Identity-Based Encryption Secure Against Selective Opening Attack

Mihir Bellare[1], Brent Waters[2], and Scott Yilek[3]

[1] University of California at San Diego
mihir@cs.ucsd.edu
[2] University of Texas at Austin
bwaters@cs.utexas.edu
[3] University of St. Thomas
syilek@stthomas.edu

Abstract. We present the first Identity-Based Encryption (IBE) schemes that are proven secure against selective opening attack (SOA). This means that if an adversary, given a vector of ciphertexts, adaptively corrupts some fraction of the senders, exposing not only their messages *but also their coins*, the privacy of the unopened messages is guaranteed. Achieving security against such attacks is well-known to be challenging and was only recently solved in the PKE case, but the techniques used there do not solve the IBE case. Our solutions illustrate two techniques to achieving SOA-secure IBE, one based on the Boyen-Waters anonymous IBE and the other based on Waters' dual-system approach.

Keywords: Identity-based encryption, pairings.

1 Introduction

Security against selective-opening attack (SOA) is arguably the most paradoxical and vexing open question in the theory of encryption. Recently (and 10 years after the problem was identified), we have seen solutions [2]. These and followups [24, 22], however, have been for the case of Public-Key Encryption (PKE). Another domain where the problem arises, and is important for applications, is Identity-Based Encryption (IBE). The techniques used for PKE do not yield solutions here. (That SOA-secure IBE remains open and challenging even with SOA-secure PKE achieved is not surprising since even basic IBE required new approaches compared to PKE and took much longer to achieve [9, 19, 5, 6, 34].) This paper initiates a treatment of IBE secure under SOA, providing definitions of security and the first solutions. Our solutions do not use random oracles.

Background. A selective-opening attack on a PKE scheme imagines n senders and receivers. Sender i encrypts a message $\mathbf{m}[i]$ under fresh, random coins $\mathbf{r}[i]$ and the public key $\mathbf{pk}[i]$ of the i-th receiver to get a ciphertext $\mathbf{c}[i]$. An adversary given the vector $\mathbf{c}$ corrupts some subset of the senders and learns not only their messages but also their coins. SOA-security requires that the remaining, unopened messages retain their privacy. SOA-security is required when implementing the assumed secure channels in an adaptively-secure multi-party computation protocol. More pragmatically, it would be required to distribute shares in a distributed file-system that is using secret-sharing for privacy.

IND-CPA and IND-CCA, widely-accepted as the "right" notions of encryption privacy, are not known to imply security under SOA. The difficulty of establishing SOA-security stems from the fact that the adversary gets the coins and also that the messages $\mathbf{m}[1], \ldots, \mathbf{m}[n]$ may be related. Constructions of SOA secure schemes also remained elusive, the area colored by negative results for commitment schemes [21, 2, 29]. Finally, Bellare, Hofheinz, and Yilek (BHY) [2] showed a large class of encryption schemes, which they call *lossy* [2, 26, 31], are SOA secure. Schemes they show to be lossy include variants of El Gamal [28], the IND-CPA scheme built from lossy trapdoor functions by Peikert and Waters [32], and even the original Goldwasser-Micali encryption scheme [23]. Hemenway, Libert, Ostrovsky and Vergnaud [24] showed that re-randomizable encryption and statistically hiding, two-round oblivious transfer imply lossy encryption, yielding still more examples of SOA secure PKE schemes via the lossy-implies-SOA-secure connection of BHY. Fehr, Hofheinz, Kiltz, and Wee (FHKW) [22] use a deniable encryption [13] approach to achieve CC-SOA (Chosen-Ciphertext SOA) secure PKE.

SOA for IBE. We can adapt the SOA framework to IBE in a natural way. A vector $\mathbf{id}$ of adversarially-chosen target receiver identities replaces the vector $\mathbf{pk}$ of public receiver keys. Sender i encrypts message $\mathbf{m}[i]$ under coins $\mathbf{r}[i]$ for identity $\mathbf{id}[i]$ to get a ciphertext $\mathbf{c}[i]$. As before the adversary, given $\mathbf{c}$, corrupts a subset of the senders and learns their messages and coins, and SOA-security requires that the unopened messages are secure. At any time, the adversary can query **Extract** with any identity not in the vector $\mathbf{id}$ and obtain its decryption key.

There are two elements here, new compared to PKE, that will be central to the technical challenges in achieving the goal. The first is the **Extract** oracle, a feature of IBE security formalizations since the pioneering work of Boneh and Franklin [9], that allows the adversary to obtain the decryption key of any (non-target) receiver of its choice. The second is that the target identities are chosen by the adversary. (We will achieve full, rather than selective-id security [15].)

IBE can conveniently replace PKE in applications such as those mentioned above, making its SOA-security important. Beyond this, we feel that determining whether SOA-secure IBE is possible is a question of both foundational and technical interest.

Contributions in brief. We provide a simulation-based, semantic security formalization of SOA-secure IBE. (This means our results do not need to assume conditional re-sampleability of message spaces, in contrast to some of the results of [2] for IND-style notions.) We provide a general paradigm to achieve SOA-secure IBE based on IBE schemes that are IND-CPA and have a property we call 1-Sided

Scheme	Pars	Ctxt	Keys	Enc	Dec	F/S	Assumption
LoR	$n+6$	5	5	5 exp	5 pr	F	DLIN
BBoR	4	2	2	2 exp	2 pr	F	GSD

Fig. 1. Our 1SPO IND-CPA IBE schemes. These encrypt 1-bit messages. Bit-by-bit encryption yields SOA-secure IBE schemes encrypting full messages. "Pars" is the size of the public parameters, "Ctxt" of the ciphertext and "Keys" of the decryption keys, all in group elements, with n the length of identities. (In practice $n = 160$ by hashing identities.) "Enc" and "Dec" are the encryption and decryption costs with "exp" standing for an exponentiation or multi-exponentiation and "pr" for a pairing. "F/S" indicates whether we get Full or Selective-id security. "GSD" stands for the General Subgroup Decision assumption.

Public Openability (1SPO). We discuss why obtaining 1SPO IND-CPA IBE schemes without random oracles is not immediate and then illustrate two ways to do it. The first, adapting the anonymous IBE scheme of Boyen and Waters [12], yields a SOA-secure IBE scheme based on the DLIN (Decision Linear) assumption of [7]. The second, using the dual-system approach of [33], yields a SOA-secure IBE scheme in the Boneh-Boyen style [5] based on a subgroup decision assumption in composite order groups. Attributes of the schemes are summarized in Figure 1. We now expand on these contributions.

1SPO IBE implies SOA-secure IBE. There are fundamental obstacles to extending BHY's lossy-implies-SOA-secure approach, that worked for SOA-secure PKE, to the IBE setting. (Briefly, we cannot make the encryption undetectably lossy on all challenge identities because the adversary has an **Extract** oracle and we wish to achieve full, not selective-id [15] security.) Instead, following FHKY [22], we return to ideas from non-committing [14] and deniable [13] encryption. We define IBE schemes that have a property we call *one-sided public openability* (1SPO) and is an IBE-analogue of a weak form of deniable PKE [13]. In short, an IBE scheme for 1-bit messages is 1SPO if it is possible, given the public parameters par, an identity id, and the encryption c of message 1 under par and id, to efficiently open the encryption, meaning find correctly-distributed randomness r such that encrypting a 1 using $\mathsf{par}, \mathsf{id}, r$ results in the ciphertext c. We emphasize that this opening must be done without the aid of any secret information. Bit-by-bit encryption then results in a scheme that can encrypt long messages. We show in Theorem 1 that if the starting 1-bit 1SPO scheme is also IND-CPA secure then the constructed IBE scheme is SOA-secure. This reduces the task of obtaining SOA-secure IBE schemes to obtaining IND-CPA secure 1SPO schemes.

Finding 1SPO IBE schemes. Known Random Oracle (RO) model IBE schemes [9, 19] can be adapted to be 1SPO secure, yielding SOA-secure IBE in the RO model. Achieving it without ROs, however, turns out not to be straightforward. The natural approach, extending that used for PKE [13, 22], is to build IBE schemes that are what we call 1SIS (1-sided invertibly sampleable). Here, encryptions of 0 to a certain identity would have a certain structure. This structure should be detectable with the secret key associated to the identity, but *not* without it, and thus not by an attacker. On the other hand, encryptions of 1 would be random, but in a special way, namely there is a public procedure that given an encryption c of a 1 can compute randomness (coins) under which the encryption algorithm applied to 1 would produce c. Any such scheme is 1SPO. The challenge that emerges is to find 1SIS IND-CPA IBE schemes. Existing IBE schemes do not have the property, and nor do direct adaptations work. The Boneh-Boyen approach [5] is probably the most widely used in IBE design. (Waters' IBE scheme [34] is one instance.) However, ciphertexts in BB-schemes contain group elements that obey relations an attacker can test and thus cannot be undetectably replaced with random group elements.

We will obtain our first solution by a different approach. Then, however, we will go back to show how the dual-system approach can be used to make a BB-style scheme work if we use composite order groups.

The linear scheme. In our "Linear or Random" (LoR) scheme, an encryption of 0 to a given identity, id, is done using (a modification of) the Boyen-Waters (BW) encryption algorithm [12]. This output of the encryption will be five group elements that share a certain structure that is only detectable to a user with the private key for id. To encrypt a 1 we simply choose five random group elements. This, however, must be done using what we call a publicly invertible process (see below). The main feature of this encryption scheme is that an encryption of 0 can always be claimed as just five random group elements, and thus as an encryption of 1. This reveals why we choose to build of the BW anonymous IBE scheme as opposed to other simpler IBE systems without random oracles. The main feature of the BW ciphertexts is that they have no detectable structure from an attacker that does not have a private key for id. In contrast, in BB-style IBE systems [5, 34] the attacker can test for structure between two group elements in well formed ciphertexts. Therefore we cannot create a secure encryption system simply by replacing these with random group elements.

We prove LoR is 1SPO directly. We must also, however, prove it is IND-CPA. We adapt techniques from [12, 3] to do this under the DLIN assumption. The proof technique of [3] allows us to avoid Waters' artificial abort step [34] thereby resulting in a more efficient reduction.

The dual-system scheme. Waters' introduced a new approach to IBE called the dual system approach in which both the challenge ciphertext and keys are replaced in the proof by "semi-functional" versions [33]. We adapt this approach to get a 1SIS (and thus 1SPO) IND-CPA IBE scheme and thus a SOA-secure IBE scheme. An interesting feature of the scheme is that ciphertexts have a BB-form, showing that the dual-system approach can surmount the above-mentioned difficulties in making BB-style systems 1SIS. We accordingly call the scheme BBoR (BB or Random). In addition this is interesting because it illustrates a quite different technique and yields a scheme based on a different assumption (subgroup decision in a composite group, not known to imply or be implied by DLIN in a prime-order group). As Figure 1 shows, the main pragmatic difference compared to LoR is short public parameters. (Those of LoR are long due to the Waters' hash function [34] which is required to get full security.) Others costs have dropped as well (from 5 to 2) but the group is larger so a closer analysis would be needed to determine whether this translates to actual efficiency gains.

Our starting points are the dual-system based Lewko-Waters (LW) IBE scheme [27] and its anonymous extension by De Caro, Iovino and Persiano (DIP) [17]. We modify these to get a 1SIS scheme where an encryption of a 0 is BB-ciphertext but in a subgroup while an encryption of a 1 is a pair of random points in the full group. While extending these schemes we manage simultaneously to make the assumptions simpler, more natural and fewer. Specifically, all these schemes rely on a pairing $\mathrm{e}\colon\ \mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$ where $\mathbb{G},\mathbb{G}_T$ have composite order N. LW make three different assumptions (numbered 1,2,3), the first two being about subgroup decision in $\mathbb{G}$ and the third in $\mathbb{G}_T$. DIP also make three assumptions, with the third being quite ad hoc and tailored to the scheme. We eliminate the third assumption in both cases and unify the rest, formulating what we call the general subgroup decision assumption, which is only in $\mathbb{G}$, and basing the proof solely on this single assumption.

Publicly invertible sampling. We have said that encryptions of a 1 in our 1SIS schemes are random group elements. This, however, is not enough. They have to be invertibly sampled. As we explained above, this means there is a public procedure that given an encryption c of a 1 can compute randomness (coins) under which the encryption algorithm applied to 1 would produce c. To illustrate the subtleties of the notion, consider a scheme in which the encryption c of a 1 is computed by picking an exponent x at random and returning g^x where g is a generator of a group $\mathbb{G}$. Although the ciphertext is random, this is not invertibly sampleable since we cannot recover x from c. Instead, a ciphertext

must be sampled "directly" as $c \leftarrow\!\!\$\ \mathbb{G}$. The difficulty is that whether or not this is possible depends on the group. In the PKE case, it is possible to stay within simple groups such as $\mathbb{Z}_p^*$ for prime p, where such sampling is easy. (Pick a a random integer in the range $1, \ldots, p-1$.) In our case, however, $\mathbb{G}$ is a complex group, namely a subgroup of the points on an elliptic curve. We show how to sample invertibly nonetheless, relying on the structure of the elliptic curve groups in question. Specifically, we modify some methods used to implement the hash function of the BLS signature scheme [11].

Extensions and remarks. After seeing a preliminary version of our work, Peikert [30] showed that the lattice-based IBE schemes of [18, 1] adapt to yield 1SPO schemes, whence, by our results, SOA-secure IBE. Furthermore our second scheme above, as well as some of these lattice schemes, can be extended to SOA-secure HIBEs. Despite this we remark that we do not achieve CC-SOA (Chosen-ciphertext SOA) secure IBE which remains an interesting open question. (The BCHK transform [8] does not work in this setting.) Applying the results of [13] to our 1SPO IBE schemes gives us the first deniable IBE schemes, which may be of independent interest.

Related work. Canetti, Feige, Goldreich and Naor [14] introduced non-committing encryption (NCE) to achieve adaptively secure multi-party computation in the computational (as opposed to secure channels) setting without erasures. In their treatment, NCE is an interactive protocol, and their definition of security is in the MPC framework. The model allows corruption of both senders and receivers. They show how to achieve NCE but, viewed as a public-key system, they would have keys larger than the total number of message bits that may be securely encrypted. Damgård and Nielsen [20] introduced more efficient schemes but this restriction remained, and Nielsen [29] showed it was necessary. With partial erasures, more efficient solutions were provided by Canetti, Halevi and Katz [16].

Dwork, Naor, Reingold and Stockmeyer [21] extracted out a stand-alone notion of committment secure against selective opening defined directly by a game rather than via the MPC framework. Corruptions allow the adversary to obtain the committer's coins along with its message. This was adapted to public-key encryption in [2], who focused on sender (as opposed to receiver) corruptions and were then able to obtain solutions based on lossy encryption.

Canetti, Dwork, Naor and Ostrovsky [13] introduced deniable encryption, where a sender may open a ciphertext to an arbitrary message by providing coins produced by a faking algorithm. The authors explain that this is stronger than NCE because in the latter only a simulator can open in this way. A weak form of their requirement is that encryptions of 1 can be opened as encryptions of 0 even if not vice versa. 1SPO IBE is an IBE analogue of this notion.

Sender versus receiver corruptions. We clarify that our model and results are for adaptive sender corruptions, not adaptive receiver corruptions. (The latter would correspond to being allowed to query to **Extract** identites in the challenge vector **id**.) Security against adaptive receiver corruptions seems out of reach of current techniques for PKE let alone for IBE. (Without either erasures or keys as long as the total number of messages bits ever encrypted.) We do allow receiver corruptions via the **Extract** oracle but these are non-adaptive. We view this as retaining (meaning neither weakening nor strengthening) the guarantees against receiver corruption already provided by the basic definition of IND-CPA-secure IBE [9]. Our notion, security against adaptive sender and non-adaptive receiver corruptions, is still very strong.

2 Preliminaries

Notation. We use boldface to denote vectors, i.e., $\mathbf{m}$. For vector $\mathbf{m}$, we let $|\mathbf{m}|$ denote the number of components in the vector. When $\mathbf{m}[i] \in \{0,1\}^*$, we denote by $\mathbf{m}[i][j]$ the jth bit of the ith component of $\mathbf{m}$, i.e., the jth bit of $\mathbf{m}[i]$. On the other hand, when $\mathbf{c}[i]$ is a sequence, we let $\mathbf{c}[i][j]$ denote the jth

value in the sequence $\mathbf{c}[i]$. We sometimes abuse notation and treat vectors as sets. Specifically, if S is a set we may write $S \cup \mathbf{m}$ to denote $S \cup \{\mathbf{m}[1]\} \cup \{\mathbf{m}[2]\} \ldots$. If two adversaries $\mathcal{A}$ and $\mathcal{B}$ have access to different oracles with the same name (e.g., **NewMesg**) we sometimes write $\mathbf{NewMesg}_{\mathcal{B}}$ to mean $\mathcal{B}$'s version of the oracle. For $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$.

We fix pairing parameters $\mathsf{GP} = (\mathbb{G}, \mathbb{G}_T, p, \mathrm{e})$ where $\mathbb{G}, \mathbb{G}_T$ are groups of order prime p and the map $\mathrm{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficiently computable non-degenerate bilinear map. We let $\mathsf{T}_{\mathrm{exp}}(\mathbb{G})$ be the time to compute an exponentiation in the group $\mathbb{G}$. We let $\mathsf{T}_{\mathrm{op}}(\mathbb{G})$ be the time to compute a group operation in $\mathbb{G}$. For any group $\mathbb{G}$, let $\mathbb{G}^*$ denote the generators of $\mathbb{G}$.

Code-Based Games. We use code based games [4] for our security definitions. A game consists of numerous procedures including an **Initialize** procedure and a **Finalize** procedure. When an adversary $\mathcal{A}$ executes with the game, the **Initialize** procedure is executed first and its outputs are the initial inputs to adversary $\mathcal{A}$. Then $\mathcal{A}$ executes and its oracle queries are answered by the corresponding procedures of the game. When the adversary halts with some final output, this output is given as input to the **Finalize** procedure. The output of the **Finalize** procedure is then considered the output of the game. We let $G^{\mathcal{A}} \Rightarrow y$ be the event that game G, when executed with adversary $\mathcal{A}$, has output y. We abbreviate "$G^{\mathcal{A}} \Rightarrow \mathsf{true}$" by "$G^{\mathcal{A}}$". We let $\mathsf{BD}(G^{\mathcal{A}})$ denote the event that the execution of G with $\mathcal{A}$ sets flag bad, and $\mathsf{GD}(G^{\mathcal{A}})$ its complement. The running time of the adversary while playing the game is considered to be the running time of the adversary while playing the game plus the time to execute all of the game procedures during the execution.

Randomized Algorithms and Sampling from Groups. We have to model randomized algorithms carefully and in a particular way to define invertible sampling. We assume that all algorithms have access to a RNG Rand that is the only source of randomness in the system. On input a positive integer n, function Rand returns a value uniformly distributed in $\mathbb{Z}_n$. We stress that Rand is not viewed as having an underlying source of coins in the form of bits as in complexity-theoretic/Turing machine models. Rather, its operation is atomic and its outputs *are* the coins.

When we write $a \leftarrow\!\!\$\ \mathbb{G}$ we mean that we run $i \leftarrow\!\!\$\ \mathsf{Rand}(p)$, where $p = |\mathbb{G}|$, and let $a = g^i$ where g is a generator of $\mathbb{G}$. However, we also want to use publicly reversible sampling. A publicly reversible (PR) sampler Sample takes no input and, via access to Rand, outputs a point in $\mathbb{G}$ or the failure symbol $\perp$. It has sampling failure probability ζ if the probability that it outputs $\perp$ is at most ζ. We require that $\Pr[a' = a \mid a' \neq \perp] = 1/|\mathbb{G}|$ for all $a \in \mathbb{G}$, where the probability is over $a' \leftarrow\!\!\$\ \mathsf{Sample}$.

If $(r_1, \ldots, r_s)$ is a sequence of non-negative integers, we let $\mathsf{Sample}[r_1, \ldots, r_s]$ be the result of running Sample with Rand replaced by the subroutine that returns r_i in response to the i-th query made to it, for $1 \leq i \leq s$. We require that there is an algorithm Sample^{-1} which on input $a \in \mathbb{G}$ outputs a sequence $(r_1, \ldots, r_s)$ such that $\mathsf{Sample}[r_1, \ldots, r_s] = a$. ($\mathsf{Sample}^{-1}$, as with any other algorithm, has access to Rand.) Sample^{-1} also might fail (and output $\perp$). We call this the reverse sampling failure probability and denote it with θ.

Identity-Based Encryption. An Identity-based encryption scheme (IBE) is a tuple of algorithms $\Pi = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with identity space IdSp, message space MsgSp, and the following properties. The parameter generation algorithm Pg takes no input and outputs a public parameter string par and a master secret key msk. The identity key generation algorithm Kg takes as input the public parameter string par, the master secret key msk, and an identity id, and outputs a secret key sk for identity id. The encryption algorithm Enc takes as input the public parameters par, an identity id, and a message M, and outputs a ciphertext C. Lastly, the decryption algorithm Dec takes as input the public parameters par, an identity secret key sk, and a ciphertext C, and outputs either a message M or a failure symbol $\perp$. We say that an IBE scheme has completeness error ϵ if the probability that $\mathsf{Dec}(\mathsf{par}, sk, \mathsf{id}, \mathsf{Enc}(\mathsf{par}, \mathsf{id}, M)) = M$ is $\geq 1 - \epsilon$ for all $\mathsf{id} \in \mathsf{IdSp}$, all $M \in \mathsf{MsgSp}$, all $(\mathsf{par}, \mathsf{msk}) \in [\mathsf{Pg}]$, and all $sk \in [\mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})]$, where the probability is taken over the coins used in encryption.

proc. Initialize: $(\mathsf{par}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Pg}$; $b \leftarrow\!\!\$\ \{0,1\}$ Return par **proc. Extract**(id): Return $\mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$	**proc. LR**(id, M_0, M_1): Return $\mathsf{Enc}(\mathsf{par}, \mathsf{id}, M_b)$ **proc. Finalize**(b'): Return $(b = b')$	INDCPA_{Π}

Fig. 2. The IBE IND-CPA Game

A one-bit IBE scheme $\Pi = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ is one with $\mathsf{MsgSp} = \{0,1\}$, while an ℓ-bit IBE scheme has $\mathsf{MsgSp} = \{0,1\}^\ell$. We will build ℓ-bit IBE schemes from one-bit IBE schemes as follows. Given one-bit IBE scheme Π as above, let $\Pi^\ell = (\mathsf{Pg}^\ell, \mathsf{Kg}^\ell, \mathsf{Enc}^\ell, \mathsf{Dec}^\ell)$ be an ℓ-bit IBE scheme defined as follows: parameter and key generation are unchanged, i.e., $\mathsf{Pg}^\ell = \mathsf{Pg}$ and $\mathsf{Kg}^\ell = \mathsf{Kg}$. The encryption algorithm Enc^ℓ, on input par, id, $M \in \{0,1\}^\ell$, outputs $\mathsf{Enc}(\mathsf{par}, \mathsf{id}, M[1]) \,\|\, \ldots \,\|\, \mathsf{Enc}(\mathsf{par}, \mathsf{id}, M[\ell])$, where $M[i]$ is the ith bit of M. In other words, encryption encrypts each bit separately and concatenates the resulting ciphertexts. Decryption works in the obvious way: decrypt each ciphertext component separately to learn individual bits. It is easy to see that if Π has ϵ completeness error, then the resulting ℓ-bit scheme has completeness error at most $\ell \cdot \epsilon$.

The standard notion of security for IBE schemes is indistinguishability under chosen plaintext attack (IND-CPA) [9]. We define the IND-CPA advantage of an IND-CPA adversary A against IBE scheme Π to be

$$\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(A) = 2 \cdot \Pr\left[\, \mathrm{INDCPA}^{A}_{\Pi} \Rightarrow \mathsf{true} \,\right] - 1 \ ,$$

where game INDCPA can be found in Figure 2. An IND-CPA adversary interacts with game INDCPA, querying **LR** only once and on an identity $\mathsf{id}^* \in \mathsf{IdSp}$ that is never queried to **Extract** and on equal length messages $M_0, M_1 \in \mathsf{MsgSp}$. We note that adversaries may query the same identity id to **Extract** multiple times, since key generation is randomized.

We associate to encryption algorithm Enc the set $\mathsf{Coins}(\mathsf{par}, m)$. This is the set from which Enc draws its coins when encrypting message m using parameters par. Similarly, we let $\mathsf{Coins}(\mathsf{par}, \mathsf{id}, c, 1)$ be the set of coins $\{r \mid c = \mathsf{Enc}(\mathsf{par}, \mathsf{id}, 1; r)\}$.

3 Security against Selective Opening Attacks

In this section we formalize SOA security for IBE, closely following the formalizations from [2]. Before proceeding, we need two definitions. A (k, ℓ)-message sampler is a randomized algorithm $\mathcal{M}$ that on input string $\alpha \in \{0,1\}^*$ outputs a vector of messages $\mathbf{m}$ such that $|\mathbf{m}| = k$ and each $\mathbf{m}[i] \in \{0,1\}^\ell$. A relation $\mathcal{R}$ is any randomized algorithm that outputs a single bit.

An soa-adversary is one that runs with game IBESOAREAL making one query to **NewMesg** before making one query to **Corrupt**; it may make one or more queries to **Extract** at any time during the game. An soa-simulator is an adversary that runs with game IBESOASIM, makes one query to **NewMesg** and later makes one query to **Corrupt**. It makes no **Extract** queries. We define the soa-advantage of soa-adversary $\mathcal{A}$ against an IBE scheme Π with respect to a (k, ℓ)-message sampler $\mathcal{M}$, relation $\mathcal{R}$, and soa-simulator $\mathcal{S}$ as

$$\mathbf{Adv}^{\text{soa}}_{\Pi,k,\mathcal{S},\mathcal{M},\mathcal{R}}(\mathcal{A}) = \Pr\left[\, \mathrm{IBESOAREAL}^{\mathcal{A}}_{\Pi,k,\mathcal{M},\mathcal{R}} \Rightarrow 1 \,\right] - \Pr\left[\, \mathrm{IBESOASIM}^{\mathcal{S}}_{\Pi,k,\mathcal{M},\mathcal{R}} \Rightarrow 1 \,\right] \ .$$

Discussion. In game IBESOAREAL (shown in Figure 3), the **Initialize** procedure runs the parameter generation algorithm and returns the scheme parameters to the adversary. The adversary then runs with oracles **NewMesg**, **Corrupt**, and **Extract**. The adversary may never query an identity to **Extract** that appears in a query to **NewMesg**.

proc. Initialize:	**proc. NewMesg**($\mathbf{id}, \alpha$):	**proc. Corrupt**(I):
$(\mathsf{par}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Pg}$	If $\mathbf{id} \cap \text{ExID} \neq \emptyset$ then return $\perp$	Return $\mathbf{r}[I]$, $\mathbf{m}[I]$
Return par	$\text{ChID} \leftarrow \text{ChID} \cup \mathbf{id}$	**proc. Finalize**(out):
proc. Extract(id):	$\mathbf{m} \leftarrow\!\!\$\ \mathcal{M}(\alpha)$	Return $\mathcal{R}(\mathbf{m}, \text{ChID}, I, out)$
If $\mathsf{id} \in \text{ChID}$ then return $\perp$	For i in 1 to k	
$\text{ExID} \leftarrow \text{ExID} \cup \{\mathsf{id}\}$	$\quad \mathbf{r}[i] \leftarrow\!\!\$\ \mathsf{Coins}(\mathsf{par}, \mathbf{m}[i])$	
$sk \leftarrow\!\!\$\ \mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$	$\quad \mathbf{c}[i] \leftarrow \mathsf{Enc}(\mathsf{par}, \mathbf{id}[i], \mathbf{m}[i]; \mathbf{r}[i])$	
Return sk	Return $\mathbf{c}$	

Fig. 3. Game $\text{IBESOAREAL}_{\Pi,k,\mathcal{M},\mathcal{R}}$.

proc. Initialize:	**proc. NewMesg**($\mathbf{id}, \alpha$):	**proc. Corrupt**(I):
Return $\perp$	$\text{ChID} \leftarrow \text{ChID} \cup \mathbf{id}$; $\mathbf{m} \leftarrow\!\!\$\ \mathcal{M}(\alpha)$	Return $\mathbf{m}[I]$
	Return $\perp$	**proc. Finalize**(out):
		Return $\mathcal{R}(\mathbf{m}, \text{ChID}, I, out)$

Fig. 4. Game $\text{IBESOASIM}_{\Pi,k,\mathcal{M},\mathcal{R}}$.

The adversary may query the **NewMesg** oracle once with a vector of identities **id** and a string α that is meant to capture state to pass on to the message sampler. Procedure **NewMesg**, on input **id** and α, samples a vector of messages from the message sampling algorithm $\mathcal{M}$ and encrypts the entire vector using independent coins to the identities specified in **id**. This means that the ith component of the resulting ciphertext vector **c** is $\mathsf{Enc}(\mathsf{par}, \mathbf{id}[i], \mathbf{m}[i]; \mathbf{r}[i])$, the encryption of the ith message to the ith identity with the ith coins.

After querying the **NewMesg** oracle, the adversary may make one query to **Corrupt** with a set of indices $I \subseteq [k]$. These indices specify which ciphertexts from the vector **c** returned by **NewMesg** the adversary would like opened. The **Corrupt** procedure returns the messages and randomness used in **NewMesg** corresponding to indices in I. Additionally, at any time the adversary may query the **Extract** oracle on an identity of its choice and learn a secret key for that identity. We do not allow the adversary to query **Extract** on any identity appearing in the vector id queried to **NewMesg**.

Finally, the adversary halts with output out and the output of the game is the relation $\mathcal{R}$ applied to the message vector **m**, the set of challenge IDs ChID, the corrupt set I, and the output out.

In game IBESOASIM (shown in Figure 4), the **Initialize** procedure does nothing and returns $\perp$ to the simulator. The simulator then runs with two oracles, **NewMesg** and **Corrupt**. On input an identity vector **id** and a string α, oracle **NewMesg** samples a vector **m** of messages using the message sampling algorithm $\mathcal{M}$ applied to the state string α. Nothing is returned to the simulator. The simulator is only allowed one **NewMesg** query. At a later time, the simulator may then make a single query to oracle **Corrupt** with a set of indices I and as a result will learn the messages in **m** corresponding to I. Finally, the simulator halts with output out and the output of the game is the relation $\mathcal{R}$ applied to the message vector **m**, the set of challenge IDs ChID, the corrupt set I, and the output out.

As noted at the end of Section 1, we model adaptive sender corruptions while retaining standard IBE security against non-adaptive receiver corruptions. (Adaptive receiver corruptions would correspond to removing the restriction that **Extract** return $\perp$ when queried on a challenge identity.) Security against adaptive receiver corruptions seems out of reach of current techniques for PKE let alone IBE.

4 Selective Opening Security from One-Sided Publicly Openable IBE

We first formalize the key property we will need to prove SOA security. A perfect one-sided public (1SP) opener for one-bit IBE scheme $\Pi = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ is an algorithm $\mathsf{OpenToOne}$ that takes input parameters par, identity id, and ciphertext c, and has the following property: for all $\mathsf{par} \in [\mathsf{Pg}]$, all $\mathsf{id} \in \mathsf{IdSp}$, every $c \in [\mathsf{Enc}(\mathsf{par}, \mathsf{id}, 1)]$, and every $\bar{r} \in \mathsf{Coins}(\mathsf{par}, \mathsf{id}, c, 1)$,

$$\Pr\left[\, r \leftarrow\!\!\$\ \mathsf{OpenToOne}(\mathsf{par}, \mathsf{id}, c)\ :\ r = \bar{r}\,\right] = \frac{1}{|\mathsf{Coins}(\mathsf{par}, \mathsf{id}, c, 1)|}\ .$$

We can weaken this definition slightly by considering opening algorithms that can fail with some probability δ, but in the case of success their output distribution is identical to the actual coin distribution. This is reflected as for all $\mathsf{par} \in [\mathsf{Pg}]$, all $\mathsf{id} \in \mathsf{IdSp}$, every $c \in [\mathsf{Enc}(\mathsf{par}, \mathsf{id}, 1)]$, and every $\bar{r} \in \mathsf{Coins}(\mathsf{par}, \mathsf{id}, c, 1)$,

$$\Pr\left[\, r \leftarrow\!\!\$\ \mathsf{OpenToOne}(\mathsf{par}, \mathsf{id}, c)\ :\ r = \bar{r} \mid r \neq \bot\,\right] = \frac{1}{|\mathsf{Coins}(\mathsf{par}, \mathsf{id}, c, 1)|}\ .$$

Notice that the probability is only over the coins used by $\mathsf{OpenToOne}$. We call such an $\mathsf{OpenToOne}$ algorithm a δ-1SP opener and we also call an IBE scheme with a δ-1SP opener δ-one-sided publicly openable (δ-1SPO).

The idea of constructing encryption schemes with such one-sided opening is originally due to Canetti, Dwork, Naor, and Ostrovsky [13]. They used PKE schemes with this property to build deniable public-key encryption schemes. To achieve what we call the 1SPO property, they built PKE schemes from translucent sets with the property that an encryption of a 1 was pseudorandom, while the encryption of a 0 was perfectly random; it was then always possible to simply claim the encryption of a 1 was random to open to a 0. The secret key allowed one to tell the difference between pseudorandom and random values, allowing correct decryption. More recently, in independent work, Fehr, Hofheinz, Kiltz, and Wee [22] used PKE schemes with a 1SPO property as a building block to achieve CCA SOA public-key encryption security. Of course, both of these works focus on PKE. Our focus, on the other hand, is on building secure IBE schemes. In the next section we will show how to achieve the 1SPO property in the IBE setting.

From 1SPO to SOA. We now state our main result: IND-CPA 1SPO one-bit IBE schemes lead to many-bit SOA-secure IBE schemes. Let $\Pi = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ denote a one-bit IBE scheme that is δ-one sided openable and let $\Pi^\ell = (\mathsf{Pg}^\ell, \mathsf{Kg}^\ell, \mathsf{Enc}^\ell, \mathsf{Dec}^\ell)$ the ℓ-bit IBE scheme built from Π as described in Section 2.

Theorem 1. *Let Π be a one-bit IBE scheme with a δ one-sided opener $\mathsf{OpenToOne}$, and let Π^ℓ be the ℓ-bit scheme built from it. Let k be an integer, $\mathcal{A}$ an soa-adversary making at most q queries to **Extract**, $\mathcal{R}$ a relation, and $\mathcal{M}$ a (k, ℓ)-message sampler. Then there exists an soa-simulator $\mathcal{S}$ and an IND-CPA adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathrm{soa}}_{\Pi^\ell, k, \mathcal{M}, \mathcal{R}, \mathcal{S}}(\mathcal{A}) \leq k\ell \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(\mathcal{B}) + k\ell \cdot \delta\ ,$$

where $\mathsf{T}(\mathcal{S}) = \mathcal{O}(\mathsf{T}(\mathcal{A}) + k\ell \cdot \mathsf{T}(\mathsf{OpenToOne}) + q \cdot \mathsf{T}(\mathsf{Kg}^\ell) + k \cdot \mathsf{T}(\mathsf{Enc}^\ell) + \mathsf{T}(\mathsf{Pg}^\ell))$ *and* $\mathsf{T}(\mathcal{B}) = \mathcal{O}(\mathsf{T}(\mathcal{A}) + \mathsf{T}(\mathcal{M}) + k\ell \cdot \mathsf{T}(\mathsf{Enc}) + k\ell \cdot \mathsf{T}(\mathsf{OpenToOne}) + \mathsf{T}(\mathcal{R}))$. □

We prove the theorem in Appendix B. We briefly sketch the proof. Recall that to show SOA security we need to be able to create an efficient simulator. Given an soa-adversary $\mathcal{A}$, we can construct a simulator $\mathcal{S}$ that runs $\mathcal{A}$ and gives it encryptions of all 0s. At some point $\mathcal{A}$ will ask for some of the ciphertexts to be opened. When this happens, the simulator queries its own **Corrupt** oracle, learns the messages it needs to open the ciphertexts to, and proceeds to open bit-by-bit. If it needs to open a ciphertext component to a 0, it simply gives $\mathcal{A}$ the coins it used when originally creating the ciphertext. If, however, $\mathcal{S}$ needs to open a ciphertex to a 1, it uses the scheme's $\mathsf{OpenToOne}$ algorithm to find

proc. Initialize:	DLIN$_{\mathbb{G}}$
$g, g_1, g_2 \leftarrow\!\!\$\ \mathbb{G}^*$; $a_1, a_2 \leftarrow\!\!\$\ \mathbb{Z}_p$; $d \leftarrow\!\!\$\ \{0,1\}$; $h_1 \leftarrow g_1^{a_1}$; $h_2 \leftarrow g_2^{a_2}$ If $d = 1$ then $W = g^{a_1+a_2}$ Else $W \leftarrow\!\!\$\ \mathbb{G}$ Return $(g, g_1, g_2, h_1, h_2, W)$	**proc. Finalize**(d'): Return $(d = d')$

Fig. 5. The DLIN game for the decisional linear assumption.

the coins. The simulator then outputs the same output as $\mathcal{A}$. In short, the IND-CPA security of the scheme will allow us to argue that the simulator is successful. To prove this, we need to do a hybrid over the ℓ individual components of the k messages sampled from $\mathcal{M}$, where in the ith hybrid game the first i bits sampled from $\mathcal{M}$ are ignored and 0s are encrypted in their place. Thus, in the first hybrid game all bits sampled from $\mathcal{M}$ are accurately encrypted, while in the last hybrid game only 0s are encrypted. This hybrid causes the loss of a factor $k \cdot \ell$ in the theorem.

5 A First Attempt

As a first attempt at constructing a 1SPO IBE scheme, we try to adapt techniques from deniable PKE [13], in particular the idea that an encryption of a 0 should be "pseudorandom" while the encryption of a 1 is "random". The decryption algorithm should then be able to use secret information to distinguish between the cases.

A typical IBE scheme has ciphertexts consisting of a tuple of group elements with some structure. To make such a scheme 1SPO, a natural idea is to make the honestly-generated ciphertext tuple the encryption of a 0, and a tuple of random group elements an encryption of a 1. The secret key for an identity then contains information which helps test for this structure. Since a tuple of random group elements is unlikely to give the same structure as an honestly-generated ciphertext, decryption should be possible. Let us see what happens if we apply this idea to the IBE scheme of Boneh and Boyen (BB) [5], which has been the basis for many other IBE schemes (e.g., the Waters (W) IBE [34]).

Recall in the BB scheme (and its variants) a ciphertext has the form $(C_1, C_2, C_3) = (\mathrm{e}(g_1, g_2)^s \cdot M, g^s, H(\mathbf{u}, \mathsf{id})^s)$, where different variants define the hash function H differently, $g, g_1, g_2, \mathbf{u}$ are part of the parameters, and s is chosen randomly by the encryptor. Now, if we follow the ideas described above for making the scheme 1SPO, encryptions of 0 would be $(C, C') = (g^s, H(\mathbf{u}, \mathsf{id})^s)$, while encryptions of 1 would be a pair of group elements chosen uniformly at random from $\mathbb{G} \times \mathbb{G}$.

Syntactically the scheme works, but it is unfortunately not IND-CPA secure. The reason is that distinguishing an encryption of a 0 from an encryption of a 1 is exactly the DDH problem in $\mathbb{G}$. In the 0 case, the tuple $(g, H(\mathbf{u}, \mathsf{id}), g^s, H(\mathbf{u}, \mathsf{id})^s)$ is a Diffie-Hellman tuple; in the 1 case, the tuple is just four random group elements. Since the DDH problem is easy in groups with an efficiently-computable pairing, this gives us a simple test for whether a ciphertext is the encryption of a 0 or a 1: take (C, C') and output 0 if $\mathrm{e}(g, C') = \mathrm{e}(C, H(\mathbf{u}, \mathsf{id}))$ and 1 otherwise.

The fundamental issue is that in the BB scheme, the structure of the ciphertexts can be detected given only public parameters. The key idea in our two schemes is to destroy any structure that is publicly detectable.

6 A 1SPO IBE Scheme Based on the Decisional Linear Assumption

The security of our first scheme will rely on the Decisional Linear Assumption [7]. The decisional linear game DLIN is found in Figure 5. We say the DLIN-advantage of an adversary A against GP is

$$\mathbf{Adv}^{\mathrm{dlin}}_{\mathsf{GP}}(A) = 2 \cdot \Pr\left[\, \mathrm{DLIN}^{A}_{\mathsf{GP}} \Rightarrow \mathsf{true} \,\right] - 1 \,.$$

Algorithm Pg:
$g \leftarrow\!\!\$\ \mathbb{G}^*$; $\mathbf{u} \leftarrow\!\!\$\ \mathbb{G}^{n+1}$
$t_1, t_2, t_3, t_4 \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$v_1 \leftarrow g^{t_1}$; $v_2 \leftarrow g^{t_2}$; $v_3 \leftarrow g^{t_3}$; $v_4 \leftarrow g^{t_4}$
$\mathsf{par} \leftarrow (g, \mathbf{u}, v_1, v_2, v_3, v_4)$
$\mathsf{msk} \leftarrow (t_1, t_2, t_3, t_4)$
Return $(\mathsf{par}, \mathsf{msk})$

Algorithm $\mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$:
$(g, \mathbf{u}, v_1, v_2, v_3, v_4) \leftarrow \mathsf{par}$
$(t_1, t_2, t_3, t_4) \leftarrow \mathsf{msk}$
$r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_p$
$d_0 \leftarrow g^{r_1 t_1 t_2 + r_2 t_3 t_4}$
$d_1 \leftarrow H(\mathbf{u}, \mathsf{id})^{-r_1 t_2}$; $d_2 \leftarrow H(\mathbf{u}, \mathsf{id})^{-r_1 t_1}$
$d_3 \leftarrow H(\mathbf{u}, \mathsf{id})^{-r_2 t_4}$; $d_4 \leftarrow H(\mathbf{u}, \mathsf{id})^{-r_2 t_3}$
Return $(d_0, d_1, d_2, d_3, d_4)$

Algorithm $\mathsf{Enc}(\mathsf{par}, \mathsf{id}, M)$:
$(g, \mathbf{u}, v_1, v_2, v_3, v_4) \leftarrow \mathsf{par}$
If $M = 0$ then
  $s, s_1, s_2 \leftarrow\!\!\$\ \mathbb{Z}_p$; $C_0 \leftarrow H(\mathbf{u}, \mathsf{id})^s$
  $C_1 \leftarrow v_1^{s-s_1}$; $C_2 \leftarrow v_2^{s_1}$; $C_3 \leftarrow v_3^{s-s_2}$; $C_4 \leftarrow v_4^{s_2}$
Else
  For $i = 0$ to 4 do $C_i \leftarrow\!\!\$\ \mathsf{Sample}_{\mathbb{G}}()$
Return $(C_0, C_1, C_2, C_3, C_4)$

Algorithm $\mathsf{Dec}(\mathsf{par}, sk, C)$:
$(g, \mathbf{u}, v_1, v_2, v_3, v_4) \leftarrow \mathsf{par}$
$(d_0, d_1, d_2, d_3, d_4) \leftarrow sk$
$(C_0, C_1, C_2, C_3, C_4) \leftarrow C$
If $\prod_{i=0}^{4} \mathrm{e}(C_i, d_i) = 1_{\mathbb{G}_T}$ then
  Return 0
Else return 1

Fig. 6. Scheme LoR based on Boyen-Waters IBE.

Scheme Description. We now describe the details of our IBE scheme $\mathsf{LoR} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$. We call it LoR, which is short for "Linear or Random" to represent the fact that in our scheme the encryption of a 0 consists of five group elements whose relationship is similar to that of the group elements in the decisional linear assumption, while an encryption of a 1 consists of five random group elements. Our scheme is a one-bit version of the anonymous IBE scheme from Boyen and Waters [12] and using the Waters' hash function [34] for adaptive security. The scheme will use a cyclic group $\mathbb{G}$ of prime order p with an efficiently computable pairing $\mathrm{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We also require the group has a PR sampler Sample with failure probability ζ and corresponding inverse sampler Sample^{-1} with reverse failure probability θ. (Appendix A gives details on how to instantiate such groups.) Let $\mathbb{G}^*$ denote the generators of $\mathbb{G}$. Let $1_{\mathbb{G}_T}$ be the identity element of the target group $\mathbb{G}_T$. Define hash function $H : \mathbb{G}^{n+1} \times \{0,1\}^n \to \mathbb{G}$ as $H(\mathbf{u}, \mathsf{id}) = \mathbf{u}[0] \prod_{i=1}^{n} \mathbf{u}[i]^{\mathsf{id}[i]}$, where $\mathsf{id}[i]$ is the ith bit of string id. This is the Waters' hash function [34]. The scheme LoR, shown in Figure 6, has message space $\{0,1\}$ and identity space $\{0,1\}^n$. The scheme has completeness error $1/p^2$.

We claim the scheme is δ-1SPO where $\delta \le 5\theta$. The algorithm $\mathsf{OpenToOne}$ simply runs Sample^{-1} on each of the five ciphertext components with independent failure probabilities θ.

It remains to show LoR is IND-CPA. The following theorem establishes this based on the Decisional Linear Assumption.

Theorem 2. Fix pairing parameters $\mathsf{GP} = (\mathbb{G}, \mathbb{G}_T, p, \mathrm{e})$ and an integer $n \ge 1$, and let $\mathsf{LoR} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be the one-bit IBE scheme associated to GP and $\mathsf{IdSp} = \{0,1\}^n$. Assume $\mathbb{G}$ is PR-samplable with sampling failure probability ζ. Let $\mathcal{A}$ be an IND-CPA adversary against LoR which has advantage $\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{LoR}}(\mathcal{A}) > 2^{n+1}/p + 5\zeta$ and makes at most $q \in [1\,..\,p\epsilon/9n]$ queries to its **Extract** oracle. Let

$$\delta = \frac{1}{2}\left(\frac{\epsilon}{2} - \frac{2^n}{p} - 5\zeta\right) .$$

Then there is a DLIN-adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{dlin}}_{\mathsf{GP}}(\mathcal{B}) \ge \frac{\delta^2}{9qn + 3\delta} \quad \text{and} \quad \mathsf{T}(\mathcal{B}) = \mathsf{T}(\mathcal{A}) + \mathsf{T}_{\text{sim}}(n, q) \tag{1}$$

proc. Initialize:	**proc. Gen**(S):	**proc. Ch**(S_0, S_1):
$(\pi, \bar{\pi}) \leftarrow\!\!\$\ \mathsf{Gen}$; $b \leftarrow\!\!\$\ \{0,1\}$	If $(S \cap S_0 = \emptyset) \wedge (S \cap S_1 \neq \emptyset)$	If $(S_0 = \emptyset$ or $S_1 = \emptyset)$
$(\langle\mathbb{G}\rangle, \langle\mathbb{G}_T\rangle, \langle \mathrm{e}\rangle, N) \leftarrow \pi$	then return $\perp$	then return $\perp$
$(p_1, \ldots, p_n) \leftarrow \bar{\pi}$	If $(S \cap S_0 \neq \emptyset) \wedge (S \cap S_1 = \emptyset)$	$T \leftarrow\!\!\$\ \mathbb{G}(S_b)$
Return π	then return $\perp$	Return T
	$g \leftarrow\!\!\$\ \mathbb{G}(S)$	**proc. Finalize**(b'):
	Return g	Return $(b = b')$

Fig. 7. Game $\mathrm{GSD}_{\mathsf{Gen}}$

where $\mathsf{T}_{\mathrm{sim}}(n, q) = \mathcal{O}(qn + (n+q)\mathsf{T}_{\mathrm{exp}}(\mathbb{G}))$. □

The proof of the theorem combines techniques from [12, 34, 3] and can be found in Appendix C.

7 A Scheme based on Dual System IBE

General Subgroup Decision. An order-n group generator with security parameter k is an algorithm Gen that returns a pair $(\pi, \bar{\pi})$, where $\pi = (\langle\mathbb{G}\rangle, \langle\mathbb{G}_T\rangle, \langle \mathrm{e}\rangle, N)$ and $\bar{\pi} = (p_1, \ldots, p_n)$ with $p_1 < \ldots < p_n$ primes; $\mathbb{G}, \mathbb{G}_T$ groups and $\mathrm{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a non-degenerate bilinear map; $p_i \in \{2^{k-1}, \ldots, 2^k - 1\}$ for $1 \le i \le n$; $N = p_1 \cdots p_n = |\mathbb{G}| = |\mathbb{G}_T|$. For $S \subseteq [n]$ we let $\mathbb{G}(S)$ denote the unique subgroup of $\mathbb{G}$ of order $\prod_{i \in S} p_i$. By $\mathbb{H}^*$ we denote the set of generators of a cyclic group $\mathbb{H}$.

The orthogonality property is that if $S_1, S_2 \subseteq [n]$ are disjoint and $g_i \in \mathbb{G}(S_i)$ $(i = 1, 2)$, then $\mathrm{e}(g_1, g_2) = 1_{\mathbb{G}_T}$. Now suppose $S_0, S_1 \subseteq [n]$ and given $T \in \mathbb{G}(S_b)$ we wish to determine $b \in \{0, 1\}$. Orthogonality makes this easy if we possess $g \in \mathbb{G}(S)$ where one of $S \cap S_0, S \cap S_1$ is empty and the other is not. The general subgroup decision assumption is that it is hard without such a g, even if we possess elements of any $G(S)$ for which $S \cap S_0, S \cap S_1$ are both empty or both not empty. Our formalization uses game $\mathrm{GSD}_{\mathsf{Gen}}$ of Figure 7. Adversary $\mathcal{A}$ must make exactly one **Ch** query, consisting of a pair $S_0, S_1 \subseteq [n]$, and this must be its first oracle query. Subsequently it can query **Gen**(S) on any $S \subseteq [n]$ and is allowed multiple queries to this oracle. It terminates by outputting a bit b' and its advantage is

$$\mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{A}) = 2 \cdot \Pr\left[\, \mathrm{GSD}^{\mathcal{A}}_{\mathsf{Gen}} \Rightarrow \mathsf{true} \,\right] - 1 \ .$$

Discussion. Lewko and Waters [27] make several different subgroup decision assumptions about order-n group generators with $n = 3$, and [17] do the same with $n = 4$. Each of these corresponds to a particular choice of S_0, S_1 queried to **Ch**, and particular queries to **Gen**, in our game. (And hence can be formulated without these oracles. We note these papers also make other assumptions, some pertaining to $\mathbb{G}_T$, that we will not need or consider.) Although the authors make only a few specific assumptions, it is apparent that they would be willing to make any "allowed" one in the family, where "allowed" means that the adversary can get elements of $\mathbb{G}(S)$ only as long as $S \cap S_0, S \cap S_1$ are both empty or both not empty. Our aim in formulating GSD has been to make this more transparent, namely, to make the full family of potential choices explicit, thereby generalizing, unifying and explaining subgroup decisions assumptions from [10, 27, 17].

GSD may at first glance look like an "interactive" assumption. It isn't. The value n will be a fixed constant, eg. $n = 3$ for [27] and $n = 4$ for us. The GSD assumption is then just a compact way of stating a constant number —one for each subset $\{S_0, S_1\}$ of $2^{[n]}$ with $S_0, S_1 \neq \emptyset$— of non-interactive assumptions. (By non-interactive we mean the game has only **Initialize** and **Finalize** procedures, no oracles.)

Algorithm Pg:
$(\pi, \bar{\pi}) \leftarrow\!\!\$\ \mathsf{Gen}$
$(\langle\mathbb{G}\rangle, \langle\mathbb{G}_T\rangle, \langle \mathrm{e}\rangle, N) \leftarrow \pi$; $(p_1, p_2, p_3, p_4) \leftarrow \bar{\pi}$
$g_1 \leftarrow\!\!\$\ \mathbb{G}(\{1\})^*$; $g_3 \leftarrow\!\!\$\ \mathbb{G}(\{3\})^*$; $g_4 \leftarrow\!\!\$\ \mathbb{G}(\{4\})^*$
$u_1 \leftarrow\!\!\$\ \mathbb{Z}_N$; $U_1 \leftarrow g_1^{u_1}$; $u_4 \leftarrow\!\!\$\ \mathbb{Z}_N$; $U_4 \leftarrow g_4^{u_4}$
$x_1 \leftarrow\!\!\$\ \mathbb{Z}_N$; $X_1 \leftarrow g_1^{x_1}$; $x_4 \leftarrow\!\!\$\ \mathbb{Z}_N$; $X_4 \leftarrow g_4^{x_4}$
$w_4 \leftarrow\!\!\$\ \mathbb{Z}_N$; $W_4 \leftarrow g_4^{w_4}$; $U_{14} \leftarrow U_1 U_4$
$W_{14} \leftarrow g_1 W_4$; $X_{14} \leftarrow X_1 X_4$
$\mathsf{par} \leftarrow (\pi, U_{14}, X_{14}, W_{14}, g_4)$
$\mathsf{msk} \leftarrow (g_1, U_1, X_1, g_3)$

Algorithm $\mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$: // $\mathsf{id} \in \mathbb{Z}_{2^{4k-4}} \subseteq Z_N$
$((\langle\mathbb{G}\rangle, \langle\mathbb{G}_T\rangle, \langle \mathrm{e}\rangle, N), U_{14}, X_{14}, W_{14}, g_4) \leftarrow \mathsf{par}$
$(g_1, U_1, X_1, g_3) \leftarrow \mathsf{msk}$
$r, r_3, r_3' \leftarrow\!\!\$\ \mathbb{Z}_N$; $K \leftarrow g_1^r g_3^{r_3}$; $K' \leftarrow (U_1^{\mathsf{id}} X_1)^r g_3^{r_3'}$
Return (K, K')

Algorithm $\mathsf{Enc}(\mathsf{par}, \mathsf{id}, M)$:
$((\langle\mathbb{G}\rangle, \langle\mathbb{G}_T\rangle, \langle \mathrm{e}\rangle, N), U_{14}, X_{14}, W_{14}, g_4) \leftarrow \mathsf{par}$
If $M = 0$ then
  $s \leftarrow\!\!\$\ \mathbb{Z}_N$; $t_4, t_4' \leftarrow\!\!\$\ \mathbb{Z}_N$
  $C \leftarrow (U_{14}^{\mathsf{id}} X_{14})^s g_4^{t_4}$; $C' \leftarrow W_{14}^s g_4^{t_4'}$
Else $C, C' \leftarrow\!\!\$\ \mathsf{Sample}_{\mathbb{G}}()$
Return (C, C')

Algorithm $\mathsf{Dec}(\mathsf{par}, (K, K'), (C, C'))$:
$((\langle\mathbb{G}\rangle, \langle\mathbb{G}_T\rangle, \langle \mathrm{e}\rangle, N), U_{14}, X_{14}, W_{14}, g_4) \leftarrow \mathsf{par}$
If $\mathrm{e}(C, K) = \mathrm{e}(C', K')$ then return 0
Else return 1

Fig. 8. Scheme BBoR based on composite order pairing groups.

We don't really need the full strength of GSD. As in previous works, we only need a few special cases, namely a few particular choices of queries S_0, S_1 to **Ch** and queries S to **Gen**. But we feel that stating GSD better elucidates the source of the assumptions, and it will allow more compact assumption and theorem statements.

Scheme Description. For our scheme we require a 4 group generator Gen with the property that the group $\mathbb{G}$ described by the first output of Gen has a PR sampler Sample with failure probability ζ and corresponding inverse sampler Sample^{-1} with reverse failure probability θ. (Appendix A describes how we can instantiate such groups.) The scheme $\mathsf{BBoR} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ associated to a order 4 group generator Gen is shown in Figure 8, where $\mathsf{IdSp} = \mathbb{Z}_{2^{4k-4}}$ and $\mathsf{MsgSp} = \{0, 1\}$. (We use identity space $\mathbb{Z}_{2^{4k-4}}$ since N will vary but will always be at least 2^{4k-4}.) If $(C, C') \in [\mathsf{Enc}(\mathsf{par}, \mathsf{id}, 0)]$ and $(K, K') \in [\mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})]$, then $\mathrm{e}(C, K)$ equals

$$\mathrm{e}((U_{14}^{\mathsf{id}} X_{14})^s g_4^{t_4}, g_1^r g_3^{r_3}) = \mathrm{e}((U_1^{\mathsf{id}} X_1)^s, g_1^r) = \mathrm{e}((U_1^{\mathsf{id}} X_1)^r, g_1^s) = \mathrm{e}((U_1^{\mathsf{id}} X_1)^r g_3^{r_3'}, W_{14}^s g_4^{t_4'}) = \mathrm{e}(C', K')$$

so decryption always succeeds. On the other hand, if $(C, C') \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{par}, \mathsf{id}, 1)$ and $(K, K') \leftarrow\!\!\$\ \mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$ then $\Pr[\,\mathrm{e}(C, K) = \mathrm{e}(C', K')\,] \le \frac{8}{2^{2k}}$ where k is the security parameter associated to Gen.

We claim the scheme is δ-1SPO with $\delta \le 2\theta$. The algorithm $\mathsf{OpenToOne}$ runs Sample^{-1} on each of the two ciphertext components. Each component will give independent reverse sample failure probability of θ. The IND-CPA security of the scheme is captured by the following theorem, proven in Appendix D.

Theorem 3. *Let* Gen *be an order 4 group generator and let the resulting group* $\mathbb{G}$ *be PR-samplable with sampling failure probability* ζ*. Let* $\mathsf{BBoR} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ *the associated IBE scheme defined above. For all adversaries* $\mathcal{A}'$ *making* q ***Extract*** *queries there exists an adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{BBoR}}(\mathcal{A}') \le (9 + 2q) \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}) + 4 \cdot \zeta\ .$$

Adversary $\mathcal{B}$ *makes at most 5 queries to* ***Gen*** *and runs in time at most* $\mathsf{T}(\mathcal{B}) = \mathsf{T}(\mathcal{A}') + \mathcal{O}(q \cdot \mathsf{T}_{\exp}(\mathbb{G}) + q \cdot \mathsf{T}(\gcd))$. □

References

1. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *Advances in Cryptology – EUROCRYPT 2010*. Springer, 2010.
2. Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 1–35, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany.
3. Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 407–424, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany.
4. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany.
5. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.
6. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.
7. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.
8. Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):915–942, 2006.
9. Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
10. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer, Berlin, Germany.
11. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.
12. Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.
13. Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 90–104, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Berlin, Germany.
14. Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648, Philadephia, Pennsylvania, USA, May 22–24, 1996. ACM Press.
15. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany.
16. Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 150–168, Cambridge, MA, USA, February 10–12, 2005. Springer, Berlin, Germany.
17. Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Fully secure anonymous HIBE with short ciphertexts. IACR ePrint Archive Report 2010/197.
18. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *Advances in Crytpology – EUROCRYPT 2010*. Springer, 2010.
19. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363, Cirencester, UK, December 17–19, 2001. Springer, Berlin, Germany.
20. Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 432–450, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Berlin, Germany.
21. Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.
22. Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In *Advances in Cryptology – EUROCRYPT 2010*. Springer, 2010. To Appear.

23. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
24. Brett Hemenway, Benoit Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. IACR ePrint Archive Report 2009/088.
25. Thomas Icart. How to hash into elliptic curves. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 303–316, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.
26. Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 320–339, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.
27. Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479, Zurich, Switzerland, February 9–11, 2010. Springer, Berlin, Germany.
28. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457, Washington, DC, USA, January 7–9, 2001. ACM-SIAM.
29. Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.
30. C. Peikert. Private Communication. May 2010.
31. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.
32. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.
33. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.
34. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.

A Pairing Groups and PR Sampling

The missing piece in our two constructions is which groups to use and how to efficiently instantiate a corresponding PR sampler Sample (and, for our proofs, its inverse Sample^{-1}). Recall that we want to sample random points from a group $\mathbb{G}$ in a publicly-reversible way. In other words, given a group element $g \in \mathbb{G}$, it should be possible with access to only public information to efficiently find coins that explain how g was sampled. This rules out sampling group elements by simply exponentiating with a generator, since explaining such a group element would entail computing a discrete logarithm. Further complicating matters, our schemes require that we use groups with efficiently computable pairings.

To solve these issues, we rely on the techniques for hashing into elliptic curve groups, specifically the techniques from Boneh, Lynn, and Shacham (BLS) [11]. Hashing techniques from [9] and [25] could also potentially be used.

The BLS Hash Function. The BLS hash function has two stages. The first stage hashes into $\mathbb{Z}_q \times \mathbb{Z}_2$ and the second stage maps a value in $\mathbb{Z}_q \times \mathbb{Z}_2$ into a group element. We will only need the second stage. This second stage can then be further broken into two parts. In the first part, choose a random x value and find a corresponding y value on the curve (if such a y exists[4]), resulting in a random point $\tilde{g}$ in the order-m group $\mathbb{H} = E(\mathsf{F}_q)$ of points on the curve. The second part maps $\tilde{g}$ into the order-n subgroup $\mathbb{G}$ of $\mathbb{H}$ by computing $\tilde{g}^{(m/n)}$, where n divides m, and n and m/n are relatively prime.

Consider an elliptic curve with points in $\mathbb{F}_q$, where $\mathbb{F}_q$ is a field with characteristic greater than 2. Let the curve be defined by equation $y^2 = f(x)$ such that the group $E(\mathbb{F}_q)$ is cyclic and has order m and let n be such that $m = n\ell$ for integer ℓ and $\gcd(n, \ell) = 1$. (The supersingular curves used in [9, 10]

[4] Because of the Hasse-Weil bound such a y will exist with probability very close to 1/2.

are one example of curves that would work.) We note that in [11] n is a prime p such that p divides m but p^2 does not divide m, making $\gcd(p, m/p) = 1$. We, however, need to hash into composite order subgroups for the BBoR scheme. As long as n and ℓ are relatively-prime the BLS procedures we describe below still work as required.

Now, to build a hash function that hashes into the subgroup of $E(\mathbb{F}_q)$ of order n (call it $\mathbb{G}$), BLS first hash onto $\mathbb{F}_q \times \{0,1\}$ and then apply a procedure (which we slightly modify and call FqBitToGroup) to that value to try and get a point in $\mathbb{G}$. They then use $\mathbb{G}$ to construct their signature scheme[5]. We now define FqBitToGroup and an inverse procedure.

FqBitToGroup(x, b)
If $x = q + 1$ return $h \leftarrow \mathcal{O}$
If $f(x)$ is not a quadratic residue in $\mathbb{F}_q$ then Return $\perp$
Compute square roots $y_0, y_1 \in \mathbb{F}_q$ of $f(x)$ such that $y_0 < y_1$ by some fixed ordering.
$\tilde{g} \leftarrow (x, y_b)$; $h \leftarrow (\tilde{g})^{\ell}$
Return h

The algorithm has approximately a 1/2 probability of failure. Note that we slightly deviate from BLS in that we do not fail when $h = 1_{\mathbb{G}}$, since we do not need h to be a generator. In fact, we draw values from $\mathbb{Z}_{q+1}$ instead of $\mathbb{Z}_q$ to make sure FqBitToGroup outputs the identity group element with the correct probability. BLS also describe an inverse procedure that, given a group element $h \in \langle g \rangle$, finds a random (x, b) such that FqBitToGroup$(x, b) = h$. In the procedure, let $z = (\ell)^{-1} \mod n$, which exists since $\gcd(n, \ell) = 1$. We will describe the procedure as taking input an element $h \in \mathbb{G}$ and an element $u \in E(\mathbb{F}_q)$ (think of u as the randomness used by the procedure).

FqBitToGroup$^{-1}(h, u)$
$(x, y) = \tilde{h} \leftarrow u^n \cdot h^z$
If $(x, y) = \mathcal{O}$ then $b \leftarrow\!\!\$ \{0,1\}$; return $(q + 1, b)$
Compute square roots y_0, y_1 of $f(x)$, where $y_0 < y_1$
If $y = y_0$ then return $(x, 0)$ else return $(x, 1)$

If u is a uniform point in $E(\mathbb{G}_q)$ then the output (x, b) of FqBitToGroup$^{-1}(h, u)$ is such that x is uniformly distributed in $\mathbb{Z}_{q+1}$ and b is uniform in $\{0, 1\}$, conditioned on FqBitToGroup(x, b) resulting in h.

Our PR Sampler. We can use the above two algorithms to create our PR sampler Sample and its inverse Sample^{-1}. We note that this PR sampler and inverse procedure work for both of our 1SPO schemes, though the underlying groups might be different (and thus the exact values used in subroutine FqBitToGroup).

[5] Actually, they explicitly use asymmetric pairings in their paper and mention symmetric pairings (which we are using in this paper) could be used instead.

$\underline{\mathsf{Sample}}$
$h \leftarrow \perp$
For $i = 1$ to ρ do
  $(x_i \parallel b_i) \leftarrow\!\!\$\ \mathbb{Z}_{q+1} \times \mathbb{Z}_2$
  $h' \leftarrow \mathsf{FqBitToGroup}(x_i, b_i)$
  If $(h' \neq \perp \wedge h = \perp)$ then
    $h \leftarrow h'$
  Continue
Return h

$\underline{\mathsf{Sample}^{-1}(h)}$
$j \leftarrow \perp$
For $i = 1$ to ρ do
  $(x_i \parallel b_i) \leftarrow\!\!\$\ \mathbb{Z}_{q+1} \times \mathbb{Z}_2$
  If $j = \perp \wedge \mathsf{FqBitToGroup}(x_i, b_i) \neq \perp$ then $j \leftarrow i$
If $j = \perp$ then Return $\perp$
Else
  Compute square roots $y_{j,0}, y_{j,1}$ of $f(x_j)$.
  $u \leftarrow (x_j, y_{j,b_j})$
  $(x_j, b_j) \leftarrow\!\!\$\ \mathsf{FqBitToGroup}^{-1}(h, u)$
Return $(x_1, b_1), \ldots, (x_\rho, b_\rho)$

The Sample algorithm has failure probability $\zeta \approx 1/2^\rho$. Notice the set of coins used by Sample is $(\mathbb{Z}_{q+1} \times \mathbb{Z}_2)^\rho$. The inverse operation Sample^{-1} is similar to the way random oracle queries are answered in the BLS proof [11]. Algorithm Sample^{-1} on input h generates a random sequence of values in $\mathbb{Z}_{q+1} \times \mathbb{Z}_2$, finds the first such value that does not make FqBitToGroup fail, and replaces that value in the sequence with one that maps to h. It has failure probability $\theta \approx 1/2^\rho$. Note that we cannot simply invert the point h but also need to simulate all ρ attempts Sample makes, including any failed attempts that precede finding h. If Sample^{-1} does not fail on input $h \in \mathbb{G}$ it outputs correctly-distributed randomness for Sample.

B Proof of Theorem 1

Let $\mathcal{A}$ be an arbitrary soa-adversary against Π^ℓ. We describe a simulator $\mathcal{S}$ for $\mathcal{A}$ in Figure 9. The simulator runs Pg to generate par and msk. It will run $\mathcal{A}$ on input par, answering oracle queries as follows. On oracle query **NewMesg**($\mathbf{id}, \alpha$) from $\mathcal{A}$, $\mathcal{S}$ forwards the query to its own **NewMesg** oracle, receiving nothing in response. $\mathcal{S}$ then generates a vector of ciphertexts, where each ciphertext is an encryption of the all-zero message $\{0,1\}^\ell$, and returns it to the adversary as the output of **NewMesg**. Later, on query **Corrupt**(I) from $\mathcal{A}$, $\mathcal{S}$ queries its own **Corrupt** oracle on I and learns $\mathbf{m}[I]$. For each index $i \in I$ and each $j \in [\ell]$, if $\mathbf{m}[i][j]$ (the jth bit of the ith message) is 0 then $\mathcal{S}$ will return the actual randomness it used to generate $\mathbf{c}[i]$ (in answering the previous **NewMesg** oracle query). If $\mathbf{m}[i][j] = 1$, however, $\mathcal{S}$ will run OpenToOne on the jth component of the ciphertext $\mathbf{c}[i]$ to find coins that can open that component to a 1. In addition to the randomness, of course, $\mathcal{S}$ also returns the messages $\mathbf{m}[I]$. On extract queries from $\mathcal{A}$, $\mathcal{S}$ simply uses msk to answer correctly. Lastly, when $\mathcal{A}$ halts with output *out*, $\mathcal{S}$ halts with the same output.

We prove the theorem through a series of game transitions. Their precise code can be found in Figures 9,10, and 11. The transitions are summarized as follows.

- G_0: The IBESOAREAL game.
- G_1: Change **Corrupt** procedure to resample coins before opening.
- G_2: Replace resampling from G_1 with call to OpenToOne. If OpenToOne fails, set bad flag and resample as in G_1.
- G_3: Same as above except if OpenToOne fails do not do any other resampling.
- H_v: The first v bits sampled from $\mathcal{M}$ (possibly across many messages) are ignored by **NewMesg** and replaced by 0s. However, **Corrupt** still uses OpenToOne to open ciphertexts to 1 depending on $\mathcal{M}$.
- $H_{k\ell}$: all messages from $\mathcal{M}$ are completely ignored by **NewMesg** and only all-0 messages are encrypted.
- G_4: Since **NewMesg** ignores the messages sampled from $\mathcal{M}$, the message sampling is moved to the **Corrupt** procedure.

alg.$\mathcal{S}$() :

$(\mathsf{par}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Pg}^{\ell}$
Run $\mathcal{A}(\mathsf{par})$.

On query **NewMesg**$(\mathbf{id}, \alpha)$:
$\perp \leftarrow \mathbf{NewMesg}_S(\mathbf{id}, \alpha)$
For i in 1 to k
  $\mathbf{r}[i] \leftarrow \mathsf{Coins}(\mathsf{par}, 0^{\ell})$
  $\mathbf{c}[i] \leftarrow \mathsf{Enc}^{\ell}(\mathsf{par}, \mathbf{id}[i], 0^{\ell}; \mathbf{r}[i])$
Return $\mathbf{c}$

On query **Corrupt**(I):
$\mathbf{m}[I] \leftarrow \mathbf{Corrupt}_S(I)$
For $i \in I$
  For j in 1 to ℓ
    if $\mathbf{m}[i][j] = 1$ then
    $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{OpenToOne}(\mathsf{par}, \mathbf{id}[i], \mathbf{c}[i][j])$
Return $(\mathbf{r}[I], \mathbf{m}[I])$

On query **Extract**(id):
Return $\mathsf{Kg}^{\ell}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$

When $\mathcal{A}$ halts with output *out*, halt and output *out*.

proc. Initialize: All Games
$(\mathsf{par}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Pg}^{\ell}$
Return par

proc. NewMesg$(\mathbf{id}, \alpha)$: G_0, G_1, G_2, G_3
If $\mathbf{id} \cap \mathrm{ExID} \neq \emptyset$ then return $\perp$
$\mathrm{ChID} \leftarrow \mathrm{ChID} \cup \mathbf{id}$
$\mathbf{m} \leftarrow\!\!\$\ \mathcal{M}(\alpha)$
For i in 1 to k
  For j in 1 to ℓ
    $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{Coins}(\mathsf{par}, \mathbf{m}[i][j])$
    $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(\mathsf{par}, \mathbf{id}[i], \mathbf{m}[i][j]; \mathbf{r}[i][j])$
Return $\mathbf{c}$

proc. Corrupt(I): G_0
Return $\mathbf{r}[I]$, $\mathbf{m}[I]$

proc. Extract(id): All Games
Return $\mathsf{Kg}^{\ell}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$

proc. Finalize(out): All Games
Return $\mathcal{R}(\mathbf{m}, \mathrm{ChID}, I, out)$

Fig. 9. Simulator $\mathcal{S}$ and the start of the game sequence.

More formally, we first claim that

$$\Pr\left[\,\mathrm{IBESOAREAL}^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] = \Pr\left[\,G_0^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] .$$

Next, we claim

$$\Pr\left[\,G_0^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] = \Pr\left[\,G_1^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] ,$$

since from the coins returned in Corrupt are identically distributed given the view of $\mathcal{A}$. Next, we can see that

$$\Pr\left[\,G_1^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] = \Pr\left[\,G_2^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] ,$$

since by definition when $\mathsf{OpenToOne}$ does not fail its output is identically distributed to as in G_1, and when it does fail G_1 ignores its output and resamples as in G_1. Now, the Fundamental Lemma of game playing justifies

$$\Pr\left[\,G_2^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] - \Pr\left[\,G_3^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] \leq \Pr\left[\,\mathsf{BD}(G_2^{\mathcal{A}})\,\right] ,$$

and since the probability that any execution of $\mathsf{OpenToOne}$ fails is at most δ (over just the coins of $\mathsf{OpenToOne}$) and there are at most $\ell \cdot k$ bits that have to be opened by **Corrupt**, we see that

$$\Pr\left[\,\mathsf{BD}(G_2^{\mathcal{A}})\,\right] \leq k\ell\delta .$$

Next, H_0 is just a rewriting of G_3. Next, we claim that

$$\Pr\left[\,H_0^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] - \Pr\left[\,H_{k\ell}^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] \leq k\ell \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(\mathcal{B}) , \tag{2}$$

where adversary $\mathcal{B}$ is described in Figure 12. To justify this claim, let $\mathsf{M0}_{v+1}(H_{v'}^{\mathcal{A}})$ be the event that in the execution of $H_{v'}^{\mathcal{A}}$ the $v+1$st bit sampled by $\mathcal{M}$ in **NewMesg** is a 0. The complement event is denoted by $\mathsf{M1}$. Notice that in the case that the event that the $v+1$st bit sampled in H_v is a 0, the games H_v and H_{v+1} are identical, since H_v will encrypt the actual $v+1$st bit (which is a 0 since the event is true) and H_{v+1} will ignore the actual bit and encrypt a 0. In both cases, 0 is encrypted. Also notice that in both H_v and H_{v+1} the message sampled is independent of whether it is game v or $v+1$

proc. Corrupt(I): G_1
For i in I
  For j in 1 to ℓ
    If $\mathbf{m}[i][j] = 1$ then
      $\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{Coins}(\mathsf{par}, \mathbf{id}[i], \mathbf{c}[i][j], 1)$
Return $\mathbf{r}[I]$, $\mathbf{m}[I]$

proc. NewMesg$(\mathbf{id}, \alpha)$: H_v
If $\mathbf{id} \cap \mathrm{ExID} \neq \emptyset$ then return $\perp$
$\mathrm{ChID} \leftarrow \mathrm{ChID} \cup \mathbf{id}$; $\mathbf{m} \leftarrow\!\$\ \mathcal{M}(\alpha)$
For i in 1 to k
  For j in 1 to ℓ
    if $((i-1)\cdot \ell + j) \leq v$ then
      $\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{Coins}(\mathsf{par}, 0)$
      $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(\mathsf{par}, \mathbf{id}[i], 0; \mathbf{r}[i][j])$
    Else
      $\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{Coins}(\mathsf{par}, \mathbf{m}[i][j])$
      $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(\mathsf{par}, \mathbf{id}[i], \mathbf{m}[i][j]; \mathbf{r}[i][j])$
Return $\mathbf{c}$

proc. Corrupt(I): H_v
For i in I
  For j in 1 to ℓ
    If $\mathbf{m}[i][j] = 1$ then
      $\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{OpenToOne}(\mathsf{par}, \mathbf{id}[i], \mathbf{c}[i][j])$
Return $\mathbf{r}[I]$, $\mathbf{m}[I]$

proc. Corrupt(I): $\boxed{G_2}$, G_3
For i in I
  For j in 1 to ℓ
    If $\mathbf{m}[i][j] = 1$ then
      $\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{OpenToOne}(\mathsf{par}, \mathbf{id}[i], \mathbf{c}[i][j])$
      If $\mathbf{r}[i][j] = \perp$ then
        $\mathsf{bad} \leftarrow \mathsf{true}$
        $\boxed{\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{Coins}(\mathsf{par}, \mathbf{id}[i], \mathbf{c}[i][j], 1)}$
Return $\mathbf{r}[I]$, $\mathbf{m}[I]$

proc. NewMesg$(\mathbf{id}, \alpha)$: $H_{k\ell}$
If $\mathbf{id} \cap \mathrm{ExID} \neq \emptyset$ then return $\perp$
$\mathrm{ChID} \leftarrow \mathrm{ChID} \cup \mathbf{id}$; $\mathbf{m} \leftarrow\!\$\ \mathcal{M}(\alpha)$
For i in 1 to k
  For j in 1 to ℓ
    $\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{Coins}(\mathsf{par}, 0)$
    $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(\mathsf{par}, \mathbf{id}[i], 0; \mathbf{r}[i][j])$
Return $\mathbf{c}$

proc. Corrupt(I): $H_{k\ell}$
For i in I
  For j in 1 to ℓ
    If $\mathbf{m}[i][j] = 1$ then
      $\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{OpenToOne}(\mathsf{par}, \mathbf{id}[i], \mathbf{c}[i][j])$
Return $\mathbf{r}[I]$, $\mathbf{m}[I]$

Fig. 10. Games for the proof of Theorem 1.

proc. NewMesg$(\mathbf{id}, \alpha)$:
If $\mathbf{id} \cap \mathrm{ExID} \neq \emptyset$ then return $\perp$
$\mathrm{ChID} \leftarrow \mathrm{ChID} \cup \mathbf{id}$
For i in 1 to k
  For j in 1 to ℓ
    $\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{Coins}(\mathsf{par}, 0)$
    $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(\mathsf{par}, \mathbf{id}[i], 0; \mathbf{r}[i][j])$
Return $\mathbf{c}$

proc. Corrupt(I): G_4
$\mathbf{m} \leftarrow\!\$\ \mathcal{M}(\alpha)$
For i in I
  For j in 1 to ℓ
    If $\mathbf{m}[i][j] = 1$ then
      $\mathbf{r}[i][j] \leftarrow\!\$\ \mathsf{OpenToOne}(\mathsf{par}, \mathbf{id}[i], \mathbf{c}[i][j])$
Return $\mathbf{r}[I]$, $\mathbf{m}[I]$

Fig. 11. The last game G_4.

since v is not used until after the message has been sampled. Thus, $\mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v+1}) = \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v})$.

$$
\begin{aligned}
&\Pr\left[\, H^{\mathcal{A}}_{v} \,\right] - \Pr\left[\, H^{\mathcal{A}}_{v+1} \,\right] \\
&= \left(\Pr\left[\, H^{\mathcal{A}}_{v} \wedge \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] + \Pr\left[\, H^{\mathcal{A}}_{v} \wedge \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v}) \,\right]\right) - \\
&\quad \left(\Pr\left[\, H^{\mathcal{A}}_{v+1} \wedge \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v+1}) \,\right] + \Pr\left[\, H^{\mathcal{A}}_{v+1} \wedge \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v+1}) \,\right]\right) \\
&= \Pr\left[\, \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] \cdot \left(\Pr\left[\, H^{\mathcal{A}}_{v} \mid \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] - \Pr\left[\, H^{\mathcal{A}}_{v+1} \mid \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v+1}) \,\right]\right) + \\
&\quad \Pr\left[\, \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] \cdot \left(\Pr\left[\, H^{\mathcal{A}}_{v} \mid \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] - \Pr\left[\, H^{\mathcal{A}}_{v+1} \mid \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v+1}) \,\right]\right) \ . \qquad (3)
\end{aligned}
$$

Now as we said above, in the event $\mathsf{M0}_{v+1}$ both H_{v+1} and H_v are identical, thus the first term in (3) becomes zero. This means that,

$$
\begin{aligned}
&\Pr\left[\, H^{\mathcal{A}}_{v} \,\right] - \Pr\left[\, H^{\mathcal{A}}_{v+1} \,\right] \\
&= \Pr\left[\, \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] \cdot \left(\Pr\left[\, H^{\mathcal{A}}_{v} \mid \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] - \Pr\left[\, H^{\mathcal{A}}_{v+1} \mid \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v+1}) \,\right]\right) \ , \qquad (4)
\end{aligned}
$$

$\underline{\textbf{alg.}\mathcal{B}(\mathsf{par})}$:
$v \leftarrow\!\!\$\ \{0,\ldots,k\ell-1\}$
Run $\mathcal{A}(\mathsf{par})$.

On query $\mathbf{NewMesg}(\mathbf{id},\alpha)$:
$\mathbf{m} \leftarrow\!\!\$\ \mathcal{M}(\alpha)$
For i in 1 to k
  For j in 1 to ℓ
    if $((i-1)\cdot\ell+j) \le v$ then
      $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{Coins}(\mathsf{par},0)$; $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(\mathsf{par},\mathbf{id}[i],0;\mathbf{r}[i][j])$
    Else if $((i-1)\cdot\ell+j) = v+1$ then
      If $\mathbf{m}[i][j]=1$ then $\mathbf{c}[i][j] \leftarrow \mathbf{LR}_{\mathcal{B}}(\mathbf{id}[i],0,1)$
      Else
        $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{Coins}(\mathsf{par},\mathbf{m}[i][j])$; $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(\mathsf{par},\mathbf{id}[i],\mathbf{m}[i][j];\mathbf{r}[i][j])$
    Else
      $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{Coins}(\mathsf{par},\mathbf{m}[i][j])$; $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(\mathsf{par},\mathbf{id}[i],\mathbf{m}[i][j];\mathbf{r}[i][j])$
Return $\mathbf{c}$

On query $\mathbf{Corrupt}(I)$:
$\mathbf{m}[I] \leftarrow \mathbf{Corrupt}_S(I)$
For $i \in I$
  For j in 1 to ℓ
    if $\mathbf{m}[i][j]=1$ then $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{OpenToOne}(\mathsf{par},\mathbf{id}[i],\mathbf{c}[i][j])$
Return $(\mathbf{r}[I],\mathbf{m}[I])$

On query $\mathbf{Extract}(\mathsf{id})$:
Return $\mathbf{Extract}_{\mathcal{B}}(\mathsf{id})$

When $\mathcal{A}$ halts with output out, halt and output $\mathcal{R}(\mathbf{m},\mathrm{ChID},I,out)$.

Fig. 12. IND-CPA adversary $\mathcal{B}$

and thus

$$\Pr\left[\,H_0^{\mathcal{A}}\,\right] - \Pr\left[\,H_{k\ell}^{\mathcal{A}}\,\right] = \sum_{v=0}^{k\ell-1} \Pr\left[\,H_v^{\mathcal{A}}\,\right] - \Pr\left[\,H_{v+1}^{\mathcal{A}}\,\right]$$
$$= \sum_{v=0}^{k\ell-1} \Pr\left[\,\mathsf{M1}_{v+1}(H_v^{\mathcal{A}})\,\right] \cdot \left(\Pr\left[\,H_v^{\mathcal{A}} \mid \mathsf{M1}_{v+1}(H_v^{\mathcal{A}})\,\right] - \Pr\left[\,H_{v+1}^{\mathcal{A}} \mid \mathsf{M1}_{v+1}(H_{v+1}^{\mathcal{A}})\,\right]\right)\ , \tag{5}$$

Consider again IND-CPA adversary $\mathcal{B}$ in Figure 12. The adversary picks a random integer v and runs $\mathcal{A}$ while simulating its environment as in either H_v or H_{v-1}, depending on whether its **LR** oracle encrypts the left or right message, respectively. Notice that $\mathcal{B}$ only uses its **LR** oracle in the event $\mathsf{M1}_{v+1}$, and thus all of its advantage comes from this case. It is thus easy to see that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(\mathcal{B}) = \frac{1}{k\ell}\sum_{v=0}^{k\ell-1} \Pr\left[\,\mathsf{M1}_{v+1}(H_v^{\mathcal{A}})\,\right] \cdot \left(\Pr\left[\,H_v^{\mathcal{A}} \mid \mathsf{M1}_{v+1}(H_v^{\mathcal{A}})\,\right] - \Pr\left[\,H_{v+1}^{\mathcal{A}} \mid \mathsf{M1}_{v+1}(H_{v+1}^{\mathcal{A}})\,\right]\right)\ ,$$

which combined with (5) justifies (2).

Next, we see that

$$\Pr\left[\,H_{k\ell}^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] = \Pr\left[\,G_4^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right]\ ,$$

which is true since G_4 is identical to $H_{k\ell}$ except the message sampling is moved to **Corrupt**. This can be done since the messages sampled in **NewMesg** are completely ignored in $H_{k\ell}$ (only 0s are encrypted).

Finally, the simulator $\mathcal{S}$ described in Figure 9 can run identically to G_4 (where it learns $\mathbf{m}[I]$ through a **Corrupt** query instead of sampling as in G_4), so we have that

$$\Pr\left[\, G_4^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] = \Pr\left[\, \text{IBESOASIM}^{\mathcal{S}} \Rightarrow \mathsf{true} \,\right] .$$

Combining all of the above equations we get that

$$\Pr\left[\, \text{IBESOAREAL}^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] - \Pr\left[\, \text{IBESOASIM}^{\mathcal{S}} \Rightarrow \mathsf{true} \,\right] \leq k\ell \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(\mathcal{B}) + k\ell\delta ,$$

which proves the theorem. □

C Proof of Theorem 2

In this section, we prove Theorem 2, establishing the IND-CPA security of our scheme LoR based on the decisional linear assumption. We will closely follow the proof of security from [3] which gave an alternative proof to Waters' IBE scheme [34]. Consider the games in Figures 13,14, and 15. We have

$$\Pr\left[\, G_2^{\mathcal{A}} \,\right] - \Pr\left[\, G_3^{\mathcal{A}} \,\right] \leq \Pr\left[\, \mathsf{BAD}(G_3^{A}) \,\right] \tag{6}$$

$$\leq \frac{2^n}{p} \tag{7}$$

Games G_2, G_3 are identical until bad so (6) is by the Fundamental Lemma of game playing [4]. To justify (7) first note $H(\mathbf{u}, \mathsf{id}) = g^{t_1 \mathsf{F}(\mathbf{x},\mathsf{id}) + \mathsf{G}(\mathbf{y},\mathsf{id})}$. Thus, the event $\mathsf{BD}(G_3^{\mathcal{A}})$ happens when $t_1\mathsf{F}(\mathbf{x},\mathsf{id}) + \mathsf{G}(\mathbf{y},\mathsf{id}) \equiv 0 \pmod p$. Fix t_1 and $\mathbf{x}$. For any particular id, the probability over $\mathbf{y}$ that $\mathsf{G}(\mathbf{y},\mathsf{id}) + \mathsf{F}(\mathbf{x},\mathsf{id})t_1 \equiv 0 \pmod p$ is $1/p$. But the number of choices of id is 2^n so we conclude by the union bound. Now we have

$$\begin{aligned}
\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{LoR}}(A) &\leq 2\left(\Pr\left[\, G_0^{\mathcal{A}} \,\right] + 5\zeta\right) - 1 \\
&= 2\left(\Pr\left[\, G_0^{A} \,\right] - \frac{1}{2}\right) + 10\zeta \\
&= 2\Bigg(\Pr\left[\, G_0^{A} \,\right] - \Pr\left[\, G_1^{A} \,\right] + \Pr\left[\, G_1^{A} \,\right] - \Pr\left[\, G_2^{A} \,\right] + \Pr\left[\, G_2^{A} \,\right] - \Pr\left[\, G_3^{A} \,\right] + \Pr\left[\, G_3^{A} \,\right] - \frac{1}{2}\Bigg) + 10\zeta
\end{aligned}$$

However $\Pr\left[\, G_3^{A} \,\right] = \frac{1}{2}$. Using this and (7) we have

$$\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(A) \leq 2\epsilon_1 + 2\epsilon_2 + \frac{2^{n+1}}{p} + 10\zeta$$

where

$$\epsilon_1 = \Pr\left[\, G_0^{A} \,\right] - \Pr\left[\, G_1^{A} \,\right] \text{ and } \epsilon_2 = \Pr\left[\, G_1^{A} \,\right] - \Pr\left[\, G_2^{A} \,\right]$$

We consider two cases. The first is when

$$\epsilon_1 \geq \frac{1}{2}\left(\frac{\epsilon}{2} - \frac{2^n}{p} - 5\zeta\right)$$

and the second is when

$$\epsilon_2 \geq \frac{1}{2}\left(\frac{\epsilon}{2} - \frac{2^n}{p} - 5\zeta\right) .$$

In the first case we design $\mathcal{B}_1$ so that

$$\mathbf{Adv}^{\text{dlin}}_{\mathsf{GP}}(\mathcal{B}_1) \geq \frac{\epsilon_1^2}{9qn + 3\epsilon_1} \tag{8}$$

and in the second case we design $\mathcal{B}_2$ so that

$$\mathbf{Adv}^{\text{dlin}}_{\mathsf{GP}}(\mathcal{B}_2) \geq \frac{\epsilon_2^2}{9qn + 3\epsilon_2} .$$

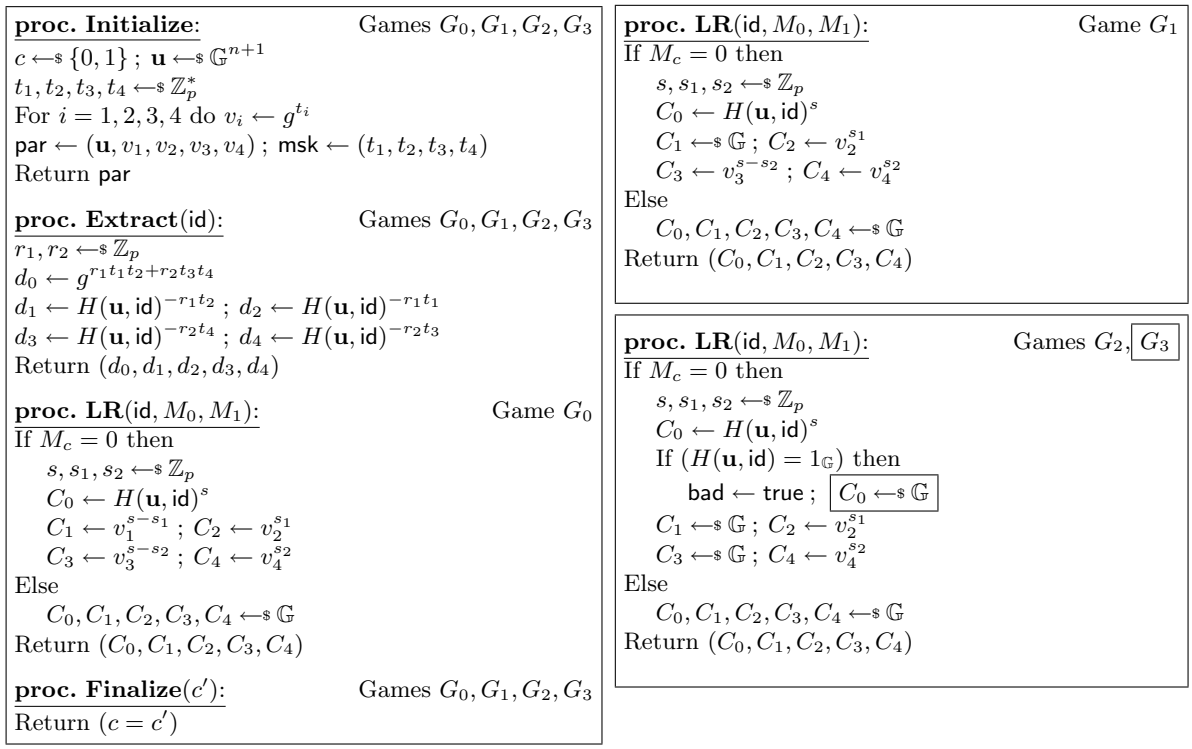

Fig. 13. Game transitions

The adversary $\mathcal{B}$ of the theorem statement is either $\mathcal{B}_1$ or $\mathcal{B}_2$ depending which case is true. In the sequel we describe $\mathcal{B}_1$ and sketch the proof of (8) based on [12, 3]. The design and analysis of $\mathcal{B}_2$ is similar and omitted.

Simulation subroutines. Towards the proof we begin with some definitions from [3]. Let $m = \lceil 3q/\epsilon_1 \rceil$ and let $X = [-n(m-1)\,..\,0] \times [0\,..\,m-1] \times \cdots \times [0\,..\,m-1]$ where the number of copies of $[0\,..\,m-1]$ is n. For $\mathbf{x} \in X$, $\mathbf{y} \in \mathbb{Z}_p^{n+1}$ and $\mathsf{id} \in \{0,1\}^n$ we let

$$\mathsf{F}(\mathbf{x}, \mathsf{id}) = \mathbf{x}[0] + \sum_{i=1}^{n} \mathbf{x}[i]\mathsf{id}[i] \quad \text{and} \quad \mathsf{G}(\mathbf{y}, \mathsf{id}) = \mathbf{y}[0] + \sum_{i=1}^{n} \mathbf{y}[i]\mathsf{id}[i] \bmod p\,. \tag{9}$$

In the above, the computation of G is over $\mathbb{Z}_p$, while the computation of F is over $\mathbb{Z}$. Adversary $\mathcal{B}_1$ is shown in Figure 14. We now describe the subroutines it utilizes to answer **Extract** and **LR** queries. We define the following procedures:

Adversary $\mathcal{B}_1(g, v_1, v_2, C_1, C_2, W)$:
$c \leftarrow\$ \{0,1\}$
$t_3, t_4 \leftarrow\$ \mathbb{Z}_p^*$; $v_3 \leftarrow g^{t_3}$; $v_4 \leftarrow g^{t_4}$

For $j = 0, \ldots, n$ do
  $\mathbf{y}[j] \leftarrow\$ \mathbb{Z}_p$
  If $j = 0$ then $\mathbf{x}[j] \leftarrow\$ [-n(m-1) \,..\, 0]$
  Else $\mathbf{x}[j] \leftarrow\$ [0 \,..\, m-1]$
  $\mathbf{u}[j] \leftarrow v_1^{\mathbf{x}[j]} g^{\mathbf{y}[j]}$
$\mathsf{par} \leftarrow (g, \mathbf{u}, v_1, v_2, v_3, v_4)$
Run $\mathcal{A}(\mathsf{par})$, answering queries by

On **query Extract**(id):
$sk(\mathsf{id}) \leftarrow \bot$
If $\mathsf{F}(\mathbf{x}, \mathsf{id}) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else $sk(\mathsf{id}) \leftarrow\$ \mathsf{KgS}(g, v_1, v_2, t_3, t_4, \mathbf{u}, \mathbf{x}, \mathbf{y}, \mathsf{id})$
Return $sk(\mathsf{id})$

On **query LR**(id, M_0, M_1):
$C \leftarrow \bot$
If $\mathsf{F}(\mathbf{x}, \mathsf{id}) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else
  $C \leftarrow\$ \mathsf{EncS}(g, C_1, C_2, v_3, v_4, t_3, W, \mathbf{y}, \mathsf{id}, M_c)$
Return C

$\mathcal{A}$ finishes, returning bit c'
If $\mathsf{bad} = \mathsf{true}$ then $c' \leftarrow\$ \{0,1\}$
If $c = c'$ then return 1 else return 0

proc. Initialize: Games G_4, G_5, G_6
$c, d \leftarrow\$ \{0,1\}$; $t_1, t_2, t_3, t_4 \leftarrow\$ \mathbb{Z}_p^*$; $s, s_1, s_2 \leftarrow\$ \mathbb{Z}_p$
$v_1 \leftarrow g^{t_1}$; $v_2 \leftarrow g^{t_2}$; $v_3 \leftarrow g^{t_3}$; $v_4 \leftarrow g^{t_4}$
$C_1 \leftarrow v_1^{s-s_1}$; $C_2 \leftarrow v_2^{s_1}$
If $d = 1$ then $W \leftarrow g^s$ else $W \leftarrow\$ \mathbb{G}$
For $j = 0, \ldots, n$ do
  $\mathbf{y}[j] \leftarrow\$ \mathbb{Z}_p$
  If $j = 0$ then $\mathbf{x}[j] \leftarrow\$ [-n(m-1) \,..\, 0]$
  Else $\mathbf{x}[j] \leftarrow\$ [0 \,..\, m-1]$
  $\mathbf{u}[j] \leftarrow v_1^{\mathbf{x}[j]} g^{\mathbf{y}[j]}$
$\mathsf{par} \leftarrow (g, \mathbf{u}, v_1, v_2, v_3, v_4)$; $\mathsf{msk} \leftarrow (t_1, t_2, t_3, t_4)$
Return par

proc. Extract(id): Games G_4, $\boxed{G_5}$
$sk(\mathsf{id}) \leftarrow \bot$
If $\mathsf{F}(\mathbf{x}, \mathsf{id}) = 0$ then
  $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{sk(\mathsf{id}) \leftarrow\$ \mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})}$
Else $sk(\mathsf{id}) \leftarrow\$ \mathsf{KgS}(g, v_1, v_2, t_3, t_4, \mathbf{u}, \mathbf{x}, \mathbf{y}, \mathsf{id})$
Return $sk(\mathsf{id})$

proc. LR(id, M_0, M_1): Games G_4, $\boxed{G_5}$
$C \leftarrow \bot$
If $\mathsf{F}(\mathbf{x}, \mathsf{id}) \neq 0$ then
  $\mathsf{bad} \leftarrow \mathsf{true}$

  If $M_c = 0$ then
    $t \leftarrow\$ \mathbb{Z}_p$; If $d = 1$ then $t \leftarrow s$
    $C_0 \leftarrow H(\mathbf{u}, \mathsf{id})^t$
    $C_3 \leftarrow v_3^{s-s_2}$; $C_4 \leftarrow v_4^{s_2}$
  Else
    $C_0, C_3, C_4 \leftarrow\$ \mathbb{G}$
  $C \leftarrow (C_0, C_1, C_2, C_3, C_4)$

Else $C \leftarrow\$ \mathsf{EncS}(g, C_1, C_2, v_3, v_4, t_3, W, \mathbf{y}, \mathsf{id}, M_c)$
Return C

proc. Finalize(c'): Games G_4, $\boxed{G_5}$
$c'' \leftarrow c'$
If $\mathsf{bad} = \mathsf{true}$ then $c'' \leftarrow\$ \{0,1\}$; $\boxed{c'' \leftarrow c'}$
If $(c = c'')$ then return 1 else return 0

Fig. 14. Adversary $\mathcal{B}$ and the continuation of the game sequence.

proc. $\mathsf{KgS}(g, v_1, v_2, t_3, t_4, \mathbf{u}, \mathbf{x}, \mathbf{y}, \mathsf{id})$:
$r_1, r_2 \leftarrow\$ \mathbb{Z}_p$
$d_0 \leftarrow g^{r_2 t_3 t_4} v_2^{r_1}$
If $H(\mathbf{u}, \mathsf{id}) \neq 1_{\mathbb{G}}$ then
  $d_1 \leftarrow v_2^{-r_1 \cdot \mathsf{F}(\mathbf{x}, \mathsf{id})}$
  $d_2 \leftarrow v_1^{-r_1 \cdot \mathsf{F}(\mathbf{x}, \mathsf{id})}$
  $d_3 \leftarrow H(\mathbf{u}, \mathsf{id})^{-r_2 t_4} v_2^{-r_1 \cdot \mathsf{G}(\mathbf{y}, \mathsf{id})/t_3}$
  $d_4 \leftarrow H(\mathbf{u}, \mathsf{id})^{r_2 t_3} v_2^{-r_1 \cdot \mathsf{G}(\mathbf{y}, \mathsf{id})/t_4}$
Else
  $d_1, d_2, d_3, d_4 \leftarrow 1_{\mathbb{G}}$
Ret $(d_0, d_1, d_2, d_3, d_4)$

proc. $\mathsf{EncS}(g, C_1, C_2, v_3, v_4, t_3, W, \mathbf{y}, \mathsf{id}, M)$:
If $M = 0$ then
  $s_2 \leftarrow\$ \mathbb{Z}_p$
  $C_0 \leftarrow W^{\mathsf{G}(\mathbf{y}, \mathsf{id})}$
  $C_3 \leftarrow W^{t_3} v_3^{-s_2}$
  $C_4 \leftarrow v_4^{s_2}$
Else
  $C_0, C_3, C_4 \leftarrow\$ \mathbb{G}$
Return $(C_0, C_1, C_2, C_3, C_4)$

proc. Extract(id): Game G_6
If $\mathsf{F}(\mathbf{x}, \mathsf{id}) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
$sk(\mathsf{id}) \leftarrow\!\!\$\ \mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$
Return $sk(\mathsf{id})$

proc. LR(id, M_0, M_1): Game G_6
If $\mathsf{F}(\mathbf{x}, \mathsf{id}) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $M_c = 0$ then
  $t \leftarrow\!\!\$\ \mathbb{Z}_p$; If $d = 1$ then $t \leftarrow s$
  $C_0 \leftarrow H(\mathbf{u}, \mathsf{id})^t$
  $C_3 \leftarrow v_3^{s-s_2}$; $C_4 \leftarrow v_4^{s_2}$
Else
  $C_0, C_3, C_4 \leftarrow\!\!\$\ \mathbb{G}$
$C \leftarrow (C_0, C_1, C_2, C_3, C_4)$
Return $(C_0, C_1, C_2, C_3, C_4)$

proc. Finalize(c'): Game G_6
If $(c = c')$ then return 1 else return 0

proc. Initialize: Game G_7
$c, d \leftarrow\!\!\$\ \{0,1\}$; $cnt \leftarrow 0$
$t_1, t_2, t_3, t_4 \leftarrow\!\!\$\ \mathbb{Z}_p^*$; $s, s_1, s_2 \leftarrow\!\!\$\ \mathbb{Z}_p$
$v_1 \leftarrow g^{t_1}$; $v_2 \leftarrow g^{t_2}$; $v_3 \leftarrow g^{t_3}$; $v_4 \leftarrow g^{t_4}$
$C_1 \leftarrow v_1^{s-s_1}$; $C_2 \leftarrow v_2^{s_1}$
If $d = 1$ then $W \leftarrow g^s$ else $W \leftarrow\!\!\$\ \mathbb{G}$
For $j = 0, \ldots, n$ do
  $\mathbf{z}[j] \leftarrow\!\!\$\ \mathbb{Z}_p$; $\mathbf{u}[j] \leftarrow g^{\mathbf{z}[j]}$
  If $j = 0$ then $\mathbf{x}[j] \leftarrow\!\!\$\ [-n(m-1)\,..\,0]$
  Else $\mathbf{x}[j] \leftarrow\!\!\$\ [0\,..\,m-1]$
  $\mathbf{y}[j] \leftarrow \mathbf{z}[j] - t_1 \cdot \mathbf{x}[j] \mod p$
$\mathsf{par} \leftarrow (g, \mathbf{u}, v_1, v_2, v_3, v_4)$
$\mathsf{msk} \leftarrow (t_1, t_2, t_3, t_4)$
Return par

proc. Finalize(c'): Game G_7
For $j = 0$ to cnt do
  If $\mathsf{F}(\mathbf{x}, \mathsf{id}_j) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $\mathsf{F}(\mathbf{x}, \mathsf{id}_0) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $(c = c')$ then return 1 else return 0

proc. Initialize: Game G_8
$c, d \leftarrow\!\!\$\ \{0,1\}$; $cnt \leftarrow 0$
$t_1, t_2, t_3, t_4 \leftarrow\!\!\$\ \mathbb{Z}_p^*$; $s, s_1, s_2 \leftarrow\!\!\$\ \mathbb{Z}_p$
$v_1 \leftarrow g^{t_1}$; $v_2 \leftarrow g^{t_2}$; $v_3 \leftarrow g^{t_3}$; $v_4 \leftarrow g^{t_4}$
$C_1 \leftarrow v_1^{s-s_1}$; $C_2 \leftarrow v_2^{s_1}$
If $d = 1$ then $W \leftarrow g^s$ else $W \leftarrow\!\!\$\ \mathbb{G}$
For $j = 0, \ldots, n$ do
  $\mathbf{z}[j] \leftarrow\!\!\$\ \mathbb{Z}_p$; $\mathbf{u}[j] \leftarrow g^{\mathbf{z}[j]}$
$\mathsf{par} \leftarrow (g, \mathbf{u}, v_1, v_2, v_3, v_4)$
$\mathsf{msk} \leftarrow (t_1, t_2, t_3, t_4)$
Return par

proc. Extract(id): Games G_7, G_8
$cnt \leftarrow cnt + 1$; $\mathsf{id}_{cnt} \leftarrow \mathsf{id}$
$sk(\mathsf{id}) \leftarrow\!\!\$\ \mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$
Return $sk(\mathsf{id})$

proc. LR(id, M_0, M_1): Game G_7, G_8
$\mathsf{id}_0 \leftarrow \mathsf{id}$
If $M_c = 0$ then
  $t \leftarrow\!\!\$\ \mathbb{Z}_p$; If $d = 1$ then $t \leftarrow s$
  $C_0 \leftarrow H(\mathbf{u}, \mathsf{id})^t$
  $C_3 \leftarrow v_3^{s-s_2}$; $C_4 \leftarrow v_4^{s_2}$
Else
  $C_0, C_3, C_4 \leftarrow\!\!\$\ \mathbb{G}$
$C \leftarrow (C_0, C_1, C_2, C_3, C_4)$
Return $(C_0, C_1, C_2, C_3, C_4)$

proc. Finalize(c'): Game G_8
For $j = 0$ to cnt do
For $j = 0, \ldots, n$ do
  If $j = 0$ then $\mathbf{x}[j] \leftarrow\!\!\$\ [-n(m-1)\,..\,0]$
  Else $\mathbf{x}[j] \leftarrow\!\!\$\ [0\,..\,m-1]$
  If $\mathsf{F}(\mathbf{x}, \mathsf{id}_j) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $\mathsf{F}(\mathbf{x}, \mathsf{id}_0) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $(c = c')$ then return 1 else return 0

Fig. 15. Game transitions

The next lemma captures a fact about the simulation subroutine KgS, which we will use in our analysis. It is obtained by adapting a lemma in [12] for use with the Waters' hash.

Lemma 1. *Suppose* $g, v_1, v_2, v_3, v_4 \in \mathbb{G}^*$, $\mathsf{id} \in \{0,1\}^n$, $W \in \mathbb{G}_T$, $\mathbf{x} \in X$, $\mathbf{y} \in \mathbb{Z}_p^{n+1}$, $\mathbf{u}[j] = v_1^{\mathbf{x}[j]} g^{\mathbf{y}[j]}$ *for* $0 \leq j \leq n$. *Further, suppose* $\mathsf{F}(\mathbf{x}, \mathsf{id}) \neq 0$ *and let* $t_i = \log_g(v_i)$ *for* $1 \leq i \leq 4$. *Let* $\mathsf{par} = (g, \mathbf{u}, v_1, v_2, v_3, v_4)$ *and* $\mathsf{msk} = (t_1, t_2, t_3, t_4)$. *Then the outputs of* $\mathsf{KgS}(g, v_1, v_2, t_3, t_4, \mathbf{u}, \mathbf{x}, \mathbf{y}, \mathsf{id})$ *and* $\mathsf{Kg}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$ *are identically distributed.* □

Proof. First assume $t_1\mathsf{F}(\mathbf{x}, \mathsf{id}) + \mathsf{G}(\mathbf{y}, \mathsf{id}) \neq 0 \pmod p$. Define $f_1, f_2 : \mathbb{Z}_p \to \mathbb{Z}_p$ by

$$f_1(r_1) = \frac{r_1 \cdot \mathsf{F}(\mathbf{x}, \mathsf{id})}{t_1\mathsf{F}(\mathbf{x}, \mathsf{id}) + \mathsf{G}(\mathbf{y}, \mathsf{id})}$$

and

$$f_2(r_2) = r_2 + \frac{r_1 t_2 \mathsf{G}(\mathbf{y}, \mathsf{id})}{(t_3 t_4)(t_1\mathsf{F}(\mathbf{x}, \mathsf{id}) + \mathsf{G}(\mathbf{y}, \mathsf{id}))} \ .$$

These are well defined because $t_1\mathsf{F}(\mathbf{x},\mathsf{id}) + \mathsf{G}(\mathbf{y},\mathsf{id})$ was assumed $\neq 0 \pmod p$ and also we know $t_3, t_4 \neq 0 \pmod p$. Then letting $r_1' = f(r_1)$ and $r_2' = f(r_2)$, a computation shows that

$$g^{r_2 t_3 t_4} v_2^{r_1} = g^{r_1' t_1 t_2 + r_2' t_3 t_4} \ , \quad v_2^{-r_1 \mathsf{F}(\mathbf{x},\mathsf{id})} = H(\mathbf{u},\mathsf{id})^{-r_1' t_2} \ , \quad v_1^{-r_1 \mathsf{F}(\mathbf{x},\mathsf{id})} = H(\mathbf{u},\mathsf{id})^{-r_1' t_1}$$

$$H(\mathbf{u},\mathsf{id})^{-r_2 t_4} v_2^{-r_1 \mathsf{G}(\mathbf{u},\mathsf{id})/t_3} = H(\mathbf{u},\mathsf{id})^{-r_2' t_4} \ , \quad H(\mathbf{u},\mathsf{id})^{-r_2 t_3} v_2^{-r_1 \mathsf{G}(\mathbf{y},\mathsf{id})/t_4} = H(\mathbf{u},\mathsf{id})^{-r_2' t_3}$$

Given that $\mathsf{F}(\mathbf{x},\mathsf{id}) \neq 0$, the functions f_1, f_2 are permutations so from above we are done. Now, in the case

$$t_1\mathsf{F}(\mathbf{x},\mathsf{id}) + \mathsf{G}(\mathbf{y},\mathsf{id}) \equiv 0 \pmod p$$

we have $H(\mathbf{u},\mathsf{id}) = 1_{\mathbb{G}}$. We let

$$f_1(r_1) = \begin{cases} r_1/t_1 & \text{If } t_1 \neq 0 \\ r_1 & \text{otherwise} \end{cases}$$

and $f_2(r_2) = r_2$. Again, f_1, f_2 are permutations and we are done. □

Analysis. Consider games G_4–G_8 of Figures 14 and 15. We have

$$\begin{aligned}
&\Pr\left[\,\mathrm{DLIN}^{\mathcal{B}_1}_{\mathsf{GP}} \Rightarrow \mathsf{true}\,\right] = \Pr\left[\,G_4^{\mathcal{A}} \Rightarrow d\,\right] \\
&= \Pr\left[\,G_4^{\mathcal{A}} \Rightarrow d \mid \mathsf{BD}(G_4^{\mathcal{A}})\,\right] \cdot \Pr\left[\,\mathsf{BD}(G_4^{\mathcal{A}})\,\right] + \Pr\left[\,G_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}(G_4^{\mathcal{A}})\,\right] \\
&= \frac{1}{2}\Pr\left[\,\mathsf{BD}(G_4^{\mathcal{A}})\,\right] + \Pr\left[\,G_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}(G_4^{\mathcal{A}})\,\right] \\
&= \frac{1}{2}\Pr\left[\,\mathsf{BD}(G_5^{\mathcal{A}})\,\right] + \Pr\left[\,G_5^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}(G_5^{\mathcal{A}})\,\right] \ ,
\end{aligned}$$

the last because G_4, G_5 are identical until bad. We now claim

$$\Pr\left[\,\mathsf{BD}(G_5^{\mathcal{A}})\,\right] = \Pr\left[\,\mathsf{BD}(G_6^{\mathcal{A}})\,\right] \text{ and } \Pr\left[\,G_5^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}(G_5^{\mathcal{A}})\,\right] = \Pr\left[\,G_6^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}(G_6^{\mathcal{A}})\,\right] \ .$$

Lemma 1 implies that the **Extract** procedures in G_5, G_6 are equivalent. We now claim the **LR** procedures are as well. To see this consider separately the cases $d = 0$ and $d = 1$. (In the former, if $w = g^t$ is random then EncS is equivalent to the boxed code.) Additionally $c'' = c'$ in **Finalize** of G_5 hence the **Finalize** procedures of G_5, G_6 are equivalent. Standard game manipulations following [3] now give us

$$\Pr\left[\,\mathsf{BD}(G_6^{\mathcal{A}})\,\right] = \Pr\left[\,\mathsf{BD}(G_7^{\mathcal{A}})\,\right] = \Pr\left[\,\mathsf{BD}(G_8^{\mathcal{A}})\,\right]$$

$$\Pr\left[\,\mathrm{G}_6^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}(G_6^{\mathcal{A}})\,\right] = \Pr\left[\,G_6^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}(G_7^{\mathcal{A}})\,\right] = \Pr\left[\,G_8^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}(G_8^{\mathcal{A}})\,\right] \ .$$

Putting the above together we have

$$\mathbf{Adv}^{\mathrm{dlin}}_{\mathsf{GP}}(\mathcal{B}_1) = 2 \cdot \Pr\left[\,\mathrm{DLIN}^{\mathcal{B}_1}_{\mathsf{GP}}\,\right] - 1 = 2 \cdot \Pr\left[\,G_8^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}(G_8^{\mathcal{A}})\,\right] - \Pr\left[\,\mathsf{GD}(G_8^{\mathcal{A}})\,\right] \ .$$

Next

$$\begin{aligned}
\Pr\left[\,G_8^{\mathcal{A}} \Rightarrow d\,\right] &= \frac{1}{2}\Pr\left[\,G_8^{\mathcal{A}} \Rightarrow 1 \mid d = 1\,\right] - \frac{1}{2}\Pr\left[\,G_8^{\mathcal{A}} \Rightarrow 1 \mid d = 0\,\right] + \frac{1}{2} \\
&= \frac{1}{2}\Pr\left[\,G_0^{\mathcal{A}}\,\right] - \frac{1}{2}\Pr\left[\,G_1^{\mathcal{A}}\,\right] \\
&= \frac{1}{2} + \frac{\epsilon_1}{2}
\end{aligned}$$

Now use [3, Lemmas 3.4,3.5], and define $\gamma_{\min}$, $\gamma_{\max}$ as there. Then, using the fact that $m = \lceil 3q/\epsilon_1 \rceil$, calculation (omitted) shows

$$\begin{aligned}\mathbf{Adv}^{\mathrm{dlin}}_{\mathsf{GP}}(\mathcal{B}_1) &\geq \gamma_{\min}\epsilon_1 + (\gamma_{\min} - \gamma_{\max}) \\ &\geq \frac{\epsilon_1^2}{9qn + 3\epsilon_1} \ .\end{aligned}$$

□

D Proof of Theorem 3

Let $\mathcal{A}$ be an adversary that has access to oracles **LR**, **Extract**, and makes only 1 query to **LR**. Let id^* denote the identity in this query. We say $\mathcal{A}$ is *good* if for all **Extract**(id) queries it makes we have $\mathsf{id} \not\equiv \mathsf{id}^* \pmod{p_2}$, with probability 1 regardless of how oracle queries are answered.

Our first step is to convert $\mathcal{A}'$ to a good $\mathcal{A}$ without significant resource increase. $\mathcal{A}$ runs $\mathcal{A}'$. When $\mathcal{A}'$ makes **LR** query id^*, M_0, M_1, adversary $\mathcal{A}$ checks if $\mathcal{A}'$ previously made a query **Extract**(id) such that $\gcd(\mathsf{id} - \mathsf{id}^*, N) \neq 1$. If so $\mathcal{A}$ aborts and guesses a bit. Otherwise it makes the same **LR** query and returns the response to $\mathcal{A}'$. When $\mathcal{A}'$ makes query **Extract**(id) after already querying **LR** on identity id^*, adversary $\mathcal{A}$ checks if $\gcd(\mathsf{id} - \mathsf{id}^*, N) = 1$. If so it forwards the query and returns the answer to $\mathcal{A}'$, else it does not forward the query, instead halting with a random bit output. If $\mathcal{A}'$ terminates without abortion, $\mathcal{A}$ adopts its output. Clearly $\mathcal{A}$ is good. We claim its advantage is almost that of $\mathcal{A}'$.

Claim 1. *Let $\mathcal{A}$ be constructed from $\mathcal{A}'$ as above. Then there exists a $\mathcal{B}_1$ such that*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{BBoR}}(\mathcal{A}') \leq \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{BBoR}}(\mathcal{A}) + 3 \cdot \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_1) \ ,$$

*and $\mathcal{B}_1$ makes 3 **Gen** queries and runs in time $\mathsf{T}(\mathcal{B}_1) = \mathsf{T}(\mathcal{A}') + \mathcal{O}(q \cdot \mathsf{T}_{\exp}(\mathbb{G}) + q \cdot \mathsf{T}(\gcd))$.* □

Proof (Claim 1). $\mathcal{B}_1$ picks $i \leftarrow\!\!\$\ \{1, 3, 4\}$ and makes query $T \leftarrow\!\!\$\ \mathbf{Ch}(\{i\}, \{i, 2\})$. It then makes queries $g_j \leftarrow\!\!\$\ \mathbf{Gen}(\{j\})$ for all $j \in \{1, 3, 4\}$. Given g_1, g_3, g_4, it can create $\mathsf{par}, \mathsf{msk}$ and thus run $\mathcal{A}'$, answering all its queries. Whenever $\mathcal{A}'$ queries **Extract**(id), adversary $\mathcal{B}_1$ tests if $r = \gcd(\mathsf{id} - \mathsf{id}^*, N) \neq 1$ and if so has a factor of N. We want to use r to generate a point in $\mathbb{G}(S)$ where $2 \in S$ but $i \notin S$ and then test T with orthogonality. The key point that makes this possible is that i is random and not known to $\mathcal{A}'$. Let us look at the possibilities: Let $R \subseteq [n]$ be such that $r = \prod_{i \in R} p_i$. In the table below we enumerate the possiblities for R, $[n] \setminus R$, and the resulting i and S that give us what we want and allow us to win the GSD game.

R	$[n] \setminus R$	i	S
1	2 3 4	1	2 3 4
2	1 3 4	1 3 4	2
3	1 2 4	3	1 2 4
4	1 2 3	4	1 2 3
1 2	3 4	3 4	1 2
1 3	2 4	1 3	2 4
1 4	2 3	1 4	2 3
2 3	1 4	1 4	2 3
2 4	1 3	1 3	2 4
1 2 3	4	4	1 2 3
1 2 4	3	3	1 2 4
1 3 4	2	1 3 4	2
2 3 4	1	1	2 3 4

We can see from the above table that no matter what the value of R, there is at least one choice of i that allows us to use r or N/r to construct a point in $\mathbb{G}(S)$ where $2 \in S$ but $i \notin S$ (and immediately win game GSD). Since the choice of i is independent of anything $\mathcal{A}'$ sees, $\mathcal{B}_1$ has at least a $1/3$ chance of picking an i that will work. The claim follows. □

Now we bound $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{BBoR}}(\mathcal{A})$. Consider the games of Figure 16. Game H is the real game while game G_0 replaces the challenge encryption of a 0 by a semi-functional one, meaning its components are multiplied by points in $\mathbb{G}(\{2\})$. The following says this makes little difference.

Claim 2 ($H \approx G_0$). *There exists $\mathcal{B}_2$ such that*

$$\Pr\left[\, H^{\mathcal{A}} \,\right] - \Pr\left[\, G_0^{\mathcal{A}} \,\right] \leq \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_2)\ ,$$

*and $\mathcal{B}_2$ makes 3 **Gen** queries and runs in time $\mathsf{T}(\mathcal{B}_2) = \mathsf{T}(\mathcal{A}) + \mathcal{O}(q \cdot \mathsf{T}_{\exp}(\mathbb{G}))$.* □

Proof (Claim 2). $\mathcal{B}_2(\pi)$ begins with $T \leftarrow\!\!\$\ \mathbf{Ch}(\{1,2,4\},\{1,4\})$. It then queries $g_j \leftarrow\!\!\$\ \mathbf{Gen}(\{j\})$ for $k \in \{1,3,4\}$. Next it performs initialization.

$u_1, u_4, x_1, x_4, w_4 \leftarrow\!\!\$\ \mathbb{Z}_N$; $\beta \leftarrow\!\!\$\ \{0,1\}$
$U_1 \leftarrow g_1^{u_1}$; $U_4 \leftarrow g_4^{u_4}$; $X_1 \leftarrow g_1^{x_1}$; $X_4 \leftarrow g_4^{x_4}$; $W_4 \leftarrow g_4^{w_4}$
$U_{14} \leftarrow U_1 U_4$; $X_{14} \leftarrow X_1 X_4$; $W_{14} \leftarrow g_1 W_4$
$\mathsf{par} \leftarrow (\pi, U_{14}, X_{14}, W_{14}, g_4)$

It runs $A(\mathsf{par})$ answering queries as follows:

On query $\mathbf{LR}(\mathsf{id}, M_0, M_1)$:
$\mathsf{id}^* \leftarrow \mathsf{id}$
If $M_\beta = 0$ then
  $C \leftarrow T^{u_1 \mathsf{id} + x_1}$; $C' \leftarrow T$
Else
  $C, C' \leftarrow\!\!\$\ \mathbb{G}$
Return (C, C')

proc. Initialize: Games H, G_ℓ $(0 \le \ell \le q)$
$(\pi, \bar{\pi}) \leftarrow\!\!\$\ \mathsf{Gen}$
$(\langle\mathbb{G}\rangle, \langle\mathbb{G}_T\rangle, \langle \mathrm{e}\rangle, N) \leftarrow \pi$
$(p_1, p_2, p_3, p_4) \leftarrow \bar{\pi}$
$g_1 \leftarrow\!\!\$\ \mathbb{G}(\{1\})*$; $g_2 \leftarrow\!\!\$\ \mathbb{G}(\{2\})^*$
$g_3 \leftarrow\!\!\$\ \mathbb{G}(\{3\})^*$; $g_4 \leftarrow\!\!\$\ \mathbb{G}(\{4\})^*$
$u_1, u_4, x_1, x_4, w_4 \leftarrow\!\!\$\ \mathbb{Z}_N$
$U_1 \leftarrow g_1^{u_1}$; $U_4 \leftarrow g_4^{u_4}$
$X_1 \leftarrow g_1^{x_1}$; $X_4 \leftarrow g_4^{x_4}$; $W_4 \leftarrow g_4^{w_4}$
$U_{14} \leftarrow U_1 U_4$; $W_{14} \leftarrow g_1 W_4$; $X_{14} \leftarrow X_1 X_4$
$\mathsf{par} \leftarrow (\pi, U_{14}, X_{14}, W_{14}, g_4)$
$\beta \leftarrow\!\!\$\ \{0,1\}$; $i \leftarrow 0$
Return par

proc. Extract(id): Games H
$r, r_3, r_3' \leftarrow\!\!\$\ \mathbb{Z}_N$
$K \leftarrow g_1^r g_3^{r_3}$; $K' \leftarrow (U_1^{\mathsf{id}} X_1)^r g_3^{r_3'}$
Return (K, K')

proc. LR(id, M_0, M_1): Game H
$\mathsf{id}^* \leftarrow \mathsf{id}$
If $M_\beta = 0$ then
  $s, t_4, t_4' \leftarrow\!\!\$\ \mathbb{Z}_N$
  $C \leftarrow (U_{14}^{\mathsf{id}} X_{14})^s g_4^{t_4}$
  $C' \leftarrow W_{14}^s g_4^{t_4'}$
Else
  $C, C' \leftarrow\!\!\$\ \mathbb{G}$
Return (C, C')

proc. Finalize(β'): Games H, G_ℓ $(0 \le \ell \le q)$
Return $(\beta = \beta')$

proc. Extract(id): Games G_ℓ $(0 \le \ell \le q)$
$i \leftarrow i + 1$
$r, r_3, r_3', t_2, t_2' \leftarrow\!\!\$\ \mathbb{Z}_N$
If $(i \le \ell)$ then
  $K \leftarrow g_1^r g_3^{r_3} g_2^{t_2}$; $K' \leftarrow (U_1^{\mathsf{id}} X_1)^r g_3^{r_3'} g_2^{t_2'}$
Else
  $K \leftarrow g_1^r g_3^{r_3}$; $K' \leftarrow (U_1^{\mathsf{id}} X_1)^r g_3^{r_3'}$
Return (K, K')

proc. LR(id, M_0, M_1): Game G_ℓ $(0 \le \ell \le q)$
$\mathsf{id}^* \leftarrow \mathsf{id}$
If $M_\beta = 0$ then
  $s, t_4, t_4', x_2, x_2' \leftarrow\!\!\$\ \mathbb{Z}_N$
  $C \leftarrow (U_{14}^{\mathsf{id}} X_{14})^s g_4^{t_4} g_2^{x_2 x_2'}$
  $C' \leftarrow W_{14}^s g_4^{t_4'} g_2^{x_2}$
Else
  $C, C' \leftarrow\!\!\$\ \mathbb{G}$
Return (C, C')

proc. Extract(id): Games H_1, H_2
$r, r_3, r_3', t_2, t_2' \leftarrow\!\!\$\ \mathbb{Z}_N$
$K \leftarrow g_1^r g_3^{r_3} g_2^{t_2}$; $K' \leftarrow (U_1^{\mathsf{id}} X_1)^r g_3^{r_3'} g_2^{t_2'}$
Return (K, K')

proc. LR(id, M_0, M_1): Game H_1
If $M_\beta = 0$ then
  $C, C' \leftarrow\!\!\$\ \mathbb{G}(\{1,2,4\})$
Else
  $C, C' \leftarrow\!\!\$\ \mathbb{G}$
Return (C, C')

proc. LR(id, M_0, M_1): Game H_2
$C, C' \leftarrow\!\!\$\ \mathbb{G}$
Return (C, C')

Fig. 16. Games for the proof of Theorem 3.

On query **Extract**(id) it can answer as in game H since it knows msk and also H, G_0 have the same **Extract** oracle. When $\mathcal{A}$ halts with output β', adversary $\mathcal{B}_2$ returns 1 if $(\beta = \beta')$ and 0 otherwise.

We have

$$\begin{aligned}
&\frac{1}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{BBoR}}(\mathcal{A}) + \frac{1}{2} = \Pr\left[\, H^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] \\
&= \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] + (\Pr\left[\, H^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] - \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right]
\end{aligned}$$

We can write $T = (g_1 W_4)^s g_4^{t_4'} g_2^{x_2}$ where $x_2 = 0$ if challenge bit $b = 1$ and otherwise $x_2 \leftarrow\!\!\$\ \mathbb{Z}_N$, and s, t_4' are random in $\mathbb{Z}_N$ in either case. Thus, C' has the right form. Also,

$$\begin{aligned}
C = T^{u_1 \mathsf{id} + x_1} &= (g_1 g_4^{w_4})^{s(u_1 \mathsf{id} + x_1)} g_4^{t_4'(u_1 \mathsf{id} + x_1)} g_2^{x_2(u_1 \mathsf{id} + x_1)} \\
&= (U_1^{\mathsf{id}} X_1)^s g_4^{(sw_4 + t_4')(u_1 \mathsf{id} + x_1)} g_2^{x_2(u_1 \mathsf{id} + x_1)} \\
&= (U_{14}^{\mathsf{id}} X_{14})^s g_4^{(sw_4 + t_4')(u_1 \mathsf{id} + x_1) - u_4 \mathsf{id} s - x_4 s} g_2^{x_2(u_1 \mathsf{id} + x_1)}
\end{aligned}$$

This has the form $(U_{14}^{\mathsf{id}} X_{14})^s g_4^{t_4} g_2^{x_2 x_2'}$ with

$$\begin{aligned} t_4 &= (sw_4 + t_4')(u_1\mathsf{id} + x_1) - u_4 \mathsf{id} s - x_4 s \mod p_4 \\ x_2' &= x_2(u_1\mathsf{id} + x_1) \mod p_2 \end{aligned}$$

The key is that $u_1\mathsf{id} + x_1 \mod p_1$, $u_1\mathsf{id} + x_1 \mod p_2$, $u_1\mathsf{id} + x_1 \mod p_4$ are random and independent by the chinese remainder theorem (CRT) so t_4 is random over given par, T. If $b = 1$ then $x_2' = 0$ and if $b = 0$ then x_2' is random and independent of par, T, t_4.

□

To bound $\Pr\left[\, G_0^{\mathcal{A}} \,\right]$ we use a hybrid argument. Games $G_0, \ldots, G_q$ replace the keys provided by **Extract** by semi-functional ones, one at a time. Now we want to bound

$$\Pr\left[\, G_0^{\mathcal{A}} \,\right] = (\Pr\left[\, G_0^{\mathcal{A}} \,\right] - \Pr\left[\, G_q^{\mathcal{A}} \,\right]) + \Pr\left[\, G_q^{\mathcal{A}} \,\right] \ .$$

This is the step that uses the assumption that $\mathcal{A}$ is good.

Claim 3. *There exists $\mathcal{B}_3$ such that*

$$\Pr\left[\, G_0^{\mathcal{A}} \,\right] - \Pr\left[\, G_q^{\mathcal{A}} \,\right] \le q \cdot \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_3) \ ,$$

*and $\mathcal{B}_3$ makes 5 **Gen** queries and runs in time $\mathsf{T}(\mathcal{B}_3) = \mathsf{T}(\mathcal{A}) + \mathcal{O}(q \cdot \mathsf{T}_{\exp}(\mathbb{G}))$.* □

Proof (Claim 3). $\mathcal{B}_3(\pi)$ begins with $T \leftarrow\!\!\$\ \mathbf{Ch}(\{1,2,3\},\{1,3\})$. It then queries $g_j \leftarrow\!\!\$\ \mathbf{Gen}(\{j\})$ for $j \in \{1,3,4\}$ and also $D_{23} \leftarrow\!\!\$\ \mathbf{Gen}(\{2,3\})$ and $A_{12} \leftarrow\!\!\$\ \mathbf{Gen}(\{1,2\})$. It performs the same initialization as $\mathcal{B}_2$ and sets $i \leftarrow 0$ and $\ell \leftarrow\!\!\$\ \{1,\ldots,q\}$. It then runs $\mathcal{A}(\mathsf{par})$ answering queries as follows:

On query $\mathbf{LR}(\mathsf{id}, M_0, M_1)$:
$\mathsf{id}^* \leftarrow \mathsf{id}$
If $M_\beta = 0$ then
  $s_4, s_4' \leftarrow\!\!\$\ \mathbb{Z}_N$
  $c \leftarrow A_{12}^{u_1\mathsf{id}+x_1} g_4^{s_4}$; $C' \leftarrow A_{12} g_4^{s_4'}$
Else
  $C, C' \leftarrow\!\!\$\ \mathbb{G}$
Return (C, C')

On query $\mathbf{Extract}(\mathsf{id})$:
$i \leftarrow i + 1$
If $i < \ell$ then
  $r, z, z' \leftarrow\!\!\$\ \mathbb{Z}_N$
  $K \leftarrow g_1^r D_{23}^z$; $K' \leftarrow (U_1^{\mathsf{id}} X_1)^r D_{23}^{z'}$
Else if $(i = \ell)$ then
  $K \leftarrow T$; $K' \leftarrow T^{u_1\mathsf{id}+x_1}$
Else // $i > \ell$
  $r_1, r_3, r_3' \leftarrow\!\!\$\ \mathbb{Z}_N$
  $K \leftarrow g_1^r g_3^{r_3}$; $K' \leftarrow (U_1^{\mathsf{id}} X_1)^r g_3^{r_3'}$
Return (K, K')

When $\mathcal{A}$ halts with output β' adversary $\mathcal{B}_3$ returns 1 if $\beta = \beta'$ and 0 otherwise.

Let b be $\mathcal{B}_3$'s challenge bit. We need to show its responses are distributed as in $G_{\ell-1}$ when $b = 1$ and G_ℓ when $b = 0$ so that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_3) &= \frac{1}{q}\sum_{\ell=1}^{q} \Pr\left[\, G^A_{\ell-1}\,\right] - \Pr\left[\, G^A_{\ell}\,\right] \\ &= \frac{1}{q}\left(\Pr\left[\, G^A_0\,\right] - \Pr\left[\, G^A_q\,\right]\right)\ .\end{aligned}$$

First we show the ciphertext is correctly distributed regardless of b, ℓ. We can write $A_{12} = g_1^s g_2^{x_2}$ for random $s, x_3 \in \mathbb{Z}_N$. Then

$$\begin{aligned}C' &= A_{12} g_4^{s'_4} = g_1^s g_4^{s'_4} g_2^{x_2} \\ &= (g_1 g_4^{w_4})^s g_4^{s'_4 - w_4 s} g_2^{x_2}\end{aligned}$$

This has the form $(g_1 W_4)^s g_4^{t'_4} g_2^{x_2}$ with $t'_4 = s'_4 - w_4 s \mod p_4$. Now

$$\begin{aligned}C &= A_{12}^{u_1\mathsf{id}+x_1} g_4^{s_4} = (g_1^s g_2^{x_2})^{u_1\mathsf{id}+x_1} g_4^{s_4} \\ &= (U_1^{\mathsf{id}} X_1)^s g_4^{s_4} g_2^{x_2(u_1\mathsf{id}+x_1)} \\ &= ((U_1U_4)^{\mathsf{id}} X_1 X_4)^s g_4^{s_4 - u_4\mathsf{id}s - x_4 s} g_2^{x_2(u_1\mathsf{id}+x_1)}\end{aligned}$$

This has the form

$$((U_1U_4)^{\mathsf{id}} X_1 X_4)^s g_4^{t_4} g_2^{x_2 x'_2}$$

with

$$\begin{aligned}t_4 &= s_4 - u_4\mathsf{id}s - x_4 s \mod p_4 \\ x'_2 &= u_1\mathsf{id} + x_1 \mod p_2\end{aligned}$$

The first is random due to s_4 and the second by the usual CRT argument.

Next we have to show the keys are correctly distributed. They certainly are when $i > \ell$ so we look at $i < \ell$ and $i = \ell$. For the first case say $D_{23} = g_2^{y_2} g_3^{y_3}$. Then

$$K = g_1^r D_{23}^z = g_1^r g_2^{zy_2} g_3^{zy_3}$$

which has the form $g_1^r g_3^{r_3} g_2^{t_2}$ with $r_3 = zy_3 \mod p_3$ and $t_2 = zy_2 \mod p_2$. Also

$$K' = (U_1^{\mathsf{id}} X_1)^r D_{23}^{z'} = (U_1^{\mathsf{id}} X_1)^r g_1^{zy_2} g_3^{z'y_3}$$

which has the form $(U_1^{\mathsf{id}} X_1)^r g_3^{r'_3} g_2^{t'_2}$ with $r'_3 = z'y_3 \mod p_3$ and $t'_2 = z'y_2 \mod p_2$. Now if $i = \ell$ we can write $T = g_1^r g_2^{t_2} g_3^{r_3}$ where $t_2 = 0$ if $b = 1$. So $K = T$ has the right form. Now

$$K' = T^{u_1\mathsf{id}+x_1} = g_1^{r(u_1\mathsf{id}+x_1)} g_2^{t_2(u_1\mathsf{id}+x_1)} g_3^{r_3(u_1\mathsf{id}+x_1)} = (U_1^{\mathsf{id}} X_1)^r g_2^{t'_2} g_3^{r'_3}$$

with

$$\begin{aligned}t'_2 &= t_2(u_1\mathsf{id} + x_1) \mod p_2 \\ r'_3 &= r_3(u_1\mathsf{id} + x_1) \mod p_3\end{aligned}$$

Taken by itself this is OK by CRT but we also gave out $u_1\mathsf{id}^* + x_1 \mod p_2$ in the ciphertext. But $u_1\mathsf{id}^* + x_1 \mod p_2$ and $u_1\mathsf{id} + x_1 \mod p_2$ are independent as long as $\mathsf{id} \not\equiv \mathsf{id}^* \pmod{p_2}$ which is where we use the assumption that $\mathcal{A}$ is good. □

We now need to bound

$$\Pr\left[\, G_q^{\mathcal{A}} \,\right] = \Pr\left[\, H_1^{\mathcal{A}} \,\right] + \left(\Pr\left[\, G_q^{\mathcal{A}} \,\right] - \Pr\left[\, H_1^{\mathcal{A}} \,\right]\right) .$$

In game H_1, the challenge encryption of a 0 is a pair of random, independent points in $\mathbb{G}(\{1,2,4\})$, meaning the structure from game G_q is broken. (The keys remain all semi-functional.) The following says this change makes little difference.

Claim 4. *There exists $\mathcal{B}_4$ such that*

$$\Pr\left[\, G_q^{\mathcal{A}} \,\right] - \Pr\left[\, H_1^{\mathcal{A}} \,\right] \le \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_4) ,$$

*and $\mathcal{B}_4$ makes 5 **Gen** queries and runs in time $\mathsf{T}(\mathcal{B}_4) = \mathsf{T}(\mathcal{A}) + \mathcal{O}(q \cdot \mathsf{T}_{\exp}(\mathbb{G}))$.* □

Proof (Claim 4). $\mathcal{B}_4(\pi)$ begins with $T \leftarrow\!\!\$\ \mathbf{Ch}(\{2,4\},\{1,2,4\})$. It then queries $W_{14} \leftarrow\!\!\$\ \mathbf{Gen}(\{1,4\})$, $E_{12} \leftarrow\!\!\$\ \mathbf{Gen}(\{1,2\})$, $g_j \leftarrow\!\!\$\ \mathbf{Gen}(\{j\})$ for $j \in \{2,3,4\}$. Next it performs initialization

$u_1, x_1 \leftarrow\!\!\$\ \mathbb{Z}_N$; $U_{14} \leftarrow W_{14}^{u_1}$; $X_{14} \leftarrow W_{14}^{x_1}$
$\mathsf{par} \leftarrow (\pi, U_{14}, X_{14}, W_{14}, g_4)$

It runs $\mathcal{A}(\mathsf{par})$ answering oracle queries as follows:

On query $\mathbf{LR}(\mathsf{id}, M_0, M_1)$:
$s, x_2' \leftarrow\!\!\$\ \mathbb{Z}_N$
$C \leftarrow (U_{14}^{\mathsf{id}} X_{14})^s T^{x_2'}$; $C' \leftarrow W_{14}^s T$
Return (C, C')

On query $\mathbf{Extract}(\mathsf{id})$:
$t, y, y', r_3, r_3' \leftarrow\!\!\$\ \mathbb{Z}_N$
$K \leftarrow E_{12}^t g_2^y g_3^{x_3}$; $K' \leftarrow E_{12}^{t(u_1\mathsf{id}+x_1)} g_2^{y'} g_3^{r_3'}$
Return (K, K')

When $\mathcal{A}$ halts with output β', adversary $\mathcal{B}_4$ returns 1 if $\beta = \beta'$ and 0 otherwise.

If $\mathcal{B}_4$'s challenge bit is $b = 1$ then T is random in $\mathbb{G}(\{1,2,4\})$ and hence, due to this and x_2', both C, C' are random in $\mathbb{G}(\{1,2,4\})$, as in game H_1. On the other hand if $b = 0$ then $T = g_4^{t_4'} g_2^{x_2}$ and $C' = W_{14}^s g_4^{t_4'} g_2^{x_2}$ has the right form for game G_q. Also

$$\begin{aligned} C &= (U_{14}^{\mathsf{id}} X_{14})^s g_4^{x_2' t_4'} g_2^{x_2' x_2} \\ &= (U_{14}^{\mathsf{id}} X_{14})^s g_4^{t_4} g_2^{x_2 x_2'} \end{aligned}$$

where

$$t_4 = t_4' x_2' \mod p_4$$

which is random and independent of $x_2 x_2' \mod p_2$ by CRT so C has form of game G_q as well.

Next we look at responses to $\mathbf{Extract}(\mathsf{id})$ queries. Say $E_{12} = g_1^{e_1} g_2^{e_2}$. Then

$$\begin{aligned} K &= E_{12}^t g_2^y g_3^{r_3} \\ &= g_1^{te_1} g_2^{te_2+y} g_3^{r_3} \\ &= g_1^r g_3^{r_3} g_2^{t_2} \end{aligned}$$

with $r = te_1$, $t_2 = te_2 + y$. Also

$$\begin{aligned} K' &= E_{12}^{t(u_1\mathsf{id}+x_1)} g_2^{y'} g_3^{r_3'} \\ &= g_1^{u_1\mathsf{id}+x_1)te_1} g_2^{(u_1\mathsf{id}+x_1)te_2+y'} g_3^{r_3'} \\ &= (U_1^{\mathsf{id}} X_1)^{te_1} g_2^{(u_1\mathsf{id}+x_1)te_2+y'} g_3^{r_3'} \\ &= (U_1^{\mathsf{id}} X_1)^r g_3^{r_3'} g_2^{t_2'} \end{aligned}$$

with

$$t'_2 = (u_1\mathsf{id} + x_1)te_2 + y'$$

which is random due to y'. □

Now we need to bound

$$\Pr\left[\, H_1^{\mathcal{A}} \,\right] = \Pr\left[\, H_2^{\mathcal{A}} \,\right] + \left(\Pr\left[\, H_1^{\mathcal{A}} \,\right] - \Pr\left[\, H_2^{\mathcal{A}} \,\right]\right) .$$

Game H_2 replaces the challenge encryption of 0 by a pair of random points in the full group. A subtle point is that we needed two steps (games H_1, H_2) because the challenge queries made by adversaries $\mathcal{B}_4, \mathcal{B}_5$ are different, meaning subgroup decision hardness for two different pairs of subgroups is being used.

Claim 5. *There exists $\mathcal{B}_5$ such that*

$$\Pr\left[\, H_1^{\mathcal{A}} \,\right] - \Pr\left[\, H_2^{\mathcal{A}} \,\right] \leq \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_5) ,$$

*and $\mathcal{B}_5$ makes 3 **Gen** queries and runs in time $\mathsf{T}(\mathcal{B}_5) = \mathsf{T}(\mathcal{A}) + \mathcal{O}(q \cdot \mathsf{T}_{\exp}(\mathbb{G}))$.* □

Proof (Claim 5). Adversary $\mathcal{B}_5(\pi)$ begins with $T \leftarrow\!\!\$\ \mathbf{Ch}(\{1,2,3,4\},\{1,2,4\})$. It then queries $g_1 \leftarrow\!\!\$\ \mathbf{Gen}(\{1\})$, $D_{23} \leftarrow\!\!\$\ \mathbf{Gen}(\{2,3\})$, $g_4 \leftarrow\!\!\$\ \mathbf{Gen}(\{4\})$. It performs the same initialization as $\mathcal{B}_2$. It runs $\mathcal{A}(\mathsf{par})$ answering queries as follows:

On query $\mathbf{LR}(\mathsf{id}, M_0, M_1)$:
$z \leftarrow\!\!\$\ \mathbb{Z}_N$; $C \leftarrow T$; $C' \leftarrow T^z$; return (C, C')

On query $\mathbf{Extract}(\mathsf{id})$:
$r, d, d' \leftarrow\!\!\$\ \mathbb{Z}_N$
$K \leftarrow g_1^r D_{23}^d$; $K' \leftarrow (U_1^{\mathsf{id}} X_1)^r D_{23}^{d'}$

$\mathcal{B}_5$ outputs whatever $\mathcal{A}$ outputs. The challenge ciphertext components are random in $\mathbb{G}$ if $b = 0$ and in $\mathbb{G}(\{1,2,4\})$ if $b = 1$. The keys are as in games H_1, H_2. □

The following is clear since the challenge ciphertext in game H_2 does not depend on the challenge bit β.

Claim 6. $\Pr\left[\, H_2^{\mathcal{A}} \,\right] = \frac{1}{2}$. □

Combining the claims gives

$$\begin{aligned}
\frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{BBoR}}(\mathcal{A}) + \frac{1}{2} &= \Pr\left[\, H^{\mathcal{A}} \,\right] \\
&\leq \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_2) + \Pr\left[\, G_0^{\mathcal{A}} \,\right] \\
&\leq \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_2) + q \cdot \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_3) + \Pr\left[\, G_q^{\mathcal{A}} \,\right] \\
&\leq \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_2) + q \cdot \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_3) + \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_4) + \Pr\left[\, H_1^{\mathcal{A}} \,\right] \\
&\leq \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_2) + q \cdot \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_3) + \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_4) + \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_5) + \Pr\left[\, H_2^{\mathcal{A}} \,\right] \\
&\leq \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_2) + q \cdot \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_3) + \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_4) + \mathbf{Adv}^{\mathrm{gsd}}_{\mathsf{Gen}}(\mathcal{B}_5) + \frac{1}{2} .
\end{aligned}$$

Combined with our claim regarding adversary $\mathcal{A}'$ and good adversary $\mathcal{A}$, this gives

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{BBoR}}(\mathcal{A}') \leq 3 \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_1) + 2 \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_2) + 2q \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_3) + 2 \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_4) + 2 \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_5)$$

Let $v_1 = 3, v_2 = 2, v_3 = 2q, v_4 = 2, v_5 = 2$ and let $v = v_1 + v_2 + v_3 + v_4 + v_5 = (9 + 2q)$. Now, let $\mathcal{B}(\pi)$ choose $d \in \{1, 2, 3, 4, 5\}$ such that $d = i$ with probability v_i/v for $1 \leq i \leq 5$. Then, $\mathcal{B}$ runs $\mathcal{B}_d(\pi)$. Clearly

$$\begin{aligned}
&\mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}) \\
&= \frac{1}{9 + 2q} \cdot \Big(3 \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_1) + 2 \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_2) + 2q \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_3) + 2 \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_4) + 2 \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}_5)\Big)
\end{aligned}$$

Thus, we end up with

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{BBoR}}(\mathcal{A}') \leq (9 + 2q) \cdot \mathbf{Adv}^{\text{gsd}}_{\mathsf{Gen}}(\mathcal{B}) \ ,$$

and the resources used by $\mathcal{B}$ are at most those used by any of the $\mathcal{B}_i$s. Thus, $\mathcal{B}$ makes at most 5 **Gen** queries and runs in time at most $\mathsf{T}(\mathcal{B}) = \mathsf{T}(\mathcal{A}') + \mathcal{O}(q \cdot \mathsf{T}_{\text{exp}}(\mathbb{G}) + q \cdot \mathsf{T}(\text{gcd}))$.