g \in G$, and sample random element from it.

For positive integer $N'$ with $N'|N$, we use $\mathbb{G}_N$ to indicate the unique subgroup of order $N'$. Given $g \in \mathbb{G}_N$ and $h \in \mathbb{G}_m$ such that $\gcd(n, m) = 1$, we have $e(g, h) = 1_{\mathbb{G}_1}$. In fact, we can decompose $G$ as $G_{p_1} \times \cdots \times G_{p_5}$ and write $g^x$ for $x \in \mathbb{Z}_N$ as $\prod_{i=1}^5 g_i^x$, where $G_{p_i}$ is generated by $g_i \in G_{p_i}$ and $x_i \in \mathbb{Z}_{p_i}$ is unique modulo $p_i$.

Under the representation, we call $\prod_{i \in I} g_i$ for some $I \subseteq [5]$ the $\mathbb{G}_{N'}$-part of $g^x$ where $N' = \prod_{i \in I} p_i$.

We need the following six computational assumptions to reach IND-mCPA security. Assumption 1, 2, 5, 6 are concrete instantiations of General Subgroup Decision Assumption formalized by Bellare et al. [BWY11] which is inspired by several concrete assumptions used in [BGN05, DCIP10, IW10]. Assumption 3 and 4 were modified from Hofheinz et al.’s Dual System Assumption 3 [HKS15] and Chen and Wee’s second assumption in [CW13], respectively. In fact the Dual System Assumption 3 [HKS15] was also derived from Chen and Wee’s second assumption. All of them could be viewed as Diffie-Hellman Assumption on subgroups of composite-order $G$.

**Assumption 1** For any probabilistic polynomial time adversary $\mathcal{A}$, the advantage function defined as follows are negligible in $\lambda$.

$$\text{Adv}_{\mathcal{A}}^{\text{ID1}}(\lambda) := \left| \Pr \left[ \mathcal{A}(D, T_0) = 1 \right] - \Pr \left[ \mathcal{A}(D, T_1) = 1 \right] \right|$$

where

$$\left( N, G, G_T, e, p_1, \ldots, p_5 \right) \leftarrow \text{GrpGen}(1^\lambda); \quad \mathcal{G} = (N, G, G_T, e);$$

$$g_1 \leftarrow G_{p_1}; \quad g_4 \leftarrow G_{p_4}; \quad g_5 \leftarrow G_{p_5}; \quad X_1 X_2 X_3 \leftarrow G_{p_1 p_5};$$

$$D \leftarrow (g_1, g_4, g_5, X_1 X_2 X_3);$$

$$T_0 \leftarrow G_{p_1}; \quad T_1 \leftarrow G_{p_4}.$$

**Assumption 2** For any probabilistic polynomial time adversary $\mathcal{A}$, the advantage function defined as follows are negligible in $\lambda$.

$$\text{Adv}_{\mathcal{A}}^{\text{ID2}}(\lambda) := \left| \Pr \left[ \mathcal{A}(D, T_0) = 1 \right] - \Pr \left[ \mathcal{A}(D, T_1) = 1 \right] \right|$$

where

$$\left( N, G, G_T, e, p_1, \ldots, p_5 \right) \leftarrow \text{GrpGen}(1^\lambda); \quad \mathcal{G} = (N, G, G_T, e);$$

$$g_1 \leftarrow G_{p_1}; \quad g_4 \leftarrow G_{p_4}; \quad g_5 \leftarrow G_{p_5}; \quad X_1 X_2 \leftarrow G_{p_1 p_5}; \quad Y_2 Y_3 \leftarrow G_{p_1 p_5};$$

$$D \leftarrow (g_1, g_4, g_5, X_1 X_2, Y_2 Y_3);$$

$$T_0 \leftarrow G_{p_1 p_5}; \quad T_1 \leftarrow G_{p_4}.$$

**Assumption 3** For any probabilistic polynomial time adversary $\mathcal{A}$, the advantage function defined as follows are negligible in $\lambda$.

$$\text{Adv}_{\mathcal{A}}^{\text{ID3}}(\lambda) := \left| \Pr \left[ \mathcal{A}(D, T_0) = 1 \right] - \Pr \left[ \mathcal{A}(D, T_1) = 1 \right] \right|$$

where

$$\left( N, G, G_T, e, p_1, \ldots, p_5 \right) \leftarrow \text{GrpGen}(1^\lambda); \quad \mathcal{G} = (N, G, G_T, e);$$

$$g_1 \leftarrow G_{p_1}; \quad g_2 \leftarrow G_{p_2}; \quad g_3 \leftarrow G_{p_3}; \quad g_4 \leftarrow G_{p_4}; \quad g_5 \leftarrow G_{p_5};$$

$$\bar{X}_4, \bar{Y}_4, \bar{X}_4, \bar{Y}_4 \leftarrow G_{p_1}; \quad x, y, z \leftarrow \mathbb{Z}_N;$$

$$D = (g_1, g_2, g_3, g_4, g_5, \bar{X}_4, \bar{Y}_4, g_3 \bar{X}_4, g_3 \bar{Y}_4);$$

$$T_0 \leftarrow (g_2^{xy}, g_3^{xy}); \quad T_1 \leftarrow (g_2^{xy+}, g_3^{xy+}).$$
Assumption 4 For any probabilistic polynomial time adversary $\mathcal{A}$, the advantage function defined as follows are negligible in $\lambda$.

$$\text{Adv}^\text{DDH2}_\mathcal{A}(\lambda) := \left| \Pr [\mathcal{A}(D, T_0) = 1] - \Pr [\mathcal{A}(D, T_1) = 1] \right|$$

where

$$(N, G, G_T, e, p_1, \ldots, p_5) \leftarrow \text{GrpGen}(1^\lambda);\; \mathcal{G} = (N, G, G_T, e);$$

$g_1 \leftarrow G_{p_1};\; g_2 \leftarrow G_{p_2};\; g_3 \leftarrow G_{p_3};\; g_4 \leftarrow G_{p_4};\; g_5 \leftarrow G_{p_5};$

$$\bar{X}_5, \bar{Y}_5 \leftarrow G_{p_5};\; x, y, z \leftarrow \mathbb{Z}_p^*;$$

$$D = (\mathcal{G}, g_1, g_2, g_3, g_4, g_5, g_2^x \bar{X}_5, g_2^y \bar{Y}_5);$$

$$T_0 \leftarrow g_2^{xy};\; T_1 \leftarrow g_2^{x+y+z}.$$

Assumption 5 For any probabilistic polynomial time adversary $\mathcal{A}$, the advantage function defined as follows are negligible in $\lambda$.

$$\text{Adv}^\text{DD3}_\mathcal{A}(\lambda) := \left| \Pr [\mathcal{A}(D, T_0) = 1] - \Pr [\mathcal{A}(D, T_1) = 1] \right|$$

where

$$(N, G, G_T, e, p_1, \ldots, p_5) \leftarrow \text{GrpGen}(1^\lambda);\; \mathcal{G} = (N, G, G_T, e);$$

$g_2 \leftarrow G_{p_1};\; g_3 \leftarrow G_{p_1};\; g_4 \leftarrow G_{p_1};\; g_5 \leftarrow G_{p_1};$

$$X_1X_5 \leftarrow G_{p_1p_2};\; Y_1Y_5 \leftarrow G_{p_1p_2};$$

$$D = (\mathcal{G}, g_2, g_3, g_4, g_3, X_1X_5, Y_1Y_5);$$

$$T_0 \leftarrow G_{p_1p_2};\; T_1 \leftarrow G_{p_1p_2p_3}.$$

Assumption 6 For any probabilistic polynomial time adversary $\mathcal{A}$, the advantage function defined as follows are negligible in $\lambda$.

$$\text{Adv}^\text{DD4}_\mathcal{A}(\lambda) := \left| \Pr [\mathcal{A}(D, T_0) = 1] - \Pr [\mathcal{A}(D, T_1) = 1] \right|$$

where

$$(N, G, G_T, e, p_1, \ldots, p_5) \leftarrow \text{GrpGen}(1^\lambda);\; \mathcal{G} = (N, G, G_T, e);$$

$g_1 \leftarrow G_{p_1};\; g_2 \leftarrow G_{p_1};\; g_5 \leftarrow G_{p_1};\; X_2X_3X_4 \leftarrow G_{p_1p_2p_3};$

$$D = (\mathcal{G}, g_1, g_2, g_5, X_2X_3X_4);$$

$$T_0 \leftarrow G_{p_1p_2p_3};\; T_1 \leftarrow G.$$

To realize the One-sided Publicly Openability, we need $G$ to be equipped with publicly reversible sampling [BWY11, FHKW10, LDL+14]. Formally, there exist two algorithms defined as follows.

- Sample_{G}() \rightarrow G. The publicly reversible sampler takes no input and outputs a random element from group $G$ with probability $1 - \zeta$ or outputs a failure symbol $\bot$ with probability $\zeta$. In particular, for all $G' \in G$, we require that

$$\Pr [G = G' | G \neq \bot] = 1/|G|,$$

where the probability space is defined by random coins consumed by Sample_{G}.

- Sample_{G}^{-1}(G) \rightarrow r. The public re-sampler takes an element $G \in G$ as input and outputs random coins $r$ with probability $1 - \theta$ such that Sample_{G}(r) = G or outputs a failure symbol $\bot$ with probability $\theta$. In particular, we require that, for all $r' \in R_G$ where $R_G := \{ \bar{r} : \text{Sample}_{G}(\bar{r}) = G \}$,

$$\Pr [r = r' | r \neq \bot] = 1/|R_G|,$$

where the probability space is defined by random coins consumed by Sample_{G}^{-1}.

Bellare, Waters, and Yilek [BWY11] had realized these two algorithms for bilinear group $G$ with $\zeta \approx 1/2^\rho$ and $\theta \approx 1/2^\rho$ where $\rho$ is an independent parameter, based on the technique for hashing into elliptic curve groups [BLS01]. We will invoke Sample_{G} and Sample_{G}^{-1} as black box, their detailed specifications can be found in [BWY11].
4.2 Construction

Assuming a group generator GrpGen described in previous subsection, our main construction is shown in Figure 7, an IBE scheme with identity space IdSp = \{0, 1\}^n and message space MsgSp = \{0, 1\} which is also equipped with algorithm OpenToOne for achieving One-Sided Publicly Openability.

The correctness of our construction is obvious. Fix an \(id \in \text{IdSp}\). For the case \((C, C') \in [\text{Enc}(\text{mpk}, \text{id}, 0)]\), we have

\[
e(C, K) = e \left( \prod_{i=1}^{n} \text{W}[2i - \text{id}[i]] \right)^{s} g_{5}^{r}, \tilde{G}^{r_{4}}
\]

\[
= e \left( g_{1}^{(u+\sum_{i=1}^{n} \text{W}[2i - \text{id}[i]])} \right)^{s} g_{1}^{r} = e \left( g_{1}^{(u+\sum_{i=1}^{n} \text{W}[2i - \text{id}[i]])} \right)^{r}
\]

\[
= e \left( g_{1}^{r_{4}}, \tilde{U} \right) \left( \prod_{i=1}^{n} \text{W}[2i - \text{id}[i]] \right)^{r} g_{4}^{r_{4}} = e(C', K'),
\]

which means \(\text{Pr}[\text{Dec}(\text{mpk}, \text{sk}, \text{Enc}(\text{mpk}, \text{id}, 0)) = 0] = 1\). For the case \((C, C') \in [\text{Enc}(\text{mpk}, \text{id}, 1)]\), we let \(C = g^{c}\) and \(C' = g^{c'}\) where \(c, c' \in \mathbb{Z}_{N}\), and we have, with the probability space defined by sampling \(r, r_{4}, c, c' \leftarrow \mathbb{Z}_{N}\),

\[
\text{Pr}[\text{Dec}(\text{mpk}, \text{sk}, \text{Enc}(\text{mpk}, \text{id}, 1)) = 0] = \text{Pr}[e(C, K) = e(C', K')]
\]

\[
= \text{Pr}[e(g^{c}, \tilde{G}^{r_{4}}) = e(g^{c'}, (\prod_{i=1}^{n} \text{W}[2i - \text{id}[i]])^{r} g_{4}^{r_{4}})]
\]

\[
= \text{Pr}[e((g_{1}^{2}g_{3}g_{4})^{c'}, (g_{1}^{2}g_{3})^{c} g_{4}^{r_{4}}) = e((g_{1}^{2}g_{3}g_{4})^{c'}, (g_{1}^{2}g_{3}^{c})^{(u+\sum_{i=1}^{n} \text{W}[2i - \text{id}[i]])} r g_{4}^{r_{4}})]
\]

\[
\leq 64/2^{2^k}
\]

Therefore we conclude that \(\text{Pr}[\text{Dec}(\text{mpk}, \text{sk}, \text{Enc}(\text{mpk}, \text{id}, 1)) = 1] \geq 1 - 64/2^{2^k}\).

As [BWY11], algorithm OpenToOne indeed satisfies our requirement defined in Section 2 by the property of algorithm Sample_{G} and Sample_{G^{-1}} shown in previous subsection. We remark that algorithm OpenToOne runs Sample_{G^{-1}} twice independently and thus has failure probability \(\delta \leq 2\theta\).

4.3 Security Analysis

We prove the following theorem in this subsection.
Theorem 2 (Main Theorem) For any probabilistic polynomial time adversary $\mathcal{A}$ making $Q_K$ key extraction queries and $Q_C$ challenge queries, there exist adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, $\mathcal{B}_4$, $\mathcal{B}_5$ and $\mathcal{B}_6$ such that
\[
\text{Adv}_{\mathcal{A}}^\text{INDmCPA}(\lambda) \leq \text{Adv}_{\mathcal{B}_1}^\text{SD1}(\lambda) + 2n \cdot \text{Adv}_{\mathcal{B}_2}^\text{SD2}(\lambda) + (n + 1) \cdot \text{Adv}_{\mathcal{B}_3}^\text{DDH1}(\lambda)
+ \text{Adv}_{\mathcal{B}_4}^\text{DDH2}(\lambda) + \text{Adv}_{\mathcal{B}_5}^\text{SD3}(\lambda) + \text{Adv}_{\mathcal{B}_6}^\text{SD4}(\lambda) + 2\zeta.
\]
and $\max_{i\in[6]} \text{Time}(\mathcal{B}_i) \approx \text{Time}(\mathcal{A}) + (Q_C + Q_K) \cdot \text{poly}(\lambda, n)$ where poly is independent of $\mathcal{A}$.

4.3.1 Organization of Proof

Before we proceed, we define two functions for brevity:
\[
h(u, w, i\overline{d}) := u + \sum_{i=1}^{n} w[2i \cdot i\overline{d}[i]] \quad \text{and} \quad H(U, W, i\overline{d}) := U \cdot \prod_{i=1}^{n} W[2i \cdot i\overline{d}[i]]
\]
where $i\overline{d} \in \{0, 1\}^n$, $u \in \mathbb{Z}_N$, $w \in \mathbb{Z}_{N^2}$, $U \in \mathcal{G}$, and $W \in \mathcal{G}^2$. When we set $U = g^u$ and $W = g^w$ for some $g \in \mathcal{G}$, we immediately have the following relation
\[
H(U, W, i\overline{d}) = g^{h(u, w, i\overline{d})}.
\]
We also need a family of random functions $\{R_i(\cdot)\}_{i\in[0:U(a)]}$ defined as follows:
\[
R_i(i\overline{d}) : i\overline{d}[i] \rightarrow \mathbb{Z}_N, \forall i \in [0] \cup [n],
\]
where $i\overline{d}[i]$ denotes the $i$-bit prefix of $i\overline{d}$. We note that the Chinese Reminder Theorem implies that $R_i(i\overline{d})$ mod $p_2$ and $R_i(i\overline{d})$ mod $p_3$ are distributed independently, which correspond to $\bar{R}_i$ and $\tilde{R}_i$, respectively, in [HKS15]. Although the range of random functions is $\mathbb{Z}_N$, the actual working range is $R_i(i\overline{d})$ mod $p_2$ and $R_i(i\overline{d})$ mod $p_3$ corresponding to semi-functional space $\mathcal{G}_{p_2}$ and $\mathcal{G}_{p_3}$, respectively.

The proof follows hybrid arguments using a series of games, which can be roughly divided into four phases. We describe these games in a phase-by-phase fashion in Figure 8, Figure 9, Figure 10, Figure 11, respectively. In particular, we state that

- In phase 1, we introduce semi-functional components (elements in $\mathcal{G}_{p_2}$) into ciphertexts and introduce random function $R_0$ into the system.
- In phase 2, we increase the entropy of random function we have introduced following the method of Hofheinz et al. [HKS15]. In particular, by executing phase 2 for $n$ times, we replace the initial random function $R_0$ with $R_n$, whose output depends on all bits of $i\overline{d}$.
- In phase 3, we handle the multiple occurrences of a single identity in challenge ciphertexts as well as in secret keys. We finally argue that multiple ciphertexts for a single identity are computationally independent in semi-functional space and so do secret keys. The proof will also follows Hofheinz et al.’s idea for full security [HKS15].
- In phase 4, we show that all ciphertexts for $m = 0$ are computationally indistinguishable from those for $m = 1$, which is truly random following the method used in [DCIP10, BWY11].

The remaining of the section includes four parts corresponding to four phases in order. Each phase begins with games involved and then proves a series of lemmas showing computational indistinguishability of these games. Putting them together, we immediately obtain the main theorem. In the proof, we define the advantage function of adversary $\mathcal{A}$ in Game$^{\text{e,x,x}}_{x,x,x}$ as
\[
\text{Adv}_{\mathcal{A}}^{\text{e,x,x}}(\lambda) := \left| \text{Pr}[\text{Game}_{x,x,x,\mathcal{A}}(\lambda) = 1] - 1/2 \right|.
\]

4.3.2 Phase 1: Prelude

All games used in Phase 1 is defined in Figure 8.

Lemma 5 (Game$_0 \approx$ Game$_1$) For any probabilistic polynomial time adversary $\mathcal{A}$ making $Q_K$ key extraction queries and $Q_C$ challenge queries, there exists an adversary $\mathcal{B}$ such that
\[
|\text{Adv}_{\mathcal{A}}(\lambda) - \text{Adv}_{\mathcal{B}}^{\text{SD1}}(\lambda)| \leq \text{Adv}_{\mathcal{B}}^{\text{SD1}}(\lambda)
\]
and $\text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + (Q_C + Q_K) \cdot \text{poly}(\lambda, n)$ where poly is independent of $\mathcal{A}$.
Initialize
\( \mathcal{G} := (N, G, G_{\lambda}, e) \leftarrow \text{GpGen}(1^k) \)
for \( i \in [5] \) do \( g_i \leftarrow G_{p_i} \)
\( v_5 \leftarrow Z_N \)
\( G := g_1 g_5^v \), \( \overline{G} := g_1 g_2 g_3 \)
\( u, u_5 \leftarrow Z_N \), \( w, w_5 \leftarrow Z_{2^n} \)
\( U := g_1^u g_5^{w_5} \), \( \overline{U} := (g_1 g_2)^u g_5^{w_5} \)
\( U := (g_1 g_2)^u g_5^{w_5} \), \( W := g_1^w g_5^{w_5} \)
\( \beta \leftarrow \{0, 1\} \)
return \( (\mathcal{G}, G, U, W, g_5) \)

Extract(id)
if id \in ChID then return \( \bot \)
ExID := ExID \cup \{id\}
\( r, r_4, r_4' \leftarrow Z_N \)
\( K := G^{r_4} g_4 \)
\( K' := H(\overline{U}, \overline{W}, \text{ID})^{r_4} g_4' \)
\( K' := (g_2 g_3)^{R_{\beta}(m) r} \cdot H(\overline{U}, \overline{W}, \text{ID})^{r_4} g_4' \)
return \( (K, K') \)

Challenge(id^*, m_0^*, m_1^*)
if id^* \in ExID then return \( \bot \)
ChID := ChID \cup \{id^*\}
if \( m_0^* = 0 \) then
\( s, s_5, s_5' \leftarrow Z_N \)
\( C := H(U, W, \text{ID})^{s_5} g_3^{s_5} \)
\( C := g_2^{R_{\beta}(m_1)} \cdot H(\overline{U}, \overline{W}, \text{ID})^{s_5} g_3^{s_5} \)
else
\( C, C' \leftarrow \text{Sample}_G() \)
return \( (C, C') \)

Finalize(\( \beta^* \))
return \( (\beta = \beta^*) \)

Figure 8: Game_0, Game_1, Game_2.0

Proof. Given \( (\mathcal{G}, g_1, g_4, g_5, X_1 X_2 X_3, T) \) where \( T \) is a random element of either \( G_{p_i} \) or \( G_{p_{12}} \), adversary \( \mathcal{A} \) simulates the procedures as follows. We assume \( g_2 \leftarrow G_{p_2}, g_3 \leftarrow G_{p_2} \) and implicitly parse \( X_1 X_2 X_3 = (g_1 g_2 g_3) \) for some \( x \in Z_N \), and either \( T = g_1^t \) or \( T = g_1 g_2 g_3 \) from some \( t \in Z_N \).

Initialize Sample \( v_5 \leftarrow Z_N \) and set \( G := g_1 g_5^{v_5} \). We implicitly define \( \overline{G} := (g_1 g_2) \cdot g_5^{v_5} \) and \( \overline{G} := g_1 g_2 g_3 \). Sample \( u, u_5 \leftarrow Z_N \), \( w, w_5 \leftarrow Z_{2^n} \) and set \( U := g_1^u g_5^{w_5} \), \( W := g_1^w g_5^{w_5} \). We implicitly define \( \overline{U} := (g_1 g_2)^u g_5^{w_5} \), \( \overline{W} := (g_1 g_2)^w g_5^{w_5} \), \( \overline{U} := g_1 g_2 g_3 \), \( \overline{W} := (g_1 g_2 g_3) \) . Randomly pick \( \beta \leftarrow \{0, 1\} \) and output \( \text{MPK} := (\mathcal{G}, G, U, W, g_5) \).

Extract(id) Return \( \bot \) when id \in ChID. Update \( \text{ExID} := \text{ExID} \cup \{id\} \). Sample \( r, r_4, r_4' \leftarrow Z_N \) and set
\( K := (X_1 X_2 X_3)^r \cdot g_4^{s_5} \) and \( K' := (X_1 X_2 X_3)^{(r, w, \text{ID})} g_4^{s_5} \).
Output \( (K, K') \). Here we implicitly set \( r := x r' \in Z_N \).

Challenge(id^*, m_0^*, m_1^*) Return \( \bot \) when id^* \in ExID. Update \( \text{ChID} := \text{ChID} \cup \{id^*\} \). If \( m_0^* = 0 \), sample \( s, s_5, s_5' \leftarrow Z_N \) and set
\( C := t^{(i, w, \text{ID})} g_5^{s_5} \) and \( C' := t^{s_5} g_5^{s_5} \).
If \( m_0^* = 1 \), sample \( C, C' \leftarrow \text{Sample}_G() \). Output \( (C, C') \). Here we implicitly set \( s \in Z_N \) such that \( s = s_5 \) mod \( p_1 p_2 \).

Finalize(\( \beta^* \)) Output \( (\beta = \beta^*) \).

The algorithm \( \mathcal{A} \) outputs 1 if and only if \( \text{Finalize}(\beta^*) = 1 \), i.e., the adversary \( \mathcal{A} \) wins the game.

Observe that if \( T = g_1^t \in G_{p_1} \), the simulation described above is identical to Game_0, if \( T = (g_1 g_2)^t \in G_{p_{12}} \), the simulation is identical to Game_1. Therefore we can conclude that \( \text{Adv}_{\mathcal{A}}(\lambda) = \text{Adv}_{\mathcal{A}}(\lambda) \leq \text{Adv}_{\mathcal{A}}(\lambda) \).

Lemma 6 (Game_1 = Game_2.0) For any adversary \( \mathcal{A} \), we have \( \text{Adv}_{\mathcal{A}}(\lambda) = \text{Adv}_{\mathcal{A}}(\lambda) \).

Proof. Since random function \( R_0(\cdot) \) is consistent on all possible identities in IdSp, \( R_0(id) \) for all id \in IdSp is a single random variable independently distributed over \( Z_N \). If we sample \( U' \leftarrow Z_N \) and set, in Initialize
\( U := g_1 u g_5^{w_5} \) and \( \overline{U} := (g_1 g_2)^u g_5^{w_5} \) and \( U := (g_1 g_2 g_3) u (g_2 g_3) \),

15
the resulting game remains unchanged, since we in fact implicitly define
\[ u = u' \mod p_1 \] and \[ u = u' + R_0 \mod p_2 p_3, \]
which is still distributed over \( \mathbb{Z}_N \) as we required in Game_1. Observe that the simulation is also identical to Game_2, with \( u = u' \) except the definition of \( \bar{U} \) and \( \bar{U} \) in Initialize procedure. However we note that the difference is just conceptual as both \( U \) and \( \bar{U} \) are not given to adversary in \( \text{mpk} \). Therefore we can conclude that Game_1 and Game_2 are statistically identical.

4.3.3 Phase 2: From \( R_0 \) to \( R_n \)

All games used in Phase 2 is defined in Figure 9.

```
Initialize
\( \varphi := (N,G,G_T,e) \leftarrow \text{GrpGen}(1^\lambda, 5) \)
for \( i \in [5] \) do \( g_i \leftarrow G_{p_i} \)
\( v_5 \leftarrow \mathbb{Z}_N; G := g_1^v_5 \)
\( \bar{G} := (g_1 g_2)^{v_5}; \quad \bar{G} := (g_1 g_3)^{v_5} \)
\( \bar{G} := g_1 g_3 g_5 \)
\( u, u_5 \leftarrow \mathbb{Z}_N; w, w_5 \leftarrow \mathbb{Z}_N \)
\( U := g_1^{u_5} w : W := g_5^{u_5} \)
\( \bar{U} := (g_1 g_3)^{u_5} w ; \quad \bar{W} := (g_1 g_3)^{w_5} \)
\( U := (g_1 g_3)^{u_5} w ; \quad W := (g_1 g_3)^{w_5} \)
\( \beta \leftarrow \{0, 1\} \)
return (\( \varphi, G, U, W, g_5 \))

Extract(ID)
if \( ID \in \text{ChID} \) then return \( \perp \)
\( \text{ChID} := \text{ExID} \cup \{ID\} \)
\( r, r_4, r_5 \leftarrow \mathbb{Z}_N \)
\( K := \bar{G} g_4^{r_5} \)
\( K' := (g_2 g_3)^{R_ID} \cdot H(\bar{U}, W, ID)^g_5 \)
\( \bar{K}' := (g_2 g_3)^{R_ID} \cdot H(\bar{U}, W, ID)^g_5 \)
return (\( K, K' \))

Finalize(\( \beta' \))
return (\( \beta' = \beta \))
```

Figure 9: Game_{2i} (\( i \in [0, n] \)), Game_{2i,1} (\( i \in [0, n-1] \)), Game_{2i,2} (\( i \in [0, n-1] \))

**Lemma 7 (Game_{2i} \approx Game_{2i,1})** For any probabilistic polynomial time adversary \( \mathcal{A} \) making \( Q_K \) key extraction queries and \( Q_C \) challenge queries, there exists an adversary \( \mathcal{B} \) such that
\[
\left| \text{Adv}_{2i,1}(\lambda) - \text{Adv}_{2i,1}(\lambda) \right| \leq \text{Adv}_{\mathcal{B}}^{\text{SD2}}(\lambda)
\]
and \( \text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + (Q_C + Q_K) \cdot \text{poly}(\lambda, n) \) where \( \text{poly} \) is independent of \( \mathcal{A} \).

**Proof.** Given (\( \varphi, g_1, g_3, g_5, X_2X_3, Y_2Y_3, T \)) where \( T \) is a random element of either \( G_{p_3} \) or \( G_{p_3} \), adversary \( \mathcal{B} \) simulates the procedures as follows. We assume \( g_2 \leftarrow G_{p_3} \) and \( g_3 \leftarrow G_{p_3} \) and implicitly parse \( X_2X_3 = (g_2 g_3) \) for some \( x \in \mathbb{Z}_N, Y_2Y_3 = (g_2 g_3) \) for some \( y \in \mathbb{Z}_N \) and either \( T = (g_2 g_3) \) or \( T = (g_1 g_3) \) for some \( t \in \mathbb{Z}_N \).

```
Initialize Sample \( v_5 \leftarrow \mathbb{Z}_N \) and set \( \bar{G} := g_1^v_5 \). We implicitly define \( \bar{G} := (g_1 g_2)^{v_5} \), \( \bar{G} := g_1 g_3 g_5 \). Sample \( u, u_5 \leftarrow \mathbb{Z}_N \) and \( w, w_5 \leftarrow \mathbb{Z}_N \), and set \( U := g_1^u w \) and \( W := g_5^u \). We implicitly define \( \bar{U} := (g_1 g_3)^u w ; \bar{W} := (g_1 g_3)^u w \). Randomly pick \( \beta \leftarrow \{0, 1\} \) and output \( \text{mpk} := (\varphi, G, U, W, g_5) \). \( \mathcal{B} \) also maintains a random function \( R_i(\cdot) \) in an on-the-fly manner.
```

16
The algorithm queries and Q

Proof. Challenge the procedures as follows.

\[ G_1^{h(u, w, \beta)} \cdot (X_2 X_5) \] and \[ G'_1 \cdot (X_2 X_5) \] s

\[ T \cdot (R(i, w, \beta)) \cdot s \cdot g \] 2<sup>u</sup>

Here we implicitly set \( s \in \mathbb{Z}_N \) such that \( s = s' \mod p_1 \) and \( s = xs'' \mod p_2 \).

Finally, if \( \beta_1 = 1 \), sample \( C', C'' \leftarrow \text{Sample}_2() \). Output \( (C, C') \).

Finalize(β'). Output \( (β = β') \). The algorithm \( \mathcal{B} \) outputs 1 if and only if Finalize(β') = 1, i.e., the adversary \( \mathcal{A} \) wins the game.

Observe that if \( T = (g_2 g_3) \) \( \in \mathbb{Z}_{p_1} \), the simulation described above is identical to Game\(_{2,1}\); if \( T = (g_3 g_5) \), the simulation is identical to Game\(_{2,1,2}\). Therefore we can conclude that \( |Adv_{\mathcal{A}}(\lambda) - Adv_{\mathcal{A}}^{2,1}(\lambda)| \leq Adv_{\mathcal{A}}^{2,1,1}(\lambda) \).

Lemma 8 (Game\(_{2,1,1} \approx \) Game\(_{2,1,2}\)). For any probabilistic polynomial time adversary \( \mathcal{A} \) making \( Q_K \) key extraction queries and \( Q_C \) challenge queries, there exists an adversary \( \mathcal{B} \) such that

\[ |Adv_{\mathcal{A}}^{2,1,1}(\lambda) - Adv_{\mathcal{A}}^{2,1,2}(\lambda)| \leq Adv_{\mathcal{A}}^{2,1,1}(\lambda) \]

and \( \text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + (Q_C + Q_K) \cdot \text{poly}(\lambda, n) \) where \( \text{poly} \) is independent of \( \mathcal{A} \).

Proof. Given \( (g, g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, \mathcal{U}, T) \) where \( T \) is either \( (g_2 g_3, g_3 g_4) \) or \( (g_2 g_3, g_3 g_4, g_3 g_5) \), adversary \( \mathcal{B} \) generates \( Q_K \) tuples (using Many Tuple Lemma [CW13])

where \( \tilde{x}_{4,i} \), \( \tilde{X}_{4,i} \) \( \in \mathbb{Z}_{p_1} \) and \( T_j \) is either \( (g_2, g_2 g_3, g_3 g_4) \) or \( (g_2 g_3, g_3 g_4, g_3 g_5) \) for \( x_j, z_j \in \mathbb{Z}_N \), and then simulates the procedures as follows.

Initialize. Sample \( v_5 \in \mathbb{Z}_N \) and set \( G := (g_2 g_3)^{g_5} \), \( \bar{G} := (g_2 g_3) g_5 \), \( \bar{G} := (g_2 g_3) g_5 \), and \( \bar{G} := (g_1 g_3 g_5) \). Sample \( u, u_5 \in \mathbb{Z}_N \) and set \( U := (g_2 g_3)^{g_5} \), \( \bar{U} := (g_1 g_3) g_5 \) and \( \bar{U} := (g_1 g_3 g_5) \). Sample \( w, w_5 \in \mathbb{Z}_{p_1} \) and set \( W := w g_5 \). We implicitly define

\[ \tilde{W} := (g_2 g_3)^{w} \bar{g}_2^{x_{2(i+1)-1}} \bar{w}_2 \] and \( \tilde{W} := (g_2 g_3)^{w} \bar{g}_2^{x_{2(i+1)-1}} \bar{w}_2 \) and \( \tilde{W} := (g_2 g_3)^{w} \bar{g}_2^{x_{2(i+1)-1}} \bar{g}_2^{x_{2(i+1)-1}} \). Randomly pick \( \beta \leftarrow \{0, 1\} \) and output \( \text{mpk} := (g, G, U, W, g_5) \). Note that we implicitly set \( w \in \mathbb{Z}_{p_1} \) such that

\[ w = w \mod p_1 \] and \( w = w + ye_{2(i+1)} \mod p_2 \) and \( w = w + ye_{2(i+1)-1} \mod p_3 \), and note that neither \( \tilde{W}[2(i+1)-1] \) nor \( \tilde{W}[2(i+1)-1] \) is known to \( \mathcal{B} \). Besides that, \( \mathcal{B} \) also maintains a random function \( r_i(\cdot) \) in an on-the-fly manner.

Extract(α). Return \( \bot \) when \( \alpha \in \text{ChID} \). Update ExID := ExID \cup \{\alpha\}. If \( R(\text{ExID}) \) has not been used before, sample \( r', r'_4, r''_4 \in \mathbb{Z}_N \) and set

\[ K := (g_1 g_2 g_3)^{r'} (g_2 g_3)^{r''_4}, g_4^{r''_4} \] and \( K' := g_1^{R(\alpha)} g_4^{r''_4} \) \( \cdot T_j[2] \cdot g_4^{r''_4} \), if \( \alpha[i + 1] = 0 \)

\[ \begin{cases} g_1^{R(\alpha)} g_4^{r''_4} \cdot T_j[1] \cdot g_4^{r''_4} \text{, if } \alpha[i + 1] = 1 \end{cases} \]
where \( T_r[1] \) and \( T_r[2] \) refers to the first and second entry of \( T_r \), respectively. Here we implicitly set \( r \in \mathbb{Z}_N \) such that \( r = r' \mod p_1 \) and \( r = r' + x_j \mod p_2p_3 \). Output \((K', K'')\). On the other hand, if \( R_i(\text{id}) \) has been touched, we find the index \( j \) and random coins \( r', r'' \) used at the first time \( R_i(\text{id}) \) was met, and create \((K', K'')\) using the old index and these random coins but with the new \( \text{id} \) following the above method. Then we sample \( r'' \leftarrow \mathbb{Z}_N \) and output \((K'', K''')\) as reply to the query. In this case, we implicitly set \( r \) such that \( r = r' \mod p_1 \) and \( r = (r' + x_j) \mod p_2p_3 \).

**Challenge**(\( \text{id}^*, m'_0, m'_1 \)) Return \( \bot \) when \( \text{id}^* \in \text{ExID} \). Update \( \text{ChID} := \text{ChID} \cup \{\text{id}^*\} \). If \( m'_0 = 0 \) and \( \text{id}^*[i+1] = 0 \), sample \( s, s_5, s'_5 \leftarrow \mathbb{Z}_N \) and set 
\[
C := g_2^{R(\text{id}^*)} \cdot H(U, \text{id}^*) \cdot g_5^{s_5} \quad \text{and} \quad C' := g_2^{s'_5} \cdot g_5^{s'_5}.
\]

Note that we do not need \( \text{W}[2(i+1) - 1]\) which is unknown. If \( m'_0 = 0 \) and \( \text{id}^*[i+1] = 1 \), sample \( s, s_5, s'_5 \leftarrow \mathbb{Z}_N \) and set 
\[
C := g_2^{R(\text{id}^*)} \cdot H(U, \text{id}^*) \cdot g_5^{s_5} \quad \text{and} \quad C' := g_2^{s'_5} \cdot g_5^{s'_5}.
\]

Note that we do not need \( \text{W}[2(i+1) - 0]\) which is unknown. In a word, even though \( \mathcal{B} \) does not know all elements in \( \text{W} \) and \( \text{W} \), it still can compute ciphertext for \( m'_0 = 0 \) as usual. Finally, if \( m'_0 = 1 \), sample \( C, C' \leftarrow \text{Sample}_C(\cdot) \). Output \((C, C')\).

**Finalize**(\( \beta' \)). Output \((\beta = \beta'\)).

The algorithm \( \mathcal{B} \) outputs 1 if and only if **Finalize**(\( \beta' \)) = 1, i.e., the adversary \( \mathcal{A} \) wins.

Clearly, if \( T_j = (g_2^{x_j}, g_3^{y_j}) \) for all \( j \in [q] \), the simulation described above is identical to Game\(_{2,i,1}\). On the other hand, if \( T_j = (g_2^{x_{j+y}}, g_3^{x_{j+y+z}}) \) for all \( j \in [q] \), we implicitly set 
\[
R_{i+1}(\text{id}) := \begin{cases} 
R_i(\text{id}) \mod p_2, & \text{if } \text{id}[i+1] = 0 \\
R_i(\text{id}) + z_j/(r' + x_j) \mod p_2, & \text{if } \text{id}[i+1] = 1
\end{cases}
\]

and 
\[
R_{i+1}(\text{id}) := \begin{cases} 
R_i(\text{id}) \mod p_3, & \text{if } \text{id}[i+1] = 0 \\
R_i(\text{id}) + z_j/(r' + x_j) \mod p_3, & \text{if } \text{id}[i+1] = 1
\end{cases}
\]

where index \( j \in [q] \) and index \( r' \in \mathbb{Z}_N \) were selected at the first time \( R_i(\text{id}) \) was met in the simulation. Therefore the simulation in this case is identical to Game\(_{2,i,2}\). Therefore we can conclude that 
\[
\left| \text{Adv}_{\mathcal{A}}^{2,i,1}(\lambda) - \text{Adv}_{\mathcal{A}}^{2,i,2}(\lambda) \right| \leq \text{Adv}_{\mathcal{B}}^{\text{DOH}}(\lambda).
\]

**Lemma 9** (Game\(_{2,i,2} \approx \text{Game}_{2,(i+1)}\)) For any probabilistic polynomial time adversary \( \mathcal{A} \) making Q\(_k\) key extraction queries and Q\(_C\) challenge queries, there exists an adversary \( \mathcal{B} \) such that
\[
\left| \text{Adv}_{\mathcal{A}}^{2,i,2}(\lambda) - \text{Adv}_{\mathcal{A}}^{2,i,(i+1)}(\lambda) \right| \leq \text{Adv}_{\mathcal{B}}^{\text{DOH}}(\lambda)
\]

and 
\[
\text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + (Q_C + Q_K) \cdot \text{poly}(\lambda, n) \text{ where poly is independent of } \mathcal{A}.
\]

**Proof.** The proof is almost the same as the proof for Lemma 7. The only difference is to employ the high-entropy random function \( R_{i+1} \) instead of the low-entropy \( R_i \) in the simulation.

### 4.3.4 Phase 3: Handling multi-ciphertexts, multi-keys setting

All games used in Phase 3 is defined in Figure 10.

**Lemma 10** (Game\(_{2,n} \approx \text{Game}_{C}\)) For any probabilistic polynomial time adversary \( \mathcal{A} \) making Q\(_K\) key extraction queries and Q\(_C\) challenge queries, there exists an adversary \( \mathcal{B} \) such that
\[
\left| \text{Adv}_{\mathcal{A}}^{2,n}(\lambda) - \text{Adv}_{\mathcal{A}}^{2,n}(\lambda) \right| \leq \text{Adv}_{\mathcal{B}}^{\text{DOH}}(\lambda)
\]

and 
\[
\text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + (Q_C + Q_K) \cdot \text{poly}(\lambda, n) \text{ where poly is independent of } \mathcal{A}.
\]

**Proof.** Given \((g_2, g_3, g_4, g_5, g_2^{x_j}, g_2^{x_j}, T)\) where \( T \) is either \( g_2^{x_j} \) or \( g_2^{x_j+y} \), adversary \( \mathcal{B} \) generates Q\(_C\) tuples (using Many Tuple Lemma [CW13])
\[
(g_2^{x_j}, T_j), \quad \forall j \in [Q_C]
\]

where \( X_{5,i} \leftarrow \mathbb{G}_p \) and \( T_j \) is either \( g_2^{x_j} \) or \( g_2^{x_j+y} \) for \( x_j, z_j \leftarrow \mathbb{Z}_N \), and then simulates the procedures as follows.
\[ \text{Finalize}\]

Game_{2,n} \text{ and Game}_3

Game_{4,r} \text{ and Game}_5

\[ R_n^* : \text{IdSp} \to Z_N \]

\[ \text{Challenge}(1^{\ell^*}, m_0', m_1') \]

if 1^{\ell^*} \in \text{ExID} then return \bot

ChID := ChID \cup \{1^{\ell^*}\}

if m_P^* = 0 then

\[ s, s_3, s_4' \leftarrow Z_N \]

\[ s' \leftarrow Z_N; \quad s'' \leftarrow Z_N \]

\[ C := g_2^{s'} \cdot H(\hat{U}, \hat{W}, \text{id})^y g_5^z \]

\[ C' := g_2^{s'} \cdot G g_5^z \]

else

\[ C, C' \leftarrow \text{Sample}_G() \]

return \((C, C')\)

\[ \text{Finalize}(\beta') \]

return \((\beta = \beta')\)

Initialize Sample \(v_5 \leftarrow Z_N\) and set \(G := g_1 g_5^y\) and \(\bar{G} := g_1 g_2 g_3\). Sample \(u, u_5 \leftarrow Z_N\) and \(w, w_5 \leftarrow Z_N\), and set \(U := g_1^{u_5}, W := g_1^{w_5}\), \(\bar{U} := (g_1 g_2)^{u_5}, \bar{W} := (g_1 g_2)^{w_5}\), \(\bar{G} := g_1 g_2 g_3\). Besides that, \(\mathcal{B}\) also maintains a random function \(R_n(\cdot)\) in an on-the-fly manner.

Extract(\(m^*\)) Return \(\bot\) when \(m^* \in \text{ChID}\). Update \(\text{ExID} := \text{ExID} \cup \{m^*\}\). Sample \(r, r_4, r_4' \leftarrow Z_N\) and set

\[ K := \bar{G}^r g_4^{r_4} \quad \text{and} \quad K' := (g_2 g_3)^{R_n(\text{id}) \cdot r} \cdot H(\hat{U}, \hat{W}, \text{id})^y g_5^{r_4} \]

Output \((K, K')\).

Challenge(\(1^{\ell^*}, m_0', m_1') \)

Return \(\bot\) when \(1^{\ell^*} \in \text{ExID}\). Update \(\text{ChID} := \text{ChID} \cup \{1^{\ell^*}\}\). We maintain another independent random function \(R' : \text{IdSp} \to Z_N\) whose output depends on all bits of \(\text{id}\). If \(m_P^* = 0\), pick a new \(j \in [q]\), sample \(s', s_3', s_4'' \leftarrow Z_N\) and set

\[ C := g_1^{s'} h_j^{(u, w, \text{id})^{s'}} \cdot (g_2^{X_{s,j}})^{h_j^{(u, w, \text{id})^{s'}}} \cdot T_j^{R_n'(\text{id})} \cdot g_5^{s''} \quad \text{and} \quad C' := g_2^{s'} \cdot (g_2^{X_{s,j}})^{\cdot s_4''}. \]

Here we implicitly set \(s \leftarrow s' \mod p_1, s = x, \mod p_2\) and \(R_n'(\text{id}) = y \cdot R'(\text{id})^{\mod p_2}\) for \(1^{\ell^*} \in \text{ChID}\). We note that the assignment for \(R_n'\) is always consistent since the simulation ensures that \(\text{ExID} \cap \text{ChID} = \emptyset\). If \(m_P^* = 1\), sample \(C, C' \leftarrow \text{Sample}_G()\). Output \((C, C')\).

Finalize(\(\beta') \)

Output \((\beta = \beta')\).

The algorithm \(\mathcal{A}\) outputs 1 if and only if \(\text{Finalize}(\beta') = 1\), i.e., the adversary \(\mathcal{A}\) wins the game.

Observe that if \(T_j = g_2^{x_j}\) for all \(j \in [q]\), the simulation described above is identical to Game_{2,n}. On the other hand, if \(T_j = g_2^{x_j + z_j}\) for all \(j \in [q]\), the simulation is identical to Game_{3}. We implicitly set \(s' = R'(\text{id}) \cdot (x_j y + z_j)\). Therefore we can conclude that \(|\text{Adv}_{\mathcal{A}}^2(\lambda) - \text{Adv}_{\mathcal{A}}^3(\lambda)| \leq \text{Adv}_{\mathcal{A}}^4(\lambda)\).
Lemma 11 (Game₂ ≈ Game₃) For any probabilistic polynomial time adversary $\mathcal{A}$ making $Q_K$ key extraction queries and $Q_C$ challenge queries, there exists an adversary $\mathcal{B}$ such that

$$|Adv^{3}_{\mathcal{A}}(\lambda) - Adv^{\mathcal{B}}_{\mathcal{A}}(\lambda)| \leq Adv^{DDH}_{\mathcal{B}}(\lambda)$$

and $Time(\mathcal{B}) \approx Time(\mathcal{A}) + (Q_C + Q_K) \cdot poly(\lambda, n)$, where $poly$ is poly independent of $\mathcal{A}$.

**Proof.** Given $(V, g_1, g_2, g_3, g_4, x, y, z)$ where $T$ is either $(g_2^{x+y}, g_3^{x+y})$ or $(g_2^{x+y}, g_3^{x+y+1})$, adversary $\mathcal{B}$ generates a shared term and $Q_K$ tuples (using Many Tuple Lemma [GW13])

$$(g_2g_3)^v Y_4 \text{ and } \left((g_2g_3)^v X_{4,j}, T_j\right), \forall j \in [Q_K]$$

where $Y_4, X_4 \leftarrow \mathcal{G}_p$ and $T_j$ is either $(g_2g_3)^v$ or $(g_2g_3)^{x+y+1}$ for $x, y \leftarrow \mathcal{Z}_N$, and then simulates the procedures as follows.

**Initialize** Sample $v_5 \leftarrow \mathcal{Z}_N$ and set $G := g_1s_5^v \cdot G := g_1s_5^v$ and $\tilde{G} := g_1s_2g_3$. Sample $u, u_5 \leftarrow \mathcal{Z}_N$ and $w, w_5 \leftarrow \mathcal{Z}_N$, and set $U := s_1g_5^{u_5}, W := g_1s_5^{w_5}, U := g_1s_5^{u_5}, W := g_1s_5^{w_5}, \tilde{U} := g_1s_2g_3^u$ and $\tilde{W} := g_1s_2g_3^u$. Randomly pick $b \leftarrow \{0, 1\}$ and output $mpk := (V, G, U, W, \tilde{G})$.

**Extract**(b) Return 1 when $id \in \mathcal{Ch}_D$. Update $ExlD := ExlD \cup \{id\}$. We maintain a random function $R'$: $IdS_p \rightarrow \mathcal{Z}_N$ whose output depends on all bits of $id$. Pick a new $j \in [q]$, sample $r', r'_{d}, r'_{e} \leftarrow \mathcal{Z}_N$ and set

$$K := g_1^{r'} \cdot \left((g_2g_3)^v X_{4,j}\right) \cdot g_2^{r'_{d}} \text{ and } K := g_1^{r'} \cdot \left((g_2g_3)^v X_{4,j}\right) \cdot g_2^{r'_{d}} \cdot \left((g_2g_3)^v X_{4,j}\right) \cdot g_2^{r'_{d}}.$$

Here we implicitly set $r \leftarrow \mathcal{Z}_N$ such that $r = r' \mod p_1, r = x \mod p_3$ and define $R_{\lambda}(id) = y \cdot R'(id) \mod p_3$ for $id \in \mathcal{ExlD}$. We note that the assignment for $R_{\lambda}$ is consistent since $ExlD \cap ChID = \emptyset$.

**Challenge**(id, $m_{\lambda}^d, m_{\lambda}^e$) Return 1 when $id \in ExlD$. Update $ChID := ChID \cup \{id\}$. If $m_{\lambda}^d = 0$, sample $s, s', s_5, s_5' \leftarrow \mathcal{Z}_N$ and set

$$C := g_2^s \cdot H(\tilde{U}, \tilde{W}, id) \cdot g_5^{e_{exp}} \text{ and } C' := g_2^s \cdot g_5^{e_{exp}}.$$

If $m_{\lambda}^e = 1$, sample $C, C' \leftarrow Sample_{\lambda}(\cdot)$. Output $(C, C')$.

**Finalize** Output ($\beta = \beta'$).

The algorithm $\mathcal{B}$ outputs 1 if and only if $Finalize(\beta') = 1$, i.e., the adversary $\mathcal{A}$ wins the game.

Clearly, if $T_j = (g_2^{x+y}, g_3^{x+y})$ for all $j \in [q]$, the simulation described above is identical to Game₂. On the other hand, if $T_j = (g_2^{x+y+1}, g_3^{x+y+1})$ for all $j \in [q]$, the simulation is identical to Game₃ where we implicitly set $r' = R'(id) \cdot (x, y + z)$. Therefore we can conclude that $|Adv^{3}_{\mathcal{A}}(\lambda) - Adv^{\mathcal{B}}_{\mathcal{A}}(\lambda)| \leq Adv^{DDH}_{\mathcal{B}}(\lambda)$.

**Lemma 12 (Game₄ ≈ Game₅)** For any adversary $\mathcal{A}$, $Adv^{\mathcal{A}}_{\mathcal{A}}(\lambda) = Adv^{\mathcal{B}}_{\mathcal{A}}(\lambda)$.

**Proof.** The transformation from Game₄ to Game₅ is just conceptual following the Chinese Remainder Theorem. In both games, the $G_{p_1p_2}$-parts of $K$ and $K'$ are independent, and so do the $G_{p_1}$-part of $C$ and $C'$. Of course, the $G_{p_3}$-parts of them are still structural.

4.3.5 Phase 4: Epilogue

All games used in Phase 4 is defined in Figure 11.

**Lemma 13 (Game₅ ≈ Game₆)** For any probabilistic polynomial time adversary $\mathcal{A}$ making $Q_K$ key extraction queries and $Q_C$ challenge queries, there exists an adversary $\mathcal{B}$ such that

$$|Adv^{3}_{\mathcal{A}}(\lambda) - Adv^{\mathcal{B}}_{\mathcal{A}}(\lambda)| \leq Adv^{SDH}_{\mathcal{B}}(\lambda)$$

and $Time(\mathcal{B}) \approx Time(\mathcal{A}) + (Q_C + Q_K) \cdot poly(\lambda, n)$, where $poly$ is poly independent of $\mathcal{A}$.

**Proof.** Given $(V, g_2, g_3, g_4, g_5, x_1x_2y_2z_3, T)$ where $T$ is a random element of either $G_{p_1p_2}$ or $G_{p_1p_3p_4}$, adversary $\mathcal{B}$ simulates the procedures as follows. We implicitly set $g_1 = X_1$ and parse $X_1X_2g_1 = g_1s_5^v$ from some $x \in \mathcal{Z}_N$, $y_1y_2z_3 = (g_1s_2g_3)^{y_2}$ for some $y \leftarrow \mathcal{Z}_N$ and either $T = (g_2g_3)^t$ or $T = (g_1s_2g_3)^t$ for some $t \leftarrow \mathcal{Z}_N$. 

20
Given $T = (g_1 g_2 g_3)^t$ respectively. Therefore we can conclude that $1_g t$.

Observe that if $T = (g_1 g_2 g_3)^t$, the simulation is identical to Game$_5$; if $T = (g_1 g_2 g_3)^t$, the simulation is identical to Game$_6$ where the $G_{p_i}$-parts of $C$ and $C'$ are hidden by $g_1^{t'}$ and $g_1^{t''}$, respectively. Therefore we can conclude that $1_g t$.

**Lemma 14** (Game$_6$ ≈ Game$_7$) For any probabilistic polynomial time adversary $\mathcal{A}$ making $Q_k$ key extraction queries and $Q_C$ challenge queries, there exists an adversary $\mathcal{B}$ such that

$$|\text{Adv}_{\mathcal{A}}(\lambda) - \text{Adv}_{\mathcal{B}}(\lambda)| \leq \text{Adv}_{\mathcal{A}}^{\text{pol}}(\lambda)$$

and $\text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + (Q_C + Q_K) \cdot \text{poly}(\lambda, n)$ where poly is independent of $\mathcal{A}$.

**Proof.** Given $(\mathcal{G}, g_1, g_2, g_3, X_3 X_4, T)$ where $T$ is a random element of either $G_{p_i}$ or $G$, adversary $\mathcal{B}$ simulates the procedures as follows. We implicitly parse $X_3 X_4 = (g_2 g_3)^x$ for some $x \in \mathbb{Z}_p$ and either $T = (g_1 g_2 g_3)^t$ or $T = g_i^t$ for some $t \in \mathbb{Z}_N$.

---

*Figure 11: Game$_5$, Game$_6$, Game$_7$*
Final Analysis. In the last game

$$G := g_5 \cdot g_5$$ and $$\tilde{G} := g_1.$$ Sample $$u, u_5 \leftarrow \mathbb{Z}_N$$ and $$w, w_5 \leftarrow \mathbb{Z}_N^2,$$ and set $$U := g_1^{u_w w_5}, W := g_1^{w_w}, \tilde{U} := g_1^{u_5}$$ and $$\tilde{W} := g_1^w.$$ Randomly pick $$\beta \leftarrow \{0,1\}$$ and output $$\text{MPK} := (\emptyset, G, U, W, g_5).$$

Extract($$\omega$$) Return $$\bot$$ when $$\omega \in \text{ChID}.$$ Update ExID := ExID $$\cup \{\omega\}.$$ Sample $$r, r', r'' \leftarrow \mathbb{Z}_N$$ and set

$$K := \tilde{G}^r \cdot (X_2 X_3 X_4)^{r'}$$ and $$K' := H(\tilde{U}, W, \omega) \cdot (X_2 X_3 X_4)^{r''}.$$ Output $$(K, K').$$

Challenge($$\omega^*$$, $$m^*_0, m^*_1$$) Return $$\bot$$ when $$\omega^* \in \text{ExID}.$$ Update ChID := ChID $$\cup \{\omega^*\}.$$ If $$m^*_0 = 0,$$ sample $$t', t'' \leftarrow \mathbb{Z}_N$$ and set

$$C := T^{t'}$$ and $$C' := T^{t''}.$$ If $$m^*_0 = 1,$$ sample $$C, C' \leftarrow \text{Sample}_{\emptyset}().$$ Output $$(C, C').$$

Finalize($$\beta'$$) Output $$(\beta = \beta').$$

The algorithm $$\emptyset$$ outputs 1 if and only if Finalize($$\beta'$$) = 1, i.e., the adversary $$\mathcal{A}$$ wins game.

Observe that if $$T = (g_1 g_2 g_5)^{t_c} \in \mathbb{G}_{P,P'}P''_3,$$ the simulation described above is identical to Game$_4$; if $$T = g^t,$$ the simulation is identical to Game$_7.$ Therefore we can conclude that $$|\text{Adv}^\emptyset_\mathcal{A}(\lambda) - \text{Adv}^\emptyset_\mathcal{A}(\lambda)| \leq \text{Adv}^{\emptyset}_\mathcal{A}(\lambda).$$

Lemma 15 (Game$_7 \approx$ Game$_{\text{fin}}$). For adversary $$\mathcal{A}$$, $$|\text{Adv}^\emptyset_\mathcal{A}(\lambda) - \text{Adv}^{\emptyset}_{\text{fin}}(\lambda)| \leq 2\zeta.$$

Proof. These two games are exactly the same until publicly reversible sampler Sample$_{\emptyset}$ outputs $$\bot$$ when encrypting message 0 in Game$_{\text{fin}}.$ Clearly we can bound the probability of this event by $$2\zeta$$ where $$\zeta$$ is the error probability of Sample$_{\emptyset}.$ Therefore we can conclude that $$|\text{Adv}^\emptyset_\mathcal{A}(\lambda) - \text{Adv}^{\emptyset}_{\text{fin}}(\lambda)| \leq 2\zeta.$$


