Full version of an extended abstract published in Proceedings of CT-RSA 2013, Springer-Verlag, Feb. 2013. Available from the IACR Cryptology ePrint Archive as Report 2012/687.

The k-BDH Assumption Family: Bilinear Map Cryptography from Progressively Weaker Assumptions

Karyn Benson, Hovav Shacham [*]
University of California, San Diego
{kbenson, hovav}@cs.ucsd.edu

Brent Waters[†]
University of Texas at Austin
bwaters@cs.utexas.edu

February 4, 2013

Abstract

Over the past decade bilinear maps have been used to build a large variety of cryptosystems. In addition to new functionality, we have concurrently seen the emergence of many strong assumptions. In this work, we explore how to build bilinear map cryptosystems under progressively weaker assumptions.

We propose k-BDH, a new family of progressively weaker assumptions that generalizes the decisional bilinear Diffie-Hellman (DBDH) assumption. We give evidence in the generic group model that each assumption in our family is strictly weaker than the assumptions before it. DBDH has been used for proving many schemes secure, notably identity-based and functional encryption schemes; we expect that our k-BDH will lead to generalizations of many such schemes.

To illustrate the usefulness of our k-BDH family, we construct a family of selectively secure Identity-Based Encryption (IBE) systems based on it. Our system can be viewed as a generalization of the Boneh-Boyen IBE, however, the construction and proof require new ideas to fit the family. We then extend our methods to produces hierarchical IBEs and CCA security; and give a fully secure variant. In addition, we discuss the opportunities and challenges of building new systems under our weaker assumption family.

Keywords: pairings, Identity Based Encryption, weaker assumptions

[*]Supported by the MURI program under AFOSR Grant No. FA9550-08-1-0352.

[†]Supported by NSF CNS-0915361 and CNS-0952692, AFOSR Grant No: FA9550-08-1-0352, DARPA PROCEED, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship.

1 Introduction

Since the introduction of the Boneh-Franklin [BF03] Identity-Based Encryption (IBE) system a decade ago, we have seen an explosion of new cryptosystems based on bilinear maps. These systems have provided a wide range of functionality including: new signature systems, functional encryption, e-cash, "slightly" homomorphic encryption, broadcast encryption and oblivious transfer to name just a few. The focus of many of this work was to develop new (and often not realized before) functionality. While Boneh-Franklin and many first IBE systems used "core" assumptions such as the Bilinear Diffie-Hellman or decisional variants, over time there has been a trend in bilinear map based work to employ stronger assumptions in order to obtain these functionalities. Examples of these assumptions range from "q-type" [Gen06] assumptions, assumptions in composite order groups [BGN05], interactive assumptions [AP05] and proofs that appealed directly on the generic group model [BB04b, Boy08]

Interestingly, even some work that focused on tightening security (versus achieving new functionality) have had to employ relatively strong assumptions. For example, Gentry and Halevi [GH09] and Waters [Wat09] proposed two different approaches for solving the problem of achieving adaptive security for Hierarchical Identity-Based encryption. To achieve this the former used a q-type assumption where the strength of the assumption depends on the number of attacker private key queries. The latter used the decisional-Linear assumption, where the target of the assumption is in the source element of the bilinear group versus the target element. Both of these assumptions are potentially stronger than the classic decisional-BDH prior IBE and related systems were built upon.

Our Goals. In this work, we move in the opposite direction of this trend. We will build bilinear map systems that depend on *weaker* assumptions than the decisional-BDH assumption. In particular, we want to create a suitable family of assumptions that becomes progressively weaker as some parameter k is increased. Therefore one can increase k as a hedge against potential future attacks such as an n-linear map for $n > 2$.

A natural starting point for our investigation is the k-Linear family of assumptions [HK07, Sha07], which generalizes the decisional Diffie-Hellman assumption (DDH) and the decisional Linear assumption of Boneh, Boyen, and Shacham [BBS04]. For $k \geq 1$, a k-Linear problem instance is a tuple $(g, g_1, \ldots, g_k, g_1^{r_1}, \ldots, g_k^{r_k}, T)$, where the generators are random in the group $\mathbb{G}$, the exponents in its support $\mathbb{Z}_p$, and the goal is to determine whether T is equal to $g^{r_1+\cdots+r_k}$ or random. DDH is 1-Linear, and the Linear assumption is 2-Linear.

The k-Linear assumption family has been successfully used to build chosen ciphertext secure encryption [HK07, Sha07]; to construct pseudorandom functions [LW09, BMR10]; to construct public-key encryption secure in the presence of encryption cycles [BHHO08, CCS09] and public-key encryption resilient to key leakage [NS09, DHLAW10]; to construct lossy trapdoor functions [FGK$^+$10]; to construct leakage-resilient signatures [BSW11].

While the k-Linear family has been successful in the above contexts, we desire an assumption that can be used in bilinear map cryptosystems in place of where DBDH has typically been applied. Here using the k-Linear family does not appear well suited for two reasons. First, since the assumption of the family operates solely in the source group, the assumption is not even "aware" of bilinear groups. Therefore it is not clear how it might be applied in certain systems (e.g. an variant of Boneh-Boyen IBE) where we are hiding a message in the target group. Second, the Linear assumption family has an inconsistent interaction with the DBDH assumption: the $1, 2$-Linear assumptions are actually stronger than DBDH, but the the k-Linear assumptions for $k > 2$ are generically incomparable to DBDH. One reason that the (2-)Linear assumption has proved so useful is that it gives DBDH "for free," but this is lost as soon as one increases k beyond 2. If a new IBE system were based on k-Linear *and* DBDH, it is not clear that this would provide an improvement in security.[1]

[1]Similarly, for attribute-based encryption, if attribute-hiding were established based on k-Linear, but payload-hiding

Our goals, then, are to find an assumption family that meets the following criteria:

- As we increase the assumption family parameter k, we should become more confident in the security of our assumption. In particular, we would argue that our k parameterized assumption is in some sense more secure than both existing decisional assumptions in bilinear groups and more secure than the $k-1$ instance.
- Our family of assumptions should be amenable to building cryptographic systems. Ideally, for any system built using the DBDH assumption, one could find a variant built using our family.

The k-BDH Family of Assumptions. Our main contribution is a new family of assumptions that can serve as a weaker generalization of DBDH.

We propose a family of progressively weaker assumptions, the k-BDH assumptions, that generalizes the DBDH assumption. The 1-BDH assumption is equivalent to DBDH. More generally, the k-BDH assumption is as follows:

$$\text{given } g,\ g^x, g^y,\ v_1, \ldots, v_k,\ v_1^{r_1}, \ldots, v_k^{r_k} \text{ in } \mathbb{G},$$
$$\text{decide whether } T = e(g,g)^{(xy)(r_1+\cdots+r_k)} \text{ or random in } \mathbb{G}_T.$$

Here g and $\{v_i\}$ are random generators of $\mathbb{G}$ and x, y, and $\{r_i\}$ are random elements of its support $\mathbb{Z}_p$. We consider only the decisional versions of these problems; as with k-Linear, the computational versions are all equivalent to each other. (This is also why we refer to our assumption family as k-BDH and not k-DBDH; there is no interesting family of computational assumptions from which our decisional assumptions must be distinguished.)

We remark that discovering and choosing such a family turned out to be challenging. Initially, we considered the assumption family in which the adversary, given the same input values in $\mathbb{G}$, must distinguish $\prod_i e\big(g, v_i\big)^{xyr_i}$ from random in $\mathbb{G}_T$. This assumption family is easier to use than our k-BDH because the values v_i and $v_i^{r_i}$ are available to pair with g^x or g^y, the way that in DBDH we can use the pairing to compute any of $e(g,g)^{xy}$, $e(g,g)^{xz}$, $e(g,g)^{yz}$. However, it turns out that every member of this alternative assumption family is equivalent to DBDH.[2] The fact that the values $\{g^{r_i}\}$ are not supplied in the k-BDH challenge make constructing an IBE from k-BDH more challenging.

We justify our choice by arguing both that the k-BDH assumptions are no stronger than existing (decisional) assumptions in bilinear groups and that it is plausible that they are strictly weaker. The former follows in a relatively straightforward by finding appropriate reductions. We can show that in a given group the k-BDH assumption is no stronger than DBDH and for a given k the k-BDH assumption is no stronger than the k-Linear assumption, for all values of k.

Arguing that the assumptions are weaker is more nuanced. Whether certain assumptions hold or do not hold might vary with the choice of a group and clearly if $\mathcal{P} = \mathcal{NP}$ all assumptions are equally false. We give evidence that, for each k, the $(k+1)$-BDH assumption is strictly weaker than the k-BDH assumption (i.e., the $(k+1)$-BDH problem is strictly harder to solve than the k-BDH problem). As in previous proofs of this sort for Linear [BBS04] and k-Linear [HK07], we rely on an argument in the generic group model [Nec94, Sho97]. We show that the $k+1$-BDH problem is generically hard even in the presence of an oracle that solves k-BDH.

Using the k-BDH Assumption. We demonstrate the utility of our assumption family, by constructing a family of IBEs secure under k-BDH. The size of the public parameters, secret keys, and

were established based on DBDH, then one the assumption for one property would be weakened while the assumption for the other property would remain strong.

[2]The reduction makes use of the DBDH tuple $(g, \prod_i v_i^{r_i}, g^x, g^y, C \stackrel{?}{=} \prod_i e\big(g, v_i\big)^{xyr_i})$.

ciphertexts are all linear in the parameter k. One can view our family as a generalization of the Boneh-Boyen selectively secure IBE system [BB04a].

Finding a system that is provably secure under the k-BDH assumption is challenging. Our system requires a novel technique that effectively switches the base of several public parameters.

The main technical difficulty arises because of the inconvenient target value in the k-BDH assumption, $(e(g,g)^{xy\sum_{i=1}^{k} r_i})$. This might appear to be a natural embedding of k BDH problems: Given (g, g^x, g^y, g^{r_i}) embed $e(g,g)^{xyr_i}$. However, we do not have the value g^{r_i} for each i (as one might hope to have in building a straightforward analogy to the Boneh-Boyen IBE). Instead, we have the pair $(v_i, v_i^{r_i})$, where v_i is a generator not used elsewhere. To overcome this we need to use a a new cancellation trick to effectively switch the base of the $v_i^{r_i}$ elements.

In Appendix D, an extend our construction family to a hierarchical IBEs. These yield CCA-secure schemes via standard transformations [BCHK07, BMW05]. In addition, in Appendix E, we show how to produce a Waters-IBE–style variant [Wat05] that is fully secure in the standard model.

Looking Ahead. In the future, we expect that one will be able to build cryptosystems from our k-BDH assumption where DBDH was previously used. However, as our experience with IBE has taught us, this might require new insights or techniques.

One interesting challenge is whether one can build more complex systems using the k-BDH assumption where the performance overhead is *additive* in k versus a multiplicative factor (which seems more natural). For instance, in existing (Key-Policy) Attribute-Based Encryption [GPSW06, SW05] systems, the size of a private key is proportional to a policy expressed as a boolean formula. If, the cost of using the k-BDH assumption only required adding $\approx k$ more group elements, this could be a relatively small key size overhead for reasonably chosen k. This is in contrast to blowing up the entire key size by a factor of k. A similar argument holds for other parameters such as ciphertext size and decryption time. In one datapoint suggesting that this might be possible, Freeman et. al. [FGK+10] recently built Lossy Trapdoor Functions in a novel way from the k-linear assumption which were rather efficient relative to the "natural" extension of the Peikert and Waters [PW08] DDH construction.

There also exist currently exist several functionalities where there are no known systems that reduce to DBDH. These include systems that appear to inherently on assumption related to source group elements such as Decision Linear. Examples of these include Groth-Sahai NIZKs [GS08], dual system encryption proofs [Wat09], and the Boneh-Goh-Nissim [BGN05] slightly 2-homomorphic encryption system.

Finally, an interesting question is which k values one might use in practice. For very large k, it might turn out that bit by bit encryption systems built from using hard core bits [GL89] and Computational Diffie-Hellman or Computational Bilinear Diffie-Hellman have comparable efficiency. When proposing systems, it is important to keep in mind where these lines cross. However, we believe for most practical choices of k the k-BDH assumption will yield more efficient systems.

2 The k-BDH Assumption and Relationships

Throughout this paper we work in a cyclic group $\mathbb{G}$ of order p where p is a large prime. g is a generator of $\mathbb{G}$. $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ denotes an admissible bilinear map where $\mathbb{G}_T$ is another cyclic group of order p. Bilinear Maps and well known complexity assumptions BDH, DBDH, Linear and k-Linear are formally defined in Appendix A.

Definition 1. *The k-BDH problem in $\langle \mathbb{G}, \mathbb{G}_T, e\rangle$ asks given $(g,\ g^x,\ g^y,\ v_1,\ \ldots,\ v_k,\ v_1^{r_1},\ \ldots,\ v_k^{r_k},\ T)$ for $x, y, r_1, \ldots, r_k, c \in \mathbb{Z}_p^*$, $g, v_1, \ldots, v_k \in \mathbb{G}$ and $T \in \mathbb{G}_T$ does $T = e(g,g)^{xy(r_1+\cdots+r_k)}$ or is it the case that $T = e(g,g)^c$. An adversary, $\mathcal{B}$ outputs 1 if $T = e(g,g)^{xy(r_1+\cdots+r_k)}$ and 0 otherwise. $\mathcal{B}$ has advantage ϵ*

in solving k-BDH if

$$|Pr[\mathcal{B}(g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, e(g,g)^{xy(r_1+\cdots+r_k)}) = 1] - Pr[\mathcal{B}((g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, e(g,g)^c) = 1]| \geq 2\epsilon.$$

Where the probability is taken over the random choice of $x, y, r_1, \ldots, r_k, c \in \mathbb{Z}_p^*$, $g, v_1, \ldots, v_k \in \mathbb{G}$ and the random bits consumed by $\mathcal{B}$.

The k-BDH Assumption is that if no t-time algorithm can achieve advantage at least ϵ in deciding the k-BDH problem in $\mathbb{G}$ and $\mathbb{G}_T$.

This is only a decisional problem. We show that, as a corollary of Theorem 4, the computational version is equivalent to the computational BDH problem.

2.1 k-BDH's Relationship to Standard Assumptions

In this subsection we state k-BDH's relationship to standard cryptographic assumptions; the proofs are straightforward and given in Appendix B. The reductions show that the k-BDH assumption is both secure and novel. We refer the reader to Section 4 for a discussion of the relative strengths of assumptions in the k-BDH family.

We also note that k-BDH is a member of the (R,S,T,f)-Diffie Hellman uber-assumption family [Boy08]. Namely: $R = S = \{1, x, y, a_1, \ldots, a_k, a_1 r_1, \ldots, a_k r_k\}$, $T = \{1\}$ and $f = xy(r_1 + \cdots + r_k)$ where $v_i = g^{a_i}$ for $1 \leq i \leq k$. Being part of this family tells us that it is generically secure, however, the focus on our work is to understand the relative strengths of assumptions.

2.1.1 k-BDH's Relationship to k-Linear

We will use the notation L_k to denote the k-Linear problem. If we wish to specify the k-Linear assumption in a specific group $\mathbb{G}$ we write $L_k^{\mathbb{G}}$, and similarly for $\mathbb{G}_T$.

Theorem 1. *If the $L_k^{\mathbb{G}}$ assumption holds, then so does the k-BDH assumption.*

Theorem 2. *If the k-BDH assumption holds, then so does the $L_k^{\mathbb{G}_T}$ assumption.*

Evidence that k-BDH is not equivalent to either $L_k^{\mathbb{G}}$ or $L_k^{\mathbb{G}_T}$. From the above theorems, the natural question arises: Is k-BDH equivalent to the linear assumption in either $\mathbb{G}$ or $\mathbb{G}_T$? Such an equivalence would imply that k-BDH assumption is neither a new assumption nor a new tool to construct a family of IBEs. Fortunately, separation of the assumptions appears to be related to inverting a bilinear map. There is strong evidence that inverting a bilinear map is hard [Ver01, Moo09]. We show separation results for these assumptions in Appendix B.3 in the generic group model.

2.1.2 k-BDH's Relationship to BDH

Theorem 3. *If the DBDH assumption holds, then so does the k-BDH assumption.*

Theorem 4. *If the Computational k-BDH assumption holds, then so does the Computational BDH assumption.*

Corollary 1. *The Computational k-BDH assumption is equivalent to the BDH assumption.*

Corollary 2. *The DBDH assumption is equivalent to the 1-BDH assumption.*

3 A Selectively Secure IBE system from the k-BDH Assumption

For completeness, the standard definitions of IBE and the selective-ID model are given in Appendix C.

Using the k-BDH assumption in to create an IBE system is not straightforward. The main technical difficulty arises because the target in the k-BDH assumption, $(e(g,g)^{xy\sum_{i=1}^{k} r_i})$, is naturally an embedding of k Computational BDH problems: Given (g, g^x, g^y, g^{r_i}) find $e(g,g)^{xyr_i}$. However, we do not have the value g^{r_i} for each i. Instead, we have the pair $(v_i, v_i^{r_i})$, where v_i is a generator not used elsewhere.

We use a cancellation trick to effectively switch the base of the $v_i^{r_i}$. The setup algorithm will provide the values $e(g^x, v_i^{r_i})$ and v_i which are both taken to the same power in the encryption algorithm, namely y_i. The challenge needs to be crafted so that it takes $e(g^x, g^{r_i})$ to the power y instead of taking $e(g^x, v_i^{r_i})$ to the power y_i. To do this, we provide g^y in place of $v_i^{y_i}$. Since $v_i = g^{s_i}$ for some value of s_i we implicitly set $y_i = y/s_i$. Using the bilinear property of e, this effectively changes the value of the other term to $e(g^x, v_i^{r_i})^{y_i} = e(g,g)^{xr_is_i\frac{y}{s_i}} = e(g^x, g^{r_i})^y$. The product of these values is exactly the target of the k-BDH assumption.

To ensure the challenge has the proper distribution in the view of the adversary it is required to randomize g^y for every value of k.

Our IBE construction is related to the Boneh-Boyen scheme in the selective-ID model [BB04a], which is proven secure under the DBDH assumption. To prove our scheme is secure under the k-BDH assumption requires an alteration to the "Boneh-Boyen trick" for generating the private key for identities other than the target identity.

The "Boneh-Boyen trick" raises elements of the DBDH instance to cleverly selected random values to obtain a valid private key. However, constructing the same private key with the $\mathsf{KeyGen(ID)}$ algorithm is impossible as the random selections are unknown. Our construction uses the same idea but using multiple bases $(g,\, v_i)$ requires *three* components instead of two for the first term of the private key.

Specifically, we use $(v_i^{\hat{r_i}})^{-t_i/d} v_i^{t_i m_i} (g^x)^{dm_i}$ for the first term. $v_i^{\hat{r_i}}$ is the randomization of $v_i^{r_i}$ that permits the challenge have the proper distribution. The value d is a function of the target identity and the identity associated with the private key; t_i is used to randomize a public parameter; and m_i randomizes the private key. The first term is dependent on both g^x and $v_i^{r_i}$ from the k-BDH assumption.

The IBE system works as follows:

Setup : The public parameters are $(g, u = g^x, v_1 = g^{s_1}, \ldots, v_k = g^{s_k}, v_1^{\hat{r_1}}, \ldots, v_k^{\hat{r_k}}, w_1, \ldots, w_k)$. The values $s_1, \ldots, s_k, \hat{r_1}, \ldots, \hat{r_k}, x$ (chosen uniformly and independently at random) are kept as the master-key.

$\mathsf{KeyGen(ID)}$: Select random $n_1, \ldots, n_k \in \mathbb{Z}_p^*$. For each $1 \le i \le k$ output $(K_{A,i}, K_{B,i})$=$((g^{x\hat{r_i}}(w_i u^{\mathsf{ID}})^{n_i}, v_i^{n_i})$.

$\mathsf{Encrypt}(m, \mathsf{ID})$: Select random $y_1, \ldots, y_k \in \mathbb{Z}_p^*$. Output $C_0 = m \prod_{1 \le i \le k} e(g^x, v_i^{\hat{r_i}})^{y_i}$ and for each $1 \le i \le k$ output $(C_{A,i}, C_{B,i})$=$(v_i^{y_i}, (w_i u^{\mathsf{ID}})^{y_i})$ for a total of $2k+1$ values.

$\mathsf{Decrypt}(c)$: Output

$$\frac{C_0 \cdot \prod_{1 \le i \le k} e(K_{B,i}, C_{B,i})}{\prod_{1 \le i \le k} e(K_{A,i}, C_{A,i})} = \frac{m \prod_{1 \le i \le k} e(g^x, v_i^{\hat{r_i}})^{y_i} \cdot \prod_{1 \le i \le k} e(v_i^{n_i}, (w_i u^{\mathsf{ID}})^{y_i})}{\prod_{1 \le i \le k} e(g^{x\hat{r_i}}(w_i u^{\mathsf{ID}})^{n_i}, v_i^{y_i})} = m$$

3.1 Proof of Security

Theorem 5. *Suppose the k-BDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$ (precisely, no t-time algorithm has advantage at least ϵ in solving the k-BDH problem in $\mathbb{G}$ and $\mathbb{G}_T$). Then the previously defined IBE system is $(t - \Theta(\tau k q), q, \epsilon)$-Selective-ID IND-CPA secure where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the IBE system. We build $\mathcal{B}$ to solve a decisional k-BDH instance $(g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, T \stackrel{?}{=} e(g,g)^{xy(r_1+\cdots+r_k)})$. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective identity game as follows:
Init: The selective identity games begins with $\mathcal{A}$ outputting an identity to attacked ID^*.

Setup: Algorithm $\mathcal{B}$ first selects random a_i, t_i for $1 \leq i \leq k$. It then sets the public parameters to: $(g, u = g^x, v_1, \ldots, v_k, v_1^{\hat{r_1}} = (v_1^{r_1})^{1/a_1}, \ldots, v_k^{\hat{r_k}} = (v_k^{r_k})^{1/a_k}, w_1 = v_1^{t_1}(g^x)^{-\mathsf{ID}^*}, \ldots, w_k = v_k^{t_k}(g^x)^{-\mathsf{ID}^*})$. These parameters are are all independent of ID^* in the view of $\mathcal{A}$. The a_i terms will serve as the way to randomize the challenge.

Phase 1: $\mathcal{A}$ issues queries for the private key of an identity, ID. It must be the case that $\mathsf{ID} \neq \mathsf{ID}^*$. $\mathcal{B}$'s response is generated as follows for each value of $1 \leq i \leq k$:

Select random m_i. Let $d = \mathsf{ID} - \mathsf{ID}^*$. Output $(K_{A,i}, K_{B,i})$=$((v_i^{\hat{r_i}})^{-t_i/d} v_i^{t_i m_i} (g^x)^{d m_i}, (v_i^{\hat{r_i}})^{(-1/d)} v_i^{m_i})$. For $n_i = -\hat{r_i}/d + m_i$, which implies $m_i = \hat{r_i}/d + n_i$, this is the expected value:

$$\begin{aligned}
&((v_i^{\hat{r_i}})^{-t_i/d} v_i^{t_i m_i} (g^x)^{d m_i}, (v_i^{\hat{r_i}})^{(-1/d)} v_i^{m_i}) \\
&= ((v_i^{\hat{r_i}})^{-t_i/d} v_i^{t_i(\hat{r_i}/d + n_i)} (g^x)^{d(\hat{r_i}/d + n_i)}, (v_i^{\hat{r_i}})^{(-1/d)} v_i^{(\hat{r_i}/d) + n_i)}) \\
&= (v_i^{t_i n_i} (g^x)^{\hat{r_i} + d n_i}, v_i^{n_i}) \\
&= (g^{x\hat{r_i}} (v_i^{t_i} g^{xd})^{n_i}, v_i^{n_i}) \\
&= (g^{x\hat{r_i}} (v_i^{t_i} g^{x(\mathsf{ID} - \mathsf{ID}^*)})^{n_i}, v_i^{n_i}) \\
&= (g^{x\hat{r_i}} (w_i g^{x\mathsf{ID}})^{n_i}, v_i^{n_i})
\end{aligned}$$

The second term is uniformly distributed among all elements in $\mathbb{Z}_p^*$ due to the selection of m_i. Private keys can be generated for all identities except ID^*.

Challenge(m_0, m_1) : $\mathcal{B}$ picks random bit $b \in \{0, 1\}$. The response is: $(C_0, (C_{A,1}, C_{B,1}), \ldots, (C_{A,k}, C_{B,k}))$. $\mathcal{B}$ sets $C_0 = m_b T$ and for each i from 1 to k it sets:

$$C_{A,i} = (g^y)^{a_i}, \quad C_{B,i} = (g^y)^{a_i \cdot t_i}.$$

We observe that $(g^y)^{a_i} = v_i^{y_i}$ and that $(g^y)^{a_i t_i} = v_i^{t_i y_i} = (w_1 u^{\mathsf{ID}^*})^{y_i}$ from which correctness follows. The simulator's ability to construct the second term in this manner follows directly from the fact that the encrypted identity is ID^* and no g^x term appears in $w_1 u^{\mathsf{ID}^*}$.

For each value of i, this implicitly sets $y a_i = s_i y_i$ or $y_i = y a_i / s_i$. If the input is a valid k-BDH tuple then the response is drawn from a uniform distribution and $m_b T$ is the expected value:

$$m_b \prod_{1 \leq i \leq k} e(g,g)^{x y r_i} = m_b \prod_{1 \leq i \leq k} e(g^x, g^{s_i r_i / a_i})^{y a_i / s_i} = m_b \prod_{1 \leq i \leq k} e(g^x, v_i^{r_i / a_i})^{y_i} = m_b \prod_{1 \leq i \leq k} e(g^x, v_i^{\hat{r_i}})^{y_i}$$

If T is not a valid k-BDH tuple then the distribution is uniform and independent of b.

Phase 2: $\mathcal{A}$ issues more private key queries. It is exactly the same as Phase 1.

Guess: $\mathcal{A}$ outputs a guess of $b' \in \{0,1\}$. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning T is a valid k-BDH tuple. Otherwise, it is not a valid k-BDH tuple and the output is 0.

When the input is a valid k-BDH instance, $\mathcal{A}$ must satisfy $|\Pr[b = b'] - \frac{1}{2}| \geq \epsilon$. When the input is not a valid k-BDH instance, the input is uniform and independent and $\Pr[b = b'] = \frac{1}{2}$. Therefore, we have

$$|\Pr[\mathcal{B}(\text{valid } k\text{-BDH}) = 1] - \Pr[\mathcal{B}(\text{not valid } k\text{-BDH}) = 1]| \geq |(\frac{1}{2} + \epsilon) - \frac{1}{2}| \geq \epsilon$$

as required. □

3.2 Efficiency

Assume that the value $e(g^x, v_i^{r_i})$ is precomputed for all values $1 \leq i \leq k$. Each encryption takes k exponentiations and k group operations in $\mathbb{G}_T$, $2k+1$ exponentiations and k group operation in $\mathbb{G}$. Decryption requires $2k$ bilinear map computations, one inversion and $2k+2$ group operations in $\mathbb{G}_T$.

3.3 Extensions

This construction fits in Boneh-Boyen framework. We give the natural extension to a hierarchical IBE in Appendix D and to a fully secure IND-CPA scheme in the style of [Wat05] in Appendix E.

4 Relationship between k-BDH Problems

In this section we prove that the k-BDH family of problems becomes progressively weaker. Informally, this means that an oracle for k-BDH does not help in solving a $(k+1)$-BDH instance.

The proof uses the generic group model [BS84, Nec94, Sho97]. This idealized version of a group retains the important properties of the group while facilitating reasoning about its minimal possible assumptions. If a statement cannot be proven in the generic group model then it is impossible to find a group for which the statement holds. The generic group model has been used to reason about complexity assumptions both with bilinear maps [BB04b, Boy08] and without bilinear maps [Sho97].

The closely related proof for the separation of k-Linear family of assumptions [Sha07] could not be used directly. This stems from the fact that a standard multilinear map [BS03] cannot be used to solve k-BDH. We create a modified k-multilinear map that takes as input k elements in $\mathbb{G}$ and 1 element in $\mathbb{G}_T$ (which is the result of a bilinear map on two elements in $\mathbb{G}$) and produces an output in a third group $\mathbb{G}_M$ (the target group of the k-multilinear map). The modified k-multilinear map acts as an oracle for k-BDH. The main technical difficulty is showing that all inputs to the k-multilinear map fail to produce a multiple of the target element in the $(k+1)$-BDH instance.

Theorem 6. *If the k-BDH assumption holds, then so does the $(k+1)$-BDH assumption.*

Proof. Informally, this means that if $(k+1)$-BDH is easy, then k-BDH is also easy. Suppose we have an oracle $\mathcal{A}$ for $(k+1)$-BDH. $\mathcal{A}$ can be used to solve an k-BDH instance (g, g^x, g^y, v_1, …, v_k, $v_1^{r_1}$, …, $v_k^{r_k}$, T). Select random $v_{k+1} \in \mathbb{G}$ and $r_{k+1} \in \mathbb{Z}_p^*$ and run $\mathcal{A}$ on input (g, g^x, g^y, v_1, …, v_k, v_{k+1}, $v_1^{r_1}$, …, $v_k^{r_k}$, $v_{k+1}^{r_{k+1}}$, $T \cdot e(g^x, g^y)^{r_{k+1}}$). By returning the same value as $\mathcal{A}$, the simulation is perfect. □

As in the k-Linear generic group separation proof [Sha07], we prove a stronger result by means of a multilinear map [BS03, GGH12] in Theorem 7. A k multilinear map is an efficiently computable map $e_k : \mathbb{G}^k \to \mathbb{G}_M$ such that $e_k(g_1^{a_1}, \ldots, g_k^{a_k}) = e_k(g_1, \ldots, g_k)^{\prod_{i=1}^k a_i}$ for all $g_1, \ldots, g_k \in \mathbb{G}$ and $a_1, \ldots, a_k \in \mathbb{Z}_p$; and $e_k(g, \ldots, g) \neq 1$. Here, we consider a modified k-multilinear map: $\hat{e_k} : \mathbb{G}_T \times \mathbb{G}^k \to \mathbb{G}_M$ where $\mathbb{G}_T$ is the group resulting from a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We define $\hat{e_k} : (e(g_x, g_y)^{a_T}, g_1^{a_1}, \ldots, g_k^{a_k}) = \hat{e_k}(e(g_x, g_y), g_1, \ldots, g_k)^{a_T \prod_{i=1}^k a_i}$.

Lemma 1. *Given a modified k-multilinear map there is an efficient algorithm to solve k-BDH.*

Proof. On input a k-BDH instance $(g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, T)$ output "yes" if

$$\hat{e_k}(T, v_1, \ldots, v_k) \overset{?}{=} \prod_{i=1}^{k} \hat{e_k}(e(g^x, g^y), v_1, \ldots, v_{i-1}, v_i^{r_i}, v_{i+1}, v_k)$$

and no otherwise. This is correct because

$$\begin{aligned}\prod_{i=1}^{k} \hat{e_k}(e(g^x, g^y), v_1, \ldots, v_{i-1}, v_i^{r_i}, v_{i+1}, v_k) &= \prod_{i=1}^{k} \hat{e_k}(e(g, g), v_1, \ldots, v_k)^{xyr_i} \\ &= \hat{e_k}(e(g, g), v_1, \ldots, v_k)^{xy \sum_{i=1}^k r_i}\end{aligned}$$

and when $T = e(g, g)^{xy \sum_{i=1}^k r_i}$ equality holds as required. □

In the generic group model, elements of $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{G}_M$ are encoded as opaque strings such that only equality can be tested by the adversary. To perform operations in the group the adversary queries oracles. The oracles map the opaque string representations to elements of $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{G}_M$ using ξ_G,ξ_T and ξ_M respectively. In our case, we provide the adversary with oracles to perform Group Action in each group, Inversion in each group, Bilinear Map for $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and Modified k-Multilinear Map for $\mathbb{G}_T \times \mathbb{G}^k$.

Theorem 7. *Let $\mathcal{A}$ be an algorithm that solves $(k+1)$-BDH in the generic group model making a total of q queries to the oracles computing the group action in $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{G}_M$, the oracles computing inversion in $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{G}_M$, the bilinear map oracle and an oracle for modified k-multilinear map. Then $\mathcal{A}$'s probability of success is bounded by*

$$\epsilon \leq \frac{(k+5)(q+2k+5)^2}{p}.$$

Proof. Consider an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ as follows.

Let g be a randomly selected generator of $\mathbb{G}$. Select random $x, y, v_1, \ldots, v_{k+1}, r_1, \ldots, r_{k+1}, c \in \mathbb{Z}_p$ as well as random bit $d \in \{0, 1\}$. Set $T_d = e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ and $T_{1-d} = e(g, g)^c$. $\mathcal{A}$ is given $(\xi_G(g), \xi_G(g^x), \xi_G(g^y), \xi_G(g^{v_1}), \ldots, \xi_G(g^{v_{k+1}}), \xi_G(g^{v_1 r_1}), \ldots, \xi_G(g^{v_{k+1} r_{k+1}}), \xi_T(T_0), \xi_T(T_1))$ with the goal of guessing d.

$\mathcal{B}$ keeps track of the elements known to $\mathcal{A}$ as three lists: $L_G = \{(F_{G,i}, \xi_{G,i})\}$, $L_T = \{(F_{T,i}, \xi_{T,i})\}$ and $L_M = \{(F_{M,i}, \xi_{M,i})\}$. The first element of each list is the internal representation kept by $\mathcal{B}$- represented as a polynomial in the ring $\mathbb{Z}_p[1, x, y, v_1, \ldots, v_{k+1}, r_1, \ldots, r_{k+1}, c]$. The set of all elements in these rings are denoted F_G, F_T and F_M. The second element is the opaque representation known to $\mathcal{A}$. $\mathcal{B}$ handles oracle queries from $\mathcal{A}$ by calculating the correct value and checking to see if a the corresponding external representation already exists. If so, the corresponding known representation is returned; otherwise $\mathcal{B}$

generates a distinct random string to serve as the external representation and adds it to the respective list. We assume that the domains of ξ_G, ξ_T and ξ_M are sufficiently large so that the probability that algorithm $\mathcal{A}$ makes queries for an element other than one obtained through $\mathcal{B}$ is negligible.

Oracle queries from $\mathcal{A}$ are handled by $\mathcal{B}$ as follows:

Group Action: Given elements in $\mathbb{G}$ with internal representations $F_{G,i}$ and $F_{G,j}$ compute $F' = F_{G,i} + F_{G,j}$. If there does not already exist an external representation of the value F' then generate $\xi_G(F')$ and add $(F', \xi_G(F'))$ to L_G . Return $\xi_G(F')$. Group Action for $\mathbb{G}_T$ and $\mathbb{G}_M$ are handled analogously. Denote the number of Group Action queries made in $\mathbb{G}$ as q_{G_g}, the number of Group Action queries made in $\mathbb{G}_T$ as q_{T_g} and the number of Group Action queries made in $\mathbb{G}_M$ as q_{M_g}.

Inversion: Given an element in $\mathbb{G}$ with internal representation $F_{G,i}$ set $F' = -F_{G,i}$. If there does not already exist an external representation of the value F' generate $\xi_G(F')$ and add $(F', \xi_G(F'))$ to L_G . Return $\xi_G(F')$. Inversion for $\mathbb{G}_T$ and $\mathbb{G}_M$ are handled analogously. Denote the number of Group Action queries made in $\mathbb{G}$ as q_{G_i}, the number of Group Action queries made in $\mathbb{G}_T$ as q_{T_i} and the number of Group Action queries made in $\mathbb{G}_M$ as q_{M_i}.

Bilinear Map (e): Given elements in $\mathbb{G}$ with internal representations $F_{G,i}$ and $F_{G,j}$ calculate $F' = F_{G,i} \cdot F_{G,j}$. If there does not already exist an external representation of the value F' generate $\xi_T(F')$ and add $(F', \xi_T(F'))$ to L_T. Return $\xi_T(F')$. Let q_B denote the number of bilinear map queries made.

Modified k-Multilinear Map ($\hat{e_k}$): Given elements in $\mathbb{G}$ with internal representations $F_{G,v1}$, …, $F_{G,vk}$, and an element in $\mathbb{G}_T$ with internal representation $F_{T,j}$. Compute $F' = F_{T,j} \prod_{i=1}^{k} F_{G,vi}$. If there does not already exist an external representation of the value F' generate $\xi_M(F')$ and add $(F', \xi_M(F'))$ to L_M. Return $\xi_M(F')$.

Elements in F_G have at most degree 2; elements of F_T have at most degree 4; elements in F_M have degree at most $2k+4$. The input elements that are in $\mathbb{G}$ have corresponding elements in F_G with degree at most 2 and the elements in $\mathbb{G}_T$ have corresponding elements in F_T with degree at most 3. The group action and inversion operations cannot increase the degree of the polynomials in F_G,F_T or F_M. The Bilinear Map operation uses elements in $\mathbb{G}$ to produce elements of at most degree $2+2=4$ in F_T. The Modified Multilinear Map produces elements of at most degree $4+k(2)$ in F_M.

Finally, $\mathcal{A}$ halts and outputs a guess of d' for d. $\mathcal{B}$ now selects random $g^* \in \mathbb{G}$ and x^*, y^*, v_1^*, …, v_{k+1}^*, r_1^*, …, r_{k+1}^*, $c^* \in \mathbb{Z}_p$. T_b is set to $e(g^*, g^*)^{x^* y^* \sum_{i=1}^{k+1} r_i^*}$ and $T_{1-b} = e(g^*, g^*)^{c^*}$. All elements besides T_b are independent of each other. Therefore the simulation engineered by $\mathcal{B}$ is consistent with these values unless one of the following events occur:

- Two values in F_G have the same representation in $\mathbb{G}$
- Two values in F_T have the same representation in $\mathbb{G}_T$
- Two values in F_M have the same representation in $\mathbb{G}_M$
- Using a bilinear map on values in F_G, it is possible to find a multiple of $e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ in F_T.
- Using a modified k-multilinear map on values in F_G, it is possible to find a multiple of $e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ in F_M.

The input elements are all chosen independently. Since $\mathcal{A}$ makes $q_{G_g} + q_{G_i}$ group actions or inversion queries for group $\mathbb{G}$ the corresponding elements in F_G are at most degree 2 and the probability of a collision is $\binom{q_{G_g}+q_{G_i}+2(k+1)+3}{2}\frac{2}{p}$. For the elements in $\mathbb{G}_T$ there $q_{T_g} + q_{T_i} + q_B$ group actions or inversion or bilinear map queries are made resulting in elements in $\mathbb{G}_T$. Since elements in $\mathbb{G}_T$ have corresponding polynomials in F_T with degree at most 4 the probability of a collision is $\binom{q_{T_g}+q_{T_i}+q_B+2}{2}\frac{4}{p}$. For each of the group actions in $\mathbb{G}_M$, inversion in $\mathbb{G}_M$ and k-Modified Multilinear Map queries the probability of a collision is $\binom{q_{M_g}+q_{M_i}+q_K}{2}\frac{2k+4}{p}$.

Next, we show the probability of finding a multiple of $e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ from the terms in F_G is zero. If a multiple exists, it must be formed using at least one bilinear map operation. Since x, y, r_i all appear in T_b then the product of at least two of these values must appear in the same element in F_G for each value of i, $1 \leq i \leq k+1$. This is impossible by the following claim:
Claim: It is impossible for any two of x, y, r_i to appear in the same monomial in F_G:.

Proof. We show that each way to choose two of the three values to appear in the same term is impossible:

- x and y appear in the same term. This requires creating a multiple of the polynomial xy. We are initially given the polynomials x and y each of degree 1 (and polynomials that are independent of x and y). This means from polynomial of degree 1 we must create a polynomial of degree 2 also in F_G. Only the group action and inversion oracles result in new elements in F_G. However, the output of these oracles cannot increase the degree of a monomial. Thus we cannot create monomials of degree greater than 1 from x and y. In particular, xy cannot be created by the adversary.

- x and r_i appear in the same term. This requires creating a multiple of xr_i namely axr_i. Since the term r_i never appears without v_i it follows that $v_i \mid a$ and we can rewrite axr_i as $a'xr_iv_i$. This is a polynomial of degree 3. It is impossible to create a polynomial of degree greater than 2 in F_G. So x and r_i cannot appear in the same term.

- y and r_i appear in the same term. This follows from a symmetric argument that x and r_i cannot appear in the same term.

□

Finally, we claim that it is impossible to find a multiple of $e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ in F_M. In order to use the modified k-multilinear map to find a multiple of a target value, $T_d \overset{?}{=} e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$, at least one $\hat{e_k}$ operation involving a multiple of T_d is required. The only option is to use T_d as the input element in $\mathbb{G}_T$. The modified k-multilinear map produces a multiple of $xy\sum_{i=1}^{k+1} r_i$, namely $Axy\sum_{i=1}^{k+1} r_i$. $\mathcal{A}$ must then use combination of oracle calls using only values in F_G to form $Axy\sum_{i=1}^{k+1} r_i$ so that it can test equality. We call the combination of oracle calls $\mathcal{F}$.[3]
All inputs in F_G containing r_i also contain v_i. As a result, any monomial divisible by r_i is also divisible by v_i. Every type of oracle call preserves this property. In particular, consider the polynomial $Axy\sum_{i=1}^{k+1} r_i$ constructed by the adversary in $\mathcal{F}$. It is required that each monomial in the expansion of $Axyr_i$ must be divisible by v_i. It follows that for each of the $k+1$ values of v_i it is the case that $v_i|A$.[4] Specifically, A is divisible by $\prod_{i=1}^{k+1} v_i$.
For a given value i, the value $Axyr_i$ is divisible by $k+4$ values: x, y, v_1, $\ldots$, v_{k+1}, and r_i. Producing such a term requires taking the product of at least $k+3$ terms available to the adversary (x and y only

[3] $\mathcal{A}$ could first perform $\hat{e_k}(CT_d + D, n_1, \ldots, n_k)$ for constant C and a polynomial D that does not contain T_d. It would then perform some combination of oracle calls,$\mathcal{F}$, to produce a value equal to $\hat{e_k}(CT_d + D, n_1, \ldots, n_k)$. However, an equivalent test is to first perform $\hat{e_k}(CT_d, n_1, \ldots, n_k)$ and then test equality with $\mathcal{F}/\hat{e_k}(D, n_1, \ldots, n_k)$.

[4] For a more detailed argument see [Sha07]

appear on their own and it is impossible to produce a multiple of $v_i v_j$ in F_G using the group action and inversion oracles). However, the bilinear map can only take the product of 2 values and the modified k-multilinear map can only take the product from a bilinear map and k additional values for a total of $k+2$. Consequently, we deduce that the adversary cannot synthesize a multiple of $xy\sum_{i=1}^{k+1} r_i$ in F_M to cause a collision.

The probability of finding a collision is bounded by

$$\epsilon \leq \binom{q_{G_g} + q_{G_i} + 2(k+1) + 3}{2}\frac{2}{p} + \binom{q_{T_g} + q_{T_i} + q_B + 2}{2}\frac{4}{p} + \binom{q_{T_m} + q_{T_i} + q_k}{2}\frac{(2k+4)}{p}$$
$$< \frac{(q+2k+5)^2 + 2(q+2)^2 + (k+2)q^2}{p} < \frac{(k+5)(q+2k+5)^2}{p}$$

□

The combination of these two theorems implies: DBDH=1-BDH $\lneq$ 2-BDH $\lneq \ldots \lneq$ k-BDH $\lneq$ $(k+1)$-BDH $\lneq \ldots$.

4.1 Relationship Between k and the Group Size

From Theorem 7, we know that increasing k increases security. The generic attack on k-BDH appears to require $\mathrm{O}(k)$ discrete logarithm calculations, and that solving t discrete logarithm problems on a given curve appears to require $\mathrm{O}(t)$ times solving one problem; assuming that, the generic attack scales linearly with k.
Another means of increasing security is to increase the group size. An interesting question is, "what is the equivalent increase in group size if we increase k to $k+1$." We assume finding the discrete log is a function, f, of the order of the group. Then in the generic attack on k-BDH where $\mathbb{G}$ has prime order p increasing k to $k+1$ is approximately equivalent to increasing the group size from p to $f^{-1}(\frac{(k+1)f(p)}{k})$.

5 Conclusions and Future Work

We have proposed k-BDH, a family of assumptions generalizing the DBDH assumption. We have given evidence, using the generic group model, that assumptions in the k-BDH family become strictly weaker with increasing values of the parameter k. Unlike the k-Linear family of assumptions, k-BDH makes a natural tool for constructing pairing-based cryptosystems, including IBEs. We have demonstrated this by constructing a family of IBEs in which the kth member is secure based on k-BDH. Our IBE family fits in the Boneh-Boyen framework; we give a hierarchical IBE and a full secure variant (similar to the Waters IBE [Wat05]). Our k-BDH family allows IBEs to be instantiated with an assumption safety buffer for the first time.

We hope that, like k-Linear, our k-BDH assumption family will see widespread use. We believe that it will be especially well suited for constructing attribute-based encryption and other forms of functional encryption. In addition, we believe that dual system encryption techniques could be applied to k-BDH, yielding more efficient cryptosystems with tighter security reductions.

An important open problem arises from the fact that the k-BDH assumptions are all no weaker than computational BDH (just as the k-Linear assumptions are all no weaker than CDH). Because the components of our IBE grow with k, there may be a crossover point beyond which an IBE based on hard-core bits of the computational BDH problem is more efficient than one based on k-BDH.

References

[AP05] Michel Abdalla and David Pointcheval. Interactive diffie-hellman assumptions with applications to password-based authentication. In Andrew S. Patrick and Moti Yung, editors, *Proceedings of Financial Cryptography 2005*, volume 3570 of *Lecture Notes in Computer Science*, pages 341–356. Springer, March 2005.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, May 2004.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, August 2004.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[BF03] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003. Extended abstract in *Proceedings of Crypto 2001.*

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Proceedings of TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, February 2005.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *Proceedings of Crypto 2008*, volume 5157 of *LNCS*, pages 108–25. Springer-Verlag, August 2008.

[BMR10] Dan Boneh, Hart Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efciency from the augmented cascade. In Angelos Keromytis and Vitaly Shmatikov, editors, *Proceedings of CCS 2010*, pages 131–40. ACM Press, October 2010.

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *Proceedings of CCS 2005*, pages 320–329. ACM Press, November 2005.

[Boy08] Xavier Boyen. The uber-assumption family – a unified complexity framework for bilinear groups. In *2nd International Conference on Pairing-based Cryptography—PAIRING 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Berlin: Springer-Verlag, 2008.

[BR09] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme. In Antoine Joux, editor, *Proceedings of Eurocrypt 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 407–424. Springer, April 2009.

[BS84] László Babai and Endre Szemerédi. On the complexity of matrix group problems I. In *Proceedings of FOCS 1984*, pages 229–40. IEEE Computer Society, October 1984.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. In Caroline Grant Melles, Jean-Paul Brasselet, Gary Kennedy, Kristin Lauter, and Lee McEwan, editors, *Topics in Algebraic and Noncommutative Geometry: Proceedings in Memory of Ruth Michler*, volume 324 of *Contemporary Mathematics*, pages 71–90. American Mathematical Society, 2003.

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology*, EUROCRYPT'11, pages 89–108, Berlin, Heidelberg, 2011. Springer-Verlag.

[CCS09] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In Antoine Joux, editor, *Proceedings of Eurocrypt 2009*, volume 5479, pages 351–68. Springer-Verlag, April 2009.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, May 2003.

[DHLAW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In Luca Trevisan, editor, *Proceedings of FOCS 2010*, pages 511–20. IEEE Computer Society, October 2010.

[FGK$^+$10] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In Phong Nguyen and David Pointcheval, editors, *Proceedings of PKC 2010*, volume 6056 of *LNCS*, pages 279–95. Springer-Verlag, May 2010.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Proceedings of Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, May 2006.

[GGH12] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. Cryptology ePrint Archive, Report 2012/610, 2012. `http://eprint.iacr.org/`.

[GH09] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In Omer Reingold, editor, *Proceedings of TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 437–456. Springer, March 2009.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of STOC 1989*, pages 25–32. ACM, May 1989.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of ACM Conference on Computer and Communications Security 2006*, pages 89–98. ACM, November 2006.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Proceedings of Eurocrypt 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *Proceedings of Crypto 2007*, volume 4622 of *LNCS*, pages 553–71. Springer-Verlag, August 2007.

[HW10] Susan Hohenberger and Brent Waters. Constructing veriable random functions with large input spaces. Cryptology ePrint Archive, Report 2010/102, 2010. `http://eprint.iacr.org/`.

[LW09] Allison B. Lewko and Brent Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 112–120. ACM, November 2009.

[Moo09] Dustin Moody. The diffiehellman problem and generalization of verheuls theorem. *Designs, Codes and Cryptography*, 52:381–390, 2009.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–72, February 1994.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *Proceedings of Crypto 2009*, volume 5677 of *LNCS*, pages 18–35. Springer-Verlag, August 2009.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Cynthia Dwork, editor, *Proceedings of STOC 2008*, pages 187–196. ACM, May 2008.

[Sha07] Hovav Shacham. A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/`.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of Eurocrypt 1997*, pages 256–266, 1997.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, May 2005.

[Ver01] Eric R. Verheul. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. In Birgit Pfitzmann, editor, *Proceedings of Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer, May 2001.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Proceedings of Crypto 2009*, volume 5677 of *LNCS*, pages 619–36. Springer-Verlag, August 2009.

A Complexity Assumptions

The following complexity assumptions are defined for a cyclic group $\mathbb{G}$ of order p where p is a large prime. g is a generator of $\mathbb{G}$. Additionally, $\mathbb{G}_T$ is a cyclic group of order p.

We say there is an admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ between the two groups if the following conditions hold:

- *Bilinear*: $e(g^a, g^b) = e(g, g)^{ab}$ holds for all $a, b \in \mathbb{Z}_p$.
- *Non-degenerate*: If g generate G, then $e(g, g) \neq 1$.
- *Computable*: e is efficiently computable on all input.

A.1 Bilinear Diffie-Hellman Assumption

The computational Bilinear Diffie-Hellman (BDH) problem in $< \mathbb{G}, \mathbb{G}_T, e >$ asks given (g, g^a, g^b, g^c) for some $a, b, c \in \mathbb{Z}_p^*$ compute $e(g, g)^{abc}$. An adversary, $\mathcal{A}$, has advantage ϵ in solving BDH if

$$\Pr[\mathcal{A}(g, g^a, g^b, g^c) = e(g, g)^{abc}] \geq \epsilon.$$

Where the probability is taken over the random choice of $a, b, c \in \mathbb{Z}_p^*$, $g \in \mathbb{G}$ and the random bits consumed by $\mathcal{A}$.
The *Bilinear Diffie-Hellman Assumption* is that no polynomial time algorithm can achieve non-negligible advantage in computing the BDH problem.

The decisional version of the problem (DBDH) in $< \mathbb{G}, \mathbb{G}_T, e >$ asks given (g, g^a, g^b, g^c, T) for some $a, b, c, z \in \mathbb{Z}_p^*$ does $T = e(g, g)^{abc}$ or is it the case that $T = e(g, g)^z$. An adversary, $\mathcal{B}$ outputs 1 if $T = e(g, g)^{abc}$ and 0 otherwise. $\mathcal{B}$ has advantage ϵ in solving DBDH if

$$|\Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g, g)^z) = 1]| \geq 2\epsilon.$$

Where the probability is taken over the random choice of $a, b, c, z \in \mathbb{Z}_p^*$, $g \in \mathbb{G}$ and the random bits consumed by $\mathcal{B}$.
The *Decisional Bilinear Diffie-Hellman Assumption* is that no polynomial time algorithm can achieve non-negligible advantage in deciding the BDH problem.

A.2 Linear Assumption

The computational Linear problem in $\mathbb{G}$ asks given $(g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2})$ for some $r_1, r_2 \in \mathbb{Z}_p^*$ compute $g_0^{r_1+r_2}$. An adversary, $\mathcal{A}$, has advantage ϵ in solving the Linear problem if

$$\Pr[\mathcal{A}(g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2}) = g_0^{r_1+r_2}] \geq \epsilon.$$

Where the probability is taken over the random choice of $r_1, r_2 \in \mathbb{Z}_p^*$, $g_0, g_1, g_2 \in \mathbb{G}$ and the random bits consumed by $\mathcal{A}$.
The *Linear Assumption* is that no polynomial time algorithm can achieve non-negligible advantage in computing the Linear problem.

The decisional version of the Linear problem in $\mathbb{G}$ asks given $(g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2}, T)$ for some $r_1, r_2, z \in \mathbb{Z}_p^*$ if $T = g_0^{r_1+r_2}$ or is it the case that $T = g_0^z$ for some $z \in \mathbb{Z}_p^*$. An adversary, $\mathcal{B}$, outputs 1 if $T = g_0^{r_1+r_2}$ and 0 otherwise. $\mathcal{B}$ has advantage ϵ in solving the decisional Linear problem if

$$|\Pr[\mathcal{B}(g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2}, g_0^{r_1+r_2}) = 1] - \Pr[\mathcal{B}(g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2}, g_0^z) = 1]| \geq 2\epsilon$$

Where the probability is taken over the random choice of $r_1, r_2, z \in \mathbb{Z}_p^*$, $g_0, g_1, g_2 \in \mathbb{G}$ and the random bits consumed by $\mathcal{B}$.

The *Decisional Linear Assumption* is that no polynomial time algorithm can achieve non-negligible advantage in deciding the Linear problem.

The decisional k-Linear problem in $\mathbb{G}$ is a generalization of the Linear problem that asks given $(g_0, g_1, \ldots, g_k, g_1^{r_1}, \ldots, g_k^{r_k}, T)$ for $r_1, \ldots, r_k \in \mathbb{Z}_p^*$ and $g_0, g_1, \ldots, g_k \in \mathbb{G}$ if $T = g_0^{r_1+\cdots+r_k}$ or is it the case that $T = g_0^z$ for some $z \in \mathbb{Z}_p^*$. An adversary, $\mathcal{C}$, outputs 1 if $T = g_0^{r_1+\cdots+r_k}$ and 0 otherwise.
The *Decisional k-Linear Assumption* is that no polynomial time algorithm can achieve non-negligible advantage in deciding the k-Linear problem.

Shacham [Sha07] proved that the family of *Decisional k-Linear Assumptions* becomes weaker as k increases at least in the generic group model. However, in the computational version of the problem the complexity is the same for all members of the family. In particular Computational Diffie-Hellman$=L_k$ (for $k \geq 1$).

B Proofs of k-BDH's Relationship to Standard Assumptions

B.1 k-BDH's Relationship to k-Linear

We will use the notation L_k to denote the k-Linear problem. If we wish to specify the k-Linear assumption in a specific group $\mathbb{G}$ we write $L_k^{\mathbb{G}}$.

Theorem 1. *If the $L_k^{\mathbb{G}}$ assumption holds, then so does the k-BDH assumption.*

Proof. Suppose we have an adversary $\mathcal{A}$ that can decide k-BDH. We can construct $\mathcal{A}'$ that solves the k-Linear problem in $\mathbb{G}$. On input $(g_0,\, g_1,\, \ldots,\, g_k,\, g_1^{r_1},\, \ldots,\, g_k^{r_k},\, T \overset{?}{=} g_0^{r_1+\cdots+r_k})$, $\mathcal{A}'$ selects random $y \in \mathbb{Z}_p^*$ and runs $\mathcal{A}$ on input $(g,\, g_0,\, g^y,\, g_1,\, \ldots,\, g_k,\, g_1^{r_1},\, \ldots,\, g_k^{r_k},\, e(g,T)^y)$. By returning the same value as $\mathcal{A}$, the simulation is perfect. □

Theorem 2. *If the k-BDH assumption holds, then so does the $L_k^{\mathbb{G}_T}$ assumption.*

Proof. Suppose we have an adversary $\mathcal{A}$ that can decide Decisional k-Linear in $\mathbb{G}_T$. We can construct $\mathcal{A}'$ that solves the k-BDH problem. On input $(g,\, g^x,\, g^y,\, v_1,\, \ldots,\, v_k,\, v_1^{r_1},\, \ldots,\, v_k^{r_k},\, T \overset{?}{=} e(g,g)^{xy(r_1+\cdots+r_k)})$, $\mathcal{A}'$ runs $\mathcal{A}$ on input $(e(g^x, g^y),\, e(g, v_1),\, \ldots,\, e(g, v_k),\, e(g, v_1^{r_1}),\, \ldots,\, e(g, v_k^{r_k}),\, T)$. By returning the same value as $\mathcal{A}$, the simulation is perfect. □

B.2 k-BDH's Relationship to BDH

Theorem 3. *If the DBDH assumption holds, then so does the k-BDH assumption.*

Proof. Suppose we have an adversary $\mathcal{A}$ that can decide k-BDH. We can construct $\mathcal{A}'$ that solves the DBDH problem. On input $(g,\, g^a,\, g^b,\, g^c,\, T \overset{?}{=} e(g,g)^{abc})$, $\mathcal{A}'$ selects random $r_1, \ldots, r_k, x_2, \ldots, x_k \in \mathbb{Z}_p^*$ and runs $\mathcal{A}$ on input $(g,\, g^a,\, g^b,\, g^{r_1},\, \ldots,\, g^{r_k},\, g^{cr_1},\, g^{x_2 r_2},\, \ldots,\, g^{x_k r_k},\, T \cdot e(g^a, g^b)^{x_2+\cdots+x_k})$. By returning the same value as $\mathcal{A}$, the simulation is perfect. □

Theorem 4. *If the Computational k-BDH assumption holds, then so does the Computational BDH assumption.*

Proof. Suppose we have an adversary $\mathcal{A}$ that can decide Computational BDH. We can construct $\mathcal{A}'$ that solves the computational k-BDH problem. $\mathcal{A}'$ is given input $(g,\, g^x,\, g^y,\, v_1,\, \ldots,\, v_k,\, v_1^{r_1},\, \ldots,\, v_k^{r_k})$ with the goal of computing $e(g,g)^{xy\sum_{i=1}^{k} r_i}$. For each i satisfying $1 \leq i \leq k$ $\mathcal{A}$ is run on input $(v_i, v_i^{r_i}, g^x, g^y)$

to obtain output a_i. Since each $v_i = g^{s_i}$ for some s_i, if $\mathcal{A}$ is correctly computing BDH instances then the output is $a_i = e(g^{s_i}, g^{s_i})^{r_i(\frac{x}{s_i})(\frac{y}{s_i})} = e(g,g)^{xyr_i}$. The product of all values of a_i is $e(g,g)^{xy(r_1+\cdots+r_k)}$, as expected. □

Corollary 1. *The Computational k-BDH assumption is equivalent to the BDH assumption.*

Proof. We prove in Theorem 3 if the DBDH assumption holds, then so does k-BDH. The reduction also holds for the computational versions of the problems: there is no target in the input to either $\mathcal{A}$ or $\mathcal{A}'$ and the results are elements in $\mathbb{G}_T$ instead of a bit. $\mathcal{A}'$ generates the same random values and gives the same inputs (other than the target) to $\mathcal{A}$. $\mathcal{A}$'s result is divided by $e(g^a, g^b)^{x_2+\cdots+x_k}$ to obtain $e(g,g)^{abc}$. Combining the result of Theorem 4 we have Computational k-BDH = BDH. □

Corollary 2. *The DBDH assumption is equivalent to the 1-BDH assumption.*

Proof. If there is only one value of v_i and $v_i^{r_i}$ (namely v_1 and $v_1^{r_1}$) then 1-BDH is equivalent to DBDH. We showed in Theorem 3 that if the DBDH assumption holds, then so does k-BDH for all values of k. It remains to show that if the 1-BDH assumption holds then so does DBDH. We modify the arguments in Theorem 4 so that the k-BDH instance has an additional input T and the query to the adversary is $(v_1, v_1^{r_1}, g^x, g^y, T)$. □

B.3 Separation Results for k-BDH and k-Linear

We prove the k-BDH assumption is equivalent to neither $L_k^{\mathbb{G}}$ nor $L_k^{\mathbb{G}_T}$.

B.3.1 The k-BDH Assumption is *n*ot Equivalent to $L_k^{\mathbb{G}_T}$

We showed in Theorem 2 that k-BDH $\leq L_k^{\mathbb{G}_T}$. To prove that these assumptions are distinct we show, in a generic group, k-BDH $\neq L_k^{\mathbb{G}_T}$.

Theorem 8. *In a generic group, $L_k^{\mathbb{G}_T}$ is hard even if k-BDH is easy.*

Proof. Let $\mathcal{A}$ be an algorithm that solves $L_k^{\mathbb{G}_T}$ in the generic group model making a total of q queries to the oracles computing the group action in $\mathbb{G}$ and $\mathbb{G}_T$, the oracles computing inversion in $\mathbb{G}$ and $\mathbb{G}_T$, the bilinear map oracle and an oracle for k-BDH. Then $\mathcal{A}$'s probability of success is bounded by

$$\epsilon \leq \frac{2(q+2k+3)^2}{p}.$$

In the generic group model, elements of $\mathbb{G}$ and $\mathbb{G}_T$ are encoded as opaque strings such that only equality can be tested by the adversary. To perform operations in the group the adversary queries oracles. The oracles map the opaque string representations to elements of $\mathbb{G}$ and $\mathbb{G}_T$ using ξ_G and ξ_T respectively. In our case, we provide the adversary with oracles to perform Group Action in $\mathbb{G}$, Group Action in $\mathbb{G}_T$, Inversion in $\mathbb{G}$, Inversion in $\mathbb{G}_T$, Bilinear Map $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and k-BDH.

Consider an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ as follows.

Let g be a randomly selected generator of $\mathbb{G}$. Select random $z, v_1, \ldots, v_k, r_1, \ldots, r_k, c \in \mathbb{Z}_p$ as well as random bit $d \in 0, 1$. Set $T_d = e(g,g)^{z\sum_{i=1}^{k} r_i}$ and $T_{1-d} = e(g,g)^c$. $\mathcal{A}$ is given ($\xi_G(g)$, $\xi_T(e(g,g)^z)$, $\xi_T(e(g,g)^{v_1})$, …, $\xi_T(e(g,g)^{v_k})$, $\xi_T(e(g,g)^{v_1 r_1})$, …, $\xi_T(e(g,g)^{v_k r_k})$, $\xi_T(T_0)$, $\xi_T(T_1)$) with the goal of guessing d.

$\mathcal{B}$ keeps track of the elements known to $\mathcal{A}$ as two lists: $L_G = \{(F_{G,i}, \xi_{G,i})\}$ and $L_T = \{(F_{T,i}, \xi_{T,i})\}$. The first element of each list is the internal representation kept by $\mathcal{B}$- represented as a polynomial in the ring $\mathbb{Z}_p[1, z, v_1, \ldots, v_{k+1}, r_1, \ldots, r_k, c]$. The set of all elements in these rings are denoted F_G and F_T. The second element is the opaque representation known to $\mathcal{A}$. $\mathcal{B}$ handles oracle queries from $\mathcal{A}$ by calculating the correct value and checking to see if a the corresponding external representation already exists. If so, the corresponding known representation is returned; otherwise $\mathcal{B}$ generates a distinct random string to serve as the external representation and adds it to the respective list. We assume that the domains of ξ_G and ξ_T are sufficiently large so that the probability that algorithm $\mathcal{A}$ makes queries for an element other than one obtained through $\mathcal{B}$ is negligible.

Oracle queries from $\mathcal{A}$ are handled by $\mathcal{B}$ as follows:

Group Action: Given elements in $\mathbb{G}$ with internal representations $F_{G,i}$ and $F_{G,j}$ compute $F' = F_{G,i} + F_{G,j}$. Generate $\xi_G(F')$ and add $(F', \xi_G(F'))$ to L_G if there does not already exist an external representation of the value. Return $\xi_G(F')$. Group Action for $\mathbb{G}_T$ is handled analogously. Denote the number of Group Action queries made in $\mathbb{G}$ as q_{G_g} and the number of Group Action queries made in $\mathbb{G}_T$ as q_{T_g}.

Inversion: Given an element in $\mathbb{G}$ with internal representation $F_{G,i}$ set $F' = -F_{G,i}$. Generate $\xi_G(F')$ and add $(F', \xi_G(F'))$ to L_G if there does not already exist an external representation of the value. Return $\xi_G(F')$. Inversion for $\mathbb{G}_T$ is handled analogously. Denote the number of Group Action queries made in $\mathbb{G}$ as q_{G_i} and the number of Group Action queries made in $\mathbb{G}_T$ as q_{T_i}.

Bilinear Map: Given elements in $\mathbb{G}$ with internal representations $F_{G,i}$ and $F_{G,j}$ calculate $F' = F_{G,i} \cdot F_{G,j}$. Generate $\xi_T(F')$ and add $(F', \xi_T(F'))$ to L_T if there does not already exist an external representation of the value. Return $\xi_T(F')$. Let q_B denote the number of bilinear map queries made.

***k*-BDH Query:** Given elements in $\mathbb{G}$ with internal representations $F_{G,x}$, $F_{G,y}$,$F_{T,v1}$, …, $F_{T,vk}$, $F_{G,r1}$, …, $F_{G,rk}$ and an element in $\mathbb{G}_T$ with internal representation $F_{T,j}$. Set $X = \prod_{i=1}^{k} F_{G,vi}$. If $X = 0$ return 0. Check if the value of $F_{T,j}$ is correct by evaluating $XF_{T,j} \overset{?}{=} F_{G,x} \cdot F_{G,y} \sum_{1 \le i \le k} \frac{F_{G,ri} \cdot X}{F_{G,vi}}$. This is computable because $F_{G,vi} | X$ for all values of i. If the equality holds then return 1; otherwise return 0. Let q_K be the number of k-BDH queries made.

Elements in F_G have at most degree 0; elements of F_T have at most degree 2. There is one input element in $\mathbb{G}$, g, which is represented internally as the polynomial 1 which has degree 0. The input elements in $\mathbb{G}_T$ have corresponding polynomials with degree at most 2. The group action and inversion operations do not increase the degree of the polynomials in either F_G or F_T. The Bilinear Map operation takes as input elements in $\mathbb{G}$ to produce polynomials of at most degree $0 + 0 = 0$ in F_T.

The k-BDH Query produces internally elements of at most degree 2. The values in F_G are of degree 0, so any number of additions and multiplications also results in a degree 0 polynomial. Therefore the degree of left hand side of the equation comes solely from the $F_{T,j}$ element. Notice that this means the k-BDH query can only return 1 when the degrees of the left hand side and right hand side are the same. This implies the adversary is unable to gain any information about the target (or any term in F_T) from the k-BDH Query, since the target has degree greater than zero and the right hand side has degree 0.

Finally, $\mathcal{A}$ halts and outputs a guess of d' for d. $\mathcal{B}$ now selects random $g^* \in \mathbb{G}$ and z^*, v_1^*, …, v_k^*, r_1^*, …, r_k^*, $c^* \in \mathbb{Z}_p$. T_b is set to $e(g^*, g^*)^{z^* \sum_{i=1}^k r_i^*}$ and $T_{1-b} = e(g^*, g^*)^{c^*}$. All elements besides T_b are independent of each other. Therefore the simulation engineered by $\mathcal{B}$ is consistent with these values unless one of the following five things happen:

- Two values in $\mathbb{G}$ are not chosen independently
- Two values in $\mathbb{G}_T$ are not chosen independently
- The k-BDH Query returns 0 because the polynomial representation of the inputs do not satisfy the test conditions. However, for the actual integer inputs the test condition holds.
- It is possible to find a multiple of $e(g, g)^{z \sum_{i=1}^k r_i}$ from the other terms in the polynomial.
- It is possible to learn information about $e(g, g)^{z \sum_{i=1}^k r_i}$ from the k-BDH query.

There is only one value in $\mathbb{G}$ revealed to $\mathcal{A}$, so the probability of finding 2 dependent variables is 0. There $q_{T_g} + q_{T_i} + q_B$ group actions or inversion or bilinear map queries are made resulting in polynomials in F_T. Since the polynomials corresponding to elements in $\mathbb{G}_T$ have at most degree 2 the probability of a collision is $\binom{q_{T_g}+q_{T_i}+q_B+2k+3}{2}\frac{2}{p}$. For each of the k queries the probability of selecting two polynomials with the same value is $\frac{2}{p}$ for a total probability of $\binom{q_k}{2}\frac{2}{p}$.
It remains to show the probability of finding a multiple of $e(g, g)^{z \sum_{i=1}^k r_i}$ from the other terms in the polynomial. This has an internal representation of a polynomial with degree 2. It cannot be crafted T_b, otherwise it would be dependent on T_b. All other polynomials have at most degree 2 from the $\mathbb{G}_T$ representation. All operations involving the polynomial 1 (g's representation in F_G) result in a polynomial of the same degree in both F_G and F_T. Therefore it is impossible to create a polynomial of degree 2 from the other terms.

The probability of finding a collision is bounded by

$$\epsilon \leq \binom{q_{T_g} + q_{T_i} + q_B + 2k + 3}{2}\frac{2}{p} + \binom{q_k}{2}\frac{2}{p} < \frac{2(q + 2k + 3)^2 + 2q^2}{2p} < \frac{2(q + 2k + 3)^2}{p}$$

□

Combining this result and Theorem 2, we have, in a generic group, k-BDH $\lneq L_k^{\mathbb{G}_T}$.

B.3.2 The k-BDH assumption is *n*ot Equivalent to $L_k^{\mathbb{G}}$

We showed in Theorem 1 that $L_k^{\mathbb{G}} \leq k$-BDH. To prove that these assumptions are distinct we show: $L_k^{\mathbb{G}} \neq k$-BDH.

Theorem 9. *In a generic group, $L_k^{\mathbb{G}}$ is not equivalent to k-BDH.*

Proof. Let $\mathcal{A}$ be an algorithm that solves k-BDH in the generic group model making a total of q queries to the oracles computing the group action in $\mathbb{G}$ and $\mathbb{G}_T$, the oracles computing inversion in $\mathbb{G}$ and $\mathbb{G}_T$, the bilinear map oracle and an oracle for $L_k^{\mathbb{G}}$. Then $\mathcal{A}$'s probability of success is bounded by

$$\epsilon \leq \frac{(k + 3)(q + 2k + 3)^2}{p}.$$

Consider an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ as follows.

Let g be a randomly selected generator of $\mathbb{G}$. Select random $x, y, v_1, \ldots, v_k, r_1, \ldots, r_k, c \in \mathbb{Z}_p$ as well as random bit $d \in 0, 1$. Set $T_d = e(g^x, g^y)^{\sum_{i=1}^{k} r_i}$ and $T_{1-d} = e(g, g)^c$. $\mathcal{A}$ is given $(\xi_G(g), \xi_G(g^x), \xi_G(g^y), \xi_G(g^{v_1}), \ldots, \xi_G($
with the goal of guessing d.

$\mathcal{B}$ keeps track of the elements known to $\mathcal{A}$ as two lists: $L_G = \{(F_{G,i}, \xi_{G,i})\}$ and $L_T = \{(F_{T,i}, \xi_{T,i})\}$. The first element of each list is the internal representation kept by $\mathcal{B}$- represented as a polynomial in the ring $\mathbb{Z}_p[1, x, y, v_1, \ldots, v_k, r_1, \ldots, r_k, c]$. The set of all elements in these rings are denoted F_G and F_T. The second element is the opaque representation known to $\mathcal{A}$. $\mathcal{B}$ handles oracle queries from $\mathcal{A}$ by calculating the correct value and checking to see if a the corresponding external representation already exists. If so, the corresponding known representation is returned; otherwise $\mathcal{B}$ generates a distinct random string to serve as the external representation and adds it to the respective list. We assume that the domains of ξ_G and ξ_T are sufficiently large so that the probability that algorithm $\mathcal{A}$ makes queries for an element other than one obtained through $\mathcal{B}$ is negligible.

Oracle queries from $\mathcal{A}$ are handled by $\mathcal{B}$ as follows:

Group Action: Given elements in $\mathbb{G}$ with internal representations $F_{G,i}$ and $F_{G,j}$ compute $F' = F_{G,i} + F_{G,j}$. If there does not already exist an external representation of the value F' then generate $\xi_G(F')$ and add $(F', \xi_G(F'))$ to L_G . Return $\xi_G(F')$. Group Action for $\mathbb{G}_T$ is handled analogously. Denote the number of Group Action queries made in $\mathbb{G}$ as q_{G_g} and the number of Group Action queries made in $\mathbb{G}_T$ as q_{T_g}.

Inversion: Given an element in $\mathbb{G}$ with internal representation $F_{G,i}$ set $F' = -F_{G,i}$. If there does not already exist an external representation of the value F' generate $\xi_G(F')$ and add $(F', \xi_G(F'))$ to L_G . Return $\xi_G(F')$. Inversion for $\mathbb{G}_T$ is handled analogously. Denote the number of Group Action queries made in $\mathbb{G}$ as q_{G_i} and the number of Group Action queries made in $\mathbb{G}_T$ as q_{T_i}.

Bilinear Map: Given elements in $\mathbb{G}$ with internal representations $F_{G,i}$ and $F_{G,j}$ calculate $F' = F_{G,i} \cdot F_{G,j}$. If there does not already exist an external representation of the value F' generate $\xi_T(F')$ and add $(F', \xi_T(F'))$ to L_T. Return $\xi_T(F')$. Let q_B denote the number of bilinear map queries made.

$L_k^{\mathbb{G}}$ **Query:** Given elements in $\mathbb{G}$ with internal representations $F_{G,x}$, $F_{G,v1}$, …, $F_{G,vk}$, $F_{G,r1}$, …, $F_{G,rk}$ and $F_{G,Q}$. Set $X = \prod_{i=1}^{k} F_{G,vi}$. If $X = 0$ return 0. Check if the value of $F_{G,Q}$ is correct by evaluating $XF_{G,Q} \stackrel{?}{=} F_{G,x} \cdot \sum_{1 \le i \le k} \frac{F_{G,ri} \cdot X}{F_{G,vi}}$. This is computable because $F_{G,vi} | X$ for all values of i. If the equality holds then return 1; otherwise return 0. Let q_K be the number of $L_k^{\mathbb{G}}$ queries made.

Elements in F_G have at most degree 2; elements of F_T have at most degree 4. The input elements that are in $\mathbb{G}$ have corresponding elements in F_G with degree at most 2 and the elements in $\mathbb{G}_T$ have corresponding elements in F_T with degree at most 3. The group action and inversion operations do not increase the degree of the polynomials in either F_G or F_T. The Bilinear Map operation uses elements in $\mathbb{G}$ to produce elements of at most degree $2 + 2 = 4$ in F_T; this is the only oracle query that produces an external representation of an element with a larger degree than the input.

The $L_k^{\mathbb{G}}$ oracle outputs a boolean value. However, the internal representation can be a large as $2k$. Each side of the test equation will have degree at most $2k$. The terms $F_{G,x}$, $F_{G,ri}, F_{G,Q}$ all have at most degree 2. If all $F_{G,vi}$ terms have degree 2 then X has degree $2k$ and $\frac{X}{F_{G,vi}}$ has degree $2k - 2$.

Consequently, $XF_{G,Q}$ has at most degree $2k$ and $F_{G,x} \cdot \frac{F_{G,ri} \cdot X}{F_{G,vi}}$ has degree at most $2 + 2k - 2 = 2k$.

Finally, $\mathcal{A}$ halts and outputs a guess of d' for d. $\mathcal{B}$ now selects random $g^* \in \mathbb{G}$ and x^*, y^*, v_1^*, ..., v_k^*, r_1^*, ..., r_k^*, $c^* \in \mathbb{Z}_p$. T_b is set to $e(g^*, g^*)^{x^* y^* \sum_{i=1}^k r_i^*}$ and $T_{1-b} = e(g^*, g^*)^{c^*}$. All elements besides T_b are independent of each other. Therefore the simulation engineered by $\mathcal{B}$ is consistent with these values unless one of the following events occur:

- Two values in $\mathbb{G}$ are not chosen independently
- Two values in $\mathbb{G}_T$ are not chosen independently
- The k-BDH Query returns 0 because the polynomial representation of the inputs do not satisfy the test conditions. However, for the actual integer inputs the test condition holds.
- It is possible to find a multiple of $e(g^x, g^y)^{\sum_{i=1}^k r_i}$ from the other terms in the polynomial.
- It is possible to learn information about $e(g^x, g^y)^{\sum_{i=1}^k r_i}$ from the $L_k^{\mathbb{G}}$ query.

The elements in $\mathbb{G}$ are all chosen independently. Since $\mathcal{A}$ makes $q_{G_g} + q_{G_i}$ group actions or inversion queries for group $\mathbb{G}$ the corresponding elements in F_G are at most degree 2 and the probability of a collision is $\binom{q_{G_g}+q_{G_i}+2k+3}{2}\frac{2}{p}$. For the elements in $\mathbb{G}_T$ there $q_{T_g} + q_{T_i} + q_B$ group actions or inversion or bilinear map queries are made resulting in elements in $\mathbb{G}_T$. Since elements in $\mathbb{G}_T$ have corresponding polynomials in F_T with degree at most 4 the probability of a collision is $\binom{q_{T_g}+q_{T_i}+q_B+2}{2}\frac{4}{p}$. For each of the $K_k^{\mathbb{G}}$ queries the probability of selecting two polynomials with the same value is $\frac{2k}{p}$ for a total probability of $\binom{q_k}{2}\frac{2k}{p}$.

The $L_k^{\mathbb{G}}$ cannot ever answer true for a question about the target since none of the inputs to $L_k^{\mathbb{G}}$ are in the target group.

The probability of finding a collision is bounded by:

$$\begin{aligned}\epsilon \leq & \binom{q_{G_g} + q_{G_i} + 2k + 3}{2}\frac{2}{p} + \binom{q_{T_g} + q_{T_i} + q_B + 2}{2}\frac{4}{p} + \binom{q_k}{2}\frac{2k}{p} \\ < & \frac{(q + 2k + 3)^2 + 2(q + 2)^2 + kq^2}{p} < \frac{(k + 3)(q + 2k + 3)^2}{p}\end{aligned}$$

The combination of this result and Theorem 1 is $L_k^{\mathbb{G}} \lneq k$-BDH. Collectively, we have $L_k^{\mathbb{G}} \lneq k$-BDH $\lneq L_k^{\mathbb{G}_T}$ as desired. □

C Definitions

In this section we provide the standard definitions and security models for IBE. The definition for adaptive security was given by Boneh and Franklin [BF03]; the definition for selective-ID security was first given by Canetti, Halevi, and Katz [CHK03].

C.1 Identity Based Encryption

Definition 2. *Identity Based Encryption consists of the following algorithms:*

- Setup *produces the public system parameters and the private master-key.*
- KeyGen(ID) *uses the master-key to create a private key for an identity.*

- $\mathsf{Encrypt}(m, \mathsf{ID})$ *uses the system parameters to encrypt a message m for an identity* ID*. A message can be encrypted for a user with identity* ID *even if a private key does not yet exist for the identity.*

- $\mathsf{Decrypt}(c, \mathsf{ID})$ *uses the private key belonging to* ID *to decrypt the ciphertext.*

C.2 Chosen Plaintext Security for Identity-Based Encryption

CPA security for IBE is defined using the following game:

Setup: The challenger runs the Setup algorithm to generate the public parameters which are given to the adversary. It also generates the master-key which is kept secret.

Phase 1: The adversary issues at most poly(k) KeyGen queries to obtain the private key corresponding to the identities $\mathsf{ID}_1, \ldots, \mathsf{ID}_{\text{poly}(k)}$. The queries may be adaptive.

Challenge: The adversary selects an identity ID^* and two messages M_0 and M_1 on which it wishes to challenge. The challenger selects a random bit $b \in \{0, 1\}$ and responds with $\mathsf{Encrypt}(\mathsf{M}_b, \mathsf{ID}^*)$. The only restriction is that $\mathsf{ID}_i \neq \mathsf{ID}^*$ for $1 \leq i \leq \text{poly}(k)$.

Phase 2: This phase is the same as Phase 1 except a private key will not be generated when the queried identity is ID^*.

Guess: The adversary submits a guess of $b' \in \{0, 1\}$ and wins the game if $b = b'$.
The advantage of an adversary $\mathcal{A}$ attacking a scheme $\mathcal{E}$ is $\text{Adv}_{\mathcal{E},\mathcal{A}} = |\Pr[b = b'] - \frac{1}{2}|$. The probability is taken over the random bits used by the adversary and challenger.

Definition 3. *An Identity-Based Encryption scheme is chosen plaintext secure if all polynomial time adversaries have negligible advantage against the CPA Security Game.*

C.3 Selective Identity CPA Security for Identity Based Encryption

In this model of IBE, there is an additional stage(**Init**) which occurs before the Setup algorithm. In this stage the adversary selects ID^* the identity to be challenged. Consequently, the public parameters may depend on ID^*.

C.4 Chosen Ciphertext Security for Identity-Based Encryption

The CCA-Security game allows, during Phase 1 and Phase 2, for decryption queries of ciphertext encrypted for ID^*. There is a restriction that the ciphertext is not equal to the challenge ciphertext in Phase 2. There are general methods [BCHK07, BMW05] of converting a CPA-Secure IBE into a CCA-Secure scheme.

D Selective-ID Secure HIBE Under the k-BDH Assumption

In this section we show how to transform the scheme into a hierarchical identity based encryption scheme (HIBE). A L-level HIBE is constructed in the same way as the Boneh-Boyen HIBE [BB04a]. Instead of having identities represented as a single length string, there are $L' \leq L$ strings of length. L is the depth of the HIBE. We write $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_h, \ldots, \mathsf{ID}_{L'})$. The h^{th} component corresponds to the identity at level h.

Setup : The public parameters are $(g,\ u = g^x,\ v_1 = g^{s_1},\ \ldots,\ v_k = g^{s_k},\ v_1^{r_1},\ \ldots,\ v_k^{s_k},\ w_{1,1},\ \ldots,\ w_{k,L})$. The values $s_1, \ldots, s_k, r_1, \ldots, r_k, x$ are kept as the master-key.

$\mathsf{KeyGen}(\mathsf{ID})$: Select random $n_{1,1}, \ldots, n_{k,L'} \in \mathbb{Z}_p^*$. Output $K_{A,i} = g^{xr_i} \prod_{h=1}^{L'} (w_{i,h} u^{\mathsf{ID}_h})^{n_{i,h}}$ for each $1 \le i \le k$ and $K_{B,i,h} = v_i^{n_{i,h}}$ for each $1 \le i \le k$ and $1 \le h \le L' \le L$.
Note that the private key for ID can be generated from the parent identity $(\mathsf{ID}_1, \ldots, \mathsf{ID}_{L'-1})$. Suppose the parent's private key was $(K_{A,1}, \ldots, K_{A,k}, K_{B,1,1}, \ldots, K_{B,k,L'-1})$. For each value of $1 \le i \le k$, the parent select random $n_{i,L'}$, updates $K_{A,i} = K_{A,i} \cdot (w_{i,L'} u^{\mathsf{ID}_{L'}})^{n_{i,L'}}$ for all $1 \le i \le k$, and sets $K_{B,i,L'} = v_i^{n_{i,L'}}$. The parent then outputs all $K_{A,i}$ and $K_{B,i,h}$ values.
$\mathsf{Encrypt}(m, \mathsf{ID})$: Select random $y_1, \ldots, y_k \in \mathbb{Z}_p^*$. Output $C_0 = m \prod_{1 \le i \le k} e(g^x, v_i^{r_i})^{y_i}$. For each $1 \le i \le k$ output $C_{A,i} = v_i^{y_i}$. Additionally for each $1 \le i \le k$ and $1 \le h \le L' \le L$ output $C_{B,i,h} = (w_{i,h} u^{\mathsf{ID}_h})^{y_i}$.

$\mathsf{Decrypt}(c)$: Output

$$\frac{C_0 \cdot \prod_{i=1}^{k} \prod_{h=1}^{L'} e(K_{B,i,h}, C_{B,i,h})}{\prod_{i=1}^{k} e(K_{A,i}, C_{A,i})} = \frac{m \prod_{i=1}^{k} e(g^x, v_i^{r_i})^{y_i} \cdot \prod_{i=1}^{k} \prod_{h=1}^{L'} e(v_i^{n_{i,h}}, (w_{i,h} u^{\mathsf{ID}_h})^{y_i})}{\prod_{i=1}^{k} e(g^{xr_i} \prod_{h=1}^{L'} (w_{i,h} u^{\mathsf{ID}_h})^{n_{i,h}}, v_i^{y_i})} = m$$

The HIBE extension allows us to use standard transformations [BCHK07, BMW05] to construct a Selective-ID CCA-Secure L-level HIBE from a Selective-ID CPA-Secure $(L+1)$-level HIBE.

Theorem 10. *Suppose the k-BDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$ (precisely, no t-time algorithm has advantage at least ϵ in solving the k-BDH problem in $\mathbb{G}$ and $\mathbb{G}_T$). Then the previously defined HIBE system is $(t - \Theta(\tau Lkq), q, \epsilon)$-Selective-ID IND-CPA secure where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the HIBE system. We build $\mathcal{B}$ to solve a decisional k-BDH instance $(g,\ g^x,\ g^y,\ v_1,\ \ldots,\ v_k,\ v_1^{r_1},\ \ldots,\ v_k^{r_k},\ T \overset{?}{=} e(g,g)^{xy(r_1+\cdots+r_k)})$. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective identity game as follows:
Init: The selective identity games begins with $\mathcal{A}$ outputting an identity to attacked ID^*=$(\mathsf{ID}_1^*, \ldots, \mathsf{ID}_{L'}^*)$ where $L' \le L$.

Setup: Algorithm $\mathcal{B}$ first selects random a_i, $t_{i,j}$, and $\mathsf{ID}_{h-L'}^*$ for $1 \le i \le k$, $1 \le j \le L$ and $L' < h \le L$. It then sets the public parameters to: $(g,\ u = g^x,\ v_1,\ldots,v_k,\ v_1^{\hat{r_1}} = (v_1^{r_1})^{1/a_1},\ \ldots,\ v_k^{\hat{r_k}} = (v_k^{r_k})^{1/a_k},\ w_{1,1} = v_1^{t_{1,1}}(g^x)^{-\mathsf{ID}_1^*},\ \ldots,\ w_{k,L} = v_k^{t_{k,L}}(g^x)^{-\mathsf{ID}_L^*})$. These parameters are are all independent of ID^* in the view of $\mathcal{A}$. The a_i terms will serve as the way to randomize the challenge.

$\mathsf{Phase\ 1}$: $\mathcal{A}$ issues queries for the private key of an identity, $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_u)$ for $u \le L$. It must be the case that ID is not a prefix of ID^*.

Let l be the smallest index value such that $\mathsf{ID}_l \ne \mathsf{ID}_l^*$. For each value $1 \le i \le k$ and $1 \le h \le l$, select random $m_{i,h}$. Let $d_h = \mathsf{ID}_h - \mathsf{ID}_h^*$. $\mathcal{B}$ outputs: $(v_i^{\hat{r_i}})^{-t_{i,l}/d_l} \prod_{h=1}^{l} [v_i^{t_{i,h}} g^{xd_h}]^{m_{i,h}}$ for each value of i. As well as the values: $v_i^{m_{i,h}}$ for all values of i and $1 \le h \le l-1$ and $(v_i^{\hat{r_i}})^{-1/d_l} v_i^{m_{i,l}}$ for each value of i. For $n_{i,l} = -\hat{r_i}/d_l + m_{i,l}$, which implies $m_{i,l} = \hat{r_i}/d_l + n_{i,l}$ this is the expected value. The remaining terms are uniformly distributed among all elements in $\mathbb{Z}_p^*$ due to the selection of $m_{i,h}$. Private keys can be generated for all identities except prefixes of ID^*.

$\mathsf{Challenge}(m_0, m_1)$: $\mathcal{B}$ picks random bit $b \in \{0,1\}$. The response is: $(m_bT,\ (g^y)^{a_1},\ \ldots,\ (g^y)^{a_k},\ (g^y)^{a_1 t_{1,1}},$ $(g^y)^{a_k t_{k,L'}})$ For each value of i, this implicitly sets $ya_i = s_i y_i$ or $y_i = ya_i/s_i$. If T is a valid k-BDH tuple then the response is drawn from a uniform distribution and m_bT is the expected value. If T is not a valid k-BDH tuple then the distribution is uniform and independent of b.

$\mathsf{Phase\ 2}$: $\mathcal{A}$ issues more private key queries. It is exactly the same as $\mathsf{Phase1}$.

Guess: $\mathcal{A}$ outputs a guess of $b' \in \{0,1\}$. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning T is a valid k-BDH tuple. Otherwise, it is not a valid k-BDH tuple and the output is 0.

When the input is a valid k-BDH instance, $\mathcal{A}$ must satisfy $|\Pr[b = b'] - \frac{1}{2}| \geq \epsilon$. When the input is not a valid k-BDH instance, the input is uniform and independent and $\Pr[b = b'] = \frac{1}{2}$. Therefore, we have

$$|\Pr[\mathcal{B}(\text{valid } k\text{-BDH}) = 0] - \Pr[\mathcal{B}(\text{not valid } k\text{-BDH}) = 0]| \geq |(\frac{1}{2} + \epsilon) - \frac{1}{2}| \geq \epsilon$$

as required. $\square$

E Fully CPA-Secure IBE Under the k-BDH Assumption

Here we apply a transformation to the selective-ID scheme to obtain a scheme that is fully secure. It is similar to the transformation Waters applied to the [BB04a] scheme in [Wat05]. Besides growing with respect to k, this transformation is slightly more complex than Waters' because the base of the parameters may be g or v_i. In this construction, identities are represented as bit strings of length l, a separate security parameter unrelated to p. This bit string is denoted ID and the j^{th} bit of the string is ID_j. Alternatively, using a collision resistant hash function, $\mathcal{H}(\{0,1\}^*) \to \{0,1\}^l$, identities can be arbitrarily lengths. We define $\mathcal{V} \subseteq \{1,\ldots,l\}$ to be the set of all j for which $\mathsf{ID}_j = 1$.

Setup : The public parameters are $(g,\ g^x,\ v_1 = g^{s_1},\ \ldots,\ v_k = g^{s_k},\ v_1^{r_1},\ \ldots,\ v_k^{s_k},\ w_1,\ \ldots,\ w_k,\ u_{1,1},\ \ldots,$ $u_{k,l})$. The values $s_1,\ldots,s_k,r_1,\ldots,r_k,x$ are kept as the master-key.

$\mathsf{KeyGen}(\mathsf{ID})$: Select random $n_1,\ldots,n_k \in \mathbb{Z}_p^*$. For each $1 \leq i \leq k$ output

$$(K_{A,i}, K_{B,i}) = (g^{xr_i}(w_i \prod_{j\in\mathcal{V}} u_{i,j})^{n_i}, v_i^{n_i})$$

$\mathsf{Encrypt}(m, \mathsf{ID})$: Select random $y_1,\ldots,y_k \in \mathbb{Z}_p^*$. Output $C_0 = m \prod_{1\leq i\leq k} e(g^x, v_i^{r_i})^{y_i}$ and for each $1 \leq i \leq k$ output $(C_{A,i}, C_{B,i})$=$(v_i^{y_i}, (w_i \prod_{j\in\mathcal{V}} u_{i,j})^{y_i})$.

$\mathsf{Decrypt}(c)$: Output

$$\frac{C_0 \cdot \prod_{1\leq i\leq k} e(K_{B,i}, C_{B,i})}{\prod_{1\leq i\leq k} e(K_{A,i}, C_{A,i})} = \frac{m \prod_{1\leq i\leq k} e(g^x, v_i^{r_i})^{y_i} \cdot \prod_{1\leq i\leq k} e(v_i^{n_i}, (w_i \prod_{j\in\mathcal{V}} u_{i,j})^{y_i})}{\prod_{1\leq i\leq k} e(g^{xr_i}(w_i \prod_{j\in\mathcal{V}} u_{i,j})^{n_i}, v_i^{y_i})} = m$$

We chose to prove security using the artificial abort method, and our proof uses the same arguments as [Wat05, HW10]. The only deviation is in the selection of w_i and $u_{i,j}$; which are set to $(g^x)^{p-4qc+\alpha'} v_i^{\beta'_i}$

and $(g^x)^{\alpha_j} v_i^{\beta_{i,j}}$ respectively for random values $c, \alpha', \beta'_i, \alpha_j, \beta_{i,j}$ where q is the maximum number of queries made by the adversary.

Theorem 11. *Suppose the k-BDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$. Then the fully CPA-Secure IBE system is IND-CPA secure.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the IBE system after making at most q queries. We build a simulator $\mathcal{B}$ to solve a decisional k-BDH instance $(g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, T)$. The output is a bit γ where 1 corresponds to a properly formed k-BDH tuple. If $\mathcal{A}$ submits q KeyGen queries and succeeds with probability $\frac{1}{2} + \epsilon$, then $\mathcal{B}$ breaks the k-BDH assumption with probability $\frac{1}{2} + \frac{3\epsilon}{64q(l+1)}$ where l is the bit length of the identity input. Simulator $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective identity game as follows:

Setup: *Random Choices:* $\mathcal{B}$ first selects random $a_i \in \mathbb{Z}_p$ for $1 \leq i \leq k$; this will serve as a way to randomize the challenge. It also chooses integer c uniformly at random from 0 to l, α' an integer chosen uniformly at random from $\mathbb{Z}_{4q}$, and an l-length vector $\overrightarrow{\alpha} = \{\alpha_j\}$ in which the elements are chosen randomly in $\mathbb{Z}_{4q}$. Additionally, for each value $1 \leq i \leq k$ β'_i and an l-length vector $\overrightarrow{\beta}_i = \{\beta_{i,j}\}$ are chosen randomly from $\mathbb{Z}_p$.

Functions: For ease of analysis, the following functions are defined:

- $F(\mathsf{ID}) = p - 4qc + \alpha' + \sum_{j \in \mathcal{V}} \alpha_j$
- $J(i, \mathsf{ID}) = \beta'_i + \sum_{j \in \mathcal{V}} \beta_{i,j}$
- $K(\mathsf{ID}) = \begin{Bmatrix} 0 & \text{if } \alpha' + \sum_{j \in \mathcal{V}} \alpha_j \equiv 0 \bmod 4q \\ 1 & \text{otherwise} \end{Bmatrix}$

Public Parameters: It then sets the public parameters to: $(g, g^x, v_i, v_i^{\hat{r_i}} = (v_i^{r_i})^{\frac{1}{a_i}}, w_i = (g^x)^{p-4qc+\alpha'} v_i^{\beta'_i}, u_{i,j} = (g^x)^{\alpha_j} v_i^{\beta_{i,j}})$. These parameters have the correct distribution in the view of $\mathcal{A}$; the β terms randomize w_i and $u_{i,j}$

$\mathsf{Phase\ 1}$: $\mathcal{A}$ issues queries for the private key of an identity, ID. First $K(\mathsf{ID})$ is evaluated. If the result is 0 then a random guess of $\gamma \in \{0, 1\}$ is chosen and $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$'s response is generated as follows for each value of $1 \leq i \leq k$:

Select random m_i. Output $(v_i^{\hat{r_i}})^{\frac{-J(i,\mathsf{ID})}{F(\mathsf{ID})}} (w_i \prod_{j \in \mathcal{V}} u_{i,j})^{m_i}, (v_i^{\hat{r_i}})^{\frac{-1}{F(\mathsf{ID})}} v_i^{m_i}$. For $n_i = \frac{-\hat{r_i}}{F(\mathsf{ID})} + m_i \rightarrow m_i =$

$\frac{\hat{r_i}}{F(\mathsf{ID})} + n_i$ this is the expected value:

$$\begin{aligned}
&((v_i^{\hat{r_i}})^{\frac{-J(i,\mathsf{ID})}{F(\mathsf{ID})}}(w_i \prod_{j \in \mathcal{V}} u_{i,j})^{m_i}, (v_i^{\hat{r_i}})^{\frac{-1}{F(\mathsf{ID})}} v_i^{m_i}) \\
&= ((v_i^{\hat{r_i}})^{\frac{-J(i,\mathsf{ID})}{F(\mathsf{ID})}}(w_i \prod_{j \in \mathcal{V}} u_{i,j})^{\frac{\hat{r_i}}{F(\mathsf{ID})}+n_i}, (v_i^{\hat{r_i}})^{\frac{-1}{F(\mathsf{ID})}} v_i^{\frac{\hat{r_i}}{F(\mathsf{ID})}+n_i}) \\
&= ((v_i^{\hat{r_i}})^{\frac{-(\beta'_i+\sum_{j\in\mathcal{V}}\beta_{i,j})}{F(\mathsf{ID})}}(w_i \prod_{j \in \mathcal{V}} u_{i,j})^{\frac{\hat{r_i}}{F(\mathsf{ID})}}(w_i \prod_{j \in \mathcal{V}} u_{i,j})^{n_i}, v_i^{n_i}) \\
&= ((v_i^{\hat{r_i}})^{\frac{-(\beta'_i+\sum_{j\in\mathcal{V}}\beta_{i,j})}{F(\mathsf{ID})}}((g^x)^{p-4qc+\alpha'} v_i^{\beta'_i}(g^x)^{\sum_{j\in\mathcal{V}}\alpha_j} v_i^{\sum_{j\in\mathcal{V}}\beta_{i,j}})^{\frac{\hat{r_i}}{F(\mathsf{ID})}}(w_i \prod_{j \in \mathcal{V}} u_{i,j})^{n_i}, v_i^{n_i}) \\
&= (((g^x)^{p-4qc+\alpha'+\sum_{j\in\mathcal{V}}\alpha_j})^{\frac{\hat{r_i}}{F(\mathsf{ID})}}(w_i \prod_{j \in \mathcal{V}} u_{i,j})^{n_i}, v_i^{n_i}) \\
&= ((g^x)^{\hat{r_i}}(w_i \prod_{j \in \mathcal{V}} u_{i,j})^{n_i}, v_i^{n_i})
\end{aligned}$$

The second term is uniformly distributed among all elements in $\mathbb{Z}_p$ due to the selection of m_i. If $K(\mathsf{ID}) \neq 0$ this implies $F(\mathsf{ID}) \neq 0 \bmod p$ since p should be chosen such that $p > 4qc$.

$\mathsf{Challenge}(\mathsf{ID}^*, m_0, m_1)$: The simulator checks if $F(\mathsf{ID}^*) \equiv 0 \bmod p$. If not, it aborts after picking a random value of $\gamma \in \{0,1\}$. Otherwise $\mathcal{B}$ proceeds by picking random bit $b \in \{0,1\}$. Output to $\mathcal{A}$, the response:

$$(m_b T, (v_1^{y_1} = (g^y)^{a_1}, (w_1 \prod_{j \in \mathcal{V}} u_{1,j})^{y_1} = (g^y)^{a_1 J(1,\mathsf{ID}^*)}), \ldots, ((g^y)^{a_k}, (g^y)^{a_k J(k,\mathsf{ID}^*)})).$$

For each value of i, this implicitly sets $ya_i = s_i y_i$ or $y_i = ya_i/s_i$.
The $(w_i \prod u_{i,j})^{y_i}$ terms are correct. The expected value is

$$(w_i \prod_{j \in \mathcal{V}} u_{i,j})^{y_i} = ((g^x)^{p-4qc+\alpha'} v_i^{\beta'_i}(g^x)^{\sum_{j\in\mathcal{V}}\alpha_j} v_i^{\sum_{j\in\mathcal{V}}\beta_{i,j}})^{y_i}.$$

Since $F(\mathsf{ID}^*) \equiv 0 \bmod p$ the terms involving α' and α_j sum to 0 mod p and we are left with $(v_i^{\beta'_i+\sum_{j\in\mathcal{V}}\beta_{i,j}})^{y_i}$. Substituting $y_i = ya_i/s_i$ the term evaluates to $(g^{\beta'_i+\sum_{j\in\mathcal{V}}\beta_{i,j}})^{ya_i}$. Using the definition of $J(i, \mathsf{ID}^*)$ we arrive at $g^{ya_i J(i,\mathsf{ID}^*)}$.
If T is a valid k-BDH tuple then the first value of the response is drawn from a uniform distribution and $m_b T$ is the expected value:

$$m_b \prod_{1 \le i \le k} e(g,g)^{xyr_i} = m_b \prod_{1 \le i \le k} e(g^x, g^{s_i r_i/a_i})^{ya_i/s_i} = m_b \prod_{1 \le i \le k} e(g^x, v_i^{r_i/a_i})^{y_i} = m_b \prod_{1 \le i \le k} e(g^x, v_i^{\hat{r_i}})^{y_i}$$

If T is not a valid k-BDH tuple then the distribution is uniform and independent of b.

$\mathsf{Phase\ 2}$: $\mathcal{A}$ issues more private key queries. It is the same as $\mathsf{Phase1}$ except it will not answer queries for ID^*.

Guess: $\mathcal{A}$ outputs a guess of $b \in \{0,1\}$. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning T is a valid k-BDH tuple. Otherwise, it is not a valid k-BDH tuple and the output is 0.

As Waters notes about his construction in [Wat05], the simulator cannot use the output from the adversary as the probability of success may be correlated with the probability the simulator aborts. The same limitation applies to our construction. Waters introduced the novel idea of an *artificial abort* in which there is a chance that the simulator will abort on every query independent of the identity. Bellare and Ristenpart [BR09] subsequently prove security without artificial aborts. Their new proof provides a tighter reduction for low to mid range security parameters, while the original proof is better for large security parameters.

Here we use artificial aborts, but in the framework provided by Hohenberger and Waters for verifiable random functions [HW10]. This technique uses a series of games. The first game is the original security game and the last game is exactly the view of $\mathcal{A}$ when interacting with $\mathcal{B}$. We then show through a series of claims that if an adversary $\mathcal{A}$ is successful against Game j it will also be successful against Game $j+1$.

Game 1: This game is defined to be the CPA security game.

Game 2: This game is the same as Game 1 except that a record is kept of the queries made by $\mathcal{A}$ (denoted $\overrightarrow{Q} = \{Q_1, \ldots, Q_q, Q^*\}$ where Q^* is the challenge input). Upon completion of the game, we choose random integer α' and vector $\overrightarrow{\alpha} = \{\alpha_1, \ldots, \alpha_l\}$ all in $\mathbb{Z}_{4q}$. Additionally, select an integer between 0 and l. Define the following abort indicator function:

$$\tau(\overrightarrow{Q}) = \begin{Bmatrix} 1 & \text{if } F(Q^*) \neq 0 \vee_{i=1}^{q} K(Q_i) = 0 \\ 0 & \text{otherwise} \end{Bmatrix}$$

where $F(\mathsf{ID})$ and $K(\mathsf{ID})$ are defined as before. $\mathcal{A}$'s success is defined as follows:

- *Regular Abort:* If $\tau(\overrightarrow{Q}) = 1$ then flip a fair coin $\gamma \in \{0,1\}$ and say $\mathcal{A}$ wins if $\gamma = 1$ and loses otherwise. Note whenever $\tau(\overrightarrow{Q}) = 1$ the queries $\overrightarrow{Q}$ would cause an abort had the simulator set $\alpha, \overrightarrow{\alpha}, c$ accordingly. Let $\zeta(\overrightarrow{Q}) = \Pr[\tau(\overrightarrow{Q}) = 1 | \alpha, \overrightarrow{\alpha}, c]$.
- *Artificial or Balancing Abort:* We follow the same steps as [Wat05] and [HW10] to artificially abort with the correct probability. Let $\zeta_{\min} = \frac{1}{8q(l+1)}$. Set ζ'_Q be the simulator's estimate of $\zeta(\overrightarrow{Q})$ by evaluating $\tau(\overrightarrow{Q})$ with fresh random values α, $\overrightarrow{\alpha}$ and c a total of $T = O(\epsilon^{-2} ln(\epsilon^{-1}) \zeta_{\min}^{-1} ln(\zeta_{\min}^{-1}))$ times. This does not require running the distinguisher again.
With probability $\frac{\zeta'_Q - \zeta_{\min}}{\zeta'_Q}$ abort by flipping a fair coin $\gamma \in \{0,1\}$. $\mathcal{A}$ wins if $\gamma = 1$ and loses otherwise.
- If we do not abort (regularly or artificially), $\mathcal{A}$ wins if it correctly guessed b' in the real security game.

Game 3: In this game we run Game 2 but instead of choosing to abort at the end if any of the abort conditions on $\overrightarrow{Q}$ or Q^* are satisfied, the decision to abort is made inline. Specifically, this means for every query $Q_i \in \overrightarrow{Q}$, $\mathcal{B}$ first tests the abort conditions and if they are satisfied the abortion occurs immediately (by flipping a fair coin $\gamma \in \{0,1\}$ and saying $\mathcal{A}$ wins if $\gamma = 1$). If the abort conditions are not met, $\mathcal{B}$ answers the query. Note that Game 3 is exactly the game that the simulator plays.

We prove if $\mathcal{A}$ wins Game 1 with probability $\frac{1}{2} + \epsilon$, then it succeeds in winning Game 3 with probability $\frac{1}{2} + \frac{3\epsilon}{64q(l+1)}$.

Before we reason about $\mathcal{A}$'s success in the games we reiterate three claims about the probability of aborting from [HW10].

Claim 1. *Let $\zeta_{min} = \frac{1}{8q(l+1)}$. For any query vector $\overrightarrow{Q}$, $\zeta(\overrightarrow{Q}) \geq \zeta_{min}$.*

Proof. As in [Wat05, HW10] we want to find a lower bound on the probability of the simulation not triggering a regular abort. The proof is repeated here for completeness.

Without loss of generality, assume the adversary always makes the maximum number of queries q as the probability of not aborting increases with fewer queries. Fix an arbitrary $\overrightarrow{Q}$=(Q_1, …, Q_q, Q^*) $\in \mathbb{Z}\{0,1\}^l$. Then with the probability over the choice of $\overrightarrow{Q}, c$ we have that Pr[abort on $\overrightarrow{Q}$] is

$$
\begin{aligned}
&= \Pr[\wedge_{i=1}^{q} K(Q_i) = 1 \wedge F(Q^*) \equiv 0 \bmod p] && (1)\\
&= (1 - \Pr[\vee_{i=1}^{q} K(Q_i) = 0]) \cdot \Pr[F(Q^*) \equiv 0 \bmod p | \wedge_{i=1}^{q} K(Q_i) = 1] && (2)\\
&\geq (1 - \sum_{i=1}^{q} \Pr[K(Q_i) = 0]) \cdot \Pr[F(Q^*) \equiv 0 \bmod p | \wedge_{i=1}^{q} K(Q_i) = 1] && (3)\\
&= (1 - \frac{q}{m}) \cdot \Pr[F(Q^*) \equiv 0 \bmod p | \wedge_{i=1}^{q} K(Q_i) = 1] && (4)\\
&= \frac{1}{l+1} \cdot (1 - \frac{q}{m}) \cdot \Pr[K(Q^*) = 0 | \wedge_{i=1}^{q} K(Q_i) = 1] && (5)\\
&= \frac{1}{l+1} \cdot (1 - \frac{q}{m}) \cdot \frac{\Pr[K(Q^*) = 0] \cdot \Pr[\wedge_{i=1}^{q} K(Q_i) = 1 | K(Q^*) = 0]}{\Pr[\wedge_{i=1}^{q} K(Q_i) = 1]} && (6)\\
&\geq \frac{1}{m(l+1)} \cdot (1 - \frac{q}{m}) \cdot \Pr[\wedge_{i=1}^{q} K(Q_i) = 1 | K(Q^*) = 0] && (7)\\
&= \frac{1}{m(l+1)} \cdot (1 - \frac{q}{m}) \cdot \Pr[1 - \vee_{i=1}^{q} K(Q_i) = 0 | K(Q^*) = 0] && (8)\\
&\geq \frac{1}{m(l+1)} \cdot (1 - \frac{q}{m}) \cdot (1 - \sum_{i=1}^{q} \Pr[K(Q_i) = 0] | K(Q^*) = 0) && (9)\\
&= \frac{1}{m(l+1)} \cdot (1 - \frac{q}{m})^2 && (10)\\
&\geq \frac{1}{m(l+1)} \cdot (1 - \frac{2q}{m}) && (11)\\
&= \frac{1}{8q(l+1)} && (12)
\end{aligned}
$$

In the original IBE, Waters uses $\frac{1}{m}$ to quantify $\Pr[K(Q) = 0]$ for any query Q. The value is optimized to be $4q$. Equations (4) and (7) use $\Pr[K(Q) = 0]=\frac{1}{m}$. By definition $F(\mathsf{ID}) = p - mc + \alpha' + \sum_{j \in \mathcal{V}} \alpha_j$, so equation (5) gets a factor of $\frac{1}{l+1}$ from the simulator guessing the value of c. Equation (6) is derived by applying Bayes' Theorem. Equation (9) follows from the pairwise independence of the probabilities $K(Q) = 0$,$K(\hat{Q}) = 0$ for any pair of queries $Q \neq \hat{Q}$, since they will differ in at least one random ID_j value. Equation (12) is the result of applying $m = 4q$. □

In the following claims we use Chernoff bounds to find an upper and lower bounds on the probability of any abort (regular or artificial). These bounds will be used in evaluating the adversary's success in Game 2.

Claim 2. *For any set of queries $\overrightarrow{Q}$, the probability that there is an abort (regular or artificial) is $\geq 1 - \zeta_{min} - \frac{3}{8}\zeta_{min}\epsilon$.*

Proof. Let $\zeta_Q = \zeta(\overrightarrow{Q}) = \Pr[\tau(\overrightarrow{Q}) = 1 | \alpha, \overrightarrow{\alpha}, c]$ be the probability that a set of queries $\overrightarrow{Q}$ do not cause a regular abort. In Game 2, $T = O(\epsilon^{-2} ln(\epsilon^{-1})\zeta_{\min}^{-1}\ln(\zeta_{\min}^{-1}))$ samples are taken to approximate this value as ζ'_Q. By Chernoff Bounds

$$\Pr[T\zeta'_Q < T\zeta_Q(1 - \frac{\epsilon}{8})] < e^{-[64\epsilon^{-2}\ln((\epsilon/8)^{-1})\zeta_{\min}^{-1}\ln(\zeta_{\min}^{-1})(\zeta_{\min})(\epsilon/8)^2]},$$

which reduces to

$$\Pr[\zeta'_Q < \zeta_Q(1 - \frac{\epsilon}{8})] < \zeta_{\min}\frac{\epsilon}{8}.$$

Recall that for a measured ζ'_Q an artificial abort will not happen with probability $\frac{\zeta_{\min}}{\zeta'_Q}$. The probability of aborting is

$$\begin{aligned}
\Pr[\mathsf{abort}] &= 1 - \Pr[\overline{\mathsf{abort}}] = 1 - \Pr[\overline{\mathsf{RA}}]\Pr[\overline{\mathsf{AA}}] = 1 - \zeta_Q \Pr[\overline{\mathsf{AA}}] \\
&\geq 1 - \zeta_Q(\zeta_{\min}\frac{\epsilon}{8} + \frac{\zeta_{\min}}{\zeta_{\min}(1-\epsilon/8)}) \\
&\geq 1 - (\frac{\zeta_{\min}\epsilon}{8} + \frac{\zeta_{\min}}{1-\epsilon/8}) \\
&\geq 1 - (\frac{\zeta_{\min}\epsilon}{8} + \zeta_{\min}(1 + \frac{2\epsilon}{8})) \\
&\geq 1 - \zeta_{\min} - \zeta_{\min}\frac{3\epsilon}{8}
\end{aligned}$$

□

Claim 3. *For any series of queries $\overrightarrow{Q}$, the probability that there is no abort (regular or artificial) is $\geq \zeta_{min} - \frac{1}{4}\zeta_{min}\epsilon$.*

Proof. Let $\zeta_Q = \zeta(\overrightarrow{Q}) = \Pr[\tau(\overrightarrow{Q}) = 1 | \alpha, \overrightarrow{\alpha}, c]$ be the probability that a set of queries $\overrightarrow{Q}$ do not cause a regular abort. In Game 2, $T = O(\epsilon^{-2} ln(\epsilon^{-1})\zeta_{\min}^{-1}\ln(\zeta_{\min}^{-1}))$ samples are taken to approximate this value as ζ'_Q. By Chernoff Bounds

$$\Pr[T\zeta'_Q > T\zeta_Q(1 + \frac{\epsilon}{8})] < e^{-[64\epsilon^{-2}\ln((\epsilon/8)^{-1})\zeta_{\min}^{-1}\ln(\zeta_{\min}^{-1})(\zeta_{\min})(\epsilon/8)^2]},$$

which reduces to

$$\Pr[\zeta'_Q > \zeta_Q(1 + \frac{\epsilon}{8})] < \zeta_{\min}\frac{\epsilon}{8}.$$

Recall that for a measured ζ_Q an artificial abort will not happen with probability $\frac{\zeta_{\min}}{\zeta'_Q}$. The probability of aborting is

$$\begin{aligned}
\Pr[\overline{\mathsf{abort}}] &= \Pr[\overline{\mathsf{RA}}]\Pr[\overline{\mathsf{AA}}] \\
&\geq \zeta_Q(1 - \frac{\zeta_{\min}\epsilon}{8})\frac{\zeta_{\min}}{\zeta_Q(1+\epsilon/8)} \\
&\geq \zeta_{\min}(1 - \frac{\epsilon}{8})^2 \\
&\geq \zeta_{\min}(1 - \frac{\epsilon}{4})
\end{aligned}$$

□

Analyzing $\mathcal{A}$'s probability of success in the Games. We now complete the proof by analyzing the probability $\mathcal{A}$ can distinguish between the games. We denotes $\mathcal{A}$'s probability of success in Game x as $\mathbf{Adv}_{\mathcal{A}}[\text{Game } x]$

Lemma 2. *If $\mathbf{Adv}_{\mathcal{A}}[\text{Game } 1] = \frac{1}{2} + \epsilon$ then $\mathbf{Adv}_{\mathcal{A}}[\text{Game } 2] = \frac{1}{2} + \frac{3\epsilon}{64q(l+1)}$.*

Proof. We use the same argument as [HW10]. Start by observing that $\mathbf{Adv}_{\mathcal{A}}[\text{Game } 2] =$

$$
\begin{aligned}
&= \mathbf{Adv}_{\mathcal{A}}[\text{Game } 2|\mathsf{abort}] \cdot \Pr[\mathsf{abort}] + \mathbf{Adv}_{\mathcal{A}}[\text{Game } 2|\overline{\mathsf{abort}}] \cdot \Pr[\overline{\mathsf{abort}}] && (13)\\
&= \frac{1}{2}\Pr[\mathsf{abort}] + \mathbf{Adv}_{\mathcal{A}}[\text{Game } 2|\overline{\mathsf{abort}}] \cdot \Pr[\overline{\mathsf{abort}}] && (14)\\
&= \frac{1}{2}\Pr[\mathsf{abort}] + \Pr[b = b'|\overline{\mathsf{abort}}] \cdot \Pr[\overline{\mathsf{abort}}] && (15)\\
&= \frac{1}{2}\Pr[\mathsf{abort}] + \Pr[b = b'] \cdot \Pr[\overline{\mathsf{abort}}|b = b'] && (16)\\
&= \frac{1}{2}\Pr[\mathsf{abort}] + (\frac{1}{2} + \epsilon) \cdot \Pr[\overline{\mathsf{abort}}|b = b'] && (17)\\
&\geq \frac{1}{2}(1 - \zeta_{\min} - s_1) + (\frac{1}{2} + \epsilon)(\zeta_{\min} - s_2) && (18)\\
&\geq \frac{1}{2} + \epsilon \cdot \zeta_{\min} - (s_1 + s_2) && (19)\\
&= \frac{1}{2} + \frac{3\epsilon \cdot \zeta_{\min}}{8} && (20)\\
&= \frac{1}{2} + \frac{3\epsilon}{64q(l+1)} && (21)
\end{aligned}
$$

Equation (14) follows from the fact that, in the case of abort $\mathcal{A}$'s success is determined by a coin flip. It would be very convenient if we could claim $\mathbf{Adv}_{\mathcal{A}}[\text{Game } 2|\overline{\mathsf{abort}}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game } 1]$, but unfortunately this is false. The event that $\mathcal{A}$ wins Game 2 and the event of an abort are not independent; however, we have inserted the balancing abort condition in the attempt to lessen the dependence between these events. Equation (15) simply states that, when there is no abort, $\mathcal{A}$ wins if and only if it guess correctly. Equation (16) follows from Bayes' Theorem. In (17) we observe $\Pr[b = b']$ is exactly $\mathcal{A}$'s success in Game 1.

The purpose of the artificial abort is to even the probability of aborting, for all queries made by $\mathcal{A}$, to be roughly $\zeta_{\min}$. This will also get rid of the conditional dependence on $b = b'$. There will be a small error which must be taken into account. Suppose that $\Pr[\mathsf{abort}] \geq 1 - \zeta_{\min} - s_1$ and $\Pr[\overline{\mathsf{abort}}] \geq \zeta_{\min} - s_2$, which must hold for some error values s_1 and s_2, then we derive Equation (18). Algebraic manipulation and recalling that $\epsilon \leq \frac{1}{2}$ brings us to Equation (19).

From Claim 1 we set $\zeta_{\min} = \frac{1}{8q(l+1)}$. The values of $s_1 = \frac{3\zeta_{\min}\epsilon}{8}$ and $s_2 = \frac{\zeta_{\min}\epsilon}{4}$ are derived from Claim 2 and Claim 3 respectively. Plugging these values into Equations (19) and (20) establishes the lemma. □

Lemma 3. $\mathbf{Adv}_{\mathcal{A}}[\text{Game } 2] = \mathbf{Adv}_{\mathcal{A}}[\text{Game } 3]$

Proof. We make the explicit observation that these games are equivalent by observing that their only difference is the time at which the regular abort occurs. The artificial abort stage is identical. All public parameters, evaluations and proofs have the same distribution up to the point of a possible abortion. In Game 2, the simulator receives all queries $\overrightarrow{Q}$, then checks $\tau(\overrightarrow{Q}) = 1$ (with the public parameters) and aborts taking a random guess, if so. In Game 3 the simulator checks with each new query Q if $K(Q) = 0$, which implies that the ending τ will be 1, and aborts, taking a random guess if so. Therefore, the output distributions will be the same. □

□