A preliminary version of this paper appears in *Advances in Cryptology – EUROCRYPT '09*, Lecture Notes in Computer Science Vol. —, A. Joux ed., Springer-Verlag, 2009. This is the full version.

Simulation without the Artificial Abort: Simplified Proof and Improved Concrete Security for Waters' IBE Scheme

Mihir Bellare* Thomas Ristenpart†

February 2009

Abstract

Waters' variant of the Boneh-Boyen IBE scheme is attractive because of its efficency, applications, and security attributes, but suffers from a relatively complex proof with poor concrete security. This is due in part to the proof's "artificial abort" step, which has then been inherited by numerous derivative works. It has often been asked whether this step is necessary. We show that it is not, providing a new proof that eliminates this step. The new proof is not only simpler than the original one but offers better concrete security for important ranges of the parameters. As a result, one can securely use smaller groups, resulting in significant efficiency improvements.

*Dept. of Computer Science & Engineering 0404, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA. Email: mihir@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/mihir. This work was supported in part by NSF grants CNS 0524765 and CNS 0627779 and a gift from Intel corporation.

†Dept. of Computer Science & Engineering 0404, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA. Email: tristenp@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/tristenp. Supported in part by above-mentioned grants of first author.

Contents

1 Introduction

The importance of identity-based encryption (IBE) as a cryptographic primitive stems from its widespread deployment and the numerous applications enabled by it. Since the initial work on providing realizations of IBE [8, 17], improving the efficiency, security, and extensibility of the fundamental primitive has consequently received substantial attention from the research community. A challenging problem has been to arrive at a practical IBE scheme with a tight security reduction under standard assumptions. (The most attractive target being DBDH without relying on random oracles.) While a typical approach for progressing towards this goal is proposing new constructions, in this paper we take another route: improving the concrete security of existing constructions. This requires providing better proofs of security and analyzing the impact of their tighter reductions.

Why concrete security? Informally speaking, consider an IBE scheme with a security reduction showing that attacking the scheme in time t with success probability ϵ implies breaking some believed-to-be hard problem in time $t + \omega_1$ with success probability $\epsilon' \geq \epsilon/\omega_2$. Tightness of the reduction refers to the value of ω_1 (the overhead in time needed to solve the hard problem using the scheme attacker) and of ω_2 (the amount by which the success probability decreases). Unlike asymptotic treatments, provably-secure IBE has a history of utilizing concrete security, meaning specifying ω_1 and ω_2 explicitly. Concrete-security for IBE started with Boneh and Franklin [8] and has been continued in subsequent works, e.g. [6, 7, 9, 36, 21, 25] to name just a few.

As Gentry points out [21], concrete security and tight reductions are not just theoretical issues for IBE, rather they are of utmost practical import: the speed of implementations increases as ω_1 and/or ω_2 decrease. This is because security guarantees are lost unless the size of groups used to implement a scheme grow to account for the magnitude of these values. In turn group size dictates performance: exponentiations in a group whose elements can be represented in r bits takes roughly $\mathcal{O}(r^3)$ time. As a concrete example, this means that performing four 160-bit group exponentiations can be significantly faster than a *single* 256-bit group exponentiation. In practice even a factor of two efficiency slow-down is considered significant (let alone a factor of four), so finding as-tight-as-possible reductions is crucial.

Overview of IBE approaches. All practical IBE systems currently known are based on bilinear pairings. We can partition the space of such systems along two dimensions, as shown in the left table of Figure 1. In one dimension is whether one utilizes random oracles or not. In the other is the flavor of hard problem used, whether it be the basic bilinear Diffie-Hellman (BDH) assumption [8] or a q-dependent assumption such as q-BDHI [6]. Of note is that Katz and Wang [25], in the "random oracle/BDH" setting, and Gentry [21], in the "no random oracle/q-dependent setting", have essentially solved the problem of finding practical schemes with tight reductions. On the other hand, finding practical schemes with tight reductions in the "no random oracle/BDH" setting represents a hard open problem mentioned in numerous works [6, 7, 36, 21]. This last setting turns out to be attractive for two reasons. First, from a security perspective, it is the most conservative (and consequently most challenging) with regard to choice of assumptions. Second, schemes thus far proposed in this setting follow a framework due to Boneh and Boyen [6] (so-called "commutative blinding") that naturally supports many valuable extensions: hierarchical IBE [24], attribute-based IBE [34], direct CCA-secure encryption [10, 26], etc.

Progress in this setting is summarized in the right table of Figure 1. Boneh and Boyen initiated work here with the BB_1 scheme (the first scheme in [6]). They prove it secure under the decisional BDH (DBDH) assumption, but in the selective-ID attack model of [12] in which adversaries must commit to a targeted identity before seeing the IBE system's parameters. Boneh and Boyen show how to prove full security, but the reduction doing so is exponentially loose (briefly, because it

	q-dependent	BDH
RO model	SK	BF , KW
Standard model	BB_2 , Ge	BB_1 , Wa

Scheme	Security	Reduction
BB_1	selective-ID	polynomial
BB_1	full	exponential
Wa	full	polynomial

Figure 1: A comparison of practical IBE schemes. BF is the Boneh-Franklin scheme [8]; SK is the Sakai-Kasahara scheme [35, 16]; KW is the Katz-Wang scheme [25]; BB_1 and BB_2 are the first and second Boneh-Boyen schemes from [6]; Wa is Waters' scheme [36]; and Ge is Gentry's scheme [21]. **(Left)** The assumptions (an q-dependent assumption versus bilinear Diffie-Hellman) and model (random oracles or not) used to prove security of the schemes. **(Right)** Types of security offered by standard model BDH-based systems and asymptotic reduction tightness.

requires guessing the hash of the to-be-attacked identity).

Waters' proposed a variant of BB_1 that we'll call Wa [36]. This variant requires larger public parameters, but can be proven fully secure with a polynomial reduction to DBDH that does not use random oracles. The relatively complex security proof relies on a novel "artificial abort" step, that, while clever, is unintuitive. It also significantly hurts the concrete security and, thereby, efficiency of the scheme. Many researchers in the community have asked whether artificial aborts can be dispensed with, but the general consensus seems to have been that the answer is "no" and that the technique is somehow fundamental to proving security. This folklore assessment (if true) is doubly unfortunate because Wa, inheriting the flexibility of the Boneh-Boyen framework, has been used in numerous diverse applications [10, 1, 5, 30, 13, 14, 22, 26]. As observed in [26], some of these subsequent works offer difficult to understand (let alone verify) proofs, due in large part to their use of the artificial abort technique in a more-or-less black-box manner. They also inherit its concrete security overhead.

This paper. Our first contribution is to provide a novel proof of Waters' variant that completely eliminates the artificial abort step. The proof, which uses several new techniques and makes crucial use of code-based games [4], provides an alternate and (we feel) more intuitive and rigorous approach to proving the security of Wa. Considering the importance of the original proof (due to its direct or indirect use in [10, 1, 5, 30, 13, 14, 22, 26]), a more readily understood proof is already a significant contribution. Our reduction (like Waters') is not tight, but as we see below it offers better concrete security for many important parameter choices, moving us closer to the goal of standard model BDH-based schemes with tight reductions. The many Waters'-derived works [10, 1, 5, 30, 13, 14, 22, 26] inherit the improvements in concrete security. We briefly describe these derivatives in Appendix A.

We now have the BB_1 and Wa schemes, the former with an exponentially-loose reduction and the latter with now two polynomial reductions each having a complex concrete security formula. What is the most efficient approach for providing provably-secure DBDH-based IBE? Since we want to account for the impact of reduction tightness, answering this question requires work. We offer a framework for computing the concrete efficiency of reductions, adopting techniques from [28, 27, 20]. Efficiency is measured by mapping desired (provable) security levels to requisite group sizes. Not only does this approach provide a metric for comparing different reductions, it also allows comparing the resultant bit-operation speed of schemes when each is instantiated in groups of size sufficient to account for the reduction. Providing such a framework that simultaneously provides simplicity, accuracy, and fairness (i.e. not biased towards particular schemes/reductions) turned out to be very

κ	ϵ	q	s_{BB}	s_W	s_{BR}	$\mathbf{T}_{\mathsf{Enc}}(s_W)/\mathbf{T}_{\mathsf{Enc}}(s_{BR})$
60	2^{-20}	2^{20}	192	192	128	**9**
70	2^{-20}	2^{20}	256	192	128	**9**
80	2^{-30}	2^{30}	256	256	192	**5**
90	2^{-30}	2^{30}	–	256	192	**5**
100	2^{-10}	2^{10}	–	128	192	1/9
100	2^{-40}	2^{40}	–	256	192	**5**
192	2^{-40}	2^{40}	–	256	–	–

Figure 2: Table showing the security level of the pairing setups required to achieve κ-bits of security for the BB_1 and Wa encryption schemes when adversaries achieve ϵ success probability using q key extraction queries. Loosely speaking, the security level of the pairing setup is $(\log p)/2$ where p is the size of the first pairing group. Here s_{BB}, s_W, s_{BR} are, respectively, the securities of the pairing setups for BB_1, Wa under Waters' reduction, and Wa under the new reduction. The final column represents the (approximate) ratio of encryption times for Wa as specified by the two reductions. A dash signifies that one needs a pairing setup of security greater than 256.

challenging.

Let us first mention the high-level results, before explaining more. In the end our framework implies that Waters' variant usually provides faster standard model encryption (than BB_1). Our new proof provides a better reduction for low to mid range security parameters, while Waters' reduction is tighter for higher security parameters. The new reduction in fact drastically improves efficiency in the former category, offering up to 9 times faster encryption for low parameters and 5 times faster encryption for mid-range security levels. Where Waters' reduction is tighter, we can continue to choose group size via it; the new reduction never *hurts* efficiency.

BB_1 does better than Wa when identities are short, such as $n = 80$ bits. We have, however, focused on providing IBE with arbitrary identity spaces, which provides the most versatility. Supporting long identities (e.g. email addresses such as `john.doe123@anonymous.com`) requires utilizing a collision-resistant hash function to compress identities. In this case, the birthday bound mandates that the bit length n of hash outputs be double the desired security level, and this affects the BB_1 scheme more due to its reduction being loose by a factor of 2^n.

Framework details. We present some results of applying our framework in Figure 2. Let us explain briefly what the numbers signify and how we derived them. (Details are in Section 4.) By a *setup* we mean groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ admitting a bilinear map $\mathbf{e}$: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The setup provides security s (bits) if the best known algorithms to solve the discrete logarithm (DL) problem take at least 2^s time in any of the three groups. We assume (for these estimates but not for the proof!) that the best algorithm for solving DBDH is solving DL in one of the groups. An important practical issue in pairings-based cryptography is that setups for arbitrary security are not known. Accordingly, we will restrict attention to values $s = 80$, 112, 128, 192, and 256, based on information from [31, 27, 28, 18, 19]. Figure 6 in Appendix E shows representation sizes of the corresponding groups. Now we take as our target that the IBE scheme should provide κ bits of security. By this we mean that any adversary making at most $q = 1/\epsilon$ **Extract** queries and having running time at most $\epsilon 2^\kappa$ should have advantage at most ϵ. For each scheme/reduction pair we can then derive the security s of the underlying pairing setup required to support the desired level of security. See Figure 2 for BB_1 (s_{BB}), Wa under Waters' reduction (s_W), and under the new reduction (s_{BR}).

OTHER RELATED WORK AND OPEN PROBLEMS. Recently Hofheinz and Kiltz describe programmable hash functions [23]. Their main construction uses the same hash function (originally due to Chaum et al. [15]) as Waters', and they provide new proof techniques that provide a $\sqrt{n}$ (n is the length of identities) improvement on certain bounds that could be applicable to Wa. But this will only offer a small concrete security improvement compared to ours. Moreover, their results are asymptotic and hide (seemingly very large) unknown constants.

As mentioned, providing a scheme based on DBDH that has a tight security reduction (without random oracles) is a hard open problem, and one that remains after our work. (One reason we explain this is that we have heard it said that eliminating the artificial abort would solve the open problem just mentioned, but in fact the two seem to be unrelated.) Finding a tight reduction for Waters' (or another $\mathsf{BB_1}$-style scheme) is of particular interest since it would immediately give a hierarchical IBE (HIBE) scheme with security beyond a constant number of levels (the best currently achievable). From a practical point of view we contribute here, since better concrete security improves the (constant) number of levels achievable. From a theoretical perspective, this remains an open problem.

VIEWING PROOFS AS QUALITATIVE. We measure efficiency of schemes when one sets group size according to the best-known reduction. However, the fact that a proof implies the need for groups of certain size to guarantee security of the scheme does not mean the scheme is necessarily insecure (meaning there is an attack) over smaller groups. It simply means that the proof tells us nothing about security in these smaller groups. In the context of standards it is sometimes suggested one view a proof as a qualitative rather than quantitative guarantee, picking group sizes just to resist the best known attack. Our sense is that this procedure is not viewed as ideal even by its proposers but rather forced on them by the looseness of reductions. To rectify this gap, one must find tighter reductions, and our work is a step to this end.

2 Definitions and Background

NOTATION. We fix *pairing parameters* $\mathsf{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathbf{e}, \psi)$ where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are groups of prime order p; $\mathbf{e}$: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate, efficiently computable bilinear map; and ψ: $\mathbb{G}_2 \to \mathbb{G}_1$ is an efficiently computable isomorphism [8]. Let $\mathsf{T}_{\exp}(\mathbb{G})$ denote the time to compute an exponentiation in a group $\mathbb{G}$. Similarly, let $\mathsf{T}_{\mathrm{op}}(\mathbb{G})$ denote the time to compute a group operation in a group $\mathbb{G}$. Let T_ψ denote the time to compute ψ. Let $\mathbb{G}^* = \mathbb{G} - \{\mathbf{1}\}$ denote the set of generators of $\mathbb{G}$ where $\mathbf{1}$ is the identity element of $\mathbb{G}$.

Vectors are written in boldface, e.g. $\mathbf{u} \in \mathbb{Z}_p^{n+1}$ is a vector of $n+1$ values each in $\mathbb{Z}_p$. We denote the i^{th} component of a vector $\mathbf{u}$ by $\mathbf{u}[i]$. If $S \in \{0,1\}^*$ then $|S|$ denotes its length and $S[i]$ denotes its i^{th} bit. For integers i, j we let $[i\,..\,j] = \{i, \ldots, j\}$. The running time of an adversary $\mathcal{A}$ is denoted $\mathbf{T}(\mathcal{A})$. We use big-oh notation with the understanding that this hides a small, fixed, machine-dependent constant.

GAMES. Our security definitions and proofs use code-based games [4], and so we recall some background from [4]. A game (look at Figure 3 for examples) has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. A game G is executed with an adversary $\mathcal{A}$ as follows. First, **Initialize** executes, and its outputs are the inputs to $\mathcal{A}$. Then $\mathcal{A}$ executes, its oracle queries being answered by the corresponding procedures of G. When $\mathcal{A}$ terminates, its output becomes the input to the **Finalize** procedure. The output of the latter is called the output of the game, and we let $\mathrm{G}^{\mathcal{A}} \Rightarrow y$ denote the event that this game output takes value y. The boolean flag bad is assumed initialized to false. Games G_i, G_j are identical-until-bad

Game $\mathrm{DBDH}_{\mathsf{GP}}$

procedure Initialize:
$g_2 \xleftarrow{\$} \mathbb{G}_2^* \,;\, g_1 \leftarrow \psi(g_2) \,;\, a, b, s \xleftarrow{\$} \mathbb{Z}_p \,;\, d \xleftarrow{\$} \{0,1\}$
If $d = 1$ then $W \xleftarrow{\$} \mathbf{e}(g_1, g_2)^{abs}$ Else $W \xleftarrow{\$} \mathbb{G}_T$
Ret $(g_1, g_2, g_2^a, g_2^b, g_2^s, W)$

procedure Finalize(d'):
Ret $(d' = d)$

Game $\mathrm{IND\text{-}CPA}_{\mathsf{IBE}}$

procedure Initialize:
$(mpk, msk) \xleftarrow{\$} \mathsf{Pg}$; $c \xleftarrow{\$} \{0,1\}$
Ret mpk

procedure Extract(I):
Ret $\mathsf{Kg}(mpk, msk, I)$

procedure LR(I, M_0, M_1):
Ret $\mathsf{Enc}(mpk, I, M_c)$

procedure Finalize(c'):
Ret $(c' = c)$

Figure 3: The DBDH and IND-CPA games.

if their code differs only in statements that follow the setting of bad to true. (For examples, games G_0, G_1 of Figure 4 are identical-until-bad, as they differ only in the boxed statements.) We let "$\mathrm{G}_i^{\mathcal{A}}$ sets bad" denote the event that game G_i, when executed with adversary $\mathcal{A}$, sets bad to true (and similarly for "$\mathrm{G}_i^{\mathcal{A}}$ doesn't set bad"). It is shown in [4] that if $\mathrm{G}_i, \mathrm{G}_j$ are identical-until-bad and $\mathcal{A}$ is an adversary, then

$$\Pr\left[\,\mathrm{G}_i^{\mathcal{A}} \text{ sets } \mathsf{bad}\,\right] = \Pr\left[\,\mathrm{G}_j^{\mathcal{A}} \text{ sets } \mathsf{bad}\,\right]. \tag{1}$$

The fundamental lemma of game-playing [4] says that if $\mathrm{G}_i, \mathrm{G}_j$ are identical-until-bad then for any y

$$\Pr\left[\,\mathrm{G}_i^{\mathcal{A}} \Rightarrow y\,\right] - \Pr\left[\,\mathrm{G}_j^{\mathcal{A}} \Rightarrow y\,\right] \le \Pr\left[\,\mathrm{G}_i^{\mathcal{A}} \text{ sets } \mathsf{bad}\,\right].$$

This lemma is useful when the probability that bad is set is small, but in our setting this probability will be close to one. We will instead use the following variant:

Lemma 2.1 Let $\mathrm{G}_i, \mathrm{G}_j$ be identical-until-bad games and let $\mathcal{A}$ be an adversary. Then for any y

$$\Pr\left[\,\mathrm{G}_i^{\mathcal{A}} \Rightarrow y \wedge \mathrm{G}_i^{\mathcal{A}} \text{ doesn't set } \mathsf{bad}\,\right] = \Pr\left[\,\mathrm{G}_j^{\mathcal{A}} \Rightarrow y \wedge \mathrm{G}_j^{\mathcal{A}} \text{ doesn't set } \mathsf{bad}\,\right]. \ \square$$

Lemma 2.1 is implicit in the proof of the fundamental lemma of [4], but for completeness we provide a proof in Appendix B. Lemma 2.1 was also used in [3, 33].

DBDH PROBLEM. The Decisional Bilinear Diffie-Hellman (DBDH) assumption (in the asymmetric setting) [6] is captured by the game described in Figure 3. We define the dbdh-advantage of an adversary $\mathcal{A}$ against $\mathsf{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathbf{e}, \psi)$ by

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

IDENTITY-BASED ENCRYPTION. An *identity-based encryption (IBE) scheme* is a tuple of algorithms $\mathsf{IBE} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with associated identity space $\mathsf{IdSp} \subseteq \{0,1\}^*$ and message space MsgSp. The key-issuing center runs the parameter generation algorithm Pg (which takes no input) generates a master public key mpk and a master secret key msk. The former is publicly distributed. The key generation algorithm Kg takes as input mpk, msk, I, where $I \in \mathsf{IdSp}$, and outputs a secret key sk for party I. The encryption algorithm Enc takes inputs mpk, I, M, where $I \in \mathsf{IdSp}$ and $M \in \mathsf{MsgSp}$, and outputs a ciphertext C. The deterministic decryption algorithm Dec takes inputs mpk, sk, I, C and outputs either $\perp$ or a plaintext M. We require the usual consistency, namely that $\mathsf{Dec}(mpk, sk, I, \mathsf{Enc}(mpk, I, M)) = M$ with probability one for all $I \in \mathsf{IdSp}$ and $M \in \mathsf{MsgSp}$, where the probability is over $(mpk, msk) \xleftarrow{\$} \mathsf{Pg}$; $sk \xleftarrow{\$} \mathsf{Kg}(mpk, msk, I)$ and the coins used by Enc. We use the notion of privacy from [8], namely indistinguishability under chosen-plaintext attack

(ind-cpa). The ind-cpa advantage of an adversary $\mathcal{A}$ against an IBE scheme IBE is defined by

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{IBE}}(\mathcal{A}) = 2 \cdot \Pr\left[\,\text{IND-CPA}^{\mathcal{A}}_{\mathsf{IBE}} \Rightarrow \mathsf{true}\,\right] - 1\,, \tag{3}$$

where game IND-CPA is shown in Figure 3. We only allow *legitimate* adversaries, where adversary $\mathcal{A}$ is legitimate if it makes only one query (I^*, M_0, M_1) to **LR**, for some $I^* \in \mathsf{IdSp}$ and $M_0, M_1 \in \mathsf{MsgSp}$ with $|M_0| = |M_1|$, and never queries I^* to **Extract**. Here $|M|$ denotes the length of some canonical string encoding of a message $M \in \mathsf{MsgSp}$. (In the schemes we consider messages are group elements.)

Note that we do allow multiple queries to **Extract** with the same identity, which is important because key generation in the schemes we consider are randomized. When we say that $\mathcal{A}$ makes at most q oracle queries or has running time at most t we mean that this is true regardless of $\mathcal{A}$'s coins and environment, meaning even if its input and the answers to its oracle queries do not come from $\text{IND-CPA}_{\mathsf{IBE}}$.

Waters' IBE scheme. The Boneh-Boyen IBE scheme is described in Appendix F. The Waters' IBE scheme changes the hash function used by BB_1. Let n be a positive integer. Define the hash family H: $\mathbb{G}_1^{n+1} \times \{0,1\}^n \to \mathbb{G}_1$ by $H(\mathbf{u}, I) = \mathbf{u}[0] \prod_{i=1}^{n} \mathbf{u}[i]^{I[i]}$ for any $\mathbf{u} \in \mathbb{G}_1^{n+1}$ and any $I \in \{0,1\}^n$. The Waters IBE scheme $\mathsf{Wa} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ associated to GP and n has associated identity space $\mathsf{IdSp} = \{0,1\}^n$ and message space $\mathsf{MsgSp} = \mathbb{G}_T$, and its first three algorithms are as follows:

procedure Pg	**procedure** $\mathsf{Kg}(mpk, msk, I)$	**procedure** $\mathsf{Enc}(mpk, I, M)$
$A_1 \xleftarrow{\$} \mathbb{G}_1$; $g_2 \xleftarrow{\$} \mathbb{G}_2^*$	$(g_2, A_1, B_2, \mathbf{u}) \leftarrow mpk$	$(g_2, A_1, B_2, \mathbf{u}) \leftarrow mpk$
$b \xleftarrow{\$} \mathbb{Z}_p$; $B_2 \leftarrow g_2^b$; $\mathbf{u} \xleftarrow{\$} \mathbb{G}_1^{n+1}$	$K \leftarrow msk$; $r \xleftarrow{\$} \mathbb{Z}_p$	$s \xleftarrow{\$} \mathbb{Z}_p$
$mpk \leftarrow (g_2, A_1, B_2, \mathbf{u})$	Ret $(K \cdot H(\mathbf{u}, I)^r, g_2^r)$	Ret $(\mathbf{e}(A_1, B_2)^s \cdot M, g_2^s, H(\mathbf{u}, I)^s)$
$msk \leftarrow A_1^b$		
Ret (mpk, msk)		

Above, when we write $(g_2, A_1, B_2, \mathbf{u}) \leftarrow mpk$ we mean mpk is parsed into its constituent parts. We do not specify the decryption algorithm since it is not relevant to IND-CPA security; it can be found in [36].

In [36] the scheme is presented in the symmetric setting where $\mathbb{G}_1 = \mathbb{G}_2$. While this makes notation simpler, we work in the asymmetric setting because it allows pairing parameters for higher security levels [19].

The hash functions used by Boneh-Boyen and Waters' schemes have restricted domain. One can extend to $\mathsf{IdSp} = \{0,1\}^*$ by first hashing an identity with a collision-resistant hash function to derive an n-bit string. (For BB_1 this is then encoded in some canonical fashion to a point in $\mathbb{Z}_p$.)To ensure security from birthday attacks, the output length n of the CR function must have bit-length at least twice that of the desired security parameter.

Waters' result. Waters [36] proves the security of the Wa scheme associated to GP, n under the assumption that the DBDH problem in GP is hard. Specifically, let $\mathcal{A}$ be an ind-cpa adversary against Wa that runs in time at most t, makes at most $q \in [1\,..\,p/4n]$ queries to its **Extract** oracle and has advantage $\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})$. Then [36, Theorem 1] presents a dbdh-adversary $\mathcal{B}_{\mathsf{Wa}}$ such that

$$\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}_{\mathsf{Wa}}) \;\geq\; \frac{\epsilon}{32(n+1)q}\ \text{, and} \tag{4}$$

$$\mathbf{T}(\mathcal{B}_{\mathsf{Wa}}) \;=\; \mathbf{T}(\mathcal{A}) + \mathbf{T}_{\text{sim}}(n, q) + \mathbf{T}_{\text{abort}}(\epsilon, n, q) \tag{5}$$

where

$$\mathbf{T}_{\text{sim}}(n,q) = \mathcal{O}\left(\mathsf{T}_{\psi} + (n+q)\cdot\mathsf{T}_{\exp}(\mathbb{G}_1) + q\cdot\mathsf{T}_{\exp}(\mathbb{G}_2) + qn + \mathsf{T}_{\text{op}}(\mathbb{G}_T)\right) \quad (6)$$

$$\mathbf{T}_{\text{abort}}(\epsilon,n,q) = \mathcal{O}(q^2n^2\epsilon^{-2}\ln(\epsilon^{-1})\ln(qn))\,. \quad (7)$$

An important factor in the "looseness" of the reduction is the $\mathbf{T}_{\text{abort}}(\epsilon,n,q)$ term, which can be very large, making $\mathbf{T}(\mathcal{B}_{\mathsf{Wa}})$ much more than $\mathbf{T}(\mathcal{A})$. This term arises from the "artificial abort" step. (In [36], the $\mathbf{T}_{\text{abort}}(\epsilon,n,q)$ term only has a qn factor in place of the q^2n^2 factor we show. However, the step requires performing q times up to n operations over the integers modulo $4q$ for each of the $\ell = \mathcal{O}(qn\epsilon^{-2}\ln\epsilon^{-1}\ln qn)$ vectors selected, so our term represents the actual cost.)

3 New Proof of Waters' IBE without Artificial Aborts

We start with some high-level discussion regarding Waters' original proof and the reason for the artificial abort. First, one would hope to specify a simulator that, given IND-CPA adversary $\mathcal{A}$ that attacks the IBE scheme using q **Extract** queries and gains advantage ϵ, solves the DBDH problem with advantage not drastically worse than ϵ. But $\mathcal{A}$ can make **Extract** queries that force any conceivable simulator to fail, i.e. have to abort. This means that the advantage against DBDH is conditioned on $\mathcal{A}$ not causing an abort, and so it could be the case that $\mathcal{A}$ achieves ϵ advantage in the normal IND-CPA experiment but almost always causes aborts for the simulator. In this (hypothetical) case, the simulator could not effectively make use of the adversary, and the proof fails.

On the other hand, if one can argue that the lower and upper bounds on the probability of $\mathcal{A}$ causing an abort to occur are close (i.e. the case above does not occur), then the proof would go through. As Waters' points out [36], the natural simulator (that only aborts when absolutely necessarily) fails to provide such a guarantee. To compensate, Waters' introduced "artificial aborts". At the end of a successful simulation for the IBE adversary, the simulator $\mathcal{B}_{\mathsf{Wa}}$ used by Waters' generates $\mathcal{O}(qn\epsilon^{-2}\ln\epsilon^{-1}\ln qn)$ random vectors. These are used to estimate the probability that the **Extract** queries made by $\mathcal{A}$ cause an abort during any given execution of the simulator. The simulator then artificially aborts with some related probability. Intuitively, this forces the probability of aborting to be independent of $\mathcal{A}$'s particular queries. Waters' shows that $\mathcal{B}_{\mathsf{Wa}}$ provides the aforementioned guarantee of close lower and upper bounds and the proof goes through.

The artificial abort step seems strange because $\mathcal{B}_{\mathsf{Wa}}$ is forcing itself to fail even when it appears to have succeeded. The concrete security also suffers because the running time of the simulator goes up by $\mathbf{T}_{\text{abort}}(\epsilon,n,q)$ as shown in (7).

The rest of this section is devoted to proving the next theorem, which establishes the security of the Waters' IBE scheme without relying on an artificial abort step.

Theorem 3.1 Fix pairing parameters $\mathsf{GP} = (\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,p,\mathbf{e},\psi)$ and an integer $n \geq 1$, and let $\mathsf{Wa} = (\mathsf{Pg},\mathsf{Kg},\mathsf{Enc},\mathsf{Dec})$ be the Waters IBE scheme associated to GP and n. Let $\mathcal{A}$ be an ind-cpa adversary against Wa which has advantage $\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A}) > 0$ and makes at most $q \in [1\,..\,p\epsilon/9n]$ queries to its **Extract** oracle. Then there is a dbdh adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) \geq \frac{\epsilon^2}{27qn+3\epsilon}\ , \text{and} \quad (8)$$

$$\mathbf{T}(\mathcal{B}) = \mathbf{T}(\mathcal{A}) + \mathbf{T}_{\text{sim}}(n,q) \quad (9)$$

where $\mathbf{T}_{\text{sim}}(n,q)$ was defined by (6). □

The limitations on q—namely, $1 \le q \le p/4n$ in Waters' result and $1 \le q \le p\epsilon/9n$ in ours—are of little significance since in practice $p \ge 2^{160}$, $\epsilon \ge 2^{-80}$, and $n = 160$. For $q = 0$ there is a separate, tight reduction. The remainder of this section is devoted to the proof of Theorem 3.1.

SOME DEFINITIONS. Let $m = \lceil 9q/\epsilon \rceil$ 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 $I \in \{0,1\}^n$ we let

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

Note that while the computation of G above is over $\mathbb{Z}_p$, that of F is over $\mathbb{Z}$.

ADVERSARY $\mathcal{B}$. Our DBDH adversary $\mathcal{B}$ is depicted in Figure 4, where the *simulation subroutines* KgS and EncS are specified below. There are two main differences between our adversary and that of Waters'. The first is that in our case the parameter m is $\mathcal{O}(q/\epsilon)$ while in Waters' case it is $\mathcal{O}(q)$. The second difference of course is that Waters' adversary $\mathcal{B}_{\mathsf{Wa}}$, unlike ours, includes the artificial abort step. Once $\mathcal{A}$ has terminated, this step selects $l = \mathcal{O}(qn\epsilon^{-2}\ln(\epsilon^{-1})\ln(qn))$ new random vectors $\mathbf{x}_1, \ldots, \mathbf{x}_l$ from X. Letting $I_1, \ldots, I_q$ denote the identities queried by $\mathcal{A}$ to its **Extract** oracle and I_0 the identity queried to the **LR** oracle, it then evaluates $\mathsf{F}(\mathbf{x}_i, I_j)$ for all $1 \le i \le l$ and $0 \le j \le q$, and uses these values to approximate the probability that bad is set. It then aborts with some related probability. Each computation of F takes $\mathcal{O}(n)$ time, and there are q such computations for each of the l samples, accounting for the estimate of (7). In addition there are some minor differences between the adversaries. For example, $\mathbf{x}$ is chosen differently. (In [36] it is taken from $[0\,..\,m-1]^{n+1}$, and an additional value $k \in [0\,..\,n]$, which we do not have, is mixed in.)

We note that our adversary in fact *never* aborts. Sometimes, it is clearly returning incorrect answers (namely $\perp$) to $\mathcal{A}$'s queries. Adversary $\mathcal{A}$ will recognize this, and all bets are off as to what it will do. Nonetheless, $\mathcal{B}$ continues the execution of $\mathcal{A}$. Our analysis will show that $\mathcal{B}$ has the claimed properties regardless.

An analysis of the running time of $\mathcal{B}$, justifying equations (6) and (9), is given in Appendix G.

SIMULATION SUBROUTINES. We define the subroutines that $\mathcal{B}$ utilizes to answer **Extract** and **LR** queries. We say that $(g_1, g_2, A_2, A_1, B_2, B_1, \mathbf{x}, \mathbf{y}, \mathbf{u}, S, W)$ are *simulation parameters* if: $g_2 \in \mathbb{G}_2^*$; $g_1 = \psi(g_2) \in \mathbb{G}_1^*$; $A_2 \in \mathbb{G}_2$; $A_1 = \psi(A_2) \in \mathbb{G}_1$; $B_2 \in \mathbb{G}_2$; $B_1 = \psi(B_2) \in \mathbb{G}_1$; $\mathbf{x} \in X$; $\mathbf{y} \in \mathbb{Z}_p^{n+1}$; $\mathbf{u}[j] = B_1^{\mathbf{x}[j]} g_1^{\mathbf{y}[j]}$ for $j \in [0\,..\,n]$; $S \in \mathbb{G}_2$; and $W \in \mathbb{G}_T$. We define the following procedures:

procedure $\mathsf{KgS}(g_1, g_2, A_2, A_1, B_1, \mathbf{x}, \mathbf{y}, I)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $w \leftarrow \mathsf{F}(\mathbf{x}, I)^{-1} \bmod p$
$L_1 \leftarrow B_1^{\mathsf{F}(\mathbf{x},I)\cdot r} g_1^{\mathsf{G}(\mathbf{y},I)\cdot r} A_1^{-\mathsf{G}(\mathbf{y},I)w}$
$L_2 \leftarrow g_2^r A_2^{-w}$
Ret (L_1, L_2)

procedure $\mathsf{EncS}(S, W, M, \mathbf{y}, I)$
$C_1 \leftarrow W \cdot M$
$C_2 \leftarrow S$; $C_3 \leftarrow \psi(S)^{\mathsf{G}(\mathbf{y},I)}$
Ret (C_1, C_2, C_3)

Note that if $\mathsf{F}(\mathbf{x}, I) \neq 0$ then $\mathsf{F}(\mathbf{x}, I) \not\equiv 0 \pmod p$ so the quantity w computed by KgS is well-defined whenever $\mathsf{F}(\mathbf{x}, I) \neq 0$. This is because the absolute value of $\mathsf{F}(\mathbf{x}, I)$ is at most

$$n(m-1) = n\left(\left\lceil \frac{9q}{\epsilon} \right\rceil - 1\right) < \frac{9nq}{\epsilon} \le p\,, \tag{11}$$

the last because of the restriction on q in the theorem statement. The next lemma captures two facts about the simulation subroutines, which we will use in our analysis.

Lemma 3.2 Let $(g_1, g_2, A_2, A_1, B_2, B_1, \mathbf{x}, \mathbf{y}, \mathbf{u}, S, W)$ be simulation parameters. Let $I \in \{0,1\}^n$. Let $mpk = (g_2, A_1, B_2, \mathbf{u})$. Let b be the discrete log of B_1 to base g_1 and let $msk = A_1^b$. Let s be

Adversary $\mathcal{B}(g_1, g_2, A_2, B_2, S, W)$:
$c \xleftarrow{\$} \{0,1\}$; $A_1 \leftarrow \psi(A_2)$; $B_1 \leftarrow \psi(B_2)$
For $j = 0, \ldots, n$ do
  $\mathbf{y}[j] \xleftarrow{\$} \mathbb{Z}_p$
  If $j = 0$ then $\mathbf{x}[j] \xleftarrow{\$} [-n(m-1) \,..\, 0]$
  Else $\mathbf{x}[j] \xleftarrow{\$} [0 \,..\, m-1]$
  $\mathbf{u}[j] \leftarrow B_1^{\mathbf{x}[j]} g_1^{\mathbf{y}[j]}$
$mpk \leftarrow (g_2, A_1, B_2, \mathbf{u})$
Run $\mathcal{A}(mpk)$, answering queries by

query Extract(I):
$sk(I) \leftarrow \perp$
If $\mathsf{F}(\mathbf{x}, I) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else $sk(I) \xleftarrow{\$} \mathsf{KgS}(g_1, g_2, A_2, A_1, B_1, \mathbf{x}, \mathbf{y}, I)$
Ret $sk(I)$

query LR(I, M_0, M_1):
$C \leftarrow \perp$
If $\mathsf{F}(\mathbf{x}, I) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else $C \leftarrow \mathsf{EncS}(S, W, M_c, \mathbf{y}, I)$
Ret C

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

procedure Initialize: Games G_0, G_1, G_2
000 $g_2 \xleftarrow{\$} \mathbb{G}_2^*$; $g_1 \leftarrow \psi(g_2)$; $a, b, s \xleftarrow{\$} \mathbb{Z}_p$
001 $A_2 \leftarrow g_2^a$; $B_2 \leftarrow g_2^b$; $S \leftarrow g_2^s$; $c, d \xleftarrow{\$} \{0,1\}$
002 $A_1 \leftarrow \psi(A_2)$; $B_1 \leftarrow \psi(B_2)$; $K \leftarrow A_1^b$
003 For $j = 0, \ldots, n$ do
004 $\mathbf{y}[j] \xleftarrow{\$} \mathbb{Z}_p$
005 If $j = 0$ then $\mathbf{x}[j] \xleftarrow{\$} [-n(m-1) \,..\, 0]$
006 Else $\mathbf{x}[j] \xleftarrow{\$} [0 \,..\, m-1]$
007 $\mathbf{u}[j] \leftarrow B_1^{\mathbf{x}[j]} g_1^{\mathbf{y}[j]}$
008 $mpk \leftarrow (g_2, A_1, B_2, \mathbf{u})$
009 If $d = 1$ then $W \leftarrow \mathbf{e}(A_1, B_2)^s$
010 Else $W \xleftarrow{\$} \mathbb{G}_T$
011 Ret mpk

procedure Extract(I): Games G_0, $\boxed{G_1}$
020 $sk(I) \leftarrow \perp$
021 If $\mathsf{F}(\mathbf{x}, I) = 0$ then
022 $\mathsf{bad} \leftarrow \mathsf{true}$
023 $\boxed{r \xleftarrow{\$} \mathbb{Z}_p \,;\, sk(I) \leftarrow (K \cdot H(\mathbf{u}, I)^r, g_2^r)}$
024 Else $sk(I) \xleftarrow{\$} \mathsf{KgS}(g_1, g_2, A_2, A_1, B_1, \mathbf{x}, \mathbf{y}, I)$
025 Ret $sk(I)$

procedure LR(I, M_0, M_1): Games G_0, $\boxed{G_1}$
030 $C \leftarrow \perp$
031 If $\mathsf{F}(\mathbf{x}, I) \neq 0$ then
032 $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{C \leftarrow (W \cdot M_c, S, H(\mathbf{u}, I)^s)}$
033 Else $C \leftarrow \mathsf{EncS}(S, W, M_c, \mathbf{y}, I)$
034 Ret C

procedure Finalize(c'): Games G_0, $\boxed{G_1}$
040 $c'' \leftarrow c'$
041 If $\mathsf{bad} = \mathsf{true}$ then $c'' \xleftarrow{\$} \{0,1\}$; $\boxed{c'' \leftarrow c'}$
042 If $c = c''$ then Ret 1 else Ret 0

Figure 4: Adversary $\mathcal{B}$ and the start of the game sequence. Game G_1 includes the boxed statements in procedures **Extract**, **LR**, and **Finalize** while G_0 does not.

the discrete log of S to base g_2. Then if $\mathsf{F}(\mathbf{x}, I) \neq 0$ the outputs of $\mathsf{KgS}(g_1, g_2, A_2, A_1, B_1, \mathbf{x}, \mathbf{y}, I)$ and $\mathsf{Kg}(mpk, msk, I)$ are identically distributed. Also if $\mathsf{F}(\mathbf{x}, I) = 0$ then for any $M \in \mathsf{MsgSp}$, the output of $\mathsf{EncS}(S, W, M, \mathbf{y}, I)$ is $(W \cdot M, S, H(\mathbf{u}, I)^s)$. $\square$

The proof of Lemma 3.2, which follows arguments given in [36], is given in Appendix C.

Overview. Consider executing $\mathcal{B}$ in game $\mathrm{DBDH}_{\mathsf{GP}}$. If $d = 1$, then Lemma 3.2 implies that adversary $\mathcal{B}$ correctly answers oracle queries as long as it does not set bad. On the other hand if $d = 0$ then $\mathcal{B}$'s output is a random bit. Attempting to conclude by showing that bad is seldom set fails, however, because in fact it will be set with probability close to 1. Alternatively, if one

procedure Extract(I): Game G_2

220 If $\mathsf{F}(\mathbf{x}, I) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
221 $r \xleftarrow{\$} \mathbb{Z}_p$; Ret $sk(I) \leftarrow (K \cdot H(\mathbf{u}, I)^r, g_2^r)$

procedure LR(I, M_0, M_1): Game G_2

230 If $\mathsf{F}(\mathbf{x}, I) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
231 Ret $C \leftarrow (W \cdot M_c, S, H(\mathbf{u}, I)^s)$

procedure Finalize(c'): Game G_2

240 If $c = c'$ then Ret 1 else Ret 0

procedure Initialize: Game G_3

300 $A_1 \xleftarrow{\$} \mathbb{G}_1$; $g_2 \xleftarrow{\$} \mathbb{G}_2^*$; $b, s \xleftarrow{\$} \mathbb{Z}_p$; $cnt \leftarrow 0$
301 $B_2 \leftarrow g_2^b$; $S \leftarrow g_2^s$; $c, d \xleftarrow{\$} \{0,1\}$; $K \leftarrow A_1^b$
302 For $j = 0, \ldots, n$ do
303 $\quad \mathbf{z}[j] \xleftarrow{\$} \mathbb{Z}_p$; $\mathbf{u}[j] \leftarrow g_1^{\mathbf{z}[j]}$
304 $\quad$ If $j = 0$ then $\mathbf{x}[j] \xleftarrow{\$} [-n(m-1) \,..\, 0]$
305 $\quad$ Else $\mathbf{x}[j] \xleftarrow{\$} [0 \,..\, m-1]$
306 $\quad \mathbf{y}[j] \leftarrow \mathbf{z}[j] - b \cdot \mathbf{x}[j] \bmod p$
307 $mpk \leftarrow (g_2, A_1, B_2, \mathbf{u})$
308 If $d = 1$ then $W \leftarrow \mathbf{e}(A_1, B_2)^s$
309 Else $W \xleftarrow{\$} \mathbb{G}_T$
310 Ret mpk

procedure Finalize(c'): Game G_3

340 For $j = 1, \ldots, cnt$ do
341 $\quad$ If $\mathsf{F}(\mathbf{x}, I_j) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
342 If $\mathsf{F}(\mathbf{x}, I_0) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
343 If $c = c'$ then Ret 1 else Ret 0

procedure Initialize: Game G_4

400 $A_1 \xleftarrow{\$} \mathbb{G}_1$; $g_2 \xleftarrow{\$} \mathbb{G}_2^*$; $b, s \xleftarrow{\$} \mathbb{Z}_p$; $i \leftarrow 0$
401 $B_2 \leftarrow g_2^b$; $S \leftarrow g_2^s$; $c, d \xleftarrow{\$} \{0,1\}$; $K \leftarrow A_1^b$
402 For $j = 0, \ldots, n$ do
403 $\quad \mathbf{z}[j] \xleftarrow{\$} \mathbb{Z}_p$; $\mathbf{u}[j] \leftarrow g^{\mathbf{z}[j]}$
404 $mpk \leftarrow (g, A_1, B_2, \mathbf{u})$
405 If $d = 1$ then $W \leftarrow \mathbf{e}(A_1, B_2)^s$
406 Else $W \xleftarrow{\$} \mathbb{G}_T$
407 Ret mpk

procedure Extract(I): Games G_3, G_4

320 $cnt \leftarrow cnt + 1$; $I_{cnt} \leftarrow I$
321 $r \xleftarrow{\$} \mathbb{Z}_p$; Ret $sk(I) \leftarrow (K \cdot H(\mathbf{u}, I)^r, g_2^r)$

procedure LR(I, M_0, M_1): Games G_3, G_4

330 $I_0 \leftarrow I$
331 Ret $C \leftarrow (W \cdot M_c, S, H(\mathbf{u}, I)^s)$

procedure Finalize(c'): Game G_4

440 For $j = 0, \ldots, n$ do
441 $\quad$ If $j = 0$ then $\mathbf{x}[j] \xleftarrow{\$} [-n(m-1) \,..\, 0]$
442 $\quad$ Else $\mathbf{x}[j] \xleftarrow{\$} [0 \,..\, m-1]$
443 For $j = 1, \ldots, cnt$ do
444 $\quad$ If $\mathsf{F}(\mathbf{x}, I_j) = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
445 If $\mathsf{F}(\mathbf{x}, I_0) \neq 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$
446 If $c = c'$ then Ret 1 else Ret 0

Figure 5: Continuation of the game sequence. Games G_2 and G_1 (see Figure 4) have identical **Initialize** procedures. Games G_3 and G_4 have identical **Extract** and **LR** procedures.

could show that the setting of bad is independent of the correctness of $\mathcal{B}$'s output, then one could conclude by multiplying the probabilities of these events. The difficulty in the proof is that this independence does not hold. Waters' artificial abort step is one way to compensate. However, we have dropped this (expensive) step and propose nonetheless to push an argument through. We will first use the game sequence G_0–G_4 to arrive at a game where the choice of $\mathbf{x}$ is independent of the game output. The subtle point is that this *still* does not provide independence between setting bad and the game output, because the identities chosen by $\mathcal{A}$ for its oracle queries affect both events. The first step in addressing this is a conditioning argument based on Lemma 3.4 which allows us to express a lower bound on the advantage of $\mathcal{B}$ in terms of probabilities $\gamma(\mathbf{I})$ associated to different queried identities. The crucial insight is that Lemma 3.5 gives upper and lower bounds on these probabilities that are very close, specifically within a factor of $1 - \epsilon$ of each other, due to our choice of $m = \mathcal{O}(q/\epsilon)$ rather than merely the $m = \mathcal{O}(q)$ of [36]. Using this allows us to conclude easily.

The game playing sequence. Assume without loss of generality that $\mathcal{A}$ always makes exactly q queries to its **Extract** oracle rather than at most q. The proof starts using a sequence of games

$G_0 - G_4$ to move from $\mathcal{B}$ running in the DBDH experiment to a game G_4 (shown in Figure 5) that is essentially the IND-CPA experiment, though with some additional bookkeeping. This transition is critical since it moves to a setting where the choice of $\mathbf{x}$ is clearly independent of $\mathcal{A}$'s choices. We capture this game playing sequence via the following lemma. Let GD_4 denote the event that $G_4^{\mathcal{A}}$ does not set bad.

Lemma 3.3 $\mathbf{Adv}^{\mathrm{dbdh}}_{\mathsf{GP}}(\mathcal{B}) = 2 \cdot \Pr\left[\, G_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4 \,\right] - \Pr\left[\, \mathsf{GD}_4 \,\right]$ $\square$

Proof: Figures 4 and 5 compactly describe four games showing next to each procedure the games in which it appears. For example, the **Initialize** procedure of Figure 4 is common to games G_0, G_1, and G_2. Let BD_i denote the event that $G_i^{\mathcal{A}}$ sets bad and GD_i the event that $G_i^{\mathcal{A}}$ does not set bad, for $0 \le i \le 4$. The **Initialize** procedure of game G_0 includes the code of the **Initialize** procedure of game $\mathrm{DBDH}_{\mathsf{GP}}$. Game G_0 and adversary $\mathcal{B}$ answer $\mathcal{A}$'s oracle queries in the same way (remember that G_0 omits the boxed statements). The output of G_0 (as computed by its **Finalize** procedure) is the same as $\mathcal{B}$'s. (The code is written differently but is equivalent.) The bit d in G_0 represents the challenge bit of the same name in game $\mathrm{DBDH}_{\mathsf{GP}}$. Thus we have,

$$\begin{aligned}\Pr\left[\, \mathrm{DBDH}^{\mathcal{B}}_{\mathsf{GP}} \Rightarrow \mathsf{true} \,\right] &= \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow d \,\right] \\ &= \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow d \mid \mathsf{BD}_0 \,\right] \Pr\left[\, \mathsf{BD}_0 \,\right] + \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_0 \,\right] .\end{aligned}$$

If $G_0^{\mathcal{A}}$ sets bad then line 041 ensures that c'' is random. Line 042 then ensures that the output of G_0 is random as well and thus equals d with probability $1/2$. This means that the conditional probability above is $1/2$ and hence we have

$$\Pr\left[\, \mathrm{DBDH}^{\mathcal{B}}_{\mathsf{GP}} \Rightarrow \mathsf{true} \,\right] = \frac{1}{2} \cdot \Pr\left[\, \mathsf{BD}_0 \,\right] + \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_0 \,\right] . \tag{12}$$

The boxed statements of lines 023 and 032 of G_1 are "corrections" which ensure that no oracle query ever returns $\perp$, even in the case that a subroutine would fail, meaning when bad is set. (Note this code uses the secret key K and the discrete log s of S, neither of which were given to $\mathcal{B}$ or used by G_0 to respond to oracle queries.) But G_0,G_1 are identical-until-bad, so by (1) and Lemma 2.1 we have

$$\Pr\left[\, \mathsf{BD}_0 \,\right] = \Pr\left[\, \mathsf{BD}_1 \,\right] \quad \text{and} \quad \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_0 \,\right] = \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_1 \,\right] . \tag{13}$$

Let us now consider how G_1 relates to G_2. Lemma 3.2 tells us that KgS at line 024 and the code of line 023 compute the same value $sk(I)$. Lemma 3.2 also tells us that EncS at line 033 and the code of line 032 compute the same tuple C. (Recall $\mathcal{A}$ is allowed only one **LR** query.) Thus, G_1 always answers oracle queries in the same manner, meaning one can equivalently define its **Extract** and **LR** procedures as in game G_2. Since the boxed code of line 041 is included in G_1, we always have $c'' = c'$ and hence the output of the game is simply 1 if $c = c'$ and 0 otherwise. This is how G_2 computes its output at 240. It follows that

$$\Pr\left[\, \mathsf{BD}_1 \,\right] = \Pr\left[\, \mathsf{BD}_2 \,\right] \quad \text{and} \quad \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_1 \,\right] = \Pr\left[\, G_2^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_2 \,\right] . \tag{14}$$

The **Extract** and **LR** procedures of G_2 may set bad but do not use it. Throwing in the bookkeeping of lines 300, 320, and 330, we can thus move the code setting bad into lines 340–342 in G_3. We then observe that lines 004–007 of G_2 can equivalently be written as lines 303–306 of G_3. Finally, game G_2 no longer uses g_1, A_2, and B_1. Thus we re-write lines 000–002 of G_2 as 300–301 of G_3. The distribution of A_1 is equivalent in both games. At this point we have

$$\Pr\left[\, \mathsf{BD}_2 \,\right] = \Pr\left[\, \mathsf{BD}_3 \,\right] \quad \text{and} \quad \Pr\left[\, G_2^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_2 \,\right] = \Pr\left[\, G_3^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_3 \,\right] . \tag{15}$$

But now $\mathbf{x}$ is not used in G_3 until the **Finalize** procedure and $\mathbf{y}$ is not used at all. Game G_4 delays

picking $\mathbf{x}$ until **Finalize** and drops $\mathbf{y}$, but otherwise its code matches that of G_3. Thus we have

$$\Pr[\mathsf{BD}_3] = \Pr[\mathsf{BD}_4] \quad \text{and} \quad \Pr\left[G_3^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_3\right] = \Pr\left[G_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4\right]. \tag{16}$$

Using equations (2), (12), (13), (14), (15), and (16) we have

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{dbdh}}_{\mathsf{GP}}(\mathcal{B}) &= 2 \cdot \Pr\left[\mathrm{DBDH}^{\mathcal{B}}_{\mathsf{GP}} \Rightarrow \mathsf{true}\right] - 1 \\
&= \Pr[\mathsf{BD}_4] + 2 \cdot \Pr\left[G_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4\right] - 1 \\
&= 2 \cdot \Pr\left[G_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4\right] - \Pr[\mathsf{GD}_4]. \qquad (17)
\end{aligned}$$

This completes the proof of Lemma 3.3. ▌

We now reach a subtle point. Consider the following argument: "The event GD_4 depends only on $\mathbf{x}$, which is chosen at lines 440–442 *after* the adversary and game outputs are determined. So GD_4 is independent of the event that $G_4^{\mathcal{A}} \Rightarrow d$." If we buy this, the probability of the conjunct in Lemma 3.3 becomes the product of the probability of the constituent events, and it is quite easy to conclude. However, the problem is that the argument in quotes above is wrong. The reason is that GD_4 also depends on $I_0, \ldots, I_q$ and these are adversary queries whose values are not independent of the game output. Waters' compensates for this via the artificial abort step, but we do not have this step in $\mathcal{B}$ and propose to complete the analysis anyway.

Conditional independence lemma. Let

$$\mathsf{ID} = \{(I_0, \ldots, I_q) \in (\{0,1\}^n)^{q+1} \ : \ \forall i \in [1\,..\,q]\ (I_0 \neq I_i)\}.$$

For $(I_0, \ldots, I_q) \in \mathsf{ID}$ let

$$\gamma(I_0, \ldots, I_q) = \Pr[\mathsf{F}(\mathbf{x}, I_0) = 0 \wedge \mathsf{F}(\mathbf{x}, I_1) \neq 0 \wedge \cdots \wedge \mathsf{F}(\mathbf{x}, I_q) \neq 0]$$

where the probability is taken over $\mathbf{x} \xleftarrow{\$} X$. This is the probability of GD_4 under a particular sequence of queried identities $I_0, \ldots, I_q$. (We stress that here we first fix $I_0, \ldots, I_q$ and then choose $\mathbf{x}$ at random.) If $\gamma(I_0, \ldots, I_q)$ were the same for all $(I_0, \ldots, I_q) \in \mathsf{ID}$ then the problem discussed above would be resolved. The difficulty is that $\gamma(I_0, \ldots, I_q)$ varies with $I_0, \ldots, I_q$. Our next lemma is the main tool to resolve the independence problem. Roughly it says that if we consider the conditional space obtained by conditioning on a particular sequence $I_0, \ldots, I_q$ of queried identities, then independence does hold. To formalize this, let $\mathsf{Q}(\mathbf{I})$ be the event that the execution of G_4 with $\mathcal{A}$ results in the identities $I_0, \ldots, I_q$ being queried by $\mathcal{A}$, where $\mathbf{I} = (I_0, \ldots, I_q)$. Then:

Lemma 3.4 For any $\mathbf{I} \in \mathsf{ID}$,

$$\Pr\left[G_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})\right] = \gamma(\mathbf{I}) \cdot \Pr\left[G_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{Q}(\mathbf{I})\right] \tag{18}$$

$$\Pr[\mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})] = \gamma(\mathbf{I}) \cdot \Pr[\mathsf{Q}(\mathbf{I})] \ \square \tag{19}$$

Proof: The set of coin tosses underlying the execution of G_4 with $\mathcal{A}$ can be viewed as a cross product $\Omega = \Omega' \times X$, meaning each member ω of Ω is a pair $\omega = (\omega', \mathbf{x})$ where $\mathbf{x}$ is the choice made at lines 440–442 and ω' is all the rest of the game and adversary coins. For any $\mathbf{I} \in \mathsf{ID}$ let $\Omega'(\mathbf{I})$ be the set of all $\omega \in \Omega'$ such that the execution with ω produces $\mathbf{I}$ as the sequence of queried identities. (Which $\mathbf{I}$ is produced depends only on ω' since $\mathbf{x}$ is chosen after $\mathcal{A}$ has terminated.) Let Ω'_{out} be the set of all $\omega \in \Omega'$ on which the execution outputs d. (Again, this is determined only by ω' and not $\mathbf{x}$.) Let $X_{\mathrm{gd}}(\mathbf{I})$ be the set of all $\mathbf{x} \in X$ such that

$$\mathsf{F}(\mathbf{x}, I_0) = 0 \wedge \mathsf{F}(\mathbf{x}, I_i) \neq 0 \wedge \cdots \wedge \mathsf{F}(\mathbf{x}, I_q) \neq 0,$$

where $\mathbf{I} = (I_0, \ldots, I_q)$. Now observe that the set of coins leading to $\mathrm{G}_4^{\mathcal{A}} \Rightarrow d$ is $\Omega'_{\mathrm{out}} \times X$ and the set of coins leading to $\mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})$ is $\Omega'(\mathbf{I}) \times X_{\mathrm{gd}}(\mathbf{I})$. So

$$\begin{aligned}
\Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})\,\right] &= \frac{|(\Omega'_{\mathrm{out}} \times X) \cap (\Omega'(\mathbf{I}) \times X_{\mathrm{gd}}(\mathbf{I}))|}{|\Omega' \times X|} \\
&= \frac{|(\Omega'_{\mathrm{out}} \cap \Omega'(\mathbf{I})) \times X_{\mathrm{gd}}(\mathbf{I})|}{|\Omega' \times X|} = \frac{|\Omega'_{\mathrm{out}} \cap \Omega'(\mathbf{I})| \cdot |X_{\mathrm{gd}}(\mathbf{I})|}{|\Omega'| \cdot |X|} \\
&= \frac{|\Omega'_{\mathrm{out}} \cap \Omega'(\mathbf{I})| \cdot |X|}{|\Omega'| \cdot |X|} \cdot \frac{|X_{\mathrm{gd}}(\mathbf{I})|}{|X|} = \frac{|(\Omega'_{\mathrm{out}} \cap \Omega'(\mathbf{I})) \times X|}{|\Omega' \times X|} \cdot \frac{|X_{\mathrm{gd}}(\mathbf{I})|}{|X|}\,.
\end{aligned}$$

But the first term above is $\Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{Q}(\mathbf{I})\,\right]$ while the second is $\gamma(\mathbf{I})$, establishing (18). For (19) we similarly have

$$\begin{aligned}
\Pr\left[\,\mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})\,\right] &= \frac{|\Omega'(\mathbf{I}) \times X_{\mathrm{gd}}(\mathbf{I})|}{|\Omega' \times X|} = \frac{|\Omega'(\mathbf{I})|}{|\Omega'|} \cdot \frac{|X_{\mathrm{gd}}(\mathbf{I})|}{|X|} \\
&= \frac{|\Omega'(\mathbf{I})| \cdot |X|}{|\Omega'| \cdot |X|} \cdot \frac{|X_{\mathrm{gd}}(\mathbf{I})|}{|X|} = \frac{|\Omega'(\mathbf{I}) \times X|}{|\Omega' \times X|} \cdot \frac{|X_{\mathrm{gd}}(\mathbf{I})|}{|X|}\,.
\end{aligned}$$

But the final terms above are $\Pr\left[\,\mathsf{Q}(\mathbf{I})\,\right]$ and $\gamma(\mathbf{I})$, respectively, establishing (19). ▮

Analysis continued. Let $\gamma_{\min}$ be the smallest value of $\gamma(I_0, \ldots, I_q)$ taken over all $(I_0, \ldots, I_q) \in \mathsf{ID}$. Let $\gamma_{\max}$ be the largest value of $\gamma(I_0, \ldots, I_q)$ taken over all $(I_0, \ldots, I_q) \in \mathsf{ID}$. Using Lemma 3.3 we have that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{dbdh}}_{\mathsf{GP}}(\mathcal{B}) &= 2 \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4\,\right] - \Pr\left[\,\mathsf{GD}_4\,\right] \\
&= \sum_{\mathbf{I} \in \mathsf{ID}} 2 \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})\,\right] - \sum_{\mathbf{I} \in \mathsf{ID}} \Pr\left[\,\mathsf{GD}_4 \wedge \mathsf{Q}(\mathbf{I})\,\right]
\end{aligned}$$

and applying Lemma 3.4:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{dbdh}}_{\mathsf{GP}}(\mathcal{B}) &= \sum_{\mathbf{I} \in \mathsf{ID}} 2\gamma(\mathbf{I}) \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{Q}(\mathbf{I})\,\right] - \sum_{\mathbf{I} \in \mathsf{ID}} \gamma(\mathbf{I}) \cdot \Pr\left[\,\mathsf{Q}(\mathbf{I})\,\right] \\
&\geq \gamma_{\min} \underbrace{\sum_{\mathbf{I} \in \mathsf{ID}} 2 \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d \wedge \mathsf{Q}(\mathbf{I})\,\right]}_{= 2 \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d\,\right]} - \gamma_{\max} \underbrace{\sum_{\mathbf{I} \in \mathsf{ID}} \Pr\left[\,\mathsf{Q}(\mathbf{I})\,\right]}_{= 1} \\
&\geq 2\gamma_{\min} \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d\,\right] - \gamma_{\max}\,. \qquad (20)
\end{aligned}$$

Now

$$\begin{aligned}
\Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow d\,\right] &= \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1 \mid d = 1\,\right] \Pr\left[\,d = 1\,\right] + \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow 0 \mid d = 0\,\right] \Pr\left[\,d = 0\,\right] \\
&= \frac{1}{2} \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1 \mid d = 1\,\right] + \frac{1}{2} \cdot \Pr\left[\,\mathrm{G}_4^{\mathcal{A}} \Rightarrow 0 \mid d = 0\,\right] \\
&= \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})\right) + \frac{1}{2} \cdot \frac{1}{2} \qquad (21) \\
&= \frac{1}{4} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A}) + \frac{1}{2} \qquad (22)
\end{aligned}$$

where we justify (21) as follows. In the case that $d = 0$, the value W is uniformly distributed over $\mathbb{G}_T$ and hence line 331 gives $\mathcal{A}$ no information about the bit c. So the probability that $c = c'$ at line 446 is 1/2. On the other hand if $d = 1$ then G_4 implements the IND-CPA$_{\mathsf{Wa}}$ game, so $2 \cdot \Pr\left[\, \mathrm{G}_4^{\mathcal{A}} \Rightarrow 1 \mid d = 1 \,\right] - 1 = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})$ by (3). We substitute (22) into (20) and get

$$\begin{aligned} \mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) &\geq 2\gamma_{\min}\left(\frac{1}{4}\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A}) + \frac{1}{2}\right) - \gamma_{\max} \\ &= \frac{\gamma_{\min}}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A}) + (\gamma_{\min} - \gamma_{\max})\,. \end{aligned} \tag{23}$$

To finish the proof, we use the following:

Lemma 3.5 $\dfrac{1}{n(m-1)+1}\left(1 - \dfrac{q}{m}\right) \leq \gamma_{\min} \leq \gamma_{\max} \leq \dfrac{1}{n(m-1)+1}$ $\square$

The proof of Lemma 3.5, based on ideas in [36], is given in Appendix D. Let $\alpha = 1/(n(m-1)+1)$. Recall that $m = \lceil 9q/\epsilon \rceil \geq 9q/\epsilon$ where $\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})$. Then, applying Lemma 3.5 to (23) we get

$$\begin{aligned} \mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) &\geq \frac{\alpha}{2}\left(1 - \frac{q}{m}\right)\epsilon + \alpha\left(1 - \frac{q}{m}\right) - \alpha \;=\; \alpha\left[\frac{1}{2}\left(1 - \frac{q}{m}\right)\epsilon - \frac{q}{m}\right] \\ &\geq \alpha\left[\frac{1}{2}\left(1 - \frac{q\epsilon}{9q}\right)\epsilon - \frac{q\epsilon}{9q}\right] \;=\; \frac{\alpha\epsilon}{18}(7 - \epsilon) \\ &\geq \frac{\alpha\epsilon}{3}\,. \end{aligned} \tag{24}$$

Inequality (24) is justified by the fact that $\epsilon \leq 1$. Using the fact that $m = \lceil 9q/\epsilon \rceil \leq 9q/\epsilon + 1$ and substituting in for α, we complete the derivation of our lower bound for $\mathcal{B}$:

$$\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}) \geq \frac{\epsilon}{3} \cdot \frac{1}{n(m-1)+1} \geq \frac{\epsilon}{3} \cdot \frac{1}{n(9q/\epsilon)+1} = \frac{\epsilon^2}{27qn + 3\epsilon}\,.$$

4 Measuring Concrete Security

WORK FACTORS. For any adversary $\mathcal{A}$ running in time $\mathbf{T}(\mathcal{A})$ and gaining advantage ϵ we define the *work factor* of $\mathcal{A}$ to be $\mathbf{WF}(\mathcal{A}) = \mathbf{T}(\mathcal{A})/\epsilon$. The ratio of $\mathcal{A}$'s running time to its advantage provides a measure of the efficiency of the adversary. Generally speaking, to resist an adversary with work factor $\mathbf{WF}(\mathcal{A})$, a scheme should have its security parameter (bits of security) be $\kappa \geq \log \mathbf{WF}(\mathcal{A})$. Note that for a particular value of ϵ, this means a run time of $\mathbf{T}(\mathcal{A}) = \epsilon 2^{\kappa}$. Work factors were previously used in [20].

We use work factors to help tame the complexity of comparing reductions. Here we consider Waters' and our new reduction for Wa; a treatment of BB_1 is given in Appendix F. Consider an ind-cpa adversary $\mathcal{A}$ against Wa that makes q **Extract** queries, runs in time $\mathbf{T}(\mathcal{A})$, and has advantage $\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Wa}}(\mathcal{A})$. We can relate the work factor of $\mathcal{A}$ to the work factors of Waters' adversary $\mathcal{B}_{\mathsf{Wa}}$ and our new reduction's adversary $\mathcal{B}$. Namely, using Equations (4) and (5), we have that

$$\begin{aligned} \mathbf{WF}(\mathcal{B}_{\mathsf{Wa}}) &= \frac{\mathbf{T}(\mathcal{A}) + \mathbf{T}_{\text{sim}}(n,q) + \mathbf{T}_{\text{abort}}(\epsilon,n,q)}{\epsilon/(32(n+1)q)} \\ &= 32q(n+1)\left(\mathbf{WF}(\mathcal{A}) + \mathbf{T}_{\text{sim}}(n,q)\cdot\epsilon^{-1} + \mathbf{T}_{\text{abort}}(\epsilon,n,q)\cdot\epsilon^{-1}\right) \end{aligned} \tag{25}$$

				Representation Size (in bits)		
s	$\log p$	ρ	k	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$
80	160	1	10	160	1600	1600
112	224	1	10	224	2240	2240
128	256	1	12	256	3072	3072
192	384	2	10	768	7680	7680
256	512	2	15	1024	15360	15360

Figure 6: Group sizes for some pairing instantiations. Each value of s represents a common security level for pairings. Here p is the common order of the groups, i.e. $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$. The value ρ determines the size of the representation of an element in $\mathbb{G}_1$, where the size (in bits) is $\rho \cdot \log p$. Finally, k is the embedding degree and determines the size of the representation of $\mathbb{G}_2$ and $\mathbb{G}_T$, where the size (in bits) is $\rho \log p^k$.

and using Equations(8) and (9) we have that

$$\mathbf{WF}(\mathcal{B}) \;=\; \frac{\mathbf{T}(\mathcal{A}) + \mathbf{T}_{\text{sim}}(n,q)}{\epsilon^2/(27nq+3\epsilon)} = \frac{(27nq+3\epsilon)}{\epsilon} \cdot \left(\mathbf{WF}(\mathcal{A}) + \mathbf{T}_{\text{sim}}(n,q) \cdot \epsilon^{-1}\right) . \tag{26}$$

A security proof is only meaningful when the adversary it constructs has work factor less than that of the best known attack. Otherwise, the proof does not guarantee that finding a successful attacker against the scheme contradicts our confidence in the hardness of the underlying problem. Pollard's rho algorithm for finding discrete logs in $\mathbb{G}_1$ is the best known attack against DBDH. (Our instantiations will ensure that finding discrete logs in $\mathbb{G}_T$ is no easier than in $\mathbb{G}_1$.) The work factor of Pollard's algorithm ends up being

$$\mathbf{WF}(\mathcal{P}) = \frac{\mathbf{T}(\mathcal{P})}{\epsilon_p} = \frac{(0.88\sqrt{p} \cdot \mathsf{T}_{\text{op}}(\mathbb{G}_1))^2}{\mathbf{T}(\mathcal{P})} \geq 0.88\sqrt{p} \cdot \mathsf{T}_{\text{op}}(\mathbb{G}_1) .$$

where the last inequality is because $\mathcal{P}$ achieves no more advantage after running for time $0.88\sqrt{p} \cdot \mathsf{T}_{\text{op}}(\mathbb{G}_1)$. We therefore compare the efficiencies of the two reductions based on their ability to satisfy

$$\mathbf{WF}(\mathcal{B}_{\mathsf{Wa}}) \leq \mathbf{WF}(\mathcal{P}) \quad \text{or} \quad \mathbf{WF}(\mathcal{B}) \leq \mathbf{WF}(\mathcal{P})$$

for various values of $\mathbf{WF}(\mathcal{A})$, ϵ, q, and concrete pairing parameters.

Pairing instantiations. In Appendix E we discuss the (complex) topic of instantiating pairings for various security levels. Since not all security levels have efficient instantiations [19, 18], we limit our attention to several *pairing setups* that correspond to the security levels indicated (for private and public-key cryptography) by the NIST key schedule [31]. These setups and their corresponding instantiation sizes are listed in Figure 6.

Comparing the reductions. We are now ready to perform the comparison. Let $\kappa = \log \mathbf{WF}(\mathcal{A})$. The tables in Figure 7 show the pairing setup s_W needed to ensure $\mathbf{WF}(\mathcal{B}_{\mathsf{Wa}}) \leq \mathbf{WF}(\mathcal{P})$ is satisfied and the pairing setup s_N needed to ensure $\mathbf{WF}(\mathcal{B}) \leq \mathbf{WF}(\mathcal{P})$ is satisfied, for the indicated values of κ, ϵ, and q. Shown also is the pairing setup required for BB_1. Note that while one can evaluate the equations everywhere, some combinations of κ, ϵ, and q don't actually make sense, for example $-\epsilon + q \geq \kappa$ implies that the adversary spends all its time making oracle queries. We have marked such combinations with a *.

Waters' reduction provides better or equal efficiency for some choices of parameters as compared to the new reduction. This occurs for relatively large values of ϵ, small values of q, and/or very high values of κ. Moreover, if ϵ and q are held constant, then Waters' reduction has a constant level more efficiency as $\kappa \to \infty$. When choosing pairing parameters, one can use the minimum of

the two setups suggested by the reductions.

We can calculate the approximate (using conventions discussed in Appendix E) efficiency difference for encryption when Wa is instantiated with pairing setup s_W versus s_N using the following ratio

$$\frac{\mathsf{T}_{\exp}(\mathbb{G}_T) + \mathsf{T}_{\mathrm{op}}(\mathbb{G}_T) + \mathsf{T}_{\exp}(\mathbb{G}_2) + \mathsf{T}_{\exp}(\mathbb{G}_1)}{\mathsf{T}_{\exp}(\mathbb{G}'_T) + \mathsf{T}_{\mathrm{op}}(\mathbb{G}'_T) + \mathsf{T}_{\exp}(\mathbb{G}'_2) + \mathsf{T}_{\exp}(\mathbb{G}'_1)}\ .$$

Here $\mathsf{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathbf{e}, \psi)$ are the pairings as instantiated under setup s_W and $\mathsf{GP}' = (\mathbb{G}'_1, \mathbb{G}'_2, \mathbb{G}'_T, p', \mathbf{e}', \psi')$ are the pairings as instantiated under setup s_N. Figure 2 in the introduction shows this ratio for some s_W, s_N pairs.

Acknowledgments

We thank Brent Waters for pointing out a bug in the proof of an earlier version of this paper. We thank Sarah Shoup for participating in early stages of this work. We thank Dan Boneh and Xavier Boyen for comments on earlier drafts of this work.

References

[1] M. Abdalla, D. Catalano, A. W. Dent, J. Malone-Lee, G. Neven, and N. P. Smart. Identity-based encryption gone wild. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *ICALP 2006*, volume 4052 of *LNCS*, pages 300–311, Venice, Italy, July 10–14, 2006. Springer-Verlag, Berlin, Germany.

[2] M. Abdalla, E. Kiltz, and G. Neven. Generalized key delegation for hierarchical identity-based encryption. In *ESORICS*, volume 4734 of *Lecture Notes in Computer Science*, pages 139–154. Springer, 2007.

[3] M. Bellare, C. Namprempre, and G. Neven. Unrestricted aggregate signatures. In L. Arge, C. Cachin, T. Jurdziński, and A. Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 411–422, Wroclaw, Poland, July 9–13, 2007. Springer-Verlag, Berlin, Germany.

[4] M. Bellare and P. Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426, St. Petersburg, Russia, May 29 –June 1, 2006. Springer-Verlag, Berlin, Germany.

[5] J. Birkett, A. W. Dent, G. Neven, and J. C. N. Schuldt. Efficient chosen-ciphertext secure identity-based encryption with wildcards. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *ACISP 07*, volume 4586 of *LNCS*, pages 274–292, Townsville, Australia, July 2–4, 2007. Springer-Verlag, Berlin, Germany.

[6] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[7] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[8] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer-Verlag, Berlin, Germany.

[9] X. Boyen. General ad hoc encryption from exponent inversion IBE. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 394–411, Barcelona, Spain, May 20–24 2007. Springer-Verlag, Berlin, Germany.

[10] X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM CCS 05*, pages 320–329, Alexandria, Virginia, USA, Nov. 7–11, 2005. ACM Press.

[11] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271, Warsaw, Poland, May 4–8, 2003. Springer-Verlag, Berlin, Germany.

[12] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[13] S. Chatterjee and P. Sarkar. Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. In D. Won and S. Kim, editors, *ICISC 05*, LNCS, pages 424–440, Seoul, Korea, Dec. 1–2, 2005. Springer-Verlag, Berlin, Germany.

[14] S. Chatterjee and P. Sarkar. HIBE with short public parameters without random oracle. In *ASIACRYPT 2006*, LNCS, pages 145–160. Springer-Verlag, Berlin, Germany, Dec. 2006.

[15] D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In D. Chaum and W. L. Price, editors, *EUROCRYPT'87*, volume 304 of *LNCS*, pages 127–141, Amsterdam, The Netherlands, Apr. 13–15, 1987. Springer-Verlag, Berlin, Germany.

[16] L. Chen and Z. Cheng. Security proof of Sakai-Kasahara's identity-based encryption scheme. In N. P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 442–459. Springer, 2005.

[17] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 360–363, Cirencester, UK, Dec. 17–19, 2001. Springer-Verlag, Berlin, Germany.

[18] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372, `http://eprint.iacr.org/`, 2007.

[19] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165. `http://eprint.iacr.org/`, 2006.

[20] D. Galindo. The exact security of pairing based encryption and signature schemes. Based on a talk at Workshop on Provable Security, INRIA, Paris, 2004. `http://www.dgalindo.es/galindoEcrypt.pdf`, 2004.

[21] C. Gentry. Practical identity-based encryption without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464, St. Petersburg, Russia, May 29 –June 1, 2006. Springer-Verlag, Berlin, Germany.

[22] M. Green and S. Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 265–282. Springer-Verlag, Berlin, Germany.

[23] D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38, Santa Barbara, CA, USA, Aug. 17–21, 2008. Springer-Verlag, Berlin, Germany.

[24] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 466–481, Amsterdam, The Netherlands, Apr. 28 – May 2, 2002. Springer-Verlag, Berlin, Germany.

[25] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *ACM CCS 03*, pages 155–164, Washington D.C., USA, Oct. 27–30, 2003. ACM

Press.
[26] E. Kiltz and D. Galindo. Direct chosen-ciphertext secure IB-KEM without random oracles. In L. M. Batten and R. Safavi-Naini, editors, *ACISP 06*, volume 4058 of *LNCS*, pages 336–347, Melbourne, Australia, July 3–5, 2006. Springer-Verlag, Berlin, Germany.
[27] A. K. Lenstra. Unbelievable security: Matching AES security using public key systems. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 67–86, Gold Coast, Australia, Dec. 9–13, 2001. Springer-Verlag, Berlin, Germany.
[28] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001.
[29] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.
[30] D. Naccache. Secure and practical identity-based encryption. Cryptology ePrint Archive, Report 2005/369, 2005. `http://eprint.iacr.org/`.
[31] National Institute for Standards and Technology. Recommendation for Key Management Part 1: General (revised), 2005. NIST Special Publication 800-57.
[32] J. M. Pollard. Monte Carlo methods for index computation (mod p), 1987. *Mathematics of Computation*, 32:918-924.
[33] T. Ristenpart and S. Yilek. The Power of Proofs-of-Possession: Securing Multiparty Signatures from Rogue-key Attacks. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 228–245, Barcelona, Spain, May 20–24 2007. Springer-Verlag, Berlin, Germany.
[34] A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.
[35] R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054, 2003. `http://eprint.iacr.org/`.
[36] B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.

A Derivatives of Waters' IBE

A large body of research [10, 1, 5, 26, 30, 13, 14, 22] utilizes Waters' scheme. Recall that Waters' already proposed a heirarchical IBE scheme based on Wa in [36], and subsequently there have been numerous derivative works. All use the artificial abort, either due to a black-box reduction to Waters' (H)IBE or as an explicit step in a direct proof. Our new proof technique immediately benefits those schemes that utilize Waters' scheme directly (i.e. in a black-box manner). For the rest, we believe that our techniques can be applied but have not checked the details.

- Naccache [30] and Chatterjee and Sarkar [13, 14] independently and concurrently introduced a space-time trade-off for Wa that involves modifying the hash function utilized from $H(\mathbf{u}, I) = \mathbf{u}[0] \prod_{i=1}^{n} \mathbf{u}[i]^{I[i]}$ for $\mathbf{u} \in \mathbb{G}_1^{n+1}$ to $H'(\mathbf{u}, I) = \mathbf{u}[0] \prod_{i=1}^{\ell} \mathbf{u}[i]^{I[i]}$ where $\mathbf{u} \in \mathbb{G}_1^{\ell+1}$ and each $I[i]$ is now an n/ℓ-bit string. For appropriate choice of ℓ this will significantly reduce the number of elements included in the master public key. However the new choice of hash function impacts the reduction tightness, and since their proof includes just minor changes to Waters', our new reduction will increase the efficiency of this time/space trade-off for various security levels.
- In [10] BB_1- and Wa-based constructions of CCA-secure public-key encryption schemes and their proofs for the Wa case directly utilize artificial aborts.

- Kiltz and Galindo [26] propose a construction of CCA-secure identity-based key encapsulation that is a modified version of Wa.
- Wildcard IBE [1, 5] is a generalization of heirarchical IBE that allows encryption to identities that include wildcards, e.g. "*@anonymous.com". In [1] a wildcard IBE scheme is proposed that utilizes the Waters HIBE scheme, and the proof is black-box to it. In [5] a wildcard identity-based KEM is produced based (in a non-black-box manner) on Waters' IBE.
- Wicked IBE [2] allows generation of private keys for wildcard identities. These private keys can then be used to generate derivative keys that replace the wildcards with any concrete identity string. They suggest using the Waters' HIBE scheme to achieve full security in their setting.
- Blind IBE, as introduced by Green and Hohenberger [22], enables the "trusted" master key generator to generate a private key for an identity without learning anything about the identity. To prove a Waters'-based blind IBE scheme secure they utilize the Naccache proof [30] (mentioned above). They utilize blind IBE schemes to build efficient and fully-simulatable oblivious transfer protocols based on the assumptions inherited from the BDH-based IBE schemes used.

B Proof of Lemma 2.1

We prove the lemma utilizing a coin-counting argument, following the approach of [4, Lemma 1]. For a pair of games G and H and an advsersary $\mathcal{A}$ we assume that the number of random selection assignments $s \xleftarrow{\$} S$ is less than or equal to some number $b \geq 0$ and each set S sampled from is of size less than or equal to $d > 0$ (for simplicity we disallow empty sets). Then we define the coins for $(\mathrm{G}, \mathrm{H}, \mathcal{A})$ to be the set $C = [0\,..\,d!]^b$. Using coins $\mathbf{c} = (c_1, \ldots, c_b) \in C$, the i^{th} random selection statement $s \xleftarrow{\$} S = \{s_1, s_2, \ldots, s_m\}$ for $m < d$ in $\mathrm{G}^{\mathcal{A}}$ (equivalently $\mathrm{H}^{\mathcal{A}}$) is executed by assigning to s the value $s_{c_i \bmod m}$. (Note that there is no distinction between random assignment statements in the game or the adversary, they are treated the same.) Since m divides d, then selecting $\mathbf{c}$ uniformly ensures that each random assignment statement returns an indendently and uniformly chosen element from the set. We write $\mathrm{G}^{\mathcal{A}}(\mathbf{c})$ to denote running $\mathrm{G}^{\mathcal{A}}$ on the particular choice of coins $\mathbf{c}$.

Let C be the set of coins for $(\mathrm{G}, \mathrm{H}, \mathcal{A})$ where G and H are identical-until-bad games and $\mathcal{A}$ is an adverary. Let GD_{G} (resp. GD_H) be the event that bad is not set in game G (resp. H). Let $CG_0 = \{\mathbf{c} \in C \,:\, \mathrm{G}^{\mathcal{A}}(\mathbf{c}) \Rightarrow y\}$ and $CG_1 = C \backslash CG_0$. Let $CG^{\mathsf{good}} = \{\mathbf{c} \in C \,:\, \mathrm{G}^{\mathcal{A}}(\mathbf{c})$ does not set $\mathsf{bad}\}$. Let $CG_k^{\mathsf{good}} = CG^{\mathsf{good}} \cap CG_k$ for $k \in [0, 1]$. Define CH_0, CH_1, CH^{good}, and CH_k^{good} in the natural way.

Because G and H are identical-until-bad we have that $CG_1^{\mathsf{good}} = CH_1^{\mathsf{good}}$. This is true because any vector $\mathbf{c} \in CG_1^{\mathsf{good}}$ does not cause bad to be set, thereby only code that is common with H is executed. So if $\mathrm{G}^{\mathcal{A}}(\mathbf{c}) \Rightarrow y$ then $\mathrm{H}^{\mathcal{A}}(\mathbf{c}) \Rightarrow y$, making $\mathbf{c} \in CH_1^{\mathsf{good}}$. A similar argument works in the other direction. So we have that

$$\Pr\left[\,\mathrm{G}^{\mathcal{A}} \Rightarrow 1 \wedge \mathsf{GD}_{\mathrm{G}}\,\right] = \frac{|CG_1^{\mathsf{good}}|}{|C|} = \frac{|CH_1^{\mathsf{good}}|}{|C|} = \Pr\left[\,\mathrm{H}^{\mathcal{A}} \Rightarrow 1 \wedge \mathsf{GD}_{\mathrm{H}}\,\right].$$

C Proof of Lemma 3.2

As per the lemma statement, let $(g_1, g_2, A_2, A_1, B_2, B_1, \mathbf{x}, \mathbf{y}, \mathbf{u}, S, W)$ be simulation parameters. For the first claim, we recall that $\mathbf{x} \in X$ ensures that $\mathsf{F}(\mathbf{x}, I) \in [-n(m-1) \ldots n(m-1)]$. But the fact that $m = \lceil 9q/\epsilon \rceil$ and the assumption that $q \leq p\epsilon/9n$ together imply that $n(m-1) < p$. This implies that if $\mathsf{F}(\mathbf{x}, I) \neq 0$ then $\mathsf{F}(\mathbf{x}, I) \bmod p \neq 0$ as well. Thus $w = \mathsf{F}(\mathbf{x}, I)^{-1}$ is well-defined. Let

a and b be the discrete logs (base g_1) of A_1 and B_1, respectively. Define the function $f\colon\ \mathbb{Z}_p \to \mathbb{Z}_p$ by $f(r) = r - wa \bmod p$ for all $r \in \mathbb{Z}_p$. Now we claim that for any $r \in \mathbb{Z}_p$ we have

$$A_1^b H(\mathbf{u}, I)^{f(r)} = B_1^{\mathsf{F}(\mathbf{x},I)\cdot r} g_1^{\mathsf{G}(\mathbf{y},I)\cdot r} A_1^{-\mathsf{G}(\mathbf{y},I)\cdot w} \quad (27)$$

$$g_2^{f(r)} = g_2^r A_2^{-w} \quad (28)$$

In other words, the output of $\mathsf{KgS}(g_1, g_2, A_2, A_1, B_1, \mathbf{x}, \mathbf{y}, I)$ when r is the underlying randomness equals the output of $\mathsf{Kg}(mpk, msk, I)$ when $f(r)$ is the underlying randomness. But f is a permutation over $\mathbb{Z}_p$, so if r is uniformly distributed, so is $f(r)$. This proves that the outputs of $\mathsf{Kg}(mpk, msk, I)$ and $\mathsf{KgS}(g_1, g_2, A_2, A_1, B_1, \mathbf{x}, \mathbf{y}, I)$ are identically distributed as claimed. Let us now verify equations (27) and (28). We have:

$$\begin{aligned}
A_1^b H(\mathbf{u}, I)^{f(r)} &= A_1^b \cdot \left(\mathbf{u}[0] \cdot \prod_{i=1}^{n} \mathbf{u}[i]^{I[i]}\right)^{r - w\cdot a} \\
&= g_1^{ab} \cdot \left(B_1^{\mathbf{x}[0]} g_1^{\mathbf{y}[0]} \cdot \prod_{i=1}^{n} B_1^{\mathbf{x}[i]I[i]} g_1^{\mathbf{y}[i]I[i]}\right)^{r - w\cdot a} \\
&= B_1^a \cdot \left(B_1^{\mathsf{F}(\mathbf{x},I)} \cdot g_1^{\mathsf{G}(\mathbf{y},I)}\right)^{r - w\cdot a} \\
&= B_1^a \cdot B_1^{\mathsf{F}(\mathbf{x},I)\cdot r} \cdot B_1^{-a} \cdot g_1^{\mathsf{G}(\mathbf{y},I) r} \cdot g_1^{-a\mathsf{G}(\mathbf{y},I)\cdot w} \\
&= B_1^{\mathsf{F}(\mathbf{x},I)\cdot r} \cdot g_1^{\mathsf{G}(\mathbf{y},I) r} \cdot A_1^{-\mathsf{G}(\mathbf{y},I)\cdot w}\,.
\end{aligned}$$

We know that $g_1 = \psi(g_2)$ and $A_1 = \psi(A_2) = g_1^a$. This means that $A_2 = g_2^a$. So

$$g_2^{f(r)} = g_2^{r - w\cdot a} = g_2^r A_2^{-w}\,,$$

which completes the proof of the first part. For the second part, we assume that $\mathsf{F}(\mathbf{x}, I) = 0$ and let $(C_1, C_2, C_3) \leftarrow \mathsf{EncS}(S, W, M, \mathbf{y}, I)$. Let s be the discrete log of S to base g_2. Then $C_1 = W \cdot M$; $C_2 = S$; and using that $\psi(S) = g_1^s$ we have

$$C_3 = \psi(S)^{\mathrm{G}(\mathbf{y},I)} = (g_1^s)^{\mathrm{G}(\mathbf{y},I)} = \left(g_1^{\mathrm{G}(\mathbf{y},I)}\right)^s = \left(B_1^0 g_1^{\mathrm{G}(\mathbf{y},I)}\right)^s = \left(B_1^{\mathsf{F}(\mathbf{x},I)} g_1^{\mathrm{G}(\mathbf{y},I)}\right)^s = H(\mathbf{u}, I)^s\,.$$

Thus (C_1, C_2, C_3) is exactly $(W \cdot M, S, H(\mathbf{u}, I)^s)$.

D Proof of Lemma 3.5

Fix an arbitrary $(I_0, \ldots, I_q) \in \mathsf{ID}$. We will show that

$$\frac{1}{n(m-1)+1}\left(1 - \frac{q}{m}\right) \le \gamma(I_0, \ldots, I_q) \quad (29)$$

$$\frac{1}{n(m-1)+1} \ge \gamma(I_0, \ldots, I_q)\,. \quad (30)$$

The lemma follows. We first prove (29). We stress that below $(I_0, \ldots, I_q) \in \mathsf{ID}$ is fixed and the probability is only over the random choice of $\mathbf{x}$ from X. Let E_i be the event that $\mathsf{F}(\mathbf{x}, I_i) = 0$. Then

$$\begin{aligned} 1 - \gamma(I_0, \ldots, I_q) &= \Pr\left[\, \overline{\mathsf{E}}_0 \vee \mathsf{E}_1 \vee \cdots \vee \mathsf{E}_q \,\right] \\ &= \Pr\left[\, \overline{\mathsf{E}}_0 \vee (\mathsf{E}_0 \wedge \mathsf{E}_1) \vee \cdots \vee (\mathsf{E}_0 \wedge \mathsf{E}_q) \,\right] \\ &= \Pr\left[\, \overline{\mathsf{E}}_0 \,\right] + \Pr\left[\, (\mathsf{E}_0 \wedge \mathsf{E}_1) \vee \cdots \vee (\mathsf{E}_0 \wedge \mathsf{E}_q) \,\right] \qquad (31) \\ &\leq \Pr\left[\, \overline{\mathsf{E}}_0 \,\right] + \sum_{i=1}^{q} \Pr\left[\, \mathsf{E}_0 \wedge \mathsf{E}_i \,\right] \qquad (32) \end{aligned}$$

For every choice of $\mathbf{x}[1], \ldots, \mathbf{x}[n]$ there is exactly one choice of $\mathbf{x}[0]$ such that $\mathsf{F}(\mathbf{x}, I_0) = 0$, so that

$$\Pr\left[\, \mathsf{E}_0 \,\right] = \frac{1}{n(m-1)+1}\,. \qquad (33)$$

Now let $i \in [1\,..\,q]$. Since $I_0 \neq I_i$ there is a $j \in [1\,..\,n]$ such that $I_i[j] \neq I_0[j]$. Let $s \in \{0, i\}$ be such that $I_s[j] = 0$ and $I_{i-s}[j] = 1$. For any choice of $\mathbf{x}[1], \ldots, \mathbf{x}[j-1], \mathbf{x}[j+1], \ldots, \mathbf{x}[n]$ let

$$t_0 = -\sum_{l \neq j} \mathbf{x}[l] I_s[l] \quad \text{and} \quad t_j = -t_0 - \sum_{l \neq j} \mathbf{x}[l] I_{i-s}[l]\,.$$

Then $\mathsf{F}(\mathbf{x}, I_0)$ and $\mathsf{F}(\mathbf{x}, I_i)$ are both zero only if $(\mathbf{x}[0], \mathbf{x}[j]) = (t_0, t_j)$. Since $\mathbf{x}[0]$ is drawn from $[-n(m-1)\,..\,0]$ and $\mathbf{x}[j]$ from $[0\,..\,m-1]$ this means that

$$\Pr\left[\, \mathsf{E}_0 \wedge \mathsf{E}_i \,\right] \leq \frac{1}{n(m-1)+1} \cdot \frac{1}{m}\,. \qquad (34)$$

One might ask why (34) is an inequality rather than an equality; isn't the probability that $(\mathbf{x}[0], \mathbf{x}[j])$ equals (t_0, t_j) exactly the term on the right? The reason it is not is that t_j may not lie in $[0\,..\,m-1]$, so for some choices of the other coordinates of $\mathbf{x}$, the probability that $\mathbf{x}[j] = t_j$ is zero.

Putting together (32), (33), and (34) we have

$$1 - \gamma(I_0, \ldots, I_q) \;\leq\; 1 - \frac{1}{n(m-1)+1} + \sum_{i=1}^{q} \frac{1}{m} \cdot \frac{1}{n(m-1)+1} \;=\; 1 - \frac{1}{n(m-1)+1}\left(1 - \frac{q}{m}\right)$$

which implies (29).

To derive (30), we note that (31) implies that $1 - \gamma(I_0, \ldots, I_q) \geq \Pr\left[\, \overline{\mathsf{E}}_0 \,\right]$ which, when combined with (33), implies (30).

E Instantiating Pairing Parameters

Pairings and their costs. Let $\mathsf{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathbf{e}, \psi)$ be pairing parameters as per Section 2. We focus on the Type 2 [19] setting for these parameters, since the Type 1 (i.e., the symmetric setting where $\mathbb{G}_1 = \mathbb{G}_2$) is not easily instantiated for higher security levels [19]. In the Type 2 setting, $\mathbb{G}_1$ is a subgroup of $E(\mathbb{F}_r)$ where $\mathbb{F}_r$ is a finite field. Group $\mathbb{G}_2$ is a subgroup of $E(\mathbb{F}_{r^k})$ where k is called the embedding degree. Group $\mathbb{G}_T$ is a subgroup of $\mathbb{F}^*_{r^k}$. All three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are of order p for some prime p that divides $r^k - 1$. Let $\rho = (\log r)/(\log p)$, which we call the representation multiplier. Elements of $\mathbb{G}_1$ require $\rho \log p$ bits to represent while elements of $\mathbb{G}_2$ and $\mathbb{G}_T$ require $k\rho \log p$ bits to represent. We fix the following conventions regarding the computational costs of basic operations related to an instantiation of GP.

- A group operation in $E(\mathbb{F}_r)$ takes at most 16 multiplications in the underlying field $\mathbb{F}_r$ [19]. A multiplication in $\mathbb{F}_r$ takes time proportional to $\log^2 r$ [28]. Therefore, we estimate $\mathsf{T}_{\mathrm{op}}(\mathbb{G}_1) \leq$

$16 \cdot \log^2 r$.

- A group exponentiation in $E(\mathbb{F}_r)$ takes in the worst case $\log p$ operations in $E(\mathbb{F}_r)$, or $\mathsf{T}_{\text{exp}}(\mathbb{G}_1) \leq \log p \cdot \mathsf{T}_{\text{op}}(\mathbb{G}_1)$ [20].
- A group operation in $\mathbb{G}_2$ takes $\mathsf{T}_{\text{op}}(\mathbb{G}_2) = k^2 \cdot \mathsf{T}_{\text{op}}(\mathbb{G}_1)$ [19] while an operation in $\mathbb{G}_T$ takes $\mathsf{T}_{\text{op}}(\mathbb{G}_T) = k^2 \cdot \mathsf{T}_{\text{op}}(\mathbb{G}_1)/16$ [19]. Exponentiations in either group, in the worst case, require time $\mathsf{T}_{\text{exp}}(\mathbb{G}_2) \leq \log p \cdot \mathsf{T}_{\text{op}}(\mathbb{G}_2)$ and $\mathsf{T}_{\text{exp}}(\mathbb{G}_T) \leq \log p \cdot \mathsf{T}_{\text{op}}(\mathbb{G}_T)$ [20].
- A multi-exponentiation (of only a few elements) in a group $\mathbb{G}$ takes time approximately the same as a single exponentiation, i.e. $\mathsf{T}_{\text{exp}}(\mathbb{G})$ [29].
- Computing the homomorphism ψ requires time $\mathsf{T}_\psi < \mathsf{T}_{\text{exp}}(\mathbb{G}_1)$.
- Addition or multiplication over the integers (of size at most 2^p) or the integers modulo p is unit cost.

Note that our estimates above are based on generic implementations which allow for simplicity but do not consider instance-specific improvements on computational time. The running times computed might therefore be over-estimates of the true costs for some choices of parameters and particular implementations.

Concrete pairing parameters. We will consider several sets of values of $\log p$, k, and ρ (which fixes r, since $\log r = \rho \log p$), shown in Figure 6. The figure displays curve parameters for commonly desired levels of security. Here the security level s means that roughly 2^s operations should be needed to break discrete log in one of the corresponding groups ($\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$). The group sizes are selected following the guidelines of [31, 27, 28, 18, 19], and specific values for ρ and k follow the recommendations of [18]. Note that for $s = 192$ and $s = 256$, it is not possible to achieve $\rho = 1$ without significantly increasing the subgroup size or using a much larger embedding degree, and unfortunately we do not know how to construct pairings for prime order groups with such a large embedding degree [18]. Thus we follow the recommendations of [18], and choose the smallest curves with $\rho \approx 2$. We've also set $\log r^k$ to be large enough so that the resulting finite field is of size at least as large as those suggested by the NIST key size recommendations [31]. (This is meant to ensure, relative to the best known attacks, that solving DL in $\mathbb{G}_T$ is no easier than solving discrete logarithms in $\mathbb{G}_1$, as we discuss below.)

Time units. To be concrete we will require a common time unit, which we set to be the cost of the group operation for the smallest group considered. Namely, the time to do a group operation in our selected 160-bit curve: $\mathsf{T}_{\text{u}} = 16 \cdot 160^2$.

Solving DBDH. To the best of our knowledge, the best (published) attack for solving DBDH in our setting is to compute a discrete log in one of the three groups. For $\mathbb{G}_1$ and $\mathbb{G}_2$, which as outlined above are subgroups of elliptic curves, the best published attacks for solving discrete log are generic [28], e.g. Pollard's algorithm [32]. Let $\mathcal{P}$ be the dl (discrete log) adversary that implements Pollard's algorithm against a group $\mathbb{G}$. Then we have that

$$\epsilon_p = \mathbf{Adv}^{\text{dl}}_{\mathbb{G}}(\mathcal{P}) = \mathbf{T}(\mathcal{P})^2 / (.88\sqrt{p} \cdot \mathsf{T}_{\text{op}}(\mathbb{G}))^2 \tag{35}$$

where $|\mathbb{G}| = p$ and the advantage of a dl adversary is defined in the usual way (the probability that it successfully computes, given base $g \in \mathbb{G}^*$ and g^a for $a \xleftarrow{\$} \mathbb{Z}_p$, the discrete log a). Since elements of $\mathbb{G}_2$ have a larger representation than those of $\mathbb{G}_1$ (by a factor of k, the embedding degree), Pollard's algorithm will run faster against $\mathbb{G}_1$ than $\mathbb{G}_2$, so we need not consider attacking $\mathbb{G}_2$ directly. For $\mathbb{G}_T$, better attacks are known, particularly index-calculus methods [28]. Such attacks are sub-exponential in the number of group operations, meaning they work faster than attacks against $\mathbb{G}_1$ and $\mathbb{G}_2$. Security therefore requires that $\mathbb{G}_T$ has a representation that is significantly larger than

that of $\mathbb{G}_1$'s. Our choices of pairing instantiations described above ensure that the time to compute a discrete log in $\mathbb{G}_1$ (using Pollard's algorithm) is a lower bound on the time to compute a discrete log in $\mathbb{G}_T$ (using index-calculus methods). This was accomplished by choosing r^k appropriately.

F The BB_1 Scheme and its Security

BONEH-BOYEN IBE SCHEME. Fix pairing parameters $\mathsf{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathbf{e}, \psi)$ and let n be a positive integer that is less than the number of bits required to represent elements in $\mathbb{G}_1$. Define the hash family H_{BB}: $\mathbb{G}_1^2 \times \{0,1\}^n \to \mathbb{G}_1$ by $H_{BB}(\mathbf{u}, I) = \mathbf{u}[0] \cdot \mathbf{u}[1]^I$ for any $\mathbf{u} \in \mathbb{G}_1^2$ and where $I \in \{0,1\}^n$ is taken first encoded as an element of $\mathbb{Z}_p$ in some canonical way. (The first argument is the key and the second is the input.) The Boneh-Boyen IBE scheme [6] $\mathsf{BB}_1 = (\mathsf{Pg}^*, \mathsf{Kg}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ associated to GP has associated identity space $\mathsf{IdSp} = \{0,1\}^n$ and message space $\mathsf{MsgSp} = \mathbb{G}_T$. Its algorithms are defined in the same manner as those given below for Waters' scheme, modulo replacing $\mathbf{u} \xleftarrow{\$} \mathbb{G}_1^{n+1}$ in Pg with $\mathbf{u} \xleftarrow{\$} \mathbb{G}_1^2$ and replacing H with H_{BB} in Kg and Enc.

SECURITY OF BB_1. Boneh and Boyen prove that BB_1 is selective-ID secure [6]. Recall that selective-ID security [11, 12] restricts attention to adversaries that must specify the challenge ID before receiving the system parameters. To return to full security (as given in Section 2), they show that any selective-ID-secure IBE scheme is also fully secure, though the reduction loses a factor of $2^n = |\mathsf{IdSp}|$ in the advantage relation. We combine their results to assess the concrete security of BB_1. Let $\mathcal{A}$ be an ind-cpa adversary against $\mathsf{BB}_1 = (\mathsf{Pg}^*, \mathsf{Kg}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ which has advantage $\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{BB}_1}(\mathcal{A})$ and that makes q **Extract** queries. Then [6, Theorem 4.1] and [6, Theorem 7.1] together imply that there exists a dbdh-adversary $\mathcal{B}_{\mathsf{BB}_1}$ such that

$$\mathbf{Adv}^{\text{dbdh}}_{\mathsf{GP}}(\mathcal{B}_{\mathsf{BB}^*}) \geq \frac{\epsilon}{2^n} \quad \text{and} \quad \mathbf{T}(\mathcal{B}_{\mathsf{BB}^*}) = \mathbf{T}(\mathcal{A}) + \mathbf{T}'_{\text{sim}}(q)$$

where N is the size of the identity space and

$$\mathbf{T}'_{\text{sim}}(q) = \mathcal{O}((q+2) \cdot \mathsf{T}_{\text{exp}}(\mathbb{G}_1) + q \cdot \mathsf{T}_{\text{exp}}(\mathbb{G}_2) + \mathsf{T}_{\text{op}}(\mathbb{G}_T) + \mathsf{T}_\psi)\,.$$

The work factor for $\mathcal{B}_{\mathsf{BB}^*}$ is then

$$\mathbf{WF}(\mathcal{B}_{\mathsf{BB}^*}) = 2^n \cdot \frac{\mathbf{T}(\mathcal{A}) + \mathbf{T}'_{\text{sim}}(q)}{\epsilon} = 2^n \cdot \mathbf{WF}(\mathcal{A}) + \epsilon^{-1}\mathbf{T}'_{\text{sim}}(q)\,.$$

We can compare this reduction's efficiency to the two Wa reductions again by assessing when $\mathbf{WF}(\mathcal{B}_{\mathsf{BB}^*}) \leq \mathbf{WF}(\mathcal{P})$ is satisfied. Figure 7 shows the result.

G Simulation Overhead for $\mathcal{B}$

Let $\mathsf{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathbf{e}, \psi)$ be pairing parameters as per Section 2. We calculate $\mathbf{T}_{\text{sim}}$, which is the overhead of the reduction. The time used by $\mathcal{B}$ beyond that already utilized by $\mathcal{A}$ includes time to answer $\mathcal{A}$'s oracle queries. Note that all the integer operations used by $\mathcal{B}$ are on integers of absolute value less than p. This holds even for integers output by $\mathsf{F}(\mathbf{x}, I)$ as shown by Equation (11).

Adversary $\mathcal{B}$ first performs two computations of ψ and n multi-exponentiations in $\mathbb{G}_1$. Each **Extract** query may execute KgS, resulting in n additions over $\mathbb{Z}$ (to compute $\mathsf{F}(\mathbf{x}, I)$), an inversion over the integers modulo p (to compute $\mathsf{F}(\mathbf{x}, I)^{-1}$)), a multi-exponentiation in $\mathbb{G}_1$, and a multi-exponentiation in $\mathbb{G}_2$. The **LR** query may execute EncS, resulting in a multiplication in $\mathbb{G}_T$, a computation of ψ, and an exponentiation in $\mathbb{G}_1$. Thus,

$$\mathbf{T}_{\text{sim}}(n, q) = \mathcal{O}\left(\mathsf{T}_\psi + (n+q) \cdot \mathsf{T}_{\text{exp}}(\mathbb{G}_1) + q \cdot \mathsf{T}_{\text{exp}}(\mathbb{G}_2) + qn + \mathsf{T}_{\text{op}}(\mathbb{G}_T)\right)\,. \tag{36}$$

κ	ϵ	q	s_{BB}	s_W	s_N
60	2^{-10}	2^{10}	192	112	112
60	2^{-10}	2^{20}	192	112	112
60	2^{-10}	2^{30}	192	192	128
60	2^{-10}	2^{40}	192	192	192
60	2^{-20}	2^{10}	192	112	112
60	2^{-20}	2^{20}	192	192	128
60	2^{-20}	2^{30}	192	192	192
60*	2^{-20}	2^{40}	192	192	192
60	2^{-30}	2^{10}	192	192	128
60	2^{-30}	2^{20}	192	192	192
60*	2^{-30}	2^{30}	192	192	192
60*	2^{-30}	2^{40}	192	256	192
60	2^{-40}	2^{10}	192	192	192
60*	2^{-40}	2^{20}	192	192	192
60*	2^{-40}	2^{30}	192	256	192
60*	2^{-40}	2^{40}	192	256	256

κ	ϵ	q	s_{BB}	s_W	s_N
70	2^{-10}	2^{10}	256	112	112
70	2^{-10}	2^{20}	256	112	112
70	2^{-10}	2^{30}	256	192	128
70	2^{-10}	2^{40}	256	192	192
70	2^{-20}	2^{10}	256	112	112
70	2^{-20}	2^{20}	256	192	128
70	2^{-20}	2^{30}	256	192	192
70	2^{-20}	2^{40}	256	256	192
70	2^{-30}	2^{10}	256	192	128
70	2^{-30}	2^{20}	256	192	192
70	2^{-30}	2^{30}	256	256	192
70*	2^{-30}	2^{40}	256	256	192
70	2^{-40}	2^{10}	256	192	192
70	2^{-40}	2^{20}	256	256	192
70*	2^{-40}	2^{30}	256	256	192
70*	2^{-40}	2^{40}	256	256	256

κ	ϵ	q	s_{BB}	s_W	s_N
80	2^{-10}	2^{10}	256	112	112
80	2^{-10}	2^{20}	256	112	128
80	2^{-10}	2^{30}	256	192	192
80	2^{-10}	2^{40}	256	192	192
80	2^{-20}	2^{10}	256	112	128
80	2^{-20}	2^{20}	256	192	192
80	2^{-20}	2^{30}	256	192	192
80	2^{-20}	2^{40}	256	256	192
80	2^{-30}	2^{10}	256	192	192
80	2^{-30}	2^{20}	256	192	192
80	2^{-30}	2^{30}	256	256	192
80	2^{-30}	2^{40}	256	256	192
80	2^{-40}	2^{10}	256	192	192
80	2^{-40}	2^{20}	256	256	192
80	2^{-40}	2^{30}	256	256	192
80*	2^{-40}	2^{40}	256	256	256

κ	ϵ	q	s_{BB}	s_W	s_N
100	2^{-10}	2^{10}	–	128	192
100	2^{-10}	2^{20}	–	192	192
100	2^{-10}	2^{30}	–	192	192
100	2^{-10}	2^{40}	–	192	192
100	2^{-20}	2^{10}	–	128	192
100	2^{-20}	2^{20}	–	192	192
100	2^{-20}	2^{30}	–	192	192
100	2^{-20}	2^{40}	–	256	192
100	2^{-30}	2^{10}	–	192	192
100	2^{-30}	2^{20}	–	192	192
100	2^{-30}	2^{30}	–	256	192
100	2^{-30}	2^{40}	–	256	192
100	2^{-40}	2^{10}	–	192	192
100	2^{-40}	2^{20}	–	256	192
100	2^{-40}	2^{30}	–	256	192
100	2^{-40}	2^{40}	–	256	256

κ	ϵ	q	s_{BB}	s_W	s_N
128	2^{-10}	2^{10}	–	192	192
128	2^{-10}	2^{20}	–	192	192
128	2^{-10}	2^{30}	–	192	192
128	2^{-10}	2^{40}	–	192	192
128	2^{-20}	2^{10}	–	192	192
128	2^{-20}	2^{20}	–	192	192
128	2^{-20}	2^{30}	–	192	192
128	2^{-20}	2^{40}	–	256	256
128	2^{-30}	2^{10}	–	192	192
128	2^{-30}	2^{20}	–	192	192
128	2^{-30}	2^{30}	–	256	256
128	2^{-30}	2^{40}	–	256	256
128	2^{-40}	2^{10}	–	192	192
128	2^{-40}	2^{20}	–	256	256
128	2^{-40}	2^{30}	–	256	256
128	2^{-40}	2^{40}	–	256	256

κ	ϵ	q	s_{BB}	s_W	s_N
192	2^{-10}	2^{10}	–	256	256
192	2^{-10}	2^{20}	–	256	256
192	2^{-10}	2^{30}	–	256	256
192	2^{-10}	2^{40}	–	256	256
192	2^{-20}	2^{10}	–	256	256
192	2^{-20}	2^{20}	–	256	256
192	2^{-20}	2^{30}	–	256	256
192	2^{-20}	2^{40}	–	256	–
192	2^{-30}	2^{10}	–	256	256
192	2^{-30}	2^{20}	–	256	256
192	2^{-30}	2^{30}	–	256	–
192	2^{-30}	2^{40}	–	256	–
192	2^{-40}	2^{10}	–	256	256
192	2^{-40}	2^{20}	–	256	–
192	2^{-40}	2^{30}	–	256	–
192	2^{-40}	2^{40}	–	–	–

Figure 7: Comparison of pairing setups required to provably ensure security of BB_1 and Wa for various security levels κ and values of ϵ and q. Here s_{BB} represents BB_1 under Boneh and Boyen's reduction, s_W represents Wa under Waters' reduction, and s_N represents Wa under the new reduction. A dash indicates that a pairing setup greater than 256 is required.