New Trapdoor Projection Maps for Composite-Order Bilinear Groups

Sarah Meiklejohn
UC San Diego
smeiklej@cs.ucsd.edu

Hovav Shacham
UC San Diego
hovav@cs.ucsd.edu

Abstract

An asymmetric pairing over groups of composite order is a bilinear map $e\colon G_1 \times G_2 \to G_T$ for groups G_1 and G_2 of composite order $N = pq$. We observe that a recent construction of pairing-friendly elliptic curves in this setting by Boneh, Rubin, and Silverberg exhibits surprising and unprecedented structure: projecting an element of the order-N^2 group $G_1 \oplus G_2$ onto the bilinear groups G_1 and G_2 requires knowledge of a trapdoor. This trapdoor, the square root of a certain number modulo N, seems strictly weaker than the trapdoors previously used in composite-order bilinear cryptography.

In this paper, we describe, characterize, and exploit this surprising structure. It is our thesis that the additional structure available in these curves will give rise to novel cryptographic constructions, and we initiate the study of such constructions. Both the subgroup hiding and SXDH assumptions appear to hold in the new setting; in addition, we introduce custom-tailored assumptions designed to capture the trapdoor nature of the projection maps into G_1 and G_2. Using the old and new assumptions, we describe an extended variant of the Boneh-Goh-Nissim cryptosystem that allows a user, at the time of encryption, to restrict the homomorphic operations that may be performed. We also present a variant of the Groth-Ostrovsky-Sahai NIZK, and new anonymous IBE, signature, and encryption schemes.

1 Introduction

Groups with computable pairings have, over the last decade, proved to be a fruitful setting for designing cryptographic primitives and protocols. The tripartite key agreement protocol of Joux [37], followed closely by the groundbreaking identity-based encryption scheme of Boneh and Franklin [12], have led to constructions for cryptographic goals including (but by no means limited to) zero knowledge proofs [35, 36, 34], group signatures [10, 19, 20], blind signatures [9, 2, 47], functional and attribute-based encryption [39, 43, 45], and anonymous credentials [21, 6, 5].

What makes such groups so useful for cryptography is the additional structure provided by the pairing: As compared to regular groups, the presence of a bilinear map e means that cryptographers have a useful new building block for their designs. Indeed, early work on pairing-based cryptography treated the pairing as a decisional Diffie-Hellman oracle [38, 16]; additional schemes were made possible when cryptographers began taking advantage of the algebraic structure that the pairing provides beyond the DDH oracle [14].

Pairing-based cryptography is thus a striking illustration of the value of algebraic structure for constructing cryptographic schemes: A richer structure allows for a wider variety of cryptographic schemes, provided that there exist some hard problems on which security can be based. It is perhaps surprising, then, that the way in which pairings are used have become quite standard. Most often, we imagine a bilinear group G to be a cyclic group of prime order that induces a map $e\colon G \times G \to G_T$ (where G_T is treated in a similarly abstract manner).

Two lines of work seek to generalize this understanding of pairings. One line considers bilinear groups G of composite order; the other line reconsiders the mathematical structure of the group G, for example to support *asymmetric* pairings $e\colon G_1 \times G_2 \to G_T$. Both these lines of work have been exploited to construct new cryptographic schemes, and we discuss both in detail below.

In this paper, we consider one of the instantiations of pairing-friendly elliptic curves proposed in a recent paper of Boneh, Rubin, and Silverberg [17]. We show that this instantiation exhibits surprising and unprecedented new structure (explained in more detail below); namely that projecting a point from the group G onto a subgroup G_1 or G_2 requires knowledge of a trapdoor. We believe that this structure can give rise to new cryptographic schemes, and we initiate the study of such schemes by proposing new hardness assumptions and protocols that rely on them.

More precisely, Boneh et al. construct, for the first time, composite-order bilinear groups that are also *ordinary*; i.e., non-supersingular. These constructions provide a bilinear group pair G_1, G_2, with each group of order $N = pq$. As already used previously in the literature on composite-order bilinear groups, there exist maps, computable by anyone who knows the factorization of N, that project from these groups to their order-p and q subgroups. But, as we observe in this paper, one of the three constructions of Boneh et al. has surprising additional structure. Whereas in all previous constructions of pairing-friendly elliptic curves projecting from the N-torsion group $G = G_1 \oplus G_2$ to its subgroups G_1 and G_2 was either easy or infeasible, in this instantiation it is possible to project from G to G_1 and G_2 *only when given a trapdoor*. This trapdoor — knowledge of a particular square root modulo n — seems to be strictly weaker than the trapdoor required to project "at the second level" into the order-p and q subgroups of G_1 and G_2.

In our first contribution, we describe, characterize, and exploit the surprising structure of the Boneh et al. construction of pairing-friendly curves. In Section 3, we detail the mathematical structure of G and define the maps "at the first level," from G to G_1 and G_2, as well as the more familiar maps "at the second level," to the p- and q-order subgroups of G_1 and G_2.

Although the bilinear group is therefore less efficient than previous constructions of composite-order bilinear groups (as generators of G_1 and G_2 are both needed to represent G), we believe that the advantages outweigh the disadvantages, as we are able to both move away from supersingular curves and provide additional structure in the hope of supporting novel cryptographic constructions. Towards this end, we initiate the study of such constructions by first considering in Section 4 plausible hardness assumptions in our setting; in particular, we observe that both the subgroup decision (SGH) assumption and the SXDH assumption (i.e., the assumption that DDH is hard in both G_1 and G_2) appear to hold, and then propose additional new assumptions specifically tailored for arguing security of schemes in the new setting.

Using the old and new assumptions, we then describe in Section 5 an extended version of the BGN cryptosystem that allows for a wide range of flexibility in its applications, as well as an extension of the Groth-Ostrovsky-Sahai NIZK to our setting in Section 6, and new IBE, signature, and encryption schemes in Section 7. We expect that our schemes will provide roadmaps for translating other schemes that use composite-order bilinear groups into our setting; the extra structure available in our setting may then allow the translated schemes to be extended with additional properties.

Related work. As described above, two lines of work seek to expand and generalize our understanding of pairing-friendly elliptic curves.

The first line considers bilinear groups of composite order. This setting was introduced by Boneh, Goh, and Nissim in 2005 [15] and subsequently used for many cryptographic constructions. In this setting, the bilinear group G decomposes into its prime-order subgroups. This fact has been exploited to provide useful structure for cryptography: it allows one to *project* elements from G into one of the subgroups (as used, e.g., by the Boneh-Goh-Nissim encryption scheme), and to *cancel* elements from the two subgroups (as used, e.g., in the Boyen-Waters group signature [19], the traitor tracing scheme due to Boneh, Sahai, and Waters [18], and the Lewko-Waters HIBE [44]).

Subsequent work has considered whether composite-order bilinear groups have a strictly richer structure than prime-order bilinear groups. In one such paper, Freeman [29] showed how to translate schemes that used either one of the projecting or canceling properties (mentioned above) from the composite- to prime-order setting. Lewko [41] showed how to extend the methods of Freeman to work for a wider class of schemes, while Meiklejohn et al. [47] gave evidence that such translations might not always be possible, in particular in the case in which a scheme requires both the projecting and canceling properties. This evidence was later invalidated by Seo and Cheon [49] and by Lewko and Meiklejohn [42], who demonstrated that it was in fact possible to achieve, simultaneously, projecting and canceling.

The second line of work studies the mathematical structure of the bilinear group G. On supersingular curves, distortion maps make it possible to define a modified pairing $\hat{e}\colon G \times G \to G_T$ with G of prime order (see [12, Section 5] for details). On ordinary (non-supersingular) curves, such distortion maps are not available, and we take distinct groups G_1 and G_2, each of order N and define an asymmetric pairing $e\colon G_1 \times G_2 \to G_T$. For efficient representation, the group G_1 is chosen so that its elements are defined over a base field, whereas G_2 is defined over a field extension. (The degree of this extension is related to the *embedding degree*, a crucial value with an important effect on implementation efficiency [30].) Galbraith, Paterson, and Smart [31] give a taxonomy of these configuration choices. In their taxonomy, symmetric pairings correspond to "Type 1"; asymmetric pairings are "Type-2" or "Type-3," depending on whether or not a computable endomorphism ψ exists from G_2 to G_1. Recent work by Chatterjee and Menezes [24] observed that any cryptographic scheme using Type-2 pairings and relying on the endomorphism ψ can in fact be implemented using Type 3 pairings, in which such a map does not exist.

Koblitz and Menezes [40] make the case for pairing-friendly elliptic curves with embedding degree $k = 1$. In these curves, the entire N-torsion group G is already defined over the base field, and we can take G_1 and G_2 as any two orthogonal subgroups. Alternatively, we can choose to work directly with the entire group $G = G_1 \oplus G_2$. (Chen, Cheng, and Smart dub this the "Type-4" pairing [25].)

2 Mathematical Preliminaries and Notation

In this section, we provide some mathematical background on elliptic curves and review one of the ordinary composite-order curve constructions due to Boneh et al. [17]. Although we give basic definitions that do not assume any knowledge of algebraic geometry, we note that the results of Boneh et al. may nevertheless be difficult to follow without some background. We refer interested readers to either Silverman [50] or a set of notes produced by IDA/CCR [22] (the latter assumes no mathematical background).

For completeness, we also provide cryptographic security definitions in Appendix A for zero-knowledge proofs, which we construct in Section 6, and for anonymous identity-based encryption, which we construct in Section 7 (but we note here that they are just the standard definitions for these notions).

2.1 Mathematical background

We start by giving some basic definitions for elliptic curves that will be used in this section and Section 3.1; for the rest of the paper, however, no knowledge of elliptic curves is needed.

In the three definitions that follow, we use the general notion of an elliptic curve E over a field K (although we do assume for ease of exposition that E is non-singular and the characteristic of K is not 2 or 3); as we only ever use $K = \mathbb{F}_q$ for a prime power q, this can be kept in mind for concreteness.

Definition 2.1. *An* elliptic curve *defined over a field K such that char$(K) \neq 2, 3$ is the set of solutions in the projective plane $\mathbb{P}^2(\bar{K})$ to an equation of the form $y^2 = x^3 + ax + b$ for $a, b \in K$ and such that $x^3 + ax + b$ has three distinct roots (i.e., the curve is* non-singular*), together with a point at infinity $\mathcal{O}$.*

One of the most useful features of elliptic curves is that, with the addition of this point at infinity, the set of points on the curve E form a group, using addition as the group operation; this means that for any $P, Q \in E$, there is a unique point $R \in E$ such that $P + Q = R$. This group also turns out to have useful subgroups, such as the group of *K-rational points* and the N-torsion group, both of which we define here:

Definition 2.2. *If E is an elliptic curve defined over a field K, then for a field $K' \subseteq K$ we define the* K'-rational points *of E, written as $E(K')$, to be the points whose coordinates lie in K'.*

Definition 2.3. *For an elliptic curve E defined over a field K, the* N-torsion group *of E, written as $E[N]$,*[1] *is the set $\{P \in E(K) \mid N \cdot P = \mathcal{O}\}$.*

Definition 2.4. **[17]** *If q is a prime power, E is an elliptic curve over $\mathbb{F}_q$, and N is a divisor of $|E(\mathbb{F}_q)|$ such that N is relatively prime to q, then the* embedding degree *of E with respect to N is the smallest positive integer k such that $N \mid q^k - 1$.*

Finally, in terms of cryptographic constructions, one of the most useful features of elliptic curves has been their ability to support a *pairing*, or a map $e : G \times G \to G_T$ (where G is typically equal to $E[N]$ for some N, either prime or composite). In the rest of the paper, we focus exclusively on the Weil pairing. In addition to the usual notions of bilinearity and non-degeneracy that a pairing satisfies, the Weil pairing has an additional property, *alternating*, that we will be able to exploit later on in our cryptographic constructions.

Definition 2.5. **[48]** *The* Weil pairing *on an elliptic curve E is a map $e : E[N] \times E[N] \to \mu_N$ (where μ_N are the N-th roots of unity) with the following properties:*

1. *Bilinearity. If $P, Q, R \in E[N]$ then $e(P+Q, R) = e(P, R)e(Q, R)$ and $e(P, Q+R) = e(P, Q)e(P, R)$.*
2. *Alternating. If $P \in E[N]$ then $e(P, P) = 1$. This, along with bilinearity, implies that if $P, Q \in E[N]$ then $e(P, Q) = e(Q, P)^{-1}$, which is usually called* skew-symmetry.
3. *Non-degeneracy. If $\mathcal{O} \neq P \in E[N]$, there exists $Q \in E[N]$ such that $e(P, Q) \neq 1$.*

2.2 The construction of $E[N]$

As discussed in the introduction, we are interested in exploring the rich structure provided by one of the settings proposed by Boneh, Rubin, and Silverberg [17, Section 4], in which the group $G := E[N]$ decomposes into two subgroups, G_1 and G_2, both of which are distortion free. As mentioned, this setting is especially useful for cryptography, as DDH can be assumed to hold in both groups; in addition, we will see that this property allows us to construct projection maps from G into both G_1 and G_2 that require a trapdoor to compute. Finally, the construction has the additional advantage that, because the resulting curve has embedding degree $k = 1$, it is optimally efficient [30].[2]

In order to construct these ordinary composite-order curves, we follow a slightly modified version of the algorithm of Boneh et al., presented in Algorithm 1 for reference. As noted by Boneh et al., the construction guarantees that $N \mid (q - 1)$, so that the embedding degree is indeed $k = 1$. In addition, using a result of Balasubramanian and Koblitz [3, Theorem 1] and the fact that $N^2 \mid (q - 1)$, we have that $G := E[N] \subseteq E(\mathbb{F}_q)$; as G therefore has exponent N but contains N^2 points, we know that it can be represented as the product of two cyclic groups of order N. While there are many choices for this decomposition, we focus on one in particular, guided by the following proposition:[3]

[1]This is a slight abuse of notation; in fact what we mean is the group $E(K)[N]$.

[2]This is because we're using *ordinary* elliptic curves; if we instead work with supersingular curves, curves with $k = 2$ would be the most efficient.

[3]This is a heavily pared-down version of the original theorem, as it focuses only on the one case in which we are interested.

Algorithm 1 Find a prime q and a curve E over $\mathbb{F}_q$ such that $E[N] \subseteq E(\mathbb{F}_q)$

```
Input: a positive integer $N$

1. Choose a positive integer $D$ suitable for the CM method [7, Chapter VIII], and set $k = 1$.

2. Define $q := 1 + Dk^2N^2$.

3. If $q$ is prime, use the CM method to obtain an elliptic curve $E$ over $\mathbb{F}_q$ with discriminant $D$, such
   that $E(\mathbb{F}_q)$ has $q - 1$ points. If $q$ is not prime, set $k = k + 1$ and repeat Step 2.
```

Proposition 2.6. [23, Theorem 2.1] *Suppose N is a positive integer, q is a prime, and E is an ordinary elliptic curve over $\mathbb{F}_q$ such that $E[N] \subseteq \mathbb{F}_q$. Define $\mathcal{O} = End(E)$ to be an order in some imaginary quadratic field K. For each prime divisor p of N such that $p \nmid [\mathcal{O}_K \colon \mathcal{O}]Disc(K)$, if p is furthermore split in $\mathcal{O}_K$ then all but two subgroups of $E[p]$ have distortion maps; it follows that if all prime divisors p of N satisfy these properties, then all but two subgroups of $E[N]$ have distortion maps.*

This proposition guarantees that, if we choose N and E correctly, there exist distortion-free subgroups G_1 and G_2 of the N-torsion group G. To actually construct these subgroups (again, following the construction of Boneh et al.), we observe that the CM method produces a curve with endomorphism ring $\text{End}(E) = \mathcal{O}_K$ for $K = \mathbb{Q}(\sqrt{-D})$. We therefore start by picking N so that all of its prime divisors meet the two properties required; namely, for all $p \mid N$, $p \nmid \text{Disc}(K)$ and p is split in $K/\mathbb{Q}$. Next, we alter Step 1 of Algorithm 1 to require not only that D is suitable for the CM method but also that $\gcd(N, 2D) = 1$. We then compute a square root of $-D$ mod N and call it s; note that this is the only step of the setup process that requires knowing the factorization of N. After completing the algorithm and obtaining the elliptic curve E, we then define $\sigma \in \text{End}(E)$ to be the action of $\sqrt{-D}$ (see, e.g., Galbraith and Rotger [32, Algorithm 1] for a way to compute this efficiently) and pick some point $P \in G$ (recall $G = E[N]$). We then define the points $P_1 := ((\sigma + s)/2s)P$ and $P_2 := (-(\sigma - s)/2s)P$,[4] and define G_1 and G_2 to be the groups generated by these points. If both P_1 and P_2 have exact order N (which will occur with high probability, but if not we can use a different P and try again), then P_1 and P_2 jointly generate all of G; furthermore, we have that $P_1 + P_2 = ((\sigma + s)/2s)P + (-(\sigma - s)/2s)P = (2s/2s)P = P$, and so we see that $G = G_1 + G_2$.

In summary, this process gives rise to a Setup algorithm that we use throughout the rest of the paper.

- $\mathsf{Setup}(1^k)$: Use the algorithms above to generate a prime q, a composite N, and an elliptic curve E defined over $\mathbb{F}_q$ such that $G := E[N]$ contains N^2 points. Define P_1 and P_2 as above, and set $G_1 := \langle P_1 \rangle$ and $G_2 := \langle P_2 \rangle$. Finally, define e to be the Weil pairing and $G_T := \mu_N$. Output $(N, G_1, G_2, G_T, e, P_1, P_2)$.[5]

In the next two sections, we argue that this setting provides useful structural properties, and that certain cryptographic assumptions hold; to do this, we crucially rely on two properties: the alternating property of the Weil pairing, and the fact that G_1 and G_2 are both distortion free. For the latter, we see that, simply by construction, G_1 and G_2 are in fact the only two distortion-free subgroups of G. Intuitively, this holds because, while the map σ would work as a distortion map for any other subgroups of G, G_1 and G_2 are in fact the eigenspaces of σ and so are the two distortion-free subgroups of G. More formally, we prove the following lemma:

[4]We deviate slightly here from Boneh et al., who use $P_1 := (\sigma + s)P$ and $P_2 := (\sigma - s)P$, and we use the notation $\frac{1}{2s}$ to mean $(2s)^{-1}$ mod N. As we will see in the next section, our choice of points allows us to achieve useful projection properties that it seems difficult to achieve with their points.

[5]Later on, when we switch to multiplicative notation, we write P_1 and P_2 as g_1 and g_2 respectively.

Lemma 2.7. *If $P_1 := \frac{\sigma+s}{2s}P$, $P_2 := \frac{s-\sigma}{2s}P$, $G_1 := \langle P_1 \rangle$, and $G_2 := \langle P_2 \rangle$, then both G_1 and G_2 are distortion free.*

Proof. Consider any endomorphism $\phi \in \mathrm{End}(E)$; because $\mathrm{End}(E) = \mathcal{O}_K$ for $K = \mathbb{Q}(\sqrt{-D})$ we know that ϕ lies in a space generated by σ (i.e., the action of $\sqrt{-D}$) and the identity map 1. But, because $(\sigma - s)P_1 = ((\sigma-s)(\sigma+s)/2s)P = ((\sigma^2 - s^2)/2s)P = ((-D-(-D))/2s)P = 0$ we know that $\sigma(P_1) = sP_1$, which is in G_1 because P_1 generates G_1, so we have that $\sigma(G_1) = G_1$. By the same logic (just using P_2 in place of P_1), we see that $\sigma(P_2) = -sP_2$ and thus $\sigma(G_2) = G_2$. Similarly, it trivially holds that both G_1 and G_2 are preserved under the identity endomorphism. Therefore, it must be the case that G_1 and G_2 are also preserved under ϕ, as ϕ lies in a space generated by 1 and σ, so G_1 and G_2 must be distortion free. □

In addition, we would like to ensure that the alternating property is not specific to the Weil pairing, as this ensures that an adversary cannot succeed in gaining extra information about group elements (for example, if they are in all of G_1 or just a subgroup) by using a different pairing. We therefore show that the Tate pairing is alternating in our setting as well.

Lemma 2.8. *Let σ be the action of $\sqrt{-D}$; i.e., the map $\sigma \in End(E)$ such that $\sigma^2 + D = 0$. Let G_1 be an eigenspace of prime order r such that $\sigma(P) = sP$ for all $P \in G_1$, where $s^2 + D \equiv 0 \bmod r$. Then if $\gcd(r, 2D) = 1$ and e is the Tate pairing, $e(P,P) = 1$ for all $P \in G_1$.*

Proof. Theorem IX.9(4) of Blake, Seroussi, and Smart [7] tells us that for all $P, Q \in G_1$, $e(\sigma(P), \sigma(Q)) = e(P,Q)^{\deg(\sigma)}$. In our setting we have $\deg(\sigma) = D$, so $e(\sigma(P), \sigma(P)) = e(P,P)^D$. We also have that $e(\sigma(P), \sigma(P)) = e(sP, sP) = e(P,P)^{s^2} = e(P,P)^{-D}$, and thus $e(P,P)^{-D} = e(P,P)^D$. As $D \neq 0, 1$, this implies that $e(P,P) = 1$. □

3 The Structure of G

As discussed in the introduction, two of the most useful features of composite-order bilinear groups are (1) the ability to *project* into specific subgroups, and (2) the ability to *cancel* elements from different subgroups. In this section, we fully characterize the structure of the group G just constructed in Section 2.2 and use it to demonstrate how both these properties can be used; as we will see, we are able to project both from G into G_1 and G_2 and from G_1 and G_2 into their respective subgroups, as well as cancel elements in a number of interesting ways.

For the rest of the paper, we focus exclusively on the case when $N = pq$ (i.e., the product of two primes), but note that all of our subsequent arguments generalize to the case when N is a product of arbitrarily many primes as well. For the case when $N = pq$, we just saw in the previous section how to construct the N-torsion group G so as to have a decomposition into two subgroups G_1 and G_2, where G_1 and G_2 are of order N; then we know by the structure theorem for finite abelian groups that they both further decompose into their respective subgroups of order p and order q, so that we end up with a "tree" group structure as seen in Figure 1. As we discuss both the decomposition of G into G_1 and G_2 and the decomposition of G_1 and G_2 into their respective subgroups, we refer to the decomposition of G into G_1 and G_2 as the "first level" of decomposition, and the further decompositions of G_1 and G_2 as the "second level" of decomposition.

3.1 Projection at the first level of decomposition

We now consider how to map from G into its subgroups G_1 and G_2; in addition to wanting to be able to map into these subgroups, as observed above, these maps are most useful when they *project* into the subgroups. We define this type of map formally as follows:

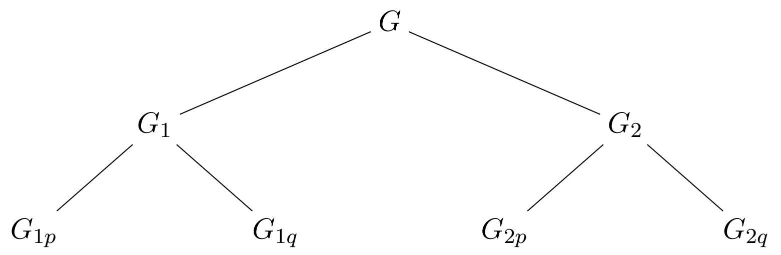

Figure 1: The structure of the N-torsion group G for the case when $N = pq$ (for the more general case when $N = p_1 \dots p_n$, we end up with an n-ary split at the second level). Each subgroup of G is cyclic and we refer to its generator using the same subscript as the group name; for example, g_{1p} generates G_{1p} and g_{2q} generates G_{2q}. We furthermore have a collection of f maps that project from G down to a corresponding subgroup; again, each map is named by the subgroup to which it maps, so we have, for example, $f_{1p} : G \to G_{1p}$ and $f_2 : G \to G_2$.

Definition 3.1 (Projection map.)**.** *We say that a function $f : G \to G'$ is a* projection map *if it is (1) efficiently computable, (2) idempotent; i.e., $f^n(x) = f(x)$ for all $x \in G$ and $n \in \mathbb{N}$, and (3) $G' \subset G$; i.e., f maps into a strict subset of G. If furthermore f is assumed to be hard to compute without the knowledge of some additional piece of information, then we say that it is a* trapdoor projection map*.*

In order to construct our trapdoor projection maps we use the values σ and s chosen in Section 2.2; recall that $\sigma \in \mathrm{End}(E)$ was the action of $\sqrt{-D}$, while s was a square root of $-D$ modulo N. We first repeat an observation used in the proof of Lemma 2.7; namely that $(\sigma+s)(\sigma-s) = \sigma^2 - s^2 = -D-(-D) = 0$. We also recall that we chose $P_1 := ((\sigma + s)/2s)P$ and $P_2 := ((s - \sigma)/2s)P$, and then set $G_1 := \langle P_1 \rangle$ and $G_2 := \langle P_2 \rangle$. Combining this with our observation, we see that σ is equivalent to s in G_1 and equivalent to $-s$ in G_2; intuitively then, G_1 and G_2 cannot have any elements in common. More formally, we prove the following lemma:

Lemma 3.2. *If $P_1 := \frac{\sigma+s}{2s}P$, $P_2 := \frac{s-\sigma}{2s}P$, $G_1 := \langle P_1 \rangle$, and $G_2 := \langle P_2 \rangle$, then $G_1 \cap G_2 = \{\mathcal{O}\}$.*

Proof. Take some element $u \in G_1 \cap G_2$; then we know we can write $u = \left(\frac{\sigma+s}{2s}\right) aP = \left(\frac{s-\sigma}{2s}\right) bP$ for some $a, b \in \mathbb{Z}/N\mathbb{Z}$. Then $(\sigma + s)u = \frac{(\sigma+s)(-(\sigma-s))aP}{2s} = \frac{0(-aP)}{2s} = \mathcal{O}$, and likewise $(\sigma - s)u = \frac{(\sigma-s)(\sigma+s)aP}{2s} = \frac{0aP}{2s} = \mathcal{O}$. We therefore have that $\sigma(u) + su = \mathcal{O}$ and that $\sigma(u) - su = \mathcal{O}$; subtracting this second equation from the first gives us that $2su = \mathcal{O}$, which implies (by the choice of s) that $u = \mathcal{O}$. □

As an initial attempt at defining our projection maps, we consider the maps defined as $f_1(x) := (\sigma+s)x$ and $f_2(x) := (\sigma - s)x$. We also use the fact that $G = G_1 \oplus G_2$ to write any element $u \in G$ uniquely as $u = aP_1 + bP_2$ for $a, b \in \mathbb{Z}/N\mathbb{Z}$, so that $f_1(u) = (\sigma+s)(aP_1+bP_2) = (\sigma+s)\left(\frac{\sigma+s}{2s}\right) aP + (\sigma+s)\left(\frac{\sigma-s}{2s}\right) bP = \frac{(\sigma+s)(\sigma+s)}{2s} aP = (\sigma + s)aP_1$. We therefore have that $f_1(G) = G_1$; we can similarly show that $f_2(G) = G_2$, so the third required property in Definition 3.1 is met. These maps are not, however, idempotent, as they do not leave their respective components unchanged (so, as seen above, f_1 alters the G_1 component in addition to eliminating the G_2 component, and f_2 behaves analogously). To meet this requirement, we alter our maps and define $f_1(x) := ((\sigma + s)/2s)x$; we then have that

$$\begin{aligned}
f_1(u) &= \left(\frac{\sigma+s}{2s}\right)(aP_1 + bP_2) = \frac{\sigma+s}{2s}\left(\frac{(\sigma+s)}{2s}aP + \frac{(\sigma-s)}{2s}bP\right) = \frac{(\sigma+s)(\sigma+s)}{2s \cdot 2s}aP + \frac{(\sigma+s)(\sigma-s)}{4s^2}bP \\
&= \frac{(\sigma^2 + 2\sigma s + s^2)}{2s \cdot 2s}aP + \frac{0}{4s^2}bP = \frac{(2s^2 + 2\sigma s)}{2s \cdot 2s}aP = \frac{(2s(\sigma+s))}{2s \cdot 2s}aP = \frac{(\sigma+s)}{2s}aP = aP_1,
\end{aligned}$$

so that f_1 gets rid of the G_2 component while leaving the G_1 component unchanged. If we define $f_2(x) := (-(\sigma - s)/2s)x$ we can go through a similar derivation to see that f_2 cancels the G_1 component and leave the G_2 component unchanged, meaning we achieve idempotence with both maps, as $f_1^n(u) = aP_1$ and $f_2^n(u) = bP_2$ for all $n \in \mathbb{N}$. In addition, recall from Section 2.2 that the value s used in computing

f_1 and f_2 is a square root modulo N, which should typically be hard to compute without knowledge of the factorization of N. Both maps therefore seemingly require this extra information in order to be efficiently computable and thus, under the assumption that s should be hard to compute given only N, both f_1 and f_2 are trapdoor projection maps.

3.2 Projection at the second level of decomposition

Because we no longer need to discuss the underlying mathematics, we now switch from additive to multiplicative notation, so that we refer to P_1 and P_2 from the previous section as g_1 and g_2 respectively, which means that any point in G can now be written as $g_1^a \cdot g_2^b$ for some $a, b \in \mathbb{Z}/N\mathbb{Z}$. As before, we focus our exposition on the case when $N = pq$ (i.e., N is an RSA modulus), although our subsequent arguments generalize to the case when N is a product of arbitrarily many primes.

As we did in the previous section with f_1 and f_2, we can attempt to construct trapdoor projection maps from G_1 into G_{1p} and G_{1q}, and similarly for G_2. Fortunately, such maps have already been used in the cryptographic literature (for example as the secret key in the Boneh-Goh-Nissim encryption scheme [15]). To project from G_1 into G_{1p}, we can use the value $\lambda_p \equiv \begin{cases} 1 \bmod p \\ 0 \bmod q, \end{cases}$ which is computable using the Chinese Remainder theorem and the factorization of N. The map $f_p : G_1 \to G_{1p}$ is then defined as $f_p(x) := x^{\lambda_p}$; as with f_1 and f_2, we can easily see that this map leaves the G_{1p} component unchanged while canceling out the G_{1q} component, so that it is also idempotent. We can similarly define $\lambda_q \equiv 0 \bmod p$ and $\equiv 1 \bmod q$ to obtain a map $f_q : G_1 \to G_{1q}$ that is defined as $f_q(x) := x^{\lambda_q}$. As these maps depend only on p and q, they work equally as well for G_2 as for G_1, so we also have $f_p : G_2 \to G_{2p}$ and $f_q : G_2 \to G_{2q}$. Note that knowledge of either λ_p or λ_q reveals the factorization of N as, e.g., $\gcd(N, \lambda_p) = q$.

We can also define maps from the full group G into the second-level subgroups; as we already have maps from G down to G_1 and G_2 and we have just constructed maps from G_1 and G_2 into their respective subgroups, however, we can define these maps as the composition of our existing maps. This means $f_{1p} := f_p \circ f_1$, $f_{1q} := f_q \circ f_1$, $f_{2p} := f_p \circ f_2$, and $f_{2q} := f_q \circ f_2$.

3.3 Cancellation of elements within G

Finally, we turn to how elements in different subgroups cancel with each other when paired; we give a definition of this property, closely related to the definition of r-cancelling given by Freeman [29, Definition 3.5], as follows:

Definition 3.3. *For a pairing $e : G \times G \to G_T$ and two subgroups $A, B \subset G$, we say that A and B are* canceling *if for all $a \in A$ and $b \in B$ it holds that $e(a, b) = 1$.*

To see what kind of cancellation we achieve, we begin at the first level of decomposition. Here, we observe that the alternating property of the bilinear map, combined with the fact that G_1 and G_2 are both cyclic, implies that $e(a, b) = 1$ for any $a, b \in G_1$ or $a, b \in G_2$. G_1 and G_2 therefore cancel with themselves, which makes the pairing functionally asymmetric. Incorporating the second level of decomposition, we also see that $e(G_{1p}, G_{2q}) = 1$, as an element of order p in G_1 is of the form $a = g_1^{q\alpha}$ for some $\alpha \in \mathbb{Z}/N\mathbb{Z}$ and an element of order q in G_2 is of the form $b = g_2^{p\beta}$ for some $\beta \in \mathbb{Z}/N\mathbb{Z}$; pairing these two yields $e(a, b) = e(g_1^{q\alpha}, g_2^{p\beta}) = e(g_1, g_2^{pq\alpha\beta}) = e(g_1, g_2^{N\alpha\beta}) = 1$.A similar argument shows that $e(G_{1q}, G_{2p}) = 1$ as well, meaning the only two pairs of subgroups that aren't canceling at the second level are G_{1p} and G_{2p}, and G_{1q} and G_{2q}.

It turns out that these canceling properties, in addition to being useful on their own, can be further exploited when combined with our projection maps. To see this, we note that by the definition of a projection map, $u = f_1(u) \cdot f_2(u)$ for any $u \in G$. This means that, as an example,

$$e(f_1(u), v) = e(f_1(u), f_1(v)f_2(v)) = e(f_1(u), f_2(v)) \cdot 1 = e(f_1(u), f_2(v))e(f_2(u), f_2(v)) = e(u, f_2(v)),$$

so that $e(f_1(u), v) = e(u, f_2(v))$ for all $u, v \in G$. In particular then, we have $e(f_1(u), g) = e(u, g_2)$, which we exploit in our constructions in Section 5 and Section 7. We also observe a similar relation at the second level, where $e(f_{1p}(u), g) = e(u, g_{2p})$ for any $u \in G$ (and similarly $e(f_{2q}(u), g) = e(u, g_{1q})$, etc.), as elements in every subgroup except G_{2p} cancel when paired with an element in G_{1p}.

4 Translating Assumptions into Our Setting

As discussed earlier, all previous cryptographic constructions using composite-order bilinear groups were constrained to use only supersingular curves; i.e., curves in which a distortion map always exists from G_2 to G_1. Among other things, this means that, for all previous composite-order settings, the f_1 and f_2 maps defined in Section 3.1 are always easy to compute and thus no assumption can be made about their hardness. For this reason, all previous constructions focused on a subgroup of the N-torsion group, as using the entire group would serve only to make the constructions less efficient. As our setting *does* use the entire N-torsion group G (and therefore does come with the caveat that it is less efficient), it generalizes these previous settings, as G_1 and G_2 together represent a composite-order bilinear group as used in previous constructions.

The rich structure provided by this generalization not only allows us to translate existing assumptions and schemes into our setting, but also provides a degree of flexibility in the process of this translation. To demonstrate this flexibility, we focus on two mild and well-established assumptions, SXDH [4] and SGH [15], and discuss the various flavors that these assumptions can take on after being translated into our setting. We then suggest two additional assumptions that capture the fact that f_1 and f_2 should be hard to compute.

Symmetric External Diffie Hellman (SXDH) Looking back at Section 2, and at Lemma 2.7 in particular, we remind ourselves that G_1 and G_2 are both subgroups in which no distortion maps exist; i.e., there are no maps $\phi \in \mathrm{End}(E)$ such that $\phi(u) \notin G_1$ for $u \in G_1$ (and similarly for G_2). In particular, this means that all known attacks on DDH do not apply to these subgroups, and so we can reasonably (or at least, as reasonably as possible) assume that the SXDH assumption holds for G_1 and G_2. As a reminder, this assumption says that DDH holds in both G_1 and G_2; i.e., given g_1 and g_2, it is hard to distinguish both (g_1^a, g_1^b, g_1^{ab}) from (g_1^a, g_1^b, g_1^c) and $(g_2^\alpha, g_2^\beta, g_2^{\alpha\beta})$ from $(g_2^\alpha, g_2^\beta, g_2^\gamma)$ for random $a, b, c, \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$.

Subgroup hiding at the second level Perhaps the most popular assumption used for composite-order bilinear groups has been the Subgroup Hiding assumption (SGH for short), which says that for a bilinear group $G = G_p \times G_q$, a random element of G_q is indistinguishable from a random element of G. In our setting, this means that a random element of G_{1q} should be indistinguishable from a random element of G_1, and that the same should be true in G_2. This is only true, however, as long as all the generators at the third level of the tree are kept hidden; if one knows, for example, g_{2p}, then breaking SGH in G_1 is simple, as $e(h, g_{2p}) = 1$ if and only if $h \in G_{1q}$.

We also easily see that, analogously, solving SGH in G_2 becomes easy when given g_{1p}. Perhaps somewhat surprisingly, however, if you are given g_{1p} rather than g_{2p} then you are no closer to solving SGH in G_1; because the Weil pairing is alternating, $e(h, g_{1p}) = 1$ regardless of whether or not h is in all of G_1.

Using this observation, we define three variants of the SGH assumption. The first, *simultaneous SGH*, says that SGH is hard in both G_1 and G_2, which as we just saw means that we can reveal only the generators g_1 and g_2. The second, *one-sided SGH*, says that if we reveal the generators g_{1p} and g_{1q} then SGH still holds in G_1 but no longer holds in G_2 (and similarly for G_1 if we reveal the generators g_{2p} and g_{2q}). Finally, we use the first level of the tree as well, and consider *extended SGH* in either G_1 or G_2, which says respectively that given an element $u \in G$, it is hard to tell whether the G_{1p} or G_{2p} component of u is empty (i.e., $f_{1p}(u) = 1$ or $f_{2p}(u) = 1$) or not.

Assumption 4.1 (Extended SGH)**.** *Given the group setting $(N, G_1, G_2, G_T, e, g_1, g_2)$ and an element $u \in G$, it is hard to tell if $f_{1p}(u) = 1$ or not.*

As with simultaneous SGH, deciding if the G_{1p} component is empty becomes easy if we are given g_{2p}, and deciding the G_{2p} component is easy if we are given g_{1p}. We also easily show that SGH in G_1 implies extended SGH in G_1 (and analogously for G_2) as follows:

Lemma 4.2. *If SGH is hard in G_1, then extended SGH is also hard in G_1.*

Proof. To show this, we take an adversary $\mathcal{A}$ that can break extended SGH in G_1 with some non-negligible advantage ϵ and use it to construct an adversary $\mathcal{B}$ that can break SGH in G_1 with the same advantage. As input, $\mathcal{B}$ receives the setting $(N, G_1, G_2, G_T, e, g_1, g_2)$ and an element $h \in G_1$. It then picks a random element $v \xleftarrow{\$} G_2$ and gives to $\mathcal{A}$ the setting and the element $h \cdot v$. $\mathcal{B}$ then guesses the same thing that $\mathcal{A}$ does. As h is random (modulo having an empty G_{1p} component or not) and v is random, the element $\mathcal{B}$ gives to $\mathcal{A}$ is distributed identically to the input that it expects. In addition, as the winning conditions for $\mathcal{A}$ and $\mathcal{B}$ are identical, $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does, and thus $\mathcal{B}$ succeeds with advantage ϵ. $\square$

We also consider one slightly stronger version of simultaneous SGH, in which we require the candidate values in G_1 and G_2 to have the same discrete logs (with respect to g_1 and g_2 respectively). We call this *duplicate SGH* and state it formally here as:

Assumption 4.3 (Duplicate SGH)**.** *Given the group setting $(N, G_1, G_2, G_T, e, g_1, g_2)$, $h_1 := g_1^\alpha$, and $h_2 := g_2^\alpha$ for $\alpha \in \mathbb{Z}/N\mathbb{Z}$, it is hard to tell whether h_1 is a random element of G_{1q} and h_2 is a random element of G_{2q}, or h_1 is a random element of G_1 and h_2 is a random element of G_2.*

As we will see in Section 6, this assumption can be used to instantiate a zero-knowledge version of Groth-Sahai proofs [36] that is closely related to their instantiation under SGH.

Subgroup hiding at the first level As the first level of decomposition is new to our setting, we now present two assumptions designed to capture the hardness of computing f_1 and f_2 (as they were constructed to be trapdoor projection maps). The first assumption we suggest is very simple: assume that f_1 and f_2 are hard to compute! (One can always compute both or neither, as $f_2(u) = u/f_1(u)$.) Formally, we have:

Assumption 4.4. *Given the group setting $(N, G_1, G_2, G_T, e, g_1, g_2)$ and random $u \xleftarrow{\$} G$ for $G := G_1 \times G_2$, it is hard to compute $f_1(u)$.*

In addition to this computational assumption, for many applications (including our construction of an anonymous IBE in Section 7) we would like a decisional variant. In G, however, there is no clear analogue to an assumption like SGH (which, as we saw, can essentially be thought of as the decisional version of the assumption that f_p is hard to compute), as the generators g_1 and g_2 are presumed to be known. While we could attempt to hide these generators, this would significantly reduce the functionality of our setting and we thus choose to instead move to the target group G_T. Here, one decisional assumption that we suggest is that given $u, v \in G$, it should be hard to distinguish $e(f_1(u), f_2(v))$ from random; note that this does indeed become easy to solve given f_1, as one can compute $e(f_1(u), v) = e(f_1(u), f_2(v))$ directly.

Assumption 4.5. *Given the group setting $(N, G_1, G_2, G_T, e, g_1, g_2)$, an element $T \in G_T$, and random $u, v \xleftarrow{\$} G$ for $G := G_1 \times G_2$, it is hard to tell if $T = e(f_1(u), f_2(v))$ or T is random.*

As the unique structure of our bilinear group makes it far from generic, proofs that these new assumptions hold in the generic group model do not seem to carry much weight. Instead, we observe that computing the map f_1 seemingly requires (from Section 3.1) knowledge of a non-trivial square root of $-D$ modulo N, just as computing the projection map into p-order subgroups (as is implicitly used in ordinary SGH) seemingly requires knowledge of the factorization of N. We therefore lend credence to these assumptions only by observing their close analogue with the corresponding assumptions for SGH, and the fact that the projection maps at the second level (i.e., the ones used for SGH) are at least as hard to compute as the projection maps at the first level.

5 An Extended Version of the BGN Encryption Scheme

In the previous section, we saw the added flexibility that can be gained when translating assumptions such as SGH into our setting. In this section, we demonstrate that flexibility can also be gained when translating existing schemes into our setting. As one example, we look at the Boneh-Goh-Nissim (BGN) encryption scheme [15], which we recall allows a user, given a set of ciphertexts, to perform any number of additions and one multiplication on plaintexts. We show that this property is preserved when the scheme is translated into our setting; beyond this, operations on specific plaintexts (for example, multiplying two plaintexts together) may be specifically disallowed by the user at the time of encryption, or even afterwards by an independent third party (i.e., someone who sees only the ciphertexts). To demonstrate why this might be useful, we describe in Appendix B a "restricted" secure two-party computation of a 2-DNF formula based on this extended encryption scheme, along with some potential applications.

5.1 The original BGN encryption scheme

We start with a reminder of the BGN encryption scheme; to keep things simple, we focus solely on the case of bit encryption. Because the scheme is homomorphic, we must describe it in terms of four algorithms: a KeyGen algorithm that outputs the public key pk and secret key sk, an Enc algorithm that takes in a public key pk and message m and outputs a ciphertext c, a Dec algorithm that takes in a secret key sk and ciphertext c and outputs a message m, and an Eval algorithm that, on input a binary operation op and a pair of ciphertexts (c_1, c_2) with respective plaintexts m_1 and m_2, outputs a new ciphertext c containing as a plaintext $\mathsf{op}(m_1, m_2)$.

- $\mathsf{KeyGen}(1^k)$: Generate a bilinear group (N, G, G_T, e, g) such that $N = pq$ for two large primes p and q. Pick a random value $r \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and set $h := g^{rp}$, so that h is a random generator of the q-order subgroup of G. Return $pk := (N, G, G_T, e, g, h)$ and $sk := f_p$.
- $\mathsf{Enc}(pk, m)$: Pick $r \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and return $c := g^m h^r$.
- $\mathsf{Dec}(sk, c)$: Compute $f_p(c)$. If this is equal to $f_p(g)$ or $f_p(e(g, g))$ then return 1; if it is equal to 1 return 0. (To extend beyond bit encryption to small messages m, we would instead compute $f_p(c)$ and then take its discrete log using, for example, Pollard's rho algorithm.)
- $\mathsf{Eval}(pk, \mathsf{op}, c_1, c_2)$: First pick a random value $r \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$. If $\mathsf{op} = +$ and $c_1, c_2 \in G$ then return $c_1 \cdot c_2 \cdot h^r$; otherwise, if $c_1, c_2 \in G_T$ then return $c_1 \cdot c_2 \cdot e(g, h)^r$, and if neither of these cases holds (i.e., c_1 and c_2 are in different groups) return $\perp$.
  If $\mathsf{op} = \times$ and $c_1, c_2 \in G$ then return $e(c_1, c_2) \cdot e(g, h)^r$. In all other cases, return $\perp$.

5.2 Our adapted scheme

In translating the BGN scheme into our setting, we exploit the fact that we have two groups instead of just one; this means that we can have a regular BGN ciphertext in G_1 (or G_2), and potentially add in a "noise" factor (i.e., some randomness) in the other group. This results in five types of ciphertexts: regular BGN ciphertexts in both G_1 and G_2 (in which the noise factor in the other group is just 0), "noisy" BGN ciphertexts in both G_1 and G_2, and ciphertexts in G_T. As we will see, both here and in Appendix B, these noisy ciphertexts make it difficult to multiply the values within; a user can thus decide, at the time of encryption, which operations should and shouldn't be performed on his particular set of ciphertexts.

As our expanded scheme deals with five types of ciphertexts rather than two, we treat the type as an explicit input to the encryption algorithm; for ease of exposition, we also focus solely on bit encryption. Our scheme is as follows:

- $\mathsf{KeyGen}(1^k)$: Use the Setup algorithm from Section 2.2 to produce $(N, G_1, G_2, G_T, e, g_1, g_2)$. Pick random elements $h_1 \xleftarrow{\$} G_{1q}$ and $h_2 \xleftarrow{\$} G_{2q}$, and output $pk := (N, G, G_T, e, g_1, g_2, h_1, h_2)$ and $sk := (f_p, f_1, f_2)$.
- $\mathsf{Enc}(pk, t, m)$: Pick random $r, s \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and return $c := g_1^m h_1^r$ if $t = 1$, $c := g_2^m h_2^r$ if $t = 2$, $g_1^m h_1^r g_2^s$ if $t = 3$, and $g_2^m h_2^r g_1^s$ if $t = 4$.
- $\mathsf{Dec}(sk, c)$: Parse $sk = (f_p, f_1, f_2)$. If $c \in G_T$ then compute $f_p(c)$; if this is equal to $f_p(e(g_1, g_2))$ then return 1 and if it is equal to 1 return 0.
  Otherwise, if $c \in G$ compute $f_{1p}(c)$ and $f_{2p}(c)$. If either is equal to 1 then return 0; otherwise, if the former is equal to $f_p(g_1)$ or the latter is equal to $f_p(g_2)$ then return 1 (and if neither of these equalities hold then output $\perp$).

The IND-CPA security of the scheme follows essentially directly from the IND-CPA security of the BGN encryption scheme; for ciphertexts of types 1 and 2 we can use SGH in G_1 and G_2 respectively, while for ciphertexts of types 3 and 4 we can use Extended SGH (Assumption 4.1) in G_1 and G_2 respectively. As we saw in Lemma 4.2, SGH in G_1 implies Extended SGH in G_1 (and analogously in G_2); we therefore have the following theorem:

Theorem 5.1. *If SGH holds in both G_1 and G_2 then the encryption scheme described above is IND-CPA secure.*

Looking at the decryption algorithm, there is one potential setback when using type 3 and 4 encryption, which is the possibility of a decryption error. For a type-1 ciphertext c, $f_{2p}(c) = 1$ and so c always decrypts perfectly; similarly, for a type-2 ciphertext $f_{1p}(c) = 1$ and decryption is again perfect, and for a type-5 ciphertext (i.e., one in G_T), decryption is perfect as well. Because type-3 and type-4 ciphertexts both inhabit the whole group G, however, we have that $f_{2p}(c) = g_{2p}^s$ for type-3 ciphertexts and $f_{1p}(c) = g_{1p}^s$ for type-4 ciphertexts. In order for a decryption error to occur, it must therefore be the case that $s = 1$ (and beyond bit encryption that s is very small), which for $s \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ happens with negligible probability and thus leads to a negligible decryption error.[6]

Finally, we turn to the homomorphic properties of our extended scheme. Just as in the original scheme, our scheme allows for arbitrarily many additions and at most one multiplication; as we will see momentarily, however, by encrypting values using certain types we may in some cases restrict these operations.

- $\mathsf{Eval}(pk, \mathsf{op}, c_1, c_2)$: First pick random values $r_1, r_2 \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$. If op is $+$ then return (1) $c := c_1 \cdot c_2 \cdot h_1^{r_1}$ if $c_1, c_2 \in G_1$, (2) $c := c_1 \cdot c_2 \cdot h_2^{r_2}$ if $c_1, c_2 \in G_2$, (3) $c := c_1 \cdot c_2 \cdot h_1^{r_1} \cdot h_2^{r_2}$ if $c_1, c_2 \in G$, or (4) $c := c_1 \cdot c_2 \cdot e(h_1, g_2)^{r_1}$ if $c_1, c_2 \in G_T$. If op is instead $\times$ then return $c := e(c_1, c_2) \cdot e(h_1, g_2)^{r_1}$ if $c_1 \in G_1$ and $c := e(c_2, c_1) \cdot e(h_1, g_2)^{r_1}$ if $c_2 \in G_1$.

As mentioned above, the goal of using noisy ciphertexts is to make it easy to perform operations on certain ciphertexts (so as to achieve the functionality provided by a homomorphic encryption scheme), but difficult to perform them on others. To formalize this, we use tables for both of our binary operations

[6]In fact, for bit encryption, we could eliminate this decryption error altogether by checking if (1) $c \in G_1$, (2) $c/g_2 \in G_1$, (3) $c \in G_2$, or (4) $c/g_1 \in G_2$. In the first case we have a type-3 ciphertext using $s = 0$, and in the second case we have $s = 1$; in either case, we can decrypt using only $f_{1p}(c)$. In the latter two cases we have a type-4 ciphertext (using $s = 0$ and $s = 1$ respectively), so can decrypt using only $f_{2p}(c)$.

$+$	1	2	3	4	5
1	1	$\perp$	3/1	$\perp$	$\perp$
2	$\perp$	2	$\perp$	4/2	$\perp$
3	3/1	$\perp$	3/1	$\perp$	$\perp$
4	$\perp$	4/2	$\perp$	4/2	$\perp$
5	$\perp$	$\perp$	$\perp$	$\perp$	5

$\times$	1	2	3	4	5
1	$\perp$	5	$\perp$	5	$\perp$
2	5	$\perp$	5	$\perp$	$\perp$
3	$\perp$	5	$\perp$	$\perp$/**5**	$\perp$
4	5	$\perp$	$\perp$/**5**	$\perp$	$\perp$
5	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

Figure 2: Tables T_+ and $T_\times$ that represent the action of a given binary operation on ciphertext types. For entries where there are two numbers, the first number represents the type that anyone can achieve (e.g., anyone can add two type-1 ciphertexts to produce a type-1 ciphertext), while the second represents the type that a user can achieve only with knowledge of f_1.

as seen in Figure 2. A table entry $T_{\mathsf{op}}(i,j)$ is equal to t if the following conditions hold: for a type-i ciphertext c_1 with corresponding plaintext m_1 and a type-j ciphertext c_2 with corresponding plaintext m_2, it must be the case that (1) $c \xleftarrow{\$} \mathsf{Eval}(pk, \mathsf{op}, c_1, c_2)$ has type t, and (2) $\mathsf{Dec}(sk, c) = \mathsf{op}(m_1, m_2)$. If these conditions hold then $T_{\mathsf{op}}(i,j) = t$, and otherwise $T_{\mathsf{op}}(i,j) = \perp$. An operation op should then be easy to perform on ciphertexts of type t_1 and t_2 if $T_{\mathsf{op}}(t_1,t_2) \neq \perp$, and hard to perform otherwise. As an example, we consider multiplying a type-1 encryption of a message m_1 with a type-4 encryption of a message m_2. Even though the latter is a noisy ciphertext, the table tells us that this should nevertheless produce a type-5 encryption of m_1m_2. To confirm this, we write the type-1 ciphertext as $c_1 = g_1^{m_1}h_1^{r_1}$ and the type-4 ciphertext as $g_2^{m_2}h_2^{r_2}g_1^{s_2}$; we then have

$$\begin{aligned}
\mathsf{Eval}(pk, \times, c_1, c_2) &= e(g_1^{m_1}h_1^{r_1}, g_2^{m_2}h_2^{r_2}g_1^{s_2}) \cdot e(g_1,h_2)^{r'} \\
&= e(g_1^{m_1}, g_2^{m_2})e(h_1^{r_1}, g_2^{m_2}h_2^{r_2}g_1^{s_2})e(g_1^{m_1}, h_2^{r_2}g_1^{s_2})e(g_1,h_2)^{r'} \\
&= g_T^{m_1m_2} \cdot e(h_1,g_2)^{r_1m_2}e(h_1,h_2)^{r_1r_2}e(g_1,h_2)^{m_1r_2}e(g_1,h_2)^{r'} \\
&= g_T^{m_1m_2} \cdot e(g_1,h_2)^{\overbrace{r_1 + \beta r_1r_2 + m_1r_2 + r'}^{r''}} \\
&= g_T^{m_1m_2} \cdot h_T^{r''},
\end{aligned}$$

which is indeed a type-5 encryption of m_1m_2. Looking at another entry in the table, however, we see that multiplying two noisy ciphertexts should be difficult, as $T_\times(3,4) = \perp$. To see this, we attempt to multiply a type-3 ciphertext $c_1 = g_1^{m_1}h_1^{r_1}g_2^{s_1}$ with a type-4 ciphertext $c_2 = g_2^{m_2}h_2^{r_2}g_1^{s_2}$ and get

$$\begin{aligned}
\mathsf{Eval}(pk, \times, c_1, c_2) &= e(g_1^{m_1}h_1^{r_1}g_2^{s_1}, g_2^{m_2}h_2^{r_2}g_1^{s_2}) \cdot e(g_1,h_2)^{r'} \\
&= e(g_1^{m_1}, g_2^{m_2})e(h_1^{r_1}g_2^{s_1}, g_2^{m_2}h_2^{r_2}g_1^{s_2})e(g_1^{m_1}, h_2^{r_2}g_1^{s_2})e(g_1,h_2)^{r'} \\
&= g_T^{m_1m_2} \cdot e(h_1,g_2)^{r_1m_2}e(h_1,h_2)^{r_1r_2}e(g_2,g_1)^{s_1s_2}e(g_1,h_2)^{m_1r_2}e(g_1,h_2)^{r'} \\
&= g_T^{m_1m_2 - s_1s_2} \cdot e(g_1,h_2)^{\overbrace{r_1 + \beta r_1r_2 + m_1r_2 + r'}^{r''}} \\
&= g_T^{m_1m_2 - s_1s_2} \cdot h_T^{r''},
\end{aligned}$$

which decrypts to $m_1m_2 - s_1s_2$ and is therefore information-theoretically independent from the original messages m_1 and m_2. Although multiplying type-3 and type-4 ciphertexts therefore seems to be difficult, we note that with knowledge of f_1 this task in fact becomes easy. To see this, we first observe that any type-1 ciphertext can be easily converted into a type-3 ciphertext (analogously, type 2 into type 4) by simply multiplying in a random value from G_2. With knowledge of f_1 then, a user can reverse this operation and turn a type-3 ciphertext into a type-1 ciphertext by computing $f_1(c)$; knowledge of f_1 therefore allows him to properly multiply the values inside type-3 and type-4 ciphertexts.

6 A NIZK Proof That a Committed Value is a Bit

For composite-order bilinear groups in the symmetric setting (i.e., where $G_1 = G_2$), the techniques for proving that a committed value is a bit are well known; the first zero-knowledge proof was given by Groth, Ostrovsky, and Sahai in 2006 [35] and has been used in a variety of constructions since.

Briefly, in the symmetric setting, if we have a commitment $c = g^b h^r$, where g generates G and h is a random element of G_q, and we call the group element in the commitment x, then we want to prove that $e(x, g^{-1}x) = 1$, as this will demonstrate that one of x or $g^{-1}x$ must be 1, which will happen only in the case that $x = g^b$ for some bit b. In the asymmetric setting, however, this strategy is clearly impossible, as we require an element from G_1 to be paired with an element from G_2, and x can only live in one or the other. We must therefore replicate the commitment in the G_2 subgroup (so assuming $c \in G_1$, construct a commitment $d \in G_2$ to the same value), prove that these are commitments to the same value, and then prove that the common value is a bit. So, if we define x_1 to be the value contained in c and x_2 to be the value contained in d, we see that to prove $x_1 = x_2$ we need to prove satisfiability of the pairing product equation $e(x_1, g_2) \cdot e(g_1, 1/x_2) = 1$. Then, to prove that their common discrete logarithm is either 0 or 1, we prove satisfiability of the equation $e(x_1, x_2/g_2) = 1$.

Fortunately, for proving these types of equations, we can directly use Groth-Sahai (GS) proofs [36] which allow a user to prove satisfiability of any pairing product equation of the form $\prod_i e(x_i, y_i) = 1$ for $x_i \in G_1$ and $y_i \in G_2$. Briefly, this is accomplished by forming commitments c_i to the values x_i and commitments d_i to the values y_i; this can be done by computing $c_i := x_i \cdot \prod_j h_{1j}^{r_j}$ for random r_j and $d_i := y_i \cdot \prod_j h_{2j}^{s_j}$ for random s_j. These values $\{h_{1j}, h_{2j}\}$ are referred to as the "commitment keys" and can be either *binding* or *hiding*; in order to prove witness indistinguishability, the two settings must be assumed to be indistinguishable. After forming the commitments, a proof $\pi = (\{\pi_{1j}\}, \{\pi_{2j}\})$ is formed such that $\prod_i e(c_i, d_i) = e(h_1, \pi_1)e(\pi_2, h_2)$.

As we argued in Section 4 that SXDH can be reasonably assumed to hold in our setting, we already have this proof system at our disposal, as Groth and Sahai provide instantiations under the Subgroup Hiding (SGH), SXDH, and Decision Linear [10] assumptions. In addition, by using the SGH-based construction in both G_1 and G_2, we obtain an instantiation under simultaneous SGH, or the assumption that SGH holds in both G_1 and G_2.

While using Groth-Sahai proofs gives us witness indistinguishability, for some applications (and in particular for our two-party computation in Appendix B) we need full zero knowledge. We therefore demonstrate here how the SGH-based proof system can be boosted to full zero knowledge, using the duplicate SGH assumption (Assumption 4.3) and the same randomization techniques as Groth, Ostrovsky, and Sahai. The protocol runs as follows:

- $\mathsf{CRSSetup}(1^k)$: Use the Setup algorithm from Section 2.2 to produce $(N, G_1, G_2, G_T, e, g_1, g_2)$. Next, compute commitment keys as h_1 and h_2, where h_1 is a random element of G_{1q} and h_2 is a random element of G_{2q}. Output $\sigma_{\text{crs}} := (N, G_1, G_2, G_T, e, g_1, g_2, h_1, h_2)$.
- $\mathcal{P}(\sigma_{\text{crs}}, (c, d), (b, r_1, r_2))$:
  1. Define x_1 to be the value in c and x_2 to be the value in d. Then, using the openings of c and d, form a GS proof for the pairing product equation $e(x_1, g_2)e(g_1, 1/x_2) = 1$; this will yield a proof $\pi'_e := (\pi_{1e}, \pi_{2e})$ such that $e(c, g_2)e(g_1, 1/d) = e(h_1, \pi_{1e})e(\pi_{2e}, h_2)$.
  2. Next, form another GS proof for the equation $e(x_1, x_2/g_2) = 1$. Call this proof $\pi'_b := (\pi_{1b}, \pi_{2b})$.[7]
  3. Now, randomize each proof as follows: for each pair π_1 and π_2, pick random values $r, s \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and compute $\pi_0 := h_1^r$ and $\pi'_1 := \pi_1^{1/r}$, as well as $\pi'_2 := \pi_2^{1/s}$ and $\pi_3 := h_2^s$. Finally, compute $\pi'_0 := g_1^r$ and $\pi'_3 := g_2^s$, and output the proof $\pi := (\pi_0, \pi'_0, \pi'_1, \pi'_2, \pi_3, \pi'_3)$.

[7]Note that if we used these proofs as is we would achieve witness indistinguishability; it is therefore the randomization in the next step that makes these proofs zero knowledge.

4. Output $\pi = (\pi_e, \pi_b)$ (where π_e and π_b are the respective versions of π'_e and π'_b run through Step 3 above).

- $\mathcal{V}(\sigma_{\text{crs}}, (c,d), (\pi_e, \pi_b))$:
  1. For the equality proof, parse $\pi_e = (\pi_0, \pi'_0, \pi'_1, \pi'_2, \pi_3, \pi'_3)$. Then check that
  $$e(\pi_0, g_2) = e(\pi'_0, h_2), \tag{1}$$
  $$e(g_1, \pi_3) = e(h_1, \pi'_3), \tag{2}$$
  and finally that
  $$e(c, g_2)e(g_1, 1/d) = e(\pi_0, \pi'_1) \cdot e(\pi'_2, \pi_3). \tag{3}$$
  2. For the bit proof, parse $\pi_b = (\pi_0, \pi'_0, \pi'_1, \pi'_2, \pi_3, \pi'_3)$. Check that Equations 1 and 2 hold for this set of proofs, and then check
  $$e(c, d/g_2) = e(\pi_0, \pi'_1) \cdot e(\pi'_2, \pi_3). \tag{4}$$
  3. Accept only if both the above checks pass.

Theorem 6.1. *If duplicate SGH holds in G, the above protocol for proving that two commitments open to the same bit is a NIZKPoK that satisfies perfect completeness, perfect soundness, computational zero knowledge, and perfect extractability.*

We show this through the following series of lemmas:

Lemma 6.2. *The above protocol satisfies perfect completeness.*

Proof. To show perfect completeness, we assume that c, d, π_b, and π_e are all formed honestly. For Equation 1, if we define $\beta := \text{dlog}_{g_1}(h_1) = \text{dlog}_{g_2}(h_2)$ then we have that

$$e(\pi_0, g_2) = e(h_1^r, g_2) = e(g_1^{\beta r}, g_2) = e(g_1^r, g_2^\beta) = e(\pi'_0, h_2),$$

so that this check will pass for both π_b and π_e. Similarly, for Equation 2, we have

$$e(g_1, \pi_3) = e(g_1, h_2^s) = e(g_1, g_2^{\beta s}) = e(g_1^\beta, g_2^s) = e(h_1, \pi'_3),$$

so that this check will pass as well. For Equation 3, we can already assume (by the completeness of Groth-Sahai proofs) that we have in place proofs π_1 and π_2 such that $e(c, g_2)e(g_1, 1/d) = e(h_1, \pi_1) \cdot e(\pi_2, h_2)$, which allows us to see that

$$e(c, g_2)e(g_1, 1/d) = e(h_1, \pi_1)^{1/r \cdot r} \cdot e(\pi_2, h_2)^{1/s \cdot s} = e(h_1^r, \pi_1^{1/r}) \cdot e(\pi_2^{1/s}, h_2^s) = e(\pi_0, \pi'_1) \cdot e(\pi'_2, \pi_3),$$

so that this check will pass as well. Using an almost identical derivation for Equation 4 allows us to see that this check will pass as well. □

Lemma 6.3. *The above protocol is perfectly sound.*

Proof. To show perfect soundness of the protocol, we can examine what each of the verification checks proves. If Equation 1 passes verification, then we know that

$$e(\pi_0^q, g_2) = e(\pi_0, g_2)^q = e(\pi'_0, h_2)^q = e(\pi'_0, h_2^q) = 1,$$

so that π_0 must have order q; we can perform a similar derivation with Equation 2 to see that π_3 must also have order q. Looking at Equation 3, we can see that

$$(e(c, g_2)e(g_1, 1/d))^q = (e(\pi_0, \pi_1') \cdot e(\pi_2', \pi_3))^q = e(\pi_0^q, \pi_1') \cdot e(\pi_2', \pi_3^q) = 1.$$

If we write $c = g_1^\gamma$ and $d = g_2^\delta$ for some $\gamma, \delta \in \mathbb{Z}/N\mathbb{Z}$, then we can see that if $e(g_1^{\gamma q}, g_2) \cdot e(g_1, g_2^{-\delta q}) = 1$ then either c and $1/d$ both have order q or $\gamma q = \delta q$; the first case implies that the value contained in both c and d is a 0, but the second implies only that c and d are commitments to the same value. Taking Equation 4 into account as well, however, we see (by the same derivation as above) that $e(c, d/g_2)^q = 1$, so that either c has order q or d/g_2 has order q, which means that either the value contained in c is 0 or the value contained in d is 1; combining this with the information above, we can see that c and d must be commitments to the same value, and that their common value must in fact be a bit, or else these equality checks will not pass. □

Lemma 6.4. *If duplicate SGH holds, the above protocol is zero knowledge.*

Proof. To show zero knowledge, we describe our simulator (S_1, S_2). To start, S_1 will output a CRS σ_{sim} so that h_1 and h_2 are in all of G_1 and G_2 respectively, and a trapdoor $\tau_s := (p, q, \beta)$, where $\beta = \text{dlog}_{g_1}(h_1) = \text{dlog}_{g_2}(h_2)$. The simulator S_2, when given τ_s and some commitments c and d, works as follows: for π_e, recall that we will need proofs that satisfy $e(c, g_2)e(g_1, 1/d) = e(\pi_0, \pi_1') \cdot e(\pi_2', \pi_3)$. To do this, S_2 will first use the factorization of N to determine which of c and $g_1^{-1}c$ is a generator for the full group G_1, and similarly which of d and $g_2^{-1}d$ is a generator for G_2; without loss of generality, assume it is c and d (and if not use $g_1^{-1}c$ and $g_2^{-1}d$ in subsequent arguments). S_2 will then pick random values $r, s \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and compute $\pi_0 := c^r$, $\pi_1' := g_2^{1/r}$, $\pi_2' := g_1^{1/s}$, and $\pi_3 := (1/d)^s$. It can compute as well $\pi_0' := \pi_0^{1/\beta}$ and $\pi_3' := \pi_3^{1/\beta}$; we then have that

$$e(\pi_0, \pi_1') \cdot e(\pi_2', \pi_3) = e(c^r, g_2^{1/r}) \cdot e(g_1^{1/s}, (1/d)^s) = e(c, g_2) \cdot e(g_1, 1/d),$$

and furthermore that

$$e(\pi_0, g_2) = e(\pi_0, g_2)^{\beta/\beta} = e(\pi_0^{1/\beta}, g_2^\beta) = e(\pi_0', h_2)$$

and

$$e(g_1, \pi_3) = e(g_1, \pi_3)^{\beta/\beta} = e(g_1^\beta, \pi_3^{1/\beta}) = e(h_1, \pi_3')$$

so that all the verification checks for π_e will pass. For π_b, S_2 can now pick new random values $r, s \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$, and a random group element $R \xleftarrow{\$} G_1$ (this can be accomplished by just picking a random exponent $t \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and computing $R := g_1^t$). It can then set $\pi_0 := (c/R)^r$, $\pi_0' := \pi_0^{1/\beta}$, $\pi_1' := (d/g_2)^{1/r}$, $\pi_2' := R^s$, $\pi_3 := (d/g_2)^{1/s}$, and $\pi_3' := \pi_3^{1/\beta}$. The checks in Equations 1 and 2 will pass for the exact same reasons as they did for π_e, so we can focus on just the check in Equation 4. This equation will be satisfied just by construction, however, as we have that

$$\begin{aligned}
e(\pi_0, \pi_1') \cdot e(\pi_2', \pi_3) &= e((c/R)^r, (d/g_2)^{1/r}) \cdot e(R^s, (d/g_2)^{1/s}) \\
&= e(c/R, d/g_2)^{r/r} \cdot e(R, d/g_2)^{s/s} \\
&= e(c/R \cdot R, d/g_2) \\
&= e(c, d/g_2).
\end{aligned}$$

Now that we have shown that S_2 can produce proofs that pass verification, we still need to argue that the output of S_1 is indistinguishable from the output of $\mathsf{CRSSetup}$ and that the proofs produced by S_2 are indistinguishable from the proofs produced by the prover. For S_1, it follows directly from duplicate

SGH that σ_{sim}, in which $h_1 \in G_1$ and $h_2 \in G_2$, is indistinguishable from the honest CRS, in which $h_1 \in G_{1q}$ and $h_2 \in G_{2q}$. For S_2, observe that if the commitments c and d are generators for G_1 and G_2, then all the π values will just be random elements in G_1 and G_2, and thus indistinguishable from honest proofs (as honest proofs will also lie in all of G_1 or G_2 in the case that h_1 and h_2 generate all of G_1 and G_2 respectively). As S_2 uses instead $g_1^{-1}c$ and $g_2^{-1}d$ in the case that c and d are in fact not generators, it will always end up with random π values and thus provide proofs that are distributed identically to those produced by the prover. $\square$

Lemma 6.5. *The above protocol is perfectly extractable.*

Proof. To show this, we describe our extractor (E_1, E_2). The algorithm E_1 will run the honest $\mathsf{CRSSetup}$ and set $\tau_e := f_p$; the σ_{ext} output by E_1 is therefore trivially distributed identically to an honest CRS. As for E_2, we know that a proof π will contain a commitment c; by perfect soundness, we know that c will be a commitment to a bit. As we are using the honest CRS and thus have that $h_1 \in G_{1q}$, we know that $f_p(h_1) = 1$ and so $f_p(c) = g_1^b$. Then to extract the bit b, the extractor can compute $f_p(c)$; if this is equal to 1 then it outputs $b = 0$ and otherwise it outputs $b = 1$. $\square$

7 An Anonymous IBE and Derivative Schemes

In this section, we see how to achieve a simple anonymous identity-based encryption (IBE) scheme in our setting. Like the Boneh-Frankin [12], Cocks [26], and Boneh-Gentry-Hamburg [13] IBEs, we use a full-domain hash function, modeled as a random oracle, to map identity strings to group elements. In our case, the target of the hash function is the entire group G. The IBE master secret is the trapdoor projection map f_1, so that the secret key corresponding to an identity id is $f_1(H(id))$. Like the Boneh-Franklin IBE, our IBE is anonymous [1]: given a ciphertext, an adversary cannot determine the identity authorized to decrypt it. Our IBE has useful mathematical structure, and we show how to adapt that structure to obtain a unique signature (in the random oracle model) and a public-key encryption scheme (in the standard model).

7.1 An anonymous IBE scheme

An identity-based encryption scheme (or IBE for short) consists of four PPT algorithms: a Setup algorithm that takes in the security parameter and outputs a set of parameters *params* and a master secret key *msk*; a KeyGen algorithm that takes in the master secret key *msk* and public identity id and outputs the secret key sk_{id} corresponding to the identity; an Enc algorithm that takes in a public identity id and a message M and outputs a ciphertext C; and a Dec algorithm that takes in a secret key sk_{id} corresponding to some identity id and a ciphertext C (encrypted to the identity id), outputs the message M contained in C. There are two security properties that are desirable for an IBE: $\mathsf{IND\text{-}ID\text{-}CPA}$ security, which is analogous to the notion of IND-CPA security for regular encryption, and *anonymity* [1], which says that it is hard to tell, given a ciphertext, which identity it is encrypted for. Formal definitions of these two properties can be found in Appendix A.2.

In this section, we see how to construct an IBE that satisfies both these properties if Assumption 4.5 holds, which we recall states that given $u, v \xleftarrow{\$} G$ it is hard to distinguish $e(f_1(u), f_2(v))$ from random.

- $\mathsf{Setup}(1^k)$: Use the Setup algorithm from Section 2.2 to produce $(N, G_1, G_2, G_T, e, g_1, g_2)$. Next, define a hash function $H : \{0,1\}^* \to G$ (note that, among others, Chatterjee and Menezes [24] demonstrate how this is possible) that maps from possible identities to elements in G, and output $params := (N, G_1, G_2, g_1, g_2, e, G_T, H)$ and $msk := f_1$.
- $\mathsf{KeyGen}(params, msk, id)$: Output $sk_{id} := f_1(H(id))$.
- $\mathsf{Enc}(params, id, M)$: Pick random values $r, s \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and compute $C_1 := g_1^r g_2^s$ and $C_2 := M \cdot e(H(id), g_2^s)$. Output $C := (C_1, C_2)$.

- $\mathsf{Dec}(params, C = (C_1, C_2), sk_{id})$: Output $M := C_2/e(sk_{id}, C_1)$.

Theorem 7.1. *The scheme outlined above is an anonymous* IND-ID-CPA*-secure IBE under Assumption 4.5 and in the random oracle model.*

Lemma 7.2. *The scheme outlined above is correct.*

Proof. Using the projection property of f_1 and the alternating property of the Weil pairing, we see that

$$\frac{C_2}{e(sk, C_1)} = \frac{M \cdot e(H(id), g_2^s)}{e(f_1(H(id)), g_1^r g_2^s)} = \frac{M \cdot e(f_1(H(id)), g_2^s) \cdot e(f_2(H(id)), g_2^s)}{e(f_1(H(id)), g_1^r) \cdot e(f_1(H(id)), g_2^s)} = \frac{M \cdot e(f_1(H(id)), g_2^s) \cdot 1}{1 \cdot e(f_1(H(id)), g_2^s)} = M,$$

so that for a properly formed ciphertext decryption will always return the correct message. □

Lemma 7.3. *If Assumption 4.5 holds, then the scheme outlined above is an* IND-ID-CPA*-secure IBE in the random oracle model.*

Proof. To show this, we consider an adversary $\mathcal{A}$ that succeeds in the game outlined in Definition A.2 with some non-negligible advantage ϵ, and use it to construct an adversary $\mathcal{B}$ that breaks Assumption 4.5 with non-negligible advantage ϵ/q, where q is the number of queries to the random oracle.

To start, $\mathcal{B}$ will receive as input the setup parameters *params* as well as the values u, v, T, where either $T = e(f_1(u), f_2(v))$ or T is random; $\mathcal{B}$ can then send *params* to $\mathcal{A}$. To answer H oracle queries, $\mathcal{B}$ will first guess at random which query it thinks $\mathcal{A}$ will pick as id^*; denote by i^* the index that $\mathcal{B}$ chooses. On all but the i^*-th query, if $\mathcal{A}$ queries on an identity id, $\mathcal{B}$ will pick random values $a, b \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and return $H(id) := g_1^a g_2^b$; it will also store $H(id)$ and g_1^a for later. On the i^*-th query, $\mathcal{B}$ will call the queried identity id^* and return $H(id^*) := u$. Meanwhile, for key extraction queries, if $\mathcal{A}$ queries on an identity id that was previously queried to H then $\mathcal{B}$ can simply return the stored G_1 component. Otherwise, $\mathcal{B}$ can pick $H(id)$ as it did earlier, return the G_1 component, and store the values for later. If $\mathcal{A}$ ever queries on id^*, $\mathcal{B}$ must simply abort. Finally, at some point $\mathcal{A}$ will issue its challenge (M_0, M_1, id^*). If id^* does not match $\mathcal{B}$'s guess then it must again abort; otherwise, it can pick a random bit $b \xleftarrow{\$} \{0, 1\}$, and return to $\mathcal{A}$ $(C_1 := v, C_2 := M_b \cdot T)$. When $\mathcal{A}$ outputs its guess bit b', $\mathcal{B}$ will guess that T is random if $b' \neq b$ and equal to $e(f_1(u), f_2(v))$ if $b' = b$.

We now need to show that if $\mathcal{A}$ has advantage ϵ, then $\mathcal{B}$ will have advantage ϵ/q; first, we show that interactions with $\mathcal{B}$ are identical to the honest interactions that $\mathcal{A}$ expects. Because the parameters for the game in Definition A.2 and Assumption 4.5 are the same, the value *params* that $\mathcal{B}$ gives to $\mathcal{A}$ will be identical to the honest one. For random oracle queries, $\mathcal{B}$ chooses completely random values for all but the identity id^*. Because u is assumed to be random, however, the value $H(id^*)$ will also look random to $\mathcal{A}$ and so the outputs $H(id)$ returned by $\mathcal{B}$ will again be identical to ones chosen completely at random. Similarly, because $\mathcal{B}$ is picking the values for $H(id)$ itself, it knows the value of $f_1(H(id))$ for every $id \neq id^*$ and so the sk_{id} values will in fact be computed correctly and are again identical to those output by an honest authority. For the ciphertext, the C_1 returned by $\mathcal{B}$ will be identically distributed to an honest one, as v is assumed to be a random element of G.

We turn finally to the value C_2 returned by $\mathcal{B}$. If T is random, then C_2 must be random as well, so in particular is information-theoretically independent from M_b and thus can reveal no information about the bit b. In this case then, $\mathcal{A}$ cannot have any advantage in guessing the bit b. If instead $T = e(f_1(u), f_2(v))$, (C_1, C_2) is an honestly computed encryption of M_b. In this case, $\mathcal{A}$ will succeed with non-negligible advantage ϵ; as $\mathcal{B}$ succeeds in guessing whenever $\mathcal{A}$ does and it correctly guesses i^*, $\mathcal{B}$ will succeed with advantage ϵ/q. □

We next turn to proving that the IBE is anonymous. As we have just shown IND-ID-CPA security, by Lemma A.5 it suffices to prove anonymity by proving that our IBE satisfies the weaker notion of anonymity defined in Definition A.4.

Lemma 7.4. *The IBE construction given in Section 7.1 satisfies weak anonymity, as defined in Definition A.4.*

Proof. To prove this, we show that, for a random message R, the ciphertext C^* given to $\mathcal{A}$ in Step 3 of the game is information-theoretically independent of the bit b, and thus $\mathcal{A}$ can have no advantage. Looking at the ciphertext, we can see that $C^* = (C_1^*, C_2^*)$, where $C_1^* = g_1^r g_2^s$ for some $r, s \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$, and $C_2^* = R \cdot e(H(id_b), g_2^s)$. As R is random, C_2^* will be distributed uniformly at random over G_T and thus its distribution will be independent of the bit b. In addition, C_1^* contains no information about the bit b (as it is always just a random element of G), and so C^* as a whole is information-theoretically independent of the bit b. □

Combining this lemma with the fact that we just proved the IND-ID-CPA security of the scheme, Lemma A.5 tells us that the scheme is also anonymous and so Theorem 7.1 follows.

7.2 A unique signature scheme

One interesting aspect of the IBE described above is the signature scheme that we can derive from it, which works as follows:

- $\mathsf{KeyGen}(1^k)$: Use the same methods as above to generate *params* and *msk*, then output $pk :=$ *params* and $sk := (pk, msk)$.
- $\mathsf{Sign}(sk, M)$: Compute and return $\sigma := f_1(H(M))$.
- $\mathsf{Verify}(pk, \sigma, M)$: Check that $e(\sigma, g_1) = 1$ and $e(\sigma, g) = e(H(M), g_2)$; return 1 if these checks pass and 0 otherwise.

The unforgeability of the signature follows from transformation, proposed by Naor [12, 27], from IBE to signatures. In fact, we can prove unforgeability for the signature scheme alone under a weaker, computational assumption (Assumption 4.4), still in the random oracle model.

Because no randomness is involved, we can see that the signing algorithm will generate only a single signature for any message. To establish that our signature is a *unique signature* [33, 46], we must prove a stronger property: that if, for some pk and M, $\mathsf{Verify}(pk, \sigma, M) = \mathsf{Verify}(pk, \sigma', M) = 1$, then $\sigma = \sigma'$.

Theorem 7.5. *For a fixed message M, if there exist two signatures σ_1 and σ_2 such that $\mathsf{Verify}(pk, \sigma_1, M) = \mathsf{Verify}(pk, \sigma_2, M) = 1$ then $\sigma_1 = \sigma_2$.*

Proof. If $\mathsf{Verify}(pk, \sigma_i, M) = 1$ for $i \in \{1, 2\}$ then, by the definition of the verification algorithm, we have that $e(\sigma_i, g_1) = 1$ and $e(\sigma_i, g) = e(H(M), g_2)$. The first of these checks ensures that, because the pairing is non-degenerate, it must be the case that $\sigma_i \in G_1$. In this case, $e(\sigma_i, g) = e(\sigma_i, g_2)$, which further implies, because $e(\sigma_i, g) = e(H(M), g_2)$, that $\sigma_i = H(M)$ for both $i = 1$ and $i = 2$; i.e., the signatures are equal. □

7.3 A simple encryption scheme

In addition to the IND-CCA-secure encryption scheme that can be derived from the IBE using the generic transformations due to Boneh, Canetti, Halevi, and Katz [11], we can also describe a multiplicatively homomorphic IND-CPA-secure scheme as follows:

- $\mathsf{Setup}(1^k)$: Use the Setup algorithm from Section 2.2 to output $params := (N, G_1, G_2, G_T, e, g_1, g_2)$.
- $\mathsf{KeyGen}(params)$: Pick random values $a, b \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and compute $u := g_1^a \cdot g_2^b$. Output $pk := u$ and $sk := g_1^a$.
- $\mathsf{Enc}(params, pk, M)$: Pick random values $r, s \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and compute $C_1 := g_1^r \cdot g_2^s$ and $C_2 := M \cdot e(pk, g_2^s)$. Output $C := (C_1, C_2)$.

- $\mathsf{Dec}(params, sk, C = (C_1, C_2))$: Output $M := C_2/e(sk, C_2)$.
- $\mathsf{Eval}(params, pk, \times, C_1, C_2)$: Parse $C_1 = (C_{11}, C_{12})$ and $C_2 = (C_{21}, C_{22})$, pick $r, s \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$, and compute $C_1' := C_{11} \cdot C_{21} \cdot g_1^r \cdot g_2^s$ and $C_2' := C_{12} \cdot C_{22} \cdot e(pk, g_2^s)$. Output $C := (C_1', C_2')$.

This scheme has a number of nice advantages. First, the keys in this scheme are group elements; as compared to BGN [15], in which keys are entire group settings, this means that keys all live in the same space, which means they are both smaller and potentially more useful in applications. For more complex protocols in which the encryption scheme is a building block, providing a user with access to f_1 makes him "all-powerful," as it allows him to recover the secret key corresponding to any public key. In addition, the encryption scheme has the advantage over the IND-CCA-secure scheme derived from the IBE that its proof of security does not rely on the random oracle model.

Theorem 7.6. *If Assumption 4.5 holds, the encryption scheme described above is IND-CPA secure.*

Proof. To show this, we take an adversary $\mathcal{A}$ that breaks the IND-CPA security of the scheme with some non-negligible advantage ϵ and use it to construct an adversary $\mathcal{B}$ that breaks Assumption 4.5 with the same non-negligible advantage.

To start, $\mathcal{B}$ will get as input a group setting $params := (N, G_1, G_2, G_T, e, g_1, g_2)$ and elements $u, v \in G$ and $T \in G_T$. It will then set $pk := u$ and give both $params$ and pk to $\mathcal{A}$. When $\mathcal{A}$ gives back its query (M_0, M_1), $\mathcal{B}$ will pick a bit $b \xleftarrow{\$} \{0, 1\}$ and return to $\mathcal{A}$ $C := (v, M_b \cdot T)$. When $\mathcal{A}$ outputs its guess bit b', $\mathcal{B}$ will guess that T is random if $b' \neq b$ and equal to $e(f_1(u), f_2(v))$ otherwise.

To see that interactions with $\mathcal{B}$ are distributed indistinguishably from those that $\mathcal{A}$ expects, we first observe that $params$ are the same in both the honest case and the interaction with $\mathcal{B}$. Similarly, as u is assumed to be a random value, the pk given to $\mathcal{A}$ will be distributed identically in both cases as well. As for the ciphertext, using $C_1 := v$ is again distributed identically to the honest distribution over C_1, as in both cases it is just a random element of G. Turning finally to the value C_2, we see that if T is random then C_2 is random as well, and is thus information-theoretically independent of the message M_b. In this case, then, $\mathcal{A}$ can have no advantage in guessing the bit b. In the case that $T = e(f_1(u), f_2(v))$, however, (C_1, C_2) is an honestly computed encryption of M_b and so $\mathcal{A}$ should have the same advantage here that it does normally. As $\mathcal{B}$ therefore succeeds in guessing whenever $\mathcal{A}$ does, $\mathcal{B}$ must succeed with advantage ϵ as well. □

Acknowledgments

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3), July 2008.

[2] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *Proceedings of Crypto 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236, 2010.

[3] R. Balasubramanian and N. Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm. *Journal of Cryptology*, 11(2):141–145, 1998.

[4] L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417, 2005. `http://eprint.iacr.org/2005/417`.

[5] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Delegatable anonymous credentials. In *Proceedings of Crypto 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125. Springer-Verlag, 2009.

[6] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Non-interactive anonymous credentials. In *Proceedings of the 5th Theory of Cryptography Conference (TCC)*, pages 356–374, 2008.

[7] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography.* Number 265 in London Mathematical Society. Cambridge University Press, 1999.

[8] M. Blum, A. de Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.

[9] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *Proceedings of PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, Jan. 2003.

[10] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proceedings of Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 2004.

[11] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal of Computing*, 36(5):1301–28, 2007.

[12] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[13] D. Boneh, C. Gentry, and M. Hamburg. Space-efficient identity based encryption without pairings. In A. Sinclair, editor, *Proceedings of FOCS 2007*, pages 647–57. IEEE Computer Society, Oct. 2007.

[14] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–32. Springer-Verlag, 2003.

[15] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proceedings of the 2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer-Verlag, 2005.

[16] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, Sept. 2004. Extended abstract in *Proceedings of Asiacrypt 2001*.

[17] D. Boneh, K. Rubin, and A. Silverberg. Finding ordinary composite order elliptic curves using the Cocks-Pinch method. *Journal of Number Theory*, 131(5):832–841, 2011.

[18] D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Proceedings of Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 573–592. Springer-Verlag, 2006.

[19] X. Boyen and B. Waters. Compact group signatures without random oracles. In *Proceedings of Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444. Springer-Verlag, 2006.

[20] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *Proceedings of PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2007.

[21] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Proceedings of Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer-Verlag, 2004.

[22] L. Charlap and D. Robbins. An elementary introduction to elliptic curves. IDA/CCR report, 1998. `www.idaccr.org/reports/reports.html`.

[23] D. Charles. On the existence of distortion maps on ordinary elliptic curves. Cryptology ePrint Archive, Report 2006/128, 2006. `http://eprint.iacr.org/2006/128`.

[24] S. Chatterjee and A. Menezes. On cryptographic protocols employing asymmetric pairings – the role of ψ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.

[25] L. Chen, Z. Cheng, and N. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6(4):213–41, 2007.

[26] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *Proceedings of IMA 2001*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–63. Springer-Verlag, Dec. 2001.

[27] Y. Cui, E. Fujisaki, G. Hanaoka, H. Imai, and R. Zhang. Formal security treatments for signatures from identity-based encryption. In W. Susilo, J. K. Liu, and Y. Mu, editors, *Proceedings of ProvSec 2007*, volume 4784 of *Lecture Notes in Computer Science*, pages 218–27. Springer-Verlag, Nov. 2007.

[28] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal of Computing*, 29(1):1–28, 1999.

[29] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *Proceedings of Eurocrypt 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 44–61. Springer-Verlag, 2010.

[30] D. M. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23:224–280, 2010.

[31] S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–21, 2008.

[32] S. Galbraith and V. Rotger. Easy decision Diffie-Hellman groups. *LMS Journal of Computation and Mathematics*, 7:201–218, 2004.

[33] S. Goldwasser and R. Ostrovsky. Invariant signatures and noninteractive zero-knowledge proofs are equivalent (extended abstract). In E. Brickell, editor, *Proceedings of Crypto 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 228–45. Springer-Verlag, Aug. 1992.

[34] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Proceedings of Asiacrypt 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer-Verlag, 2006.

[35] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero-knowledge for NP. In *Proceedings of Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer-Verlag, 2006.

[36] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Proceedings of Eurocrypt 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, 2008.

[37] A. Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–76, 2004.

[38] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *Journal of Cryptology*, 16(4):239–47, 2003.

[39] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of Eurocrypt 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer-Verlag, 2008.

[40] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In N. Smart, editor, *Proceedings of Cryptography and Coding 2005*, volume 3796 of *Lecture Notes in Computer Science*, pages 13–36. Springer-Verlag, 2005.

[41] A. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *Proceedings of Eurocrypt 2012*, 2012.

[42] A. Lewko and S. Meiklejohn. A profitable sub-prime loan: Obtaining the advantages of composite-order in prime-order bilinear groups. Cryptology ePrint Archive, Report 2013/300, 2013. `eprint.iacr.org/2013/300`.

[43] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: attributed-based encryption and (hierarchical) inner production encryption. In *Proceedings of Eurocrypt 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer-Verlag, 2010.

[44] A. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *Proceedings of the 7th Theory of Cryptography Conference (TCC)*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer-Verlag, 2010.

[45] A. Lewko and B. Waters. Decentralizing attribute-based encryption. In *Proceedings of Eurocrypt 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 568–588. Springer-Verlag, 2011.

[46] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 597–612. Springer-Verlag, Aug. 2002.

[47] S. Meiklejohn, H. Shacham, and D. M. Freeman. Limitations on transformations from composite-order to prime-order groups: the case of round-optimal blind signatures. In *Proceedings of Asiacrypt 2010*, pages 519–538, 2010.

[48] V. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, 2004.

[49] J. H. Seo and J. H. Cheon. Beyond the limitation of prime-order bilinear groups and its application to round optimal blind signature. In *Proceedings of TCC 2012*, Lecture Notes in Computer Science. Springer-Verlag, 2012.

[50] J. Silverman. *The Arithmetic of Elliptic Curves*. Number 106 in Graduate Texts in Mathematics. Springer-Verlag, 1986.

A Definitions

A.1 Definitions for zero-knowledge proofs

In Section 6 we construct a non-interactive zero-knowledge proof of knowledge that a committed value is a bit; we formally define the notions of a zero-knowledge proof and a proof of knowledge here.

A non-interactive proof system [8, 28] for a relation R consists of three PPT algorithms: a $\mathsf{CRSSetup}$ algorithm that takes in the security parameter and generates a common reference string (CRS) σ_{crs}, a $\mathcal{P}$ algorithm that takes in σ_{crs}, a statement x and a witness w and outputs a proof π that $x \in L_R$, and a $\mathcal{V}$ algorithm that takes in σ_{crs}, a statement x and a proof π and outputs 1 if the proof is valid and 0 otherwise. We have the following definition that encapsulates the notions of witness indistinguishability, zero knowledge, and extractability that can be achieved for such a proof system:

Definition A.1. *A set of algorithms* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ *constitute a* non-interactive (NI) proof system *for an efficient relation R with associated language L_R if completeness and soundness, as defined below, are satisfied; additionally, the proof system is* extractable *if extractability is satisfied,* witness indistinguishable *(WI) if witness-indistinguishability is satisfied, and* zero knowledge *if the zero-knowledge property is satisfied. A proof system that satisfies all of these properties is a non-interactive zero-knowledge proof of knowledge (NIZKPoK for short), while one that satisfies witness indistinguishability but not zero knowledge is a NIWIPoK.*

1. *Completeness [8]. For all $\sigma_{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$ and $(x,w) \in R$, $\mathcal{V}(\sigma_{crs}, x, \pi) = 1$ for all $\pi \xleftarrow{\$} \mathcal{P}(\sigma_{crs}, x, w)$.*
2. *Soundness [8]. For all PPT $\mathcal{A}$ and $\sigma_{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$, the probability that $\mathcal{A}(\sigma_{crs})$ outputs (x, π) such that $x \notin L_R$ but $\mathcal{V}(\sigma_{crs}, x, \pi) = 1$ is at most negligible.* Perfect *soundness is achieved when this probability is 0.*
3. *Extractability [35]. There exists a polynomial-time extractor (E_1, E_2) such that $E_1(1^k)$ outputs (σ_{ext}, τ_e), and $E_2(\sigma_{ext}, \tau_e, x, \pi)$ outputs a value w such that $\sigma_{ext} \xleftarrow{\$} E_1(1^k)$ is indistinguishable from $\sigma_{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$, and for all PPT $\mathcal{A}$, the probability that $\mathcal{A}(\sigma_{ext}, \tau_e)$ outputs (x, π) such that $\mathcal{V}(\sigma_{crs}, x, \pi) = 1$ but $E_2(\sigma_{ext}, \tau_e, x, \pi) = w$ and $(x, w) \notin R$ is at most negligible.* Perfect *extractability is achieved if this probability is 0, and σ_{ext} is distributed identically to σ_{crs}.*
4. *Witness indistinguishability [28]. For all (x, w_1, w_2) such that $(x, w_1) \in R$ and $(x, w_2) \in R$, for $\sigma_{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$, $\pi_1 \xleftarrow{\$} \mathcal{P}(\sigma_{crs}, x, w_1)$, and $\pi_2 \xleftarrow{\$} \mathcal{P}(\sigma_{crs}, x, w_2)$, π_1 is indistinguishable from π_2.* Perfect *witness indistinguishability is achieved when these two distributions are identical.*
5. *Zero knowledge [28]. There exists a polynomial-time simulator (S_1, S_2) such that $S_1(1^k)$ outputs (σ_{sim}, τ_s), and $S_2(\sigma_{sim}, \tau_s, x)$ outputs a value π_s such that for all $(x, w) \in R$ and PPT adversaries $\mathcal{A}$, the following two interactions are indistinguishable: in the first, we compute $\sigma_{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$ and give $\mathcal{A}$ σ_{crs} and oracle access to $\mathcal{P}(\sigma_{crs}, \cdot, \cdot)$ (where $\mathcal{P}$ will output $\perp$ on input (x, w) such that $(x, w) \notin R$); in the second, we compute $(\sigma_{sim}, \tau_s) \xleftarrow{\$} S_1(1^k)$ and give $\mathcal{A}$ σ_{sim} and oracle access to $S_2'(\sigma_{sim}, \tau_s, \cdot, \cdot)$, where, on input (x, w), S_2' outputs $S_2(\sigma_{sim}, \tau_s, x)$ if $(x, w) \in R$ and $\perp$ otherwise.* Perfect *zero-knowledge is achieved if these interactions are distributed identically.*

A.2 Definitions for identity-based encryption

In Section 7.1 we construct an IBE scheme; recall briefly that an IBE consists of four algorithms: $\mathsf{Setup}(1^k)$, $\mathsf{KeyGen}(params, msk, id)$, $\mathsf{Enc}(params, id, M)$, and $\mathsf{Dec}(params, sk_{id}, C)$. The properties required by these algorithms can be summarized in the following definition:

Definition A.2. *An IBE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is* chosen-plaintext secure under a chosen identity attack *(or $\mathsf{IND\text{-}ID\text{-}CPA}$-secure for short) if the following two properties hold:*

1. *Correctness: For all identities id, messages M, and $(params, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$, if sk_{id} is the output of $\mathsf{KeyGen}(msk, id)$ and C is the output of $\mathsf{Enc}(params, id, M)$, then $\mathsf{Dec}(params, sk_{id}, C)$ outputs M with probability* 1.

2. $\mathsf{IND\text{-}ID\text{-}CPA}$ *security: For an adversary $\mathcal{A}$ and a bit b, let $p_b^{\mathcal{A}}(k)$ be the probability of the event that $b' = 0$ in the following game:*
   - *Step 1: $(params, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$.*
   - *Step 2: $\mathcal{A}$ gets params and is allowed to issue queries to a key extraction oracle; i.e., an oracle that, on input id, returns $sk_{id} \xleftarrow{\$} \mathsf{KeyGen}(msk, id)$. At some point, $\mathcal{A}$ will output (id^*, M_0, M_1), with the constraint that it must not have queried its oracle on input id^*.*
   - *Step 3: $C^* \xleftarrow{\$} \mathsf{Enc}(params, id^*, M_b)$.*
   - *Step 4: $\mathcal{A}$ is given C^* and access to its key extraction oracle, with the continued stipulation that it cannot query it on id^*. Finally, $\mathcal{A}$ will output a guess bit b'.*

   The IBE is considered $\mathsf{IND\text{-}ID\text{-}CPA}$ secure *if for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.*

In addition, an IBE can be called *anonymous* [1] if a ciphertext does not reveal its intended recipient (i.e., the identity *id* used to compute it). We have the following security definition:

Definition A.3. [1] *For an IBE scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*, a PPT adversary $\mathcal{A}$, and a bit b, let $p_b^{\mathcal{A}}(k)$ be the probability of the event that $b' = 0$ in the following game:*

- *Step 1: $(params, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$.*
- *Step 2: $\mathcal{A}$ gets params and is allowed to issue queries to a key extraction oracle; i.e., an oracle that, on input id, returns $sk_{id} \xleftarrow{\$} \mathsf{KeyGen}(msk, id)$. At some point, $\mathcal{A}$ will output $(id_0^*, id_1^* M)$, with the constraint that it must not have queried its oracle on id_0^* or id_1^*.*
- *Step 3: $C^* \xleftarrow{\$} \mathsf{Enc}(params, id_b^*, M)$.*
- *Step 4: $\mathcal{A}$ is given C^* and access to its key extraction oracle, with the continued stipulation that it cannot query it on id_0^* or id_1^*. Finally, $\mathcal{A}$ will output a guess bit b'.*

The IBE is considered anonymous *if for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.*

In addition to introducing the above notion of anonymity, Abdalla et al. also consider a weaker notion (which they call IND-ANO-RE-CPA and we choose to call *weak anonymity*) defined as follows:

Definition A.4. [1] *For an IBE scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*, a PPT adversary $\mathcal{A}$, and a bit b, let $p_b^{\mathcal{A}}(k)$ be the probability of the event that $b' = 0$ in the following game:*

- *Step 1: $(params, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$.*
- *Step 2: $\mathcal{A}$ gets params and is allowed to issue queries to a key extraction oracle; i.e., an oracle that, on input id, returns $sk_{id} \xleftarrow{\$} \mathsf{KeyGen}(msk, id)$. At some point, $\mathcal{A}$ will output $(id_0^*, id_1^* M)$, with the constraint that it must not have queried its oracle on id_0^* or id_1^*.*
- *Step 3: $R \xleftarrow{\$} \{0,1\}^{|M|}$, $C^* \xleftarrow{\$} \mathsf{Enc}(params, id_b^*, R)$.*
- *Step 4: $\mathcal{A}$ is given C^* and access to its key extraction oracle, with the continued stipulation that it cannot query it on id_0^* or id_1^*. Finally, $\mathcal{A}$ will output a guess bit b'.*

The IBE is considered weakly anonymous *if for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.*

They then go on to prove the following lemma, which we will use in Section 7 when we prove security of our IBE construction.

Lemma A.5. [1] *If an IBE scheme is* $\mathsf{IND\text{-}ID\text{-}CPA}$ *secure and weakly anonymous then is also anonymous.*

B A Restricted Two-Party Computation for 2-DNF

In this section, we show how the adapted version of BGN encryption presented in Section 5 can be used to realize a *restricted* two-party computation for a 2-DNF formula. We first define a 2-DNF:

Definition B.1. **[15]** *A* 2-DNF formula *over the variables* $x_1, \ldots, x_n$ *is of the form* $\vee_{i=1}^{k}(\ell_{i,1} \wedge \ell_{i,2})$ *where* $\ell_{i,1}, \ell_{i,2} \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$*.*

In a regular (i.e., unrestricted) two-party computation of a 2-DNF, one party, Alice, holds the formula $\phi(x_1, \ldots, x_n)$ and another party, Bob, holds an assignment $a_1, \ldots, a_n$. In the course of the computation, we would like Bob to learn $\phi(a_1, \ldots, a_n)$ without Alice learning anything about the assignment and without Bob learning anything about the formula (beyond what he learns from his output). To accomplish this using regular BGN encryption, Boneh et al. [15] outline a protocol: briefly, Bob can encrypt his assignment $a_1, \ldots, a_n$ using the BGN encryption scheme and send the ciphertexts to Alice, who then uses the homomorphic properties of the scheme to compute an encryption of $\phi(a_1, \ldots, a_n)$ by using $\times$ in place of $\wedge$, $+$ in place of $\vee$, and $(1 - x_i)$ in place of $\bar{x}_i$.

Boneh et al. also describe [15, Section 4.2] how to use this two-party computation of a 2-DNF formula to achieve private information retrieval (PIR for short): if Alice holds a database D of size n (for n a perfect square, this can mean a table with dimensions $\sqrt{n} \times \sqrt{n}$), then the 2-DNF formula can be $\phi(x_1, \ldots, x_{\sqrt{n}}, y_1, \ldots, y_{\sqrt{n}}) := \vee_{D_{i,j}=1}(x_i \wedge y_j)$. If Bob wishes to retrieve the (i, j)-th entry of this database, he sets $x_i = y_j = 1$ and all other variables to 0; he and Alice then engage in the two-party computation outlined above and at the end Bob will learn the value $D_{i,j}$.

In a restricted two-party computation, Bob can potentially encrypt his assignment to ensure that two values are not put in the same clause; this is accomplished by encrypting these values using the "noisy" type 3 and type 4 ciphertexts discussed in Section 5.

- Bob generates $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and sends pk to Alice. For unrestricted a_i values which can be put in a clause with any other value, Bob computes $c_{1i} \xleftarrow{\$} \mathsf{Enc}(pk, 1, a_i)$ and $c_{2i} \xleftarrow{\$} \mathsf{Enc}(pk, 2, a_i)$, while for restricted values a_j Bob computes $c_{1j} \xleftarrow{\$} \mathsf{Enc}(pk, 3, a_j)$ and $c_{2j} \xleftarrow{\$} \mathsf{Enc}(pk, 4, a_j)$. Bob then sends $\{c_{1i}, c_{2i}\}$ to Alice.
- Upon receiving pk and $\{c_{1i}, c_{2i}\}$, Alice proceeds as follows: for each clause of the form (ℓ_1, ℓ_2), if $\ell_1 = x_i$ then Alice will set $y_1 := c_{1i}$ and if $\ell_1 = \bar{x}_i$ then she sets $y_1 := \mathsf{Enc}(pk, 1, 1)/c_{1i}$, so that y_1 encrypts $1 - a_i$. She then does the same for ℓ_2; i.e., if $\ell_2 = x_j$ then she sets $y_2 := c_{2j}$ and if $\ell_2 = \bar{x}_j$ then she sets $y_2 := \mathsf{Enc}(pk, 2, 1)/c_{2j}$. She then computes $c_i \xleftarrow{\$} \mathsf{Eval}(pk, \times, y_1, y_2)$ for the i-th clause, and repeats this for all clauses. She can then compute $c \xleftarrow{\$} \mathsf{Eval}(pk, +, \{c_i\})$[8] and return c to Bob.
- When he gets back c from Alice, Bob can compute $m = \mathsf{Dec}(sk, c)$ to recover the value of $\phi(a_1, \ldots, a_n)$.

Theorem B.2. *The above protocol is correct.*

Proof. To show this, we want to prove that if Alice's formula is $\phi(x_1, \ldots, x_n) = \vee_{i=1}^{k}(\ell_{1,i} \wedge \ell_{2,i})$, then if Bob uses all unrestricted values (i.e., sticks to Type 1 and Type 2 encryption), at the end he outputs $m = \phi(a_1, \ldots, a_n)$. To start, Alice computes

$$c_i \xleftarrow{\$} \mathsf{Eval}(pk, \times, y_1, y_2) = e(y_1, y_2) \cdot e(h_1, g_2)^r$$

[8]Note that while we previously defined Eval only on pairs of ciphertexts, it is straightforward to extend it to take in a set of n ciphertexts instead.

for $r \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$. If $\ell_{1,i} = x_j$ and $\ell_{2,i} = x_{j'}$, then $y_1 \xleftarrow{\$} \mathsf{Enc}(pk, 1, a_j)$ and $y_2 \xleftarrow{\$} \mathsf{Enc}(pk, 2, a_{j'})$, and thus c_i encrypts $a_j \cdot a_{j'} = a_j \wedge a_{j'}$; we can similarly argue about the cases in which $\ell_{1,i} = \bar{x_j}$ or $\ell_{2,i} = \bar{x_{j'}}$. Now, Alice computes $c \xleftarrow{\$} \mathsf{Eval}(pk, +, \{c_i\}_i)$; as c_i appropriately encrypts $a_j \wedge a_{j'}$, we have

$$c = c_1 \cdot \ldots \cdot c_k \cdot e(h_1, g_2)^r$$

for $r \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$, which encrypts the sum, or logical or, of each of the clauses. By the correctness of BGN decryption, Bob will therefore correctly pull out the value $\phi(a_1, \ldots, a_n)$ as desired. ☐

As our protocol is, modulo Bob's choice of encryption, essentially identical to that of Boneh et al., the proof of the following theorem is very similar to theirs:

Theorem B.3. *If SGH holds in G_1 and G_2, the above protocol is a secure two-party computation for a semi-honest Alice and a semi-honest Bob.*

Proof. (Sketch.) By the IND-CPA security of the encryption scheme, the ciphertexts $\{c_{1i}, c_{2i}\}_i$ that Bob sends to Alice reveal nothing about his assignment $(a_1, \ldots, a_n)$; in particular, to simulate Bob's side of the interaction, a simulator could simply give $\{c_{1i} \xleftarrow{\$} \mathsf{Enc}(pk, 1, 0), c_{2i} \xleftarrow{\$} \mathsf{Enc}(pk, 2, 0)\}_i$ to Alice. Similarly, as Bob receives only the final ciphertext c, a simulator knowing the value of $\phi(a_1, \ldots, a_n)$ could simply compute $c \xleftarrow{\$} \mathsf{Enc}(pk, 5, \phi(a_1, \ldots, a_n))$ and return this to Bob; as this is distributed identically to what Bob receives from Alice, yet the simulator knew only $\phi(a_1, \ldots, a_n)$, Bob learns nothing beyond whether the formula was satisfied or not. ☐

As observed by Boneh et al., the above protocol is not secure against a malicious Bob. As an example, if Bob sends Alice encryptions of values that are not bits, then he can potentially learn if and how these values were used by seeing if $\phi(a_1, \ldots, a_n)$ is itself a bit or not. To augment this protocol to be secure even for a malicious Bob, we again adopt the approach of Boneh et al. Now, in addition to generating the set of ciphertexts $\{c_{1i}, c_{2i}\}$, Bob can also generate, for each i, a zero-knowledge proof π_i that c_{1i} and c_{2i} are encryptions of the same value, and that that value is a bit. To see how such proofs can be generated, we refer the reader to Section 6. To also prevent against Bob giving Alice a set of ciphertexts that he does not actually know how to decrypt, we can add an extra interaction in between Steps 1 and 2 in which Alice ensures that Bob really can decrypt by sending him a set of challenge ciphertexts and asking him to return the corresponding plaintexts. She will then only continue with Step 2 if the plaintexts are correct and all of the proofs π_i verify.

Theorem B.4. *If duplicate SGH holds, the above protocol is a secure two-party computation for a semi-honest Alice and a malicious Bob.*

Proof. (Sketch.) Once again, the IND-CPA security of the encryption scheme guarantees that Alice will learn nothing about Bob's assignment. In addition, the zero-knowledge property of the proof guarantees that these also reveal no information about the assignment $(a_1, \ldots, a_n)$; i.e., to simulate the proofs in addition to the ciphertexts, it would suffice to run the zero-knowledge simulator. As for Alice's security against a malicious Bob, by the soundness of the proof, Bob is unable to generate a proof π_i that verifies and yet the ciphertexts c_{1i} and c_{2i} do not encrypt a bit. Bob is therefore constrained to behave as he would in the honest protocol, and thus security reduces to the semi-honest case. ☐

On its own, it is perhaps not immediately clear why Bob would be interested in this type of restricted evaluation. Consider, however, the following example: Bob is a government employee who has recently, for whatever reason, had his security clearance downgraded, and Alice is in possession of a database that Bob wishes to access certain elements of. Because of his downgraded clearance, however, there are now certain entries in the database that Bob is not granted permission to access. To ensure that Bob does not access these entries, a censor Carol can be inserted into the above process as follows: Bob encrypts his values just as he did in the honest PIR scheme; i.e., if he is interested in the (i, j)-th entry then he sets $x_i = y_j = 1$ and all other variables to 0. He then encrypts these values, using only type 1 and type 2 encryptions,[9] to get a set of ciphertexts $\{c_{1i}, c_{2i}\}$ and passes these ciphertexts to Carol. Now, for all entries (i', j') that Bob should not be allowed to access, Carol can add noise to the appropriate ciphertexts as follows: first, for all i' such that there is a corresponding j' such that access to the (i', j')-th entry should be disallowed (and analogously for all j' such that there is a corresponding i'), Carol can pick random values $r_1, r_2 \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$ and compute $c'_{1i'} := c_{1i'} \cdot g_2^{r_1}$ and $c'_{2i'} := c_{2i'} \cdot g_1^{r_2}$; note that this changes the type of the $c_{1i'}$ ciphertext from 1 to 3, and the type of the $c_{2i'}$ ciphertext from 2 to 4. Carol then passes these modified ciphertexts along to Alice, who can return the result straight to Bob.

By the properties discussed in Section 5, we know that if access to the (i, j)-th entry is restricted then Bob will get back an encryption of garbage rather than $D_{i,j}$. In addition, the IND-CPA security of the encryption scheme guarantees that Carol can restrict access to certain entries withough learning anything about which entry Bob is trying to access. Although Alice does learn that the request has been passed through a censor, without knowledge of f_1 she still cannot "undo" the censoring process and thus, as desired, neither Alice nor Carol will learn which entry Bob was trying to access, and Bob will get back either (1) $D_{i,j}$ if his security clearance permits access to the (i, j)-th entry, (2) nothing if Alice chooses to abort, or (3) some garbage value to indicate that he is not permitted to access the (i, j)-th entry.

[9] We note that it does not actually cause any problems for Bob to use types 3 and 4 encryption as well, but it doesn't seem to provide any benefit to Bob either.