Identity-Based Encryption Secure under Selective Opening Attack

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

[1] University of California at San Diego
{mihir,syilek}@cs.ucsd.edu
[2] University of Texas at Austin
bwaters@cs.utexas.edu

Abstract. We present the first Identity-Based Encryption (IBE) scheme that is 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 via lossy encryption. We explain why those methods won't work for IBE and instead rely on an approach based on encryption schemes that have a property we call one-sided public openability. Our SOA-secure IBE scheme is quite efficient and proven secure without random oracles based on the Decision Linear assumption.

1 Introduction

Security against selective-opening attack (SOA) has been one of the more vexing and intriguing open questions in the entire theory of encryption. However, recently (and 10 years after the problem was identified), we have seen solutions [1] for the case of Public-Key Encryption (PKE).

In this paper, we define SOA-secure IBE. We explain why the lossy encryption methods that did the trick for PKE do not work in the IBE case. Instead, we return to ideas from non-committing [10] and deniable [9] encryption. We introduce IBE with one-sided public openability (1SPO) and show that it implies SOA-secure IBE. We then show how to achieve 1SPO IBE without random oracles based on the decision-linear assumption of [5].

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 related primitives or questions [14, 18, 22]. Finally, Bellare, Hofheinz, and Yilek (BHY) [1] showed a large class of encryption schemes, which they call *lossy* [1, 20, 23], are SOA secure. Schemes they show to be lossy include variants of El Gamal [21], the IND-CPA scheme built from lossy trapdoor functions by Peikert and Waters [24], and even the original Goldwasser-Micali encryption scheme [16]. Subsequently, Hemenway and Ostrovsky [17] 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.

SOA for IBE. We can adapt the SOA framework to IBE in a natural way but there are two new elements 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 [6], 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 [11].) To expand on this, a vector **id** of adversarially-chosen target receiver identities replaces the vector **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 **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 **id** and obtain its decryption key. Our formalization is simulation-based.

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.

Our contributions. As we explain below, there are fundamental obstacles to extending the lossy-implies-SOA-secure approach, that worked for PKE, to the IBE setting. We describe a new technique for achieving SOA-security. It involves constructing one-bit 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 [9]. Bit-by-bit encryption then results in a scheme that can encrypt long messages. We show that if the 1SPO scheme is also IND-CPA *then the constructed scheme is SOA-secure*. This reduces the task of obtaining an SOA-secure IBE scheme to obtaining an IND-CPA secure 1SPO scheme. Existing non-RO schemes [25, 4] do not have the property, but we describe a 1SPO scheme based on the (anonymous) IBE scheme of Boyen and Waters [8]. We prove it is 1SPO directly, and adapt techniques from [8, 2] to show it is IND-CPA under the DLIN (Decision Linear) assumption of [5]. The proof technique of [2] allows us to avoid Water's artificial abort step [25] thereby resulting in a more efficient reduction. Let us now look at our approach in more depth, beginning with a discussion of the difficulty of extending the BHY approach to IBE.

Limitations of lossiness. In order to understand our technical approach it is instructive to review BHY's lossy-implies-SOA-secure approach for PKE (Hemenway and Ostrovsky [17] also rely on this approach, their contribution being new lossy encryption schemes) and expose the difficulties in adapting this to IBE.

Roughly, an encryption scheme is lossy if its public keys (called injective keys) are computationally indistinguishable from lossy keys, produced by an alternative lossy key generator, and encryption under the lossy keys is statistically secure. BHY's claim is that any lossy PKE encryption scheme is SOA-secure. They consider the real SOA game (with injective keys) and another in which the keys encrypting messages are changed from injective to lossy. The indistinguishability of key types means any attacker that had a noticeable advantage in the real SOA game (with injective keys) would have a noticeable advantage in the SOA game with lossy keys. But in the latter game, each ciphertext is information theoretically "ambiguous" with regard to which particular message was encrypted in the sense that every message was equally likely to have generated the ciphertext, within a negligible probability of error. A key property of this stage of the reduction is that although the public key type is unknown, the reduction algorithm will know the coins and messages used to encrypt each message and can reveal them on demand. Once the SOA game is transformed to have lossy keys, SOA-security can be established by a relatively simple information theoretic argument.

In constructing SOA secure IBE systems we would like to take a similar approach, first using a hybrid argument to make the encrypted messages ambiguous and then culminating with an information theoretic argument. A natural first step would be to find some notion and construction of lossy keys in an IBE system analogous to that in PKE. However, this direction runs into some fundamental obstacles. First, we note that we cannot simply make the IBE system lossy on all identities. If we published a set of IBE parameters that made all ciphertexts to *every* identity lossy, this would be

readily detectable by an attacker. The latter would use its **Extract** oracle to obtain the decryption key sk of some non-target identity id, encrypt a random message M to id to get a ciphertext C, and then attempt to decrypt C using sk. If M is recovered, the system was in injective mode, else lossy. What would be required is that the keys of the target identities are lossy and those of non-target identities are injective. However, this approach also runs into fundamental difficulties. To begin with, the target identity is not known in advance, meaning at the time public parameters need to be created by the reduction and there are too many possibilities to guess. It is tempting now to think we could at least achieve security in the selective-id model where the vector of target identities is supplied in advance, but even if we were willing to do this (we prefer to achieve full security) it is not possible. For a given security parameter, an IBE system's public parameters will have some maximum size, say ℓ bits, and the length of the vector of target identities is allowed to be arbitrary and independent of ℓ. If this length is more than ℓ, it is information theoretically impossible to "program" the parameters to be lossy only on the target identities and injective on all other identities! This appears to be a fundamental limitation of proving security with key lossiness in IBE systems since the entire point of IBE is to have identity spaces that are much larger than the public parameter size.

Our approach. We circumvent these limitations by establishing ambiguity of ciphertext creation at a much finer granularity than an entire public key or IBE identity at a time. In particular, we will create an encryption system where we can modify the way individual ciphertexts are created one at a time and leave the distribution of the public parameters unmolested throughout our reductions. In order to do this, we devise an encryption system with two methods of encrypting a message M. In addition to the standard method, there will exist an alternative method of encrypting a message M such that the ciphertext will information theoretically hide the message M. Achieving this property alone under the standard IND-CPA definition is not necessarily difficult, but in proving SOA security we will have the challenge that it should be difficult to tell whether the standard encryption algorithm or the alternative ambiguous algorithm was used *even when the encryption randomness is revealed*. The most natural way to meet this requirement involves introducing a negligible correctness error into the system. After we conduct a series of hybrid experiments in which all of the encryptions use the alternative ambiguous method, we can apply an information theoretic argument in a similar manner to BHY.

We create our IBE-with-alternative-associated-encryption system by introducing IBE schemes that have a property we call *one-sided public openability*. In short, a one-bit IBE scheme 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 par, id, r results in the ciphertext C. We emphasize that this opening must be done without the aid of any secret information. If a 1SPO one-bit IBE scheme is also IND-CPA, it is easy to see that encryptions of message 0 can also be opened to a 1 (or else it would be possible to distinguish encryptions of 0 from encryptions of 1). As indicated above, we then use our one-bit 1SPO IBE schemes to construct many-bit schemes in the standard way (i.e., by concatenating independent encryptions of each bit under the same parameters) and show that if the one-bit 1SPO scheme is IND-CPA secure, then the many-bit IBE scheme is SOA-secure.

In order to construct 1SPO IND-CPA schemes we apply some ideas from the Boyen and Waters [8] anonymous IBE scheme. In our SOA IBE system an encryption of 0 to a given identity, id, is the generation of a ciphertext to that identity using (a modification of) the Boyen-Waters encryption algorithm. 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 using what we call a publicly reversible 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 discussion reveals why we choose to build off of the

Boyen-Waters anonymous IBE scheme as opposed to other simpler IBE systems without random oracles. The main feature of the Boyen-Waters ciphertexts is that they have no detectable structure from an attacker that does not have a private key for *id*. In contrast, in the Boneh-Boyen IBE system [4] the attacker cannot recover the message, but it can test for structure between two group elements in well formed ciphertexts. Therefore we cannot simply create an encryption system by replacing these with random group elements.

Regarding achieving *public* openability, recall the encryption of a 1 in our scheme is five random group elements. Traditionally, one picks a random group element by choosing a random exponent and exponentiating a generator of the group to that value. Unfortunately, this is not usually a reversible process given only public parameters, as this requires computing discrete logarithms. Thus, our scheme has to use groups that have what we call publicly reversible sampling. To solve this problem we use techniques from hashing into elliptic curves. Specifically, we can use a modified version of the hash function from the BLS signature scheme [7].

Applying the results of [9] this also gives us deniable IBE, which may be of independent interest.

INDEPENDENT WORK. The PKE version of 1SPO was discovered and used by Fehr, Hofheinz, Kiltz, and Wee (FHKW) [15] to achieve SOA-CCA-secure PKE. Their work and ours are independent and concurrent. Both were submitted to Eurocrypt 2010, but their's was accepted and ours was rejected. FHKW do not consider IBE.

RELATED WORK. Canetti, Feige, Goldreich and Naor [10] 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 [13] introduced more efficient schemes but this restriction remained, and Nielsen [22] showed it was necessary. With partial erasures, more efficient solutions were provided by Canetti, Halevi and Katz [12].

Dwork, Naor, Reingold and Stockmeyer [14] 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 [1], 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 [9] 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.

In hindsight, the NCE and deniable encryption techniques are a natural approach to SOA. Yet, it has been almost 15 years before this seems to have been realized and exploited, by us for IBE and, as discussed above, by Fehr, Hofheinz, Kiltz, and Wee [15] for PKE.

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.

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 [3] 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 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 $a \xleftarrow{\$} \mathbb{G}$ we mean that we run $i \xleftarrow{\$} \mathsf{Rand}(p)$, 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 for all $a \in \mathbb{G}$

$$\Pr\left[\, a' = a \mid a' \neq \perp \,\right] = \frac{1}{|\mathbb{G}|}$$

where the probability is over $a' \xleftarrow{\$} \mathsf{Sample}$.

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

$$\mathsf{Dec}(par, sk, id, \mathsf{Enc}(par, id, M)) = M$$

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

Fig. 1. The IBE IND-CPA Game

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

Fig. 2. The DLIN game for the decisional linear assumption.

is $\geq 1-\epsilon$ for all $id \in \mathsf{IdSp}$, all $M \in \mathsf{MsgSp}$, all $(par, msk) \in [\mathsf{Pg}]$, and all $sk \in [\mathsf{Kg}(par, msk, id)]$, where the probability is taken over the coins used in encryption.

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 $\mathsf{Enc}(par, id, M[1]) \parallel \ldots \parallel \mathsf{Enc}(par, id, M[\ell])$, 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) [6]. 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 1. An IND-CPA adversary interacts with game INDCPA, querying **LR** only once and on an identity $id^* \in \mathsf{IdSp}$ 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 $\mathsf{Coins}(par, m)$. This is the set from which Enc draws its coins when encrypting message m using parameters par. Similarly, we let $\mathsf{Coins}(par, id, c, 1)$ be the set of coins $\{r \mid c = \mathsf{Enc}(par, id, 1; r)\}$.

The Decisional Linear Assumption. We will use the Decisional Linear Assumption [5] in our proofs of security. The decisional linear game DLIN is found in Figure 2. We say the DLIN-advantage of an adversary A against GP is

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

3 Security against Selective Opening Attacks

In this section we formalize SOA security for IBE, closely following the formalizations from [1]. 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

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

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**.

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 $\mathbf{c}$ is $\mathsf{Enc}(par, \mathbf{id}[i], \mathbf{m}[i]; \mathbf{r}[i])$, 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 $\mathbf{c}$ 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 $\mathbf{m}$, 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 $\mathbf{m}$ 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 $\mathbf{m}$ 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 $\mathbf{m}$, the set of challenge IDs ChID, the corrupt set I, and the output *out*.

4 One-Sided Public Opening

Before describing our IBE scheme 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

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

proc. NewMesg$(\mathbf{id}, \alpha)$:
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
  $\mathbf{r}[i] \leftarrow\!\!\$\ \mathsf{Coins}(par, \mathbf{m}[i])$
  $\mathbf{c}[i] \leftarrow \mathsf{Enc}(par, \mathbf{id}[i], \mathbf{m}[i]; \mathbf{r}[i])$
Return $\mathbf{c}$

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

proc. Extract(id):
If $id \in \mathrm{ChID}$ then return $\perp$
$\mathrm{ExID} \leftarrow \mathrm{ExID} \cup \{id\}$
$sk \leftarrow\!\!\$\ \mathsf{Kg}(par, msk, id)$
Return sk

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

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

proc. Initialize:
Return $\perp$

proc. NewMesg$(\mathbf{id}, \alpha)$:
$\mathrm{ChID} \leftarrow \mathrm{ChID} \cup \mathbf{id}$
$\mathbf{m} \leftarrow\!\!\$\ \mathcal{M}(\alpha)$
Return $\perp$

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

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

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

the following property: for all $par \in [\mathsf{Pg}]$, all $id \in \mathsf{IdSp}$, every $c \in [\mathsf{Enc}(par, id, 1)]$, and every $\bar{r} \in \mathsf{Coins}(par, id, c, 1)$,

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

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 $par \in [\mathsf{Pg}]$, all $id \in \mathsf{IdSp}$, every $c \in [\mathsf{Enc}(par, id, 1)]$, and every $\bar{r} \in \mathsf{Coins}(par, id, c, 1)$,

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

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 [9]. 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 [15] 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.

5 A 1SPO IBE Scheme

5.1 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 that are related similar to 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 [8] and using the Waters' hash function [25] 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. 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 $H(\mathbf{u}, id) = \mathbf{u}[0] \prod_{i=1}^{n} \mathbf{u}[i]^{id[i]}$, where $id[i]$ is the ith bit of string id. This is the Waters' hash function [25]. The following scheme has message space $\{0,1\}$ and identity space $\{0,1\}^n$:

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

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

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

The decryption algorithm $\mathsf{Dec}(par, sk, C)$ parses the secret key sk as $(d_0, d_1, d_2, d_3, d_4)$, parses C as $(C_0, C_1, C_2, C_3, C_4)$ and outputs 0 if $\prod_{i=0}^{4} \mathrm{e}(C_i, d_i) = 1_{\mathbb{G}_T}$ and outputs 1 otherwise. It is easy to see the scheme has completeness error $1/p^2$.

We claim the scheme is δ-1SPO where $\delta \leq 5\theta$. The algorithm $\mathsf{OpenToOne}$ simply runs Sample^{-1} on each of the five ciphertext components with independent failure probabilities θ. Further, we show in Section 7 that the scheme is IND-CPA secure.

5.2 Instantiation

The missing piece in the above construction is how to efficiently instantiate the PR sampler Sample (and, for our proofs, its inverse Sample^{-1}). For this, we rely on the techniques for hashing into elliptic curve groups, specifically the techniques from Boneh, Lynn, and Shacham (BLS) [7]. We review the relevant parts of their hash function below. Hashing techniques from [6] and [19] could also potentially be used.

The BLS Hash Function. In [7], the authors describe a hash function $H : \{0,1\}^* \to \mathbb{G}$, where $\mathbb{G}$ is a subgroup of an elliptic curve. We review the details of their hash function with minor notational differences.

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 $y^2 = f(x)$, and let $E(\mathbb{F}_q)$ denote the corresponding group of order m. Let $g \in E(\mathbb{F}_q)$ be a point with prime order p such that p^2 does not divide m. To build a hash function that hashes into the subgroup generated by g (call it $\mathbb{G}$), BLS first hash onto $\mathbb{F}_q \times \{0,1\}$ and

then apply a procedure (which we call $\mathsf{FqBitToGroup}$) to that hash value to try and get a point in $\mathbb{G}$. They then use this group to construct their signature scheme[3]. This is the group we will use for our scheme above. We now define $\mathsf{FqBitToGroup}$ and an inverse procedure, following [7].

$\underline{\mathsf{FqBitToGroup}(x, b)}$
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.
$\tilde{g} \leftarrow (x, y_b)$; $h \leftarrow (\tilde{g})^{m/p}$
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 to exponentiate with h. BLS also describe an inverse procedure that, given a group element $h \in \langle g \rangle$, finds a random (x, b) such that $\mathsf{FqBitToGroup}(x, b) = h$. In the procedure, let $z = (m/p)^{-1} \mod p$, which exists since m divides p but not p^2 (and thus $\gcd(p, m/p) = 1$). 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).

$\underline{\mathsf{FqBitToGroup}^{-1}(h, u)}$
$(x, y) = \tilde{h} \leftarrow u^p \cdot h^z$
Compute square roots y_0, y_1 of $f(x)$, where $y_0 < y_1$
If $y = y_0$ then output $(x, 0)$ else output $(x, 1)$

BLS show that if u is a uniform point in $E(\mathbb{G}_q)$ then the output (x, b) of $\mathsf{FqBitToGroup}^{-1}(h, u)$ is such that x is uniformly distributed in $\mathbb{F}_q$, b is uniform in $\{0, 1\}$, and applying $\mathsf{FqBitToGroup}(x, b)$ results in h.

Our PR Sampler. We can use the above two algorithms to create our PR sampler Sample and its inverse Sample^{-1}.

$\underline{\mathsf{Sample}}$
$h \leftarrow \perp$
For $i = 1$ to ρ do
  $(x_i \parallel b_i) \leftarrow\!\!\$\ \mathbb{Z}_q \times \mathbb{Z}_2$
  $h' \leftarrow \mathsf{FqBitToGroup}(x_i, b_i)$
  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
  $(x_i \parallel b_i) \leftarrow\!\!\$\ \mathbb{Z}_q \times \mathbb{Z}_2$
  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 $(\mathbb{Z}_q \times \mathbb{Z}_2)^\rho$. The inverse operation Sample^{-1} is similar to the way random oracle queries are answered in the BLS proof [7]. 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

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

finding h. If it does not fail on input $h \in \mathbb{G}$ it is easy to see it outputs correctly-distributed randomness for Sample.

6 Selective Opening Security from One-Side Openable IBE

We now state and prove 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. We prove the following theorem.

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* $\mathsf{T}(\mathcal{B}) = \mathcal{O}(\mathsf{T}(\mathcal{A}) + \mathsf{T}(\mathcal{M}) + k\ell \cdot \mathsf{T}(\mathsf{Enc}) + k\ell \cdot \mathsf{T}(\mathsf{OpenToOne}) + \mathsf{T}(\mathcal{R}))$. □

Proof. Let $\mathcal{A}$ be an arbitrary soa-adversary against Π^ℓ. We describe a simulator $\mathcal{S}$ for $\mathcal{A}$ in Figure 5. 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 $\mathbf{NewMesg}(\mathbf{id}, \alpha)$ 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 $\mathbf{Corrupt}(I)$ 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 $\mathsf{OpenToOne}$ 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 5,6, and 7. 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 $\mathsf{OpenToOne}$. If $\mathsf{OpenToOne}$ fails, set bad flag and resample as in G_1.
- G_3: Same as above except if $\mathsf{OpenToOne}$ 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 $\mathsf{OpenToOne}$ 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.

More formally, we first claim that

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

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

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

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

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

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

proc. Initialize: All Games
$(par, msk) \leftarrow\!\!\$\ \mathsf{Pg}^{\ell}$
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
    $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{Coins}(par, \mathbf{m}[i][j])$
    $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(par, \mathbf{id}[i], \mathbf{m}[i][j]; \mathbf{r}[i][j])$
Return $\mathbf{c}$

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

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

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

Fig. 5. Simulator $\mathcal{S}$ and the start of the game sequence.

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

$$\Pr\left[\, H_0^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] - \Pr\left[\, H_{k\ell}^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] \leq k\ell \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(\mathcal{B}) \ , \tag{1}$$

where adversary $\mathcal{B}$ is described in Figure 8. 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 $v+1$

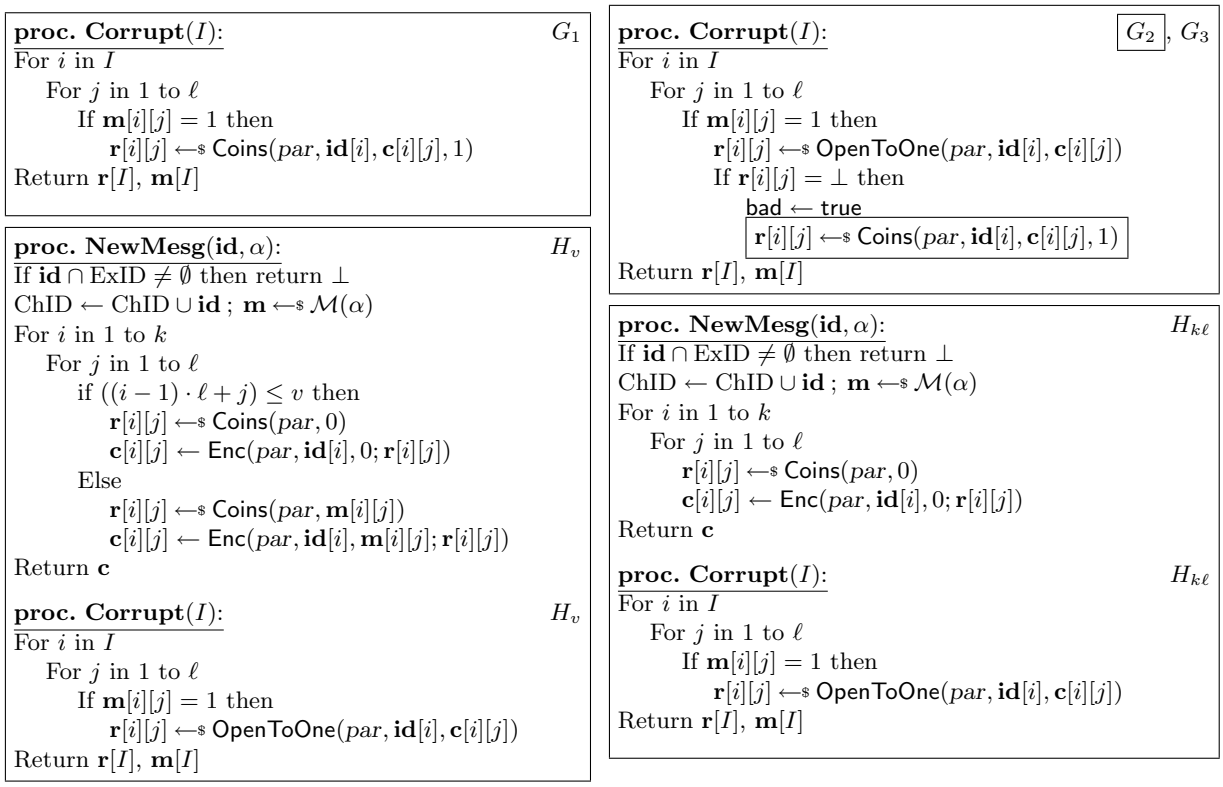

Fig. 6. Games for the proof of Theorem 1.

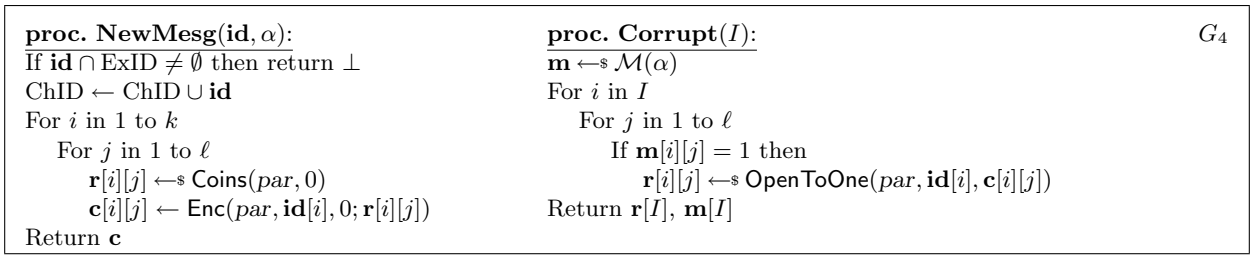

Fig. 7. The last game G_4.

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

$$\begin{aligned}
&\Pr\left[\, H^{\mathcal{A}}_{v} \,\right] - \Pr\left[\, H^{\mathcal{A}}_{v+1} \,\right] \\
&= \left(\Pr\left[\, H^{\mathcal{A}}_{v} \wedge \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] + \Pr\left[\, H^{\mathcal{A}}_{v} \wedge \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v}) \,\right]\right) - \\
&\quad \left(\Pr\left[\, H^{\mathcal{A}}_{v+1} \wedge \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v+1}) \,\right] + \Pr\left[\, H^{\mathcal{A}}_{v+1} \wedge \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v+1}) \,\right]\right) \\
&= \Pr\left[\, \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] \cdot \left(\Pr\left[\, H^{\mathcal{A}}_{v} \mid \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] - \Pr\left[\, H^{\mathcal{A}}_{v+1} \mid \mathsf{M0}_{v+1}(H^{\mathcal{A}}_{v+1}) \,\right]\right) + \\
&\quad \Pr\left[\, \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] \cdot \left(\Pr\left[\, H^{\mathcal{A}}_{v} \mid \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v}) \,\right] - \Pr\left[\, H^{\mathcal{A}}_{v+1} \mid \mathsf{M1}_{v+1}(H^{\mathcal{A}}_{v+1}) \,\right]\right) \ . \qquad (2)
\end{aligned}$$

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 (2) becomes zero. This means that,

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

alg.$\mathcal{B}(par)$:

$v \leftarrow\!\!\$\ \{0,\ldots,k\ell-1\}$
Run $\mathcal{A}(par)$.

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
      $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{Coins}(par,0)$; $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(par,\mathbf{id}[i],0;\mathbf{r}[i][j])$
    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
        $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{Coins}(par,\mathbf{m}[i][j])$; $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(par,\mathbf{id}[i],\mathbf{m}[i][j];\mathbf{r}[i][j])$
    Else
      $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{Coins}(par,\mathbf{m}[i][j])$; $\mathbf{c}[i][j] \leftarrow \mathsf{Enc}(par,\mathbf{id}[i],\mathbf{m}[i][j];\mathbf{r}[i][j])$
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 $\mathbf{r}[i][j] \leftarrow\!\!\$\ \mathsf{OpenToOne}(par,\mathbf{id}[i],\mathbf{c}[i][j])$
Return $(\mathbf{r}[I],\mathbf{m}[I])$

On query $\mathbf{Extract}(id)$:

Return $\mathbf{Extract}_{\mathcal{B}}(id)$

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

Fig. 8. IND-CPA adversary $\mathcal{B}$

and thus

$$\Pr\left[\,H_0^{\mathcal{A}}\,\right] - \Pr\left[\,H_{k\ell}^{\mathcal{A}}\,\right] = \sum_{v=0}^{k\ell-1} \Pr\left[\,H_v^{\mathcal{A}}\,\right] - \Pr\left[\,H_{v+1}^{\mathcal{A}}\,\right]$$
$$= \sum_{v=0}^{k\ell-1} \Pr\left[\,\mathsf{M1}_{v+1}(H_v^{\mathcal{A}})\,\right] \cdot \left(\Pr\left[\,H_v^{\mathcal{A}} \mid \mathsf{M1}_{v+1}(H_v^{\mathcal{A}})\,\right] - \Pr\left[\,H_{v+1}^{\mathcal{A}} \mid \mathsf{M1}_{v+1}(H_{v+1}^{\mathcal{A}})\,\right]\right)\ , \tag{4}$$

Consider again IND-CPA adversary $\mathcal{B}$ in Figure 8. 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 (4) justifies (1).

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 5 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. □

7 IND-CPA Security of LoR

Theorem 1 allows us to achieve SOA-secure IBE given a 1-bit 1SPO, IND-CPA scheme. We have seen that LoR is 1SPO. It remains to show it 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 $n \geq 1$, 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

$$\mathbf{Adv}^{\text{dlin}}_{\mathsf{GP}}(\mathcal{B}) \geq \frac{\delta^2}{9qn + 3\delta} \text{ , and} \tag{5}$$

$$\mathsf{T}(\mathcal{B}) = \mathsf{T}(\mathcal{A}) + \mathsf{T}_{\text{sim}}(n, q) \tag{6}$$

where

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

□

The proof of the theorem combines techniques from [8, 25, 2] and can be found in Appendix A.

References

1. M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, number 5479 in Lecture Notes in Computer Science, pages 1–35. Springer, 2009.
2. M. Bellare and T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme. In *Advances in Cryptology – EUROCRYPT 2009*, number 5479 in Lecture Notes in Computer Science. Springer, 2009.
3. M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. In *Advances in Cryptology – EUROCRYPT 2006*, number 4004 in Lecture Notes in Computer Science, pages 409–426. Springer, 2006.
4. D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT 2004*, pages 223–238. Springer, 2004.
5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO 2004*, number 3152 in LNCS, pages 41–55. Springer, 2004.
6. D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comp.*, 32(3):586–615, 2003.
7. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

8. X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology – CRYPTO 2006*. Springer, 2005.
9. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In *Advances in Cryptology – CRYPTO 1997*, pages 90–104. Springer, 1997.
10. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *Twenty-Eighth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1995*, pages 639–648. ACM Press, 1996.
11. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2004*. Springer, 2004.
12. R. Canetti, S. Halevi, and J. Katz. Adaptively-secure, non-interactive public-key encryption. In J. Kilian, editor, *Theory of Cryptography, Proceedings of TCC 2005*, number 3378 in Lecture Notes in Computer Science, pages 150–168. Springer-Verlag, 2005.
13. I. Damgård and J. B. Nielsen. Improved non-committing encryption schemes based on general complexity assumptions. In M. Bellare, editor, *Advances in Cryptology, Proceedings of CRYPTO 2000*, number 1880 in Lecture Notes in Computer Science, pages 432–450. Springer-Verlag, 2000.
14. C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.
15. S. Fehr, D. Hofheinz, E. Kiltz, and H. Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In *Advances in Cryptology – EUROCRYPT 2010*. Springer, 2010. To Appear.
16. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2), 1984.
17. B. Hemenway and R. Ostrovsky. Lossy encryption from general assumptions. IACR ePrint Archive Report 2009/088.
18. D. Hofheinz. Possibility and impossibility results for selective decommitments. IACR ePrint Archive, Apr. 2008.
19. T. Icart. How to hash into elliptic curves. In *Advances in Cryptology – CRYPTO 2009*, pages 303–316. Springer, 2009.
20. G. Kol and M. Naor. Cryptography and game theory: Designing protocols for exchanging information. In *TCC 2008*, pages 320–339. Springer, 2008.
21. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Twelfth Annual Symposium on Discrete Algorithms, Proceedings of SODA 2001*, pages 448–457. ACM/SIAM, 2001.
22. J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology – CRYPTO 2002*, pages 111–126. Springer, 2002.
23. C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology – CRYPTO 2008*, pages 554–571. Springer, 2008.
24. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *Fortieth Annual ACM Symposium on Theory of Computing – STOC 2008*, pages 187–196. ACM Press, 2008.
25. B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Crypology – EUROCRYPT 2005*, pages 114–127, 2005.

A 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 [2] which gave an alternative proof to Waters' IBE scheme [25]. Consider the games in Figures 9,10, and 11. We have

$$\Pr\left[\, G_2^{\mathcal{A}} \,\right] - \Pr\left[\, G_3^{\mathcal{A}} \,\right] \leq \Pr\left[\, \mathsf{BAD}(G_3^A) \,\right] \tag{7}$$

$$\leq \frac{2^n}{p} \tag{8}$$

Games G_2, G_3 are identical until bad so (7) is by the Fundamental Lemma of game playing [3]. To justify (8) first note $H(\mathbf{u}, id) = g^{t_1 \mathsf{F}(\mathbf{x},id)+\mathsf{G}(\mathbf{y},id)}$. Thus, the event $\mathsf{BD}(G_3^{\mathcal{A}})$ happens when $t_1\mathsf{F}(\mathbf{x}, id) + \mathsf{G}(\mathbf{y}, id) \equiv 0 \pmod p$. Fix t_1 and $\mathbf{x}$. For any particular id, the probability over $\mathbf{y}$ that $\mathsf{G}(\mathbf{y}, id) + \mathsf{F}(\mathbf{x}, id)t_1 \equiv 0 \pmod p$ is $1/p$. But the number of choices of id is 2^n so we conclude by the union bound. Now we have

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

proc. Initialize: Games G_0, G_1, G_2, G_3

$c \leftarrow\$ \{0,1\}$; $\mathbf{u} \leftarrow\$ \mathbb{G}^{n+1}$
$t_1, t_2, t_3, t_4 \leftarrow\$ \mathbb{Z}_p^*$
For $i = 1, 2, 3, 4$ do $v_i \leftarrow g^{t_i}$
$par \leftarrow (\mathbf{u}, v_1, v_2, v_3, v_4)$; $msk \leftarrow (t_1, t_2, t_3, t_4)$
Return par

proc. Extract(id): Games G_0, G_1, G_2, G_3

$r_1, r_2 \leftarrow\$ \mathbb{Z}_p$
$d_0 \leftarrow g^{r_1 t_1 t_2 + r_2 t_3 t_4}$
$d_1 \leftarrow H(\mathbf{u}, id)^{-r_1 t_2}$; $d_2 \leftarrow H(\mathbf{u}, id)^{-r_1 t_1}$
$d_3 \leftarrow H(\mathbf{u}, id)^{-r_2 t_4}$; $d_4 \leftarrow H(\mathbf{u}, id)^{-r_2 t_3}$
Return $(d_0, d_1, d_2, d_3, d_4)$

proc. LR(id, M_0, M_1): Game G_0

If $M_c = 0$ then
  $s, s_1, s_2 \leftarrow\$ \mathbb{Z}_p$
  $C_0 \leftarrow H(\mathbf{u}, id)^s$
  $C_1 \leftarrow v_1^{s-s_1}$; $C_2 \leftarrow v_2^{s_1}$
  $C_3 \leftarrow v_3^{s-s_2}$; $C_4 \leftarrow v_4^{s_2}$
Else
  $C_0, C_1, C_2, C_3, C_4 \leftarrow\$ \mathbb{G}$
Return $(C_0, C_1, C_2, C_3, C_4)$

proc. Finalize(c'): Games G_0, G_1, G_2, G_3

Return $(c = c')$

proc. LR(id, M_0, M_1): Game G_1

If $M_c = 0$ then
  $s, s_1, s_2 \leftarrow\$ \mathbb{Z}_p$
  $C_0 \leftarrow H(\mathbf{u}, id)^s$
  $C_1 \leftarrow\$ \mathbb{G}$; $C_2 \leftarrow v_2^{s_1}$
  $C_3 \leftarrow v_3^{s-s_2}$; $C_4 \leftarrow v_4^{s_2}$
Else
  $C_0, C_1, C_2, C_3, C_4 \leftarrow\$ \mathbb{G}$
Return $(C_0, C_1, C_2, C_3, C_4)$

proc. LR(id, M_0, M_1): Games G_2, $\boxed{G_3}$

If $M_c = 0$ then
  $s, s_1, s_2 \leftarrow\$ \mathbb{Z}_p$
  $C_0 \leftarrow H(\mathbf{u}, id)^s$
  If $(H(\mathbf{u}, id) = 1_{\mathbb{G}})$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{C_0 \leftarrow\$ \mathbb{G}}$
  $C_1 \leftarrow\$ \mathbb{G}$; $C_2 \leftarrow v_2^{s_1}$
  $C_3 \leftarrow\$ \mathbb{G}$; $C_4 \leftarrow v_4^{s_2}$
Else
  $C_0, C_1, C_2, C_3, C_4 \leftarrow\$ \mathbb{G}$
Return $(C_0, C_1, C_2, C_3, C_4)$

Fig. 9. Game transitions

However $\Pr\left[\, G_3^A \,\right] = \frac{1}{2}$. Using this and (8) we have

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

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

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

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

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

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 (9) based on [8, 2]. The design and analysis of $\mathcal{B}_2$ is similar and omitted.

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 $\mathbf{x}[j] \leftarrow\!\!\$\ [0\,..\,m-1]$
  $\mathbf{u}[j] \leftarrow v_1^{\mathbf{x}[j]} g^{\mathbf{y}[j]}$
$par \leftarrow (g, \mathbf{u}, v_1, v_2, v_3, v_4)$
Run $\mathcal{A}(par)$, answering queries by

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

On **query LR**(id, M_0, M_1):
$C \leftarrow \bot$
If $\mathsf{F}(\mathbf{x}, id) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else
  $C \leftarrow\!\!\$\ \mathsf{EncS}(g, C_1, C_2, v_3, v_4, t_3, W, \mathbf{y}, id, M_c)$
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 $\mathbf{x}[j] \leftarrow\!\!\$\ [0\,..\,m-1]$
  $\mathbf{u}[j] \leftarrow v_1^{\mathbf{x}[j]} g^{\mathbf{y}[j]}$
$par \leftarrow (g, \mathbf{u}, v_1, v_2, v_3, v_4)$; $msk \leftarrow (t_1, t_2, t_3, t_4)$
Return par

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

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

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

Else $C \leftarrow\!\!\$\ \mathsf{EncS}(g, C_1, C_2, v_3, v_4, t_3, W, \mathbf{y}, id, M_c)$
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. 10. Adversary $\mathcal{B}$ and the continuation of the game sequence.

Simulation subroutines. Towards the proof we begin with some definitions from [2]. 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 $id \in \{0,1\}^n$ we let

$$\mathsf{F}(\mathbf{x}, id) = \mathbf{x}[0] + \sum_{i=1}^{n} \mathbf{x}[i] id[i] \quad \text{and} \quad \mathsf{G}(\mathbf{y}, id) = \mathbf{y}[0] + \sum_{i=1}^{n} \mathbf{y}[i] id[i] \bmod p\,. \tag{10}$$

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 10. We now describe the subroutines it utilizes to answer **Extract** and **LR** queries. We define the following procedures:

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

proc. LR(id, M_0, M_1): Game G_6
If $\mathsf{F}(\mathbf{x}, id) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $M_c = 0$ then
  $t \leftarrow\!\!\$\ \mathbb{Z}_p$; If $d = 1$ then $t \leftarrow s$
  $C_0 \leftarrow H(\mathbf{u}, id)^t$
  $C_3 \leftarrow v_3^{s-s_2}$; $C_4 \leftarrow v_4^{s_2}$
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 $\mathbf{x}[j] \leftarrow\!\!\$\ [0\,..\,m-1]$
  $\mathbf{y}[j] \leftarrow \mathbf{z}[j] - t_1 \cdot \mathbf{x}[j] \mod p$
$par \leftarrow (g, \mathbf{u}, v_1, v_2, v_3, v_4)$
$msk \leftarrow (t_1, t_2, t_3, t_4)$
Return par

proc. Finalize(c'): Game G_7
For $j = 0$ to cnt do
  If $\mathsf{F}(\mathbf{x}, id_j) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $\mathsf{F}(\mathbf{x}, id_0) \neq 0$ 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
  $\mathbf{z}[j] \leftarrow\!\!\$\ \mathbb{Z}_p$; $\mathbf{u}[j] \leftarrow g^{\mathbf{z}[j]}$
$par \leftarrow (g, \mathbf{u}, v_1, v_2, v_3, v_4)$
$msk \leftarrow (t_1, t_2, t_3, t_4)$
Return par

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

proc. LR(id, M_0, M_1): Game G_7, G_8
$id_0 \leftarrow id$
If $M_c = 0$ then
  $t \leftarrow\!\!\$\ \mathbb{Z}_p$; If $d = 1$ then $t \leftarrow s$
  $C_0 \leftarrow H(\mathbf{u}, id)^t$
  $C_3 \leftarrow v_3^{s-s_2}$; $C_4 \leftarrow v_4^{s_2}$
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 $\mathsf{F}(\mathbf{x}, id_j) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $\mathsf{F}(\mathbf{x}, id_0) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $(c = c')$ then return 1 else return 0

Fig. 11. Game transitions

proc. $\mathsf{KgS}(g, v_1, v_2, t_3, t_4, \mathbf{u}, \mathbf{x}, \mathbf{y}, id)$:
$r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_p$
$d_0 \leftarrow g^{r_2 t_3 t_4} v_2^{r_1}$
If $H(\mathbf{u}, id) \neq 1_{\mathbb{G}}$ then
  $d_1 \leftarrow v_2^{-r_1 \cdot \mathsf{F}(\mathbf{x}, id)}$
  $d_2 \leftarrow v_1^{-r_1 \cdot \mathsf{F}(\mathbf{x}, id)}$
  $d_3 \leftarrow H(\mathbf{u}, id)^{-r_2 t_4} v_2^{-r_1 \cdot \mathsf{G}(\mathbf{y}, id)/t_3}$
  $d_4 \leftarrow H(\mathbf{u}, id)^{r_2 t_3} v_2^{-r_1 \cdot \mathsf{G}(\mathbf{y}, id)/t_4}$
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. $\mathsf{EncS}(g, C_1, C_2, v_3, v_4, t_3, W, \mathbf{y}, id, M)$:
If $M = 0$ then
  $s_2 \leftarrow\!\!\$\ \mathbb{Z}_p$
  $C_0 \leftarrow W^{\mathsf{G}(\mathbf{y}, id)}$
  $C_3 \leftarrow W^{t_3} v_3^{-s_2}$
  $C_4 \leftarrow v_4^{s_2}$
Else
  $C_0, C_3, C_4 \leftarrow\!\!\$\ \mathbb{G}$
Return $(C_0, C_1, C_2, C_3, C_4)$

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 [8] for use with the Waters' hash.

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

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

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

and

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

These are well defined because $t_1\mathsf{F}(\mathbf{x}, id) + \mathsf{G}(\mathbf{y}, id)$ 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

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

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

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

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

we have $H(\mathbf{u}, id) = 1_{\mathbb{G}}$. 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 10 and 11. 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 [2] 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 [2, 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

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

□