Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups

David Mandell Freeman[⋆]

CWI and Universiteit Leiden
`freeman@cwi.nl`

Abstract. We develop an abstract framework that encompasses the key properties of bilinear groups of composite order that are required to construct secure pairing-based cryptosystems, and we show how to use prime-order elliptic curve groups to construct bilinear groups with the same properties. In particular, we define a generalized version of the subgroup decision problem and give explicit constructions of bilinear groups in which the generalized subgroup decision assumption follows from the decision Diffie-Hellman assumption, the decision linear assumption, and/or related assumptions in prime-order groups.

We apply our framework and our prime-order group constructions to create more efficient versions of cryptosystems that originally required composite-order groups. Specifically, we consider the Boneh-Goh-Nissim encryption scheme, the Boneh-Sahai-Waters traitor tracing system, and the Katz-Sahai-Waters attribute-based encryption scheme. We give a security theorem for the prime-order group instantiation of each system, using assumptions of comparable complexity to those used in the composite-order setting. Our conversion of the last two systems to prime-order groups answers a problem posed by Groth and Sahai.

Keywords: pairing-based cryptography, composite-order groups, cryptographic hardness assumptions.

[⋆] Research supported by a National Science Foundation International Research Fellowship, with additional support from the Office of Multidisciplinary Activities in the NSF Directorate for Mathematical and Physical Sciences.

1 Introduction

Bilinear groups of composite order are a tool that has been used in the last few years to solve many problems in cryptography. The concept was introduced by Boneh, Goh, and Nissim [6], who applied the technique to the problems of private information retrieval, online voting, and universally verifiable computation. Subsequent authors have built on their work to create protocols such as non-interactive zero-knowledge proofs [19, 20], ring and group signatures [10, 31], attribute-based encryption [8, 23], traitor tracing schemes [7], and hierarchical identity-based encryption [24, 33].

Bilinear groups of composite order are pairs of abelian groups $(\mathbb{G}, \mathbb{G}_t)$, each of composite order $n = pq$, equipped with a nondegenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$. At their core, cryptosystems using bilinear groups of composite order usually base their security on variants of the *subgroup decision assumption*. Informally, this assumption says that given an element $g \in \mathbb{G}$, there is no efficient algorithm to determine whether g has order p. In particular, this assumption implies that it is infeasible to factor the group order n.

While the subgroup decision assumption is a useful tool for constructing secure protocols, it presents significant obstacles to implementing these protocols in practice. The only known instantiations of composite-order bilinear groups use elliptic curves (or more generally, abelian varieties) over finite fields. Since the elliptic curve group order n must be infeasible to factor, it must be at least (say) 1024 bits. On the other hand, the size of a prime-order elliptic curve group that provides an equivalent level of security is 160 bits [2]. As a result, group operations and especially pairing computations are prohibitively slow on composite-order curves: a Tate pairing on a 1024-bit composite-order elliptic curve is roughly 50 times slower than the same pairing on a comparable prime-order curve [29], and this performance gap will only get worse at higher security levels.

In short, requiring that the group order be infeasible to factor negates the principal advantage of elliptic curve cryptography over factoring-based systems, namely, that there is no known subexponental-time algorithm for computing discrete logarithms on an elliptic curve, while there is such an algorithm for factoring. Thus for efficient implementations we seek versions of protocols that use only prime-order elliptic curve groups. Developing these protocols is the main goal of this paper. In particular, we do the following:

- We **develop an abstract framework** that encompasses the key properties of bilinear groups of composite order, and we show how to use prime-order elliptic curves to construct bilinear groups with the same properties.
- We apply our framework and our prime-order construction to **create more efficient versions of cryptosystems** that originally used composite-order groups. Specifically, we consider:
  1. The Boneh-Goh-Nissim encryption scheme [6],
  2. The Boneh-Sahai-Waters traitor tracing system [7], and
  3. The Katz-Sahai-Waters attribute-based encryption scheme [23].

Our conversion of the last two systems to prime-order groups answers a problem posed by Groth and Sahai [20, Section 9],[1] who themselves implicitly use our framework to construct non-interactive proof systems using either composite-order or prime-order groups.

Outline and Summary of Results. The starting point for our abstract framework is the fact that the subgroup decision assumption defined by Boneh, Goh, and Nissim depends only on the existence of a group G for which it is infeasible to determine if an element $g \in G$ lies in a given

[1] Groth and Sahai ask if such a conversion can apply to traitor tracing or "Searchable Encryption," with references to [7] and [10], respectively. The latter reference should in fact be [8], which gives a result that is generalized by the construction of Katz, Sahai, and Waters [23, Section 5.2].

proper subgroup G_1 of G. This observation gives us a more general subgroup decision assumption in the language of abstract groups; see Section 2 for details.

Our construction using prime-order groups is based on the observation, used implicitly by Cramer and Shoup [13] and articulated explicitly by Gjøsteen [18], that the decision Diffie-Hellman (DDH) assumption is a generalized subgroup decision assumption. Specifically, suppose we are given a cyclic group $\mathbb{G}$ and elements $g, g^a, g^b, g^c \in \mathbb{G}$. Then the DDH assumption for $\mathbb{G}$ says exactly that it is infeasible to determine whether (g^b, g^c) is in the cyclic subgroup of $\mathbb{G} \times \mathbb{G}$ generated by (g, g^a). Thus any protocol that requires two groups $G_1 \subset G$ in which the generalized subgroup decision assumption holds can be instantiated using $G = \mathbb{G} \times \mathbb{G}$ and $G_1 = \langle (g_1, g_2) \rangle$, where $\mathbb{G}$ is a cyclic group in which the DDH assumption holds and g_1, g_2 are random elements of $\mathbb{G}$.

More generally, we can use $G = \mathbb{G}^n$ for any $n \geq 2$ and let G_1 be a rank-k subgroup for any $1 \leq k < n$. In this case the subgroup decision assumption in G follows from the *k-Linear assumption* in $\mathbb{G}$, which generalizes the DDH assumption. In particular, the 1-Linear assumption is DDH, while the 2-Linear assumption is the *decision linear assumption*. This more general construction makes explicit a relationship noticed by several previous authors (e.g., [20, 33]), namely, that functionality that can be achieved in composite-order groups under the subgroup decision assumption can also be achieved in prime-order groups under either the DDH or the decision linear assumption.

If the group $\mathbb{G}$ is equipped with a pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$, then applying $\hat{e}$ componentwise defines a pairing on $G = \mathbb{G}^n$. However, such a "symmetric" pairing (which only exists on supersingular elliptic curves) can be used to solve DDH in $\mathbb{G}$, so in this case our DDH-based construction is not secure. To get around this problem we use the fact that on ordinary elliptic curves there are two distinguished subgroups, denoted $\mathbb{G}_1$ and $\mathbb{G}_2$, in which DDH is believed to be infeasible for sufficiently large group orders. We can thus apply our construction twice to produce groups $G = \mathbb{G}_1^n$, $H = \mathbb{G}_2^n$, $G_t = \mathbb{G}_t^m$ (for some m), and an "asymmetric" pairing $e : G \times H \to G_t$. If the DDH assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$, then the subgroup decision assumption holds in G and H. (If we are using the d-Linear assumption with $d \geq 2$, then we can remain in the symmetric setting.)

While the security of composite-order group protocols depends on (variants of) the subgroup decision assumption, the correctness of these protocols depends on the groups having certain additional properties. In some cases, the groups G, H, G_t must be equipped with projection maps π_1, π_2, π_t that map them onto proper subgroups and commute with the pairing. In other cases, the groups must decompose into subgroups $G \cong \prod G_i$ and $H \cong \prod H_i$ such that the pairing restricted to $G_i \times H_j$ is trivial whenever $i \neq j$. In Section 3 we define these properties in our abstract framework and show how to instantiate them in the product groups $\mathbb{G}_1^n$, $\mathbb{G}_2^n$.

Sections 2 and 3 give us the framework and the tools necessary to convert composite-order group protocols to prime-order groups. Section 4 analyzes the efficiency gains realized in terms of the number of bits needed to represent group elements. For example, at a security level equivalent to 80-bit AES, ciphertexts in the Boneh-Goh-Nissim system can be up to three times smaller when instantiated using our prime-order construction than in the original composite-order system. At the 256-bit security level the improvement can be as large as a factor of 12.

In Sections 5, 6, and 7, we describe in detail the conversion procedure for the Boneh-Goh-Nissim cryptosystem, the Boneh-Sahai-Waters traitor tracing system, and the Katz-Sahai-Waters attribute-based encryption scheme, respectively. In each case we describe the scheme in our general framework and convert the assumptions used in the security proofs to our more general setting. We then consider the system instantiated with our prime-order group construction and give security theorems in this setting. If the original system is secure under a simple assumption (e.g., subgroup decision) then the converted scheme is also secure under a simple assumption (e.g., DDH); if the original system uses a complex assumption (as in the Katz-Sahai-Waters system) then the corresponding assumption in prime-order groups is also complex.

We note that our conversion process is not "black box": the security proof for each system must be analyzed to determine whether it carries over to our more general setting. For example, the recent identity-based encryption scheme of Lewko and Waters [24] uses explicitly in its security proof the fact that the group G has two subgroups of relatively prime order. However, we expect that our framework can be used to convert to prime-order groups other cryptosystems originally built using composite-order groups.

2 Subgroup Decision Problems

The problem of determining whether a given element g of a finite group G lies in a specified proper subgroup G_1 was used as a hardness assumption for constructing cryptosystems long before Boneh, Goh, and Nissim defined their "subgroup decision problem." Gjøsteen [18] has undertaken an extensive survey of such problems, which he calls "subgroup membership problems." For example, the *quadratic residuosity problem* is a subgroup membership problem: if we let $N = pq$ be a product of two distinct primes and define the group G to be the group of elements of $\mathbb{Z}_N^*$ with Jacobi symbol 1, the problem is to determine whether a given element in G lies in the subgroup of squares in G.

Boneh, Goh, and Nissim [6] defined their problem for pairs of groups $(\mathbb{G}, \mathbb{G}_t)$ of composite order $N = pq$ for which there exists a nondegenerate bilinear map, or "pairing," $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$. The problem is to determine whether a given element $g \in \mathbb{G}$ is in the subgroup of order p. Note that if g' generates $\mathbb{G}$, then $e(g, g')$ is a challenge element for the same problem in $\mathbb{G}_t$; thus if the subgroup decision problem is infeasible in $\mathbb{G}$ then it is in $\mathbb{G}_t$ as well.

Our general notion of a subgroup decision problem extends Gjøsteen's work to the bilinear setting. We begin by defining an object that generates the groups we will work with. We assume that the two groups input to the pairing are not identical (in the sense that there are no efficiently computable isomorphisms between them); this is known as an *asymmetric* pairing. We write all groups multiplicatively with identity element 1.

Definition 2.1. A *bilinear group generator* is an algorithm $\mathcal{G}$ that takes as input a security parameter λ and outputs a description of five abelian groups G, G_1, H, H_1, G_t, with $G_1 \subset G$ and $H_1 \subset H$. We assume that this description permits efficient[2] group operations and random sampling in each group. The algorithm also outputs an efficiently computable map $e : G \times H \to G_t$ that is

- Bilinear: $e(g_1 g_2, h_1 h_2) = e(g_1, h_1)e(g_1, h_2)e(g_2, h_1)e(g_2, h_2)$ for all $g_1, g_2 \in G$, $h_1, h_2 \in H$; and
- Nondegenerate: for any $g \in G$, if $e(g, h) = 1$ for all $h \in H$, then $g = 1$ (and similarly with G, H reversed).

Our generalized subgroup decision assumption says that it is infeasible to distinguish an element in G_1 from a random element of G, and similarly for H. More precisely, we have the following definition. (The notation $x \overset{\text{R}}{\leftarrow} X$ means x is chosen uniformly at random from the set X.)

Definition 2.2. Let $\mathcal{G}$ be a bilinear group generator. We define the following distribution:

$$\mathbb{G} = (G, G_1, H, H_1, G_t, e) \overset{\text{R}}{\leftarrow} \mathcal{G}(\lambda), \; T_0 \overset{\text{R}}{\leftarrow} G, \; T_1 \overset{\text{R}}{\leftarrow} G_1.$$

We define the *advantage* of an algorithm $\mathcal{A}$ in solving the *subgroup decision problem on the left* to be

$$\text{SDP}_{\text{L}}\text{-Adv}[\mathcal{A}, \mathcal{G}] = \Big| \Pr[\mathcal{A}(\mathbb{G}, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, T_1) = 1] \Big|.$$

[2] i.e., polynomial-time in λ.

We say that $\mathcal{G}$ *satisfies the subgroup decision assumption on the left* if $\text{SDP}_\text{L}\text{-Adv}[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$. We define the *subgroup decision problem/assumption on the right* and $\text{SDP}_\text{R}\text{-Adv}[\mathcal{A}, \mathcal{G}]$ analogously, with $T_0 \overset{\text{R}}{\leftarrow} H$ and $T_1 \overset{\text{R}}{\leftarrow} H_1$. We say $\mathcal{G}$ *satisfies the subgroup decision assumption* if it satisfies both the left and right assumptions.

Boneh, Goh, and Nissim's definition of the subgroup decision assumption does not require that the adversary be given an element of (or a means to sample from) G_1; however if G_1 is cyclic then their definition is equivalent to ours within a factor of 2 [7, Section 3].

Example 2.3 ([6, Section 2.1]). Boneh, Goh, and Nissim construct a bilinear group generator $\mathcal{G}_{BGN}$ using supersingular elliptic curves of composite order. Let $\mathcal{E}(\lambda)$ be an algorithm that outputs a product $N = p_1p_2$ of two distinct primes greater than 2^λ, a prime $q \equiv -1 \pmod{N}$, and a supersingular elliptic curve E over the finite field $\mathbb{F}_q$. Then $\#E(\mathbb{F}_q)$ is divisible by N, and we can construct $\mathcal{G}_{BGN}(\lambda)$ by running $\mathcal{E}(\lambda)$ and setting the output as follows:

- $G = H$ is the order-N subgroup of $E(\mathbb{F}_q)$;
- $G_1 = H_1$ is the order-p_1 subgroup of $E(\mathbb{F}_q)$;
- G_t is the order-N subgroup of $\mathbb{F}_{q^2}^*$; and
- $e : G \times G \to G_t$ is the modified N-Tate pairing on E [14, Section 2.1].

Each group is described by giving a generator.

It is believed that $\mathcal{G}_{BGN}$ satisfies the subgroup decision assumption when N is infeasible to factor. The construction can be extended to produce a group G whose order is a product of three or more primes, and the subgroup decision assumption is believed to hold in any nontrivial proper subgroup G_1 of G. Using the generic group analysis of Katz, Sahai, and Waters [23, Theorem A.2], one can show that any efficient generic algorithm to solve the subgroup decision problem for $\mathcal{G}_{BGN}$ can be used construct an efficient algorithm to factor N.

2.1 Product Groups, DDH, and d-Linear Assumptions

The primary motivation for our abstraction of composite-order group protocols is the observation that the decision Diffie-Hellman problem is also a subgroup decision problem [18, Section 4.5].

Let $\mathbb{G}$ be a finite cyclic group, and let $T = (g, g^a, g^b, g^c)$ be a 4-tuple of elements in $\mathbb{G}$. The *decision Diffie-Hellman (DDH) problem* is to determine whether $c \equiv ab \pmod{|g|}$; if this is infeasible then we say that $\mathbb{G}$ satisfies the *decision Diffie-Hellman assumption.* Now suppose we are given a DDH challenge T. Define G to be $\mathbb{G} \times \mathbb{G}$ and G_1 to be the cyclic subgroup of G generated by (g, g^a). Then the element $(g^b, g^c) \in G$ is in G_1 if and only if $c \equiv ab \pmod{|g|}$ — so solving the subgroup decision problem for $G_1 \subset G$ is exactly equivalent to solving DDH in $\mathbb{G}$.

Now we consider the same construction in the bilinear setting: let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ be finite cyclic groups, and let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ be a nondegenerate bilinear map. Then we can define $G = \mathbb{G}_1^2$, $H = \mathbb{G}_2^2$, and $G_t = \mathbb{G}_t^2$, and choose random elements of G and H to generate G_1 and H_1 respectively. We can define a nondegenerate pairing $e : G \times H \to \mathbb{G}_t$ by taking any nonzero matrix $A = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \text{Mat}_2(\mathbb{F}_p)$ and setting

$$e((g_1, g_2), (h_1, h_2)) := e(g_1, h_1)^a e(g_1, h_2)^b e(g_2, h_1)^c e(g_2, h_2)^d.$$

We can define a pairing mapping to $G_t = \mathbb{G}_t^m$ by choosing different coefficients a, b, c, d to define each component of the output. With this setup, if the DDH assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$, then the subgroup decision assumption holds for $G_1 \subset G$ and $H_1 \subset H$.

More generally, we consider a bilinear group generator $\mathcal{G}_k^n$ that produces two groups $G = \mathbb{G}_1^n$ and $H = \mathbb{G}_2^n$ and random rank-k subgroups $G_1 \subset G$ and $H_1 \subset H$. In this situation the natural analogue of the DDH problem is the *k-Linear problem*, introduced by Hofheinz and Kiltz [21] and Shacham [30]. The 1-Linear problem is simply DDH, while the 2-Linear problem is called the *decision linear problem* and was originally proposed by Boneh, Boyen, and Shacham [5] as a reasonable analogue for DDH in a group with a bilinear map.

The following definition and theorem formalize the relationship between subgroup decision problems and d-Linear problems. We will use the following notation: if we have a group $\mathbb{G}$ of order p, an element $g \in \mathbb{G}$, and a vector $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_p^n$, then we define $g^{\vec{x}} := (g^{x_1}, \ldots, g^{x_n}) \in \mathbb{G}^n$.

Definition 2.4. A bilinear group generator $\mathcal{P}$ is *prime-order* if the groups G, G_1, H, H_1, G_t all have prime order $p > 2^\lambda$. Then we have $G = G_1$ and $H = H_1$, and we denote the three distinct groups by $\mathbb{G}_1 = G$, $\mathbb{G}_2 = H$, and $\mathbb{G}_t = G_t$. We let $\hat{\mathbb{G}}$ denote the output $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$ of $\mathcal{P}(\lambda)$.

Let $d \geq 1$ be an integer. If $\mathcal{A}$ is an algorithm that takes as input $2d + 2$ elements of $\mathbb{G}_1$, we define the *advantage* of $\mathcal{A}$ in solving the *d-Linear problem in* $\mathbb{G}_1$ to be

$$d\text{-Lin}_{\mathbb{G}_1}\text{-Adv}[\mathcal{A}, \mathcal{P}] = \left| \Pr\left[\mathcal{A}(\hat{\mathbb{G}}, g_1, \ldots, g_d, g_1^{r_1}, \ldots, g_d^{r_d}, h, h^{r_1+\cdots+r_d}) = 1 : \begin{array}{l} \hat{\mathbb{G}} \xleftarrow{\text{R}} \mathcal{P}, g_1, \ldots, g_d \xleftarrow{\text{R}} \mathbb{G}_1, \\ r_1, \ldots, r_d \xleftarrow{\text{R}} \mathbb{F}_p \end{array}\right] - \Pr\left[\mathcal{A}(\hat{\mathbb{G}}, g_1, \ldots, g_d, g_1^{r_1}, \ldots, g_d^{r_d}, h, h^{s}) = 1 : \begin{array}{l} \hat{\mathbb{G}} \xleftarrow{\text{R}} \mathcal{P}, g_1, \ldots, g_d \xleftarrow{\text{R}} \mathbb{G}_1, \\ r_1, \ldots, r_d, s \xleftarrow{\text{R}} \mathbb{F}_p \end{array}\right] \right|,$$

and similarly for d-$\text{Lin}_{\mathbb{G}_2}$-Adv$[\mathcal{A}, \mathcal{P}]$. We say that $\mathcal{G}$ *satisfies the d-Linear assumption in* $\mathbb{G}_1$ if d-$\text{Lin}_{\mathbb{G}_1}$-Adv$[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$ (and similarly for $\mathbb{G}_2$). The *decision Diffie-Hellman (DDH) assumption* is the 1-Linear assumption. The *decision linear assumption* is the 2-Linear assumption.

Some previous authors (e.g., [1, 20]) have called the assumption that DDH is infeasible in both $\mathbb{G}_1$ and $\mathbb{G}_2$ the *symmetric external Diffie-Hellman assumption*, or SXDH. For clarity in our arguments, we prefer to call the problems DDH in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

Theorem 2.5. *Let $\mathcal{P}$ be a prime-order bilinear group generator. For integers n, k with $n \geq 2$ and $1 \leq k < n$, define $\mathcal{G}_k^n$ to be a bilinear group generator that on input λ does the following:*

1. *Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e}) \xleftarrow{\text{R}} \mathcal{P}(\lambda)$.*
2. *Let $G = \mathbb{G}_1^n$, $H = \mathbb{G}_2^n$, $G_t = \mathbb{G}_t^m$ for some m.*
3. *Choose generators $g \xleftarrow{\text{R}} \mathbb{G}_1$, $h \xleftarrow{\text{R}} \mathbb{G}_2$.*
4. *Choose random $\vec{x}_i, \vec{y}_i \xleftarrow{\text{R}} \mathbb{F}_p^n$ for $i = 1, \ldots, k$, such that the sets $\{\vec{x}_i\}$ and $\{\vec{y}_i\}$ are each linearly independent.*
5. *Let G_1 be the subgroup of G generated by $\{g^{\vec{x}_1}, \ldots, g^{\vec{x}_k}\}$ and H_1 be the subgroup of H generated by $\{h^{\vec{y}_1}, \ldots, h^{\vec{y}_k}\}$*
6. *Choose nonzero $n \times n$ matrices $A_\ell = (a_{ij}^{(\ell)})$ for $\ell = 1, \ldots, m$.*
7. *Define $e : G \times H \to G_t$ by $e((g_1, \ldots, g_n), (h_1, \ldots, h_n))_\ell := \prod_{i,j=1}^{n} e(g_i, h_j)^{a_{ij}^{(\ell)}}$.*
8. *Output the tuple $\Gamma_k^n = (G, G_1, H, H_1, G_t, e)$.*

If $\mathcal{P}$ satisfies the k-Linear assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, then $\mathcal{G}_k^n$ satisfies the subgroup decision assumption. Specifically, for any adversary $\mathcal{A}$ that solves the subgroup decision problem on the left for $\mathcal{G}_k^n$, there exists an adversary $\mathcal{B}$ that solves the k-Linear problem in $\mathbb{G}_1$ for $\mathcal{P}$, with

$$\text{SDP}_\text{L}\text{-Adv}[\mathcal{A}, \mathcal{G}_k^n] \leq (n - k) \cdot k\text{-Lin}_{\mathbb{G}_1}\text{-Adv}[\mathcal{B}, \mathcal{P}].$$

The analogous statement holds for $\mathcal{A}$ solving the subgroup decision problem on the right for $\mathcal{G}_k^n$ on the right and $\mathcal{B}$ solving the k-Linear problem in $\mathbb{G}_2$ for $\mathcal{P}$.

Proof. Let $\vec{x}_i = (x_{i,1}, \ldots, x_{i,n})$ be the vectors chosen in Step (4) of Theorem 2.5. Choose $g_0 \xleftarrow{\text{R}} \mathbb{G}_1$. For $j = 1, \ldots, n$, define distribution A_j to output a tuple $T \in \mathbb{G}_1^n$ whose first j components match those of $\prod_{i=1}^k g_0^{b_i \vec{x}_i}$ (i.e., a linear combination of the generators of $\mathbb{G}_1$), and whose last $n - j$ components are random. More precisely, we define

$$\text{Dist. } A_j\colon \left\{ \left(\mathit{\Gamma}_k^n, T = (g_0^{\sum b_i x_{i,1}}, \ldots, g_0^{\sum b_i x_{i,j}}, g_0^{r_{j+1}}, \ldots, g_0^{r_n}) \right) : \mathit{\Gamma}_k^n \leftarrow \mathcal{G}_k^n(\lambda);\ g_0 \xleftarrow{\text{R}} \mathbb{G}_1;\ \vec{b}, \vec{r} \xleftarrow{\text{R}} \mathbb{F}_p^n \right\},$$

where each sum in the exponent runs over $i = 1$ to k. Now let $(\hat{\mathbb{G}}, u_1, \ldots, u_k, v_1, \ldots, v_k, y, z)$ be a k-Linear challenge in $\mathbb{G}_1$. Fix j with $k + 1 \leq j \leq n$. Choose random $\vec{b}, \vec{r} \xleftarrow{\text{R}} \mathbb{F}_p^n$. Run $\mathcal{G}_k^n$ as before, except define $G_1 \subset G = \mathbb{G}_1^n$ to be the group generated by

$$\left\{ (u_i^{x_{i,1}}, \ldots, u_i^{x_{i,j-1}}, v_i^{1/b_i}, u_i^{x_{i,j+1}}, \ldots, u_i^{x_{i,n}}) \right\}_{i=1}^k .$$

Since the b_i are independently random, the generators G_1 remain independent and uniform in $\mathbb{G}_1^n$. Now consider the tuple

$$T' = (y^{\sum b_i x_{i,1}}, \ldots, y^{\sum b_i x_{i,j-1}}, z, u^{r_{i+1}}, \ldots, u^{r_n}).$$

Write $v_i = u_i^{s_i},\ z = y^c$. If $c = \sum_i s_i \pmod p$ then T' is distributed as the element T in distribution A_j, whereas if c is random then T' is distributed as the element T in distribution A_{j-1}. It follows that any algorithm that distinguishes distribution A_j from distribution A_{j-1} can be used to solve the k-Linear problem in $\mathbb{G}_1$.

Now observe that if $j \leq k$, then the element T in distribution A_j uses j independently random values $b_1, \ldots, b_j$. Thus distributions $A_1, \ldots, A_k$ are identical to an SDP challenge with a random element $T_0 \in G$. On the other hand, in distribution A_n the element T is equal to $g_0^{\sum b_i \vec{x}_i}$, and thus distribution A_n is equal to an SDP challenge with a random element $T_1 \in G_1$. It follows that given any $\mathcal{A}$ that attacks the SDP on the left for $\mathcal{G}_k^n$, we can construct a $\mathcal{B}$ solving the k-Linear problem in $\mathbb{G}_1$ with

$$\text{SDP}_\text{L}\text{-Adv}[\mathcal{A}, \mathcal{G}_1^n] \leq (n - k) \cdot k\text{-Lin}_{\mathbb{G}_1}\text{-Adv}[\mathcal{B}, \mathcal{P}].$$

□

Since the d-Linear assumption implies the $d+1$-Linear assumption for all $d \geq 1$ [21, Lemma 3], if $\mathcal{P}$ satisfies the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, then $\mathcal{G}_k^n$ satisfies the subgroup decision assumption for any $n \geq 1$ and $1 \leq k < n$. The converse holds when $k = 1$.

Proposition 2.6. *Let $\mathcal{P}$ be as in Theorem 2.5. If $\mathcal{G}_1^n$ satisfies the subgroup decision assumption, then $\mathcal{P}$ satisfies the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Proof. Let $(g_1, \ldots, g_n)$ be the generator of G_1 and $T = (g_1', \ldots, g_n')$ be the SDP challenge element. If $T \xleftarrow{\text{R}} G$, then $S = (g_1, g_2, g_1', g_2')$ is randomly distributed in $\mathbb{G}_1^4$, while if $T \xleftarrow{\text{R}} G$ then S is a Diffie-Hellman tuple. Thus an algorithm that solves DDH in $\mathbb{G}_1$ can solve the SDP on the left when $k = 1$. □

If we view all of the groups in the above construction as $\mathbb{F}_p$-vector spaces, then we see that the subgroup decision problem is a decisional version of the *vector decomposition problem* [34, 35, 17, 27], in which the adversary is given a decomposition $G \cong G_1 \times G_2$ and an element $x \in G$ and asked to find $y \in G_1$ and $z \in G_2$ such that $x = yz$.

The vector decomposition problem was originally proposed for use in the context of a single elliptic or hyperelliptic curve. Okamoto and Takashima [27] proposed using products of supersingular elliptic curves; our construction generalizes this idea to products of any bilinear group. In our context, the subgroup decision problem is equivalent to the "decisional subspace problem" of Okamoto and Takashima. However, Okamoto and Takashima do not indicate any relationship between their decisional problem and decisional problems in the underlying groups.

3 Pairings on Product Groups

In our construction of the bilinear group generator $\mathcal{G}_k^n$ from the prime-order bilinear group generator $\mathcal{P}$, we took the pairing e on the product groups to be any nontrivial linear combination of the componentwise pairings on the underlying prime-order group. However, the correctness proofs for protocols built in composite-order groups all use the fact that the pairings have some extra structure that arbitrary linear combinations are unlikely to have. We now investigate this structure further and determine how to replicate it in our product group context.

3.1 Projecting Pairings

The cryptosystem of Boneh, Goh, and Nissim works by taking elements $g \in G$ and $h \in G_1$ and encrypting a message M as $C = g^M h^r$, where r is random. The h term is a "blinding term" used to hide the part of the ciphertext that contains the message. Decryption is achieved by "projecting" the ciphertext away from the blinding term and taking a discrete logarithm to recover M. Specifically, when g has order $N = p_1 p_2$ and h has order p_1, the decryption can be achieved by first computing C^{p_1} to remove the h term, and then taking the discrete logarithm to the base g^{p_1} to recover M. The functionality of the cryptosystem requires that we can do this procedure either before or after the pairing; i.e., that we can construct and remove blinding terms in G_t. The following definition incorporates this concept into our abstract framework.

Definition 3.1. Let $\mathcal{G}$ be a bilinear group generator (Definition 2.1). We say that $\mathcal{G}$ is *projecting* if it also outputs a group $G'_t \subset G_t$ and three group homomorphisms π_1, π_2, π_t mapping G, H, G_t to themselves, respectively, such that

1. G_1, H_1, G'_t are contained in the kernels of π_1, π_2, π_t, respectively, and
2. $e(\pi_1(g), \pi_2(h)) = \pi_t(e(g,h))$ for all $g \in G$, $h \in H$.

Example 3.2. The bilinear group generator $\mathcal{G}_{BGN}$ of Example 2.3 is projecting: we let G'_t be the order-p_1 subgroup of G_t, let $\pi_1 = \pi_2$ be exponentiation by p_1, and let π_t be exponentiation by $(p_1)^2$.

Given a prime-order bilinear group generator $\mathcal{P}$, we wish to modify the bilinear group generator $\mathcal{G}_k^n$ constructed in Theorem 2.5 so it is projecting. To do so, we interpret the generation of G_1 and H_1 in terms of matrix actions, and we define the pairing e using a *tensor product* of matrices.

We begin by defining the projection maps π_1 and π_2. Let $G = \mathbb{G}_1^n$ and let g be a generator of $\mathbb{G}_1$. For $i = 1, \ldots, n$, let $\vec{e}_i$ be the unit vector with a 1 in the ith place and zeroes elsewhere. To construct the projection map π_1, we first observe that if G'_1 is the subgroup of G generated by $g^{\vec{e}_1}, \ldots, g^{\vec{e}_k}$, then any element of G'_1 has 1's in the last $n-k$ coordinates, so we can define a projection map π'_1 whose kernel is G'_1 by

$$\pi'_1(g_1, \ldots, g_n) := (1, \ldots, 1, g_{n-k+1}, \ldots, g_n).$$

Next we observe that the elements $g^{\vec{x}_1}, \dots, g^{\vec{x}_k}$ produced by $\mathcal{G}_k^n$ can be viewed as coming from a (right) action of an $n \times n$ matrix on the elements $g^{\vec{e}_1}, \dots, g^{\vec{e}_k}$. More precisely, for $\mathbf{g} = (g_1, \dots, g_n) \in G$ and a matrix $M = (a_{ij}) \in \mathrm{Mat}_n(\mathbb{F}_p)$, we define $\mathbf{g}^M$ by

$$\mathbf{g}^M := \left(\prod_{i=1}^n g_i^{a_{i1}}, \dots, \prod_{i=1}^n g_i^{a_{in}}\right).$$

With this definition, we have $(g^{\vec{x}})^M = g^{(\vec{x}M)}$.

Now let M be an invertible matrix whose first k rows are the vectors $\vec{x}_i$. Then $g^{\vec{x}_i} = g^{\vec{e}_i M}$. If we define U_k to be the matrix with 1's in the last $n-k$ diagonal places and zeroes elsewhere, then the map π_1' is given by $\pi_1'(\mathbf{g}) = \mathbf{g}^{U_k}$. Thus we can construct a projection map π_1 on G_1 by applying M^{-1} to map to G_1', using π_1' to project, and acting by M to map back to G_1; that is,

$$\pi_1(\mathbf{g}) = \mathbf{g}^{M^{-1}U_k M}.$$

We define π_2 analogously on H by computing an invertible matrix M' whose first k rows are the $\vec{y}_i$ produced by $\mathcal{G}_k^n$.

To define the pairing e, the subgroup G_t', and the projection map π_t, we use the *tensor product* from multilinear algebra. Recall that the tensor product of two n-dimensional vectors $\vec{x}, \vec{y}$ is

$$\vec{x} \otimes \vec{y} = (x_1\vec{y}, \dots, x_n\vec{y}) = (x_1y_1, \dots, x_1y_n, x_2y_1, \dots, x_2y_n, \dots, x_ny_1, \dots, x_ny_n).$$

We define $e : G \times H \to G_t := \mathbb{G}_t^{n^2}$ by $e(g^{\vec{x}}, h^{\vec{y}}) := \hat{e}(g,h)^{\vec{x}\otimes\vec{y}}$. That is, to pair $\mathbf{g} \in G$ and $\mathbf{h} \in H$ we take all the n^2 componentwise pairings $e(g_i, h_j)$ and write them in lexicographical order. In this case the A_ℓ of Theorem 2.5 are the n^2 matrices with a 1 in entry (i,j) and zeroes elsewhere.

Defining the pairing in this manner allows us to define the map π_t abstractly as the tensor product of the maps π_1 and π_2. In terms of the matrices we have defined, we have

$$\pi_t(\mathbf{g}_t) = \mathbf{g}_t^{(M^{-1}\otimes M'^{-1})(U_k\otimes U_k)(M\otimes M')},$$

where $\otimes$ indicates the tensor product (or *Kronecker product*) of matrices: if $A = (a_{ij})$ and $B = (b_{ij})$ are two $n \times n$ matrices, then $A \otimes B$ is the $n^2 \times n^2$ matrix which, when divided into $n \times n$ blocks, has the (i,j)th block equal to $a_{ij}B$ [22, Section 4.2].

Given this framework, we see that the constructions of Groth and Sahai [20, Section 5] are exactly the above with $(n,k) = (2,1)$ and $(3,2)$. We now give explicit details for the first case. This is the setup we will use to implement the Boneh-Goh-Nissim cryptosystem in prime-order groups.

Example 3.3. Let $\mathcal{P}$ be a prime-order bilinear group generator. Define $\mathcal{G}_P$ to be a bilinear group generator that on input λ does the following:

1. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e}) \xleftarrow{\text{R}} \mathcal{P}(\lambda)$.
2. Let $G = \mathbb{G}_1^2$, $H = \mathbb{G}_2^2$, $G_t = \mathbb{G}_t^4$.
3. Choose generators $g \xleftarrow{\text{R}} \mathbb{G}_1$, $h \xleftarrow{\text{R}} \mathbb{G}_2$, and let $\gamma = e(g,h)$.
4. Choose random $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2 \xleftarrow{\text{R}} \mathbb{F}_p$, such that $a_1d_1 - b_1c_1 = a_2d_2 - b_2c_2 = 1$.
5. Let G_1 be the subgroup of G generated by (g^{a_1}, g^{b_1}), let H_1 be the subgroup of H generated by (h^{a_1}, h^{b_1}), and let G_t' be the subgroup of G_t generated by

$$\{\gamma^{(a_1a_2, a_1b_2, b_1a_2, b_1b_2)}, \gamma^{(a_1c_2, a_1d_2, b_1c_2, b_1d_2)}, \gamma^{(c_1a_2, d_1b_2, c_1a_2, d_1b_2)}\}.$$

6. Define $e : G \times H \to G_t$ by $e((g_1,g_2),(h_1,h_2)) := (\hat{e}(g_1,h_1), \hat{e}(g_1,h_2), \hat{e}(g_2,h_1), \hat{e}(g_2,h_2))$.

7. Let $A = \begin{pmatrix} -b_1c_1 & -b_1d_1 \\ a_1c_1 & a_1d_1 \end{pmatrix}$, $B = \begin{pmatrix} -b_2c_2 & -b_2d_2 \\ a_2c_2 & a_2d_2 \end{pmatrix}$, and define

$$\begin{aligned} \pi_1((g_1, g_2)) &:= (g_1, g_2)^A = (g_1^{-b_1c_1} g_2^{a_1c_1}, g_1^{-b_1d_1} g_2^{a_1d_1}) \\ \pi_2((h_1, h_2)) &:= (h_1, h_2)^B = (h_1^{-b_2c_2} h_2^{a_2c_2}, h_1^{-b_2d_2} h_2^{a_2d_2}) \\ \pi_t((\gamma_1, \gamma_2, \gamma_3, \gamma_4)) &:= (\gamma_1, \gamma_2, \gamma_3, \gamma_4)^{A \otimes B} \end{aligned}$$

8. Output the tuple $(G, G_1, H, H_1, G_t, G'_t, e, \pi_1, \pi_2, \pi_t)$.

It is easy (though tedious) to check that $\mathcal{G}_P$ is a projecting bilinear group generator. We note that the groups output by $\mathcal{G}_P$ can be described simply by giving $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ and the pairs (g^{a_1}, g^{b_1}), (h^{a_2}, h^{b_2}). In particular, the group G'_t is generated by elements of the form $e(\mathbf{g}, \mathbf{h}_1)$ and $e(\mathbf{g}_1, \mathbf{h})$ with $\mathbf{g} \in G$, $\mathbf{g}_1 \in G_1$, $\mathbf{h} \in H$, and $\mathbf{h}_1 \in H_1$. This is important since in our application the maps π_1, π_2, π_t will be "trapdoor" information used as the system's secret key.

Proposition 3.4. *If $\mathcal{P}$ satisfies the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, then $\mathcal{G}_P$ satisfies the subgroup decision assumption.*

Proof. Since g is uniform in $\mathbb{G}_1$ and a_1, b_1, c_1, d_1 are uniformly random in $\mathbb{F}_p$, imposing the condition $a_1d_1 - b_1c_1 = 1$ does not introduce any deviations from uniformity in the generation of G_1 (and similar for G_2). We can thus apply Theorem 2.5. $\square$

3.2 Cancelling Pairings

The traitor-tracing scheme of Boneh, Sahai, and Waters [7], the predicate encryption scheme of Katz, Sahai, and Waters [23], and many other schemes based on bilinear groups of composite order use in an essential manner the fact that if two group elements g, h have relatively prime orders, then $e(g, h) = 1$. This property implies, for example, that we can use the two subgroups generated by g and h to encode different types of information, and the two components will remain distinct after the pairing operation. The following definition incorporates this concept into our framework.

Definition 3.5. Let $\mathcal{G}$ be a bilinear group generator (Definition 2.1). We say that $\mathcal{G}$ is *r-cancelling* if it also outputs groups $G_2, \ldots, G_r \subset G$ and $H_2, \ldots, H_r \subset H$, such that

1. $G \cong G_1 \times \cdots \times G_r$ and $H \cong H_1 \times \cdots \times H_r$,
2. $e(g_i, h_j) = 1$ whenever $g_i \in G_i$, $h_j \in H_j$, and $i \neq j$.

Example 3.6. The bilinear group generator $\mathcal{G}_{BGN}$ of Example 2.3 is 2-cancelling: we set $G_2 = H_2$ to be the order-p_2 subgroup of $E(\mathbb{F}_p)$. An analogous r-cancelling generator can be built by making the group order N a product of r distinct primes.

Given a prime-order bilinear group generator $\mathcal{P}$, we now show how to modify the bilinear group generator $\mathcal{G}_1^n$ constructed in Theorem 2.5 so it is n-cancelling. We begin by defining the pairing $e : G \times H \to G_t := \mathbb{G}_t$ to be

$$e((g_1, \ldots, g_n), (h_1, \ldots, h_n)) := \prod_{i=1}^n \hat{e}(g_i, h_i),$$

so in particular we have $e(g^{\vec{x}}, h^{\vec{y}}) = e(g, h)^{\vec{x} \cdot \vec{y}}$, where $\cdot$ indicates the vector dot product.

If $\mathcal{G}_1^n$ is n-cancelling, then the subgroups G_i, H_i are all cyclic of order p. Thus we need to choose generators $g^{\vec{x}_i}$ of G_i and $h^{\vec{y}_i}$ of H_i such that $\vec{x}_i \cdot \vec{y}_j = 0$ if and only if $i = j$. This is straightforward: we first choose any set of n linearly independent $\vec{x}_i$; then the equation $\vec{x}_i \cdot \vec{y}_j = 0$ for all $i \neq j$ gives a linear system n variables of rank $n-1$, so there is a one-dimensional solution space in $\mathbb{F}_p^n$. If we choose $\vec{y}_j$ in this space then with high probability we have $\vec{x}_j \cdot \vec{y}_j \neq 0$; if this is not the case then we can start again with a different set of $\vec{x}_i$. We illustrate with concrete examples for $n = 2$ and 3. We use the notation $\langle X \rangle$ to indicate the cyclic group generated by X.

Example 3.7. Let $\mathcal{P}$ be a prime-order bilinear group generator. Define $\mathcal{G}_{3C}$ to be a bilinear group generator that on input λ does the following:

1. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e}) \xleftarrow{\text{R}} \mathcal{P}(\lambda)$.
2. Let $G = \mathbb{G}_1^3$, $H = \mathbb{G}_2^3$, $G_t = \mathbb{G}_t$.
3. Choose generators $g_1, g_2, g_3 \xleftarrow{\text{R}} \mathbb{G}_1$, $h_1, h_2, h_3 \xleftarrow{\text{R}} \mathbb{G}_2$.
4. Choose random $x, y, z, u, v, w \xleftarrow{\text{R}} \mathbb{F}_p$, such that $\begin{cases} -xv-xw-yu+yw+zu+zv\neq 0, \\ xv-xw-yu+yw+zu-zv\neq 0. \end{cases}$
5. Define the subgroups
$$G_1 = \langle (g_1, g_1^x, g_1^u) \rangle,\ G_2 = \langle (g_2, g_2^y, g_2^v) \rangle,\ G_3 = \langle (g_3, g_3^z, g_3^w) \rangle,$$
$$H_1 = \langle (h_1^{zv-yw}, h_1^{w-v}, h_1^{y-z}) \rangle,\ H_2 = \langle (h_2^{zu-xw}, h_2^{w-u}, h_2^{z-x}) \rangle,\ H_3 = \langle (h_3^{yu-xv}, h_3^{v-u}, h_3^{x-y}) \rangle.$$
6. Define $e : G \times H \to G_t$ by $e((g, g', g''), (h, h', h'')) := \hat{e}(g, h)\hat{e}(g', h')\hat{e}(g'', h'')$.
7. Output the tuple $(G, G_1, G_2, H, H_1, H_2, G_t, e)$.

It is straightforward to show that $\mathcal{G}_{3C}$ is a 3-cancelling bilinear group generator. The inequalities in Step (4) guarantee that the G_i and H_i are linearly independent. Note that choosing the elements g_1, g_2, g_3 independently uniform allows us to scale the vectors $\vec{x}_1 = (1, x, u)$, $\vec{x}_2 = (1, y, v)$, $\vec{x}_3 = (1, z, w)$ so their first components are 1 without losing uniformity.

Example 3.8. We define a 2-cancelling bilinear group generator $\mathcal{G}_{2C}$ by restricting the construction in Example 3.7 to the first two components. Explicitly, we have $G = \mathbb{G}_1^2$, $H = \mathbb{G}_2^2$, $G_t = \mathbb{G}_t^2$ and we set $u = 0$, $v = 0$, $w = 1$ to obtain
$$G_1 = \langle (g_1, g_1^x) \rangle,\ G_2 = \langle (g_2, g_2^y) \rangle,\ H_1 = \langle (h_1^{-y}, h_1) \rangle,\ H_2 = \langle (h_2^{-x}, h_2) \rangle.$$
We define $e : G \times H \to G_t$ by $e((g, g'), (h, h')) := \hat{e}(g, h)\hat{e}(g', h'))$, and we output the tuple $(G, G_1, G_2, H, H_1, H_2, G_t, e)$.

Example 3.9. We can obtain an alternative 2-cancelling bilinear group generator $\mathcal{G}_L$ from the construction in Example 3.7 by letting G_1 be the subgroup spanned by the first two vectors produced. More precisely, we choose additional $a, b, c, d \xleftarrow{\text{R}} \mathbb{F}_p$ and set
$$G_1 = \langle (g_1, g_1^x, g_1^u),\ (g_2, g_2^y, g_2^v) \rangle,\ G_2 = \langle (g_3^{bc-ad}, g_3^{d-b}, g_3^{a-c}) \rangle,$$
$$H_1 = \langle (h_1, h_1^a, h_1^b),\ (h_2, h_2^c, h_2^d) \rangle,\ H_2 = \langle (h_3^{yu-xv}, h_3^{v-u}, h_3^{x-y}) \rangle.$$

Proposition 3.10. *If $\mathcal{P}$ satisfies the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, then $\mathcal{G}_{3C}$, $\mathcal{G}_{2C}$, and $\mathcal{G}_L$ satisfy the subgroup decision assumption. If $\mathcal{P}$ satisfies the decision linear assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, then $\mathcal{G}_L$ satisfies the subgroup decision assumption.*

Proof. Recall that an SDP adversary is given only G, G_1, H, H_1, and not a description of any G_i or H_i for $i \geq 2$. Since in each case the generators of G_1 and H_1 are independent and uniform, the result of Theorem 2.5 applies to $\mathcal{G}_{3C}$, $\mathcal{G}_{2C}$, and $\mathcal{G}_L$. $\square$

Remark 3.11. We observe that since the subgroup decision assumption for the generator $\mathcal{G}_L$ of Example 3.9 follows from the decision linear assumption for the prime-order group generator $\mathcal{P}$, we can use $\mathcal{G}_L$ to instantiate systems with a symmetric pairing $e : G \times G \to G_t$.

4 Performance Analysis

Our primary motivation for converting composite-order group protocols to prime-order groups is to improve efficiency in implementations. As discussed in the introduction, this improvement results from the fact that we can use smaller prime-order groups than composite-order groups at equivalent security levels. We now examine this improvement concretely. Specifically, we compare the sizes of the groups G, H, and G_t produced by the bilinear group generator $\mathcal{G}_{BGN}$ (Example 2.3) with the four examples from Section 3 of bilinear group generators built from prime-order generators.

For the generators $\mathcal{G}_P$ (Example 3.3), $\mathcal{G}_{3C}$ (Example 3.7), and $\mathcal{G}_{2C}$ (Example 3.8) we take the prime-order bilinear group generator $\mathcal{P}$ to be an algorithm that produces a "pairing-friendly" ordinary elliptic curve E over a finite field $\mathbb{F}_q$. On such curves there are two "distinguished" subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order p in which the DDH problem is presumed to be infeasible, and such that the Tate pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t \subset \mathbb{F}_{q^k}^*$ is nondegenerate. Here k is the *embedding degree*, defined to be the smallest integer such that p divides the order of $\mathbb{F}_{q^k}^*$.

The ordinary elliptic curves E that give the best performance while providing discrete log security comparable to three commonly proposed levels of AES security are as follows. The group sizes follow the 2007 NIST recommendations [2]; further details can be found in Appendix A.

80-bit security: A 170-bit Miyaji-Nakabayashi-Takano curve [26] with embedding degree 6;
128-bit security: A 256-bit Barreto-Naehrig curve [3] with embedding degree 12.
256-bit security: A 640-bit Brezing-Weng curve [11] with embedding degree 24.

The advantage of the generator $\mathcal{G}_L$ is that we can use a prime-order group with a symmetric pairing, which only exists on supersingular elliptic curves. Thus in this case we take $\mathcal{P}$ to produce a supersingular curve over $\mathbb{F}_{3^m}$ with embedding degree $k = 6$ (the maximum possible k for supersingular curves [25]). The fields that provide the best "match" for group orders at our three security levels are $\mathbb{F}_{3^{111}}$, $\mathbb{F}_{3^{323}}$, and $\mathbb{F}_{3^{1615}}$. The restriction to embedding degree $k \le 6$ means that at high security levels the group $\mathbb{G}_1$ will be much larger than the group $\mathbb{G}_1$ on an equivalent ordinary curve.

Table 1 compares the sizes of the groups produced by all five bilinear group generators at each of the three security levels. In all cases the groups G and H built using products of prime-order groups are much smaller than the groups G and H built using composite-order groups. The group G_t for the projecting generator $\mathcal{G}_P$ is twice as large as the composite-order G_t, due to the fact that elements of G_t are four elements of $\mathbb{F}_{q^k}$. However, the groups G_t for the cancelling generators $\mathcal{G}_{2C}$, $\mathcal{G}_{3C}$, $\mathcal{G}_L$ are half as large as the composite-order G_t.

Table 1. Estimated bit sizes of group elements for bilinear group generators at three different security levels.

	80-bit AES			128-bit AES			256-bit AES		
Bilinear group generator	G	H	G_t	G	H	G_t	G	H	G_t
$\mathcal{G}_{BGN}$ (Example 2.3)	1024	1024	2048	3072	3072	6144	15360	15360	30720
$\mathcal{G}_P$ (Example 3.3)	340	680	4080	512	1024	12288	1280	5120	61440
$\mathcal{G}_{3C}$ (Example 3.7)	510	1020	1020	768	1536	3072	1920	7680	15360
$\mathcal{G}_{2C}$ (Example 3.8)	340	680	1020	512	1024	3072	1280	5120	15360
$\mathcal{G}_L$ (Example 3.9)	528	528	1056	1536	1536	3072	7680	7680	15360

The pairings e for $\mathcal{G}_{2C}, \mathcal{G}_{3C}, \mathcal{G}_L$ each require two pairing computations on the curve E and the pairing e for $\mathcal{G}_P$ requires four pairings on E; the pairing e for $\mathcal{G}_{BGN}$ requires only one pairing on the curve. However, the sizes of the elliptic curve groups in the prime-order case are so much smaller that the pairings will be far more than four times faster. Indeed, the Tate pairing on a 1024-bit supersingular curve runs ≈ 50 times slower than the Tate pairing on a 170-bit MNT curve [29].

5 Application 1: The BGN Cryptosystem

Our first application of the framework developed above is to the public-key encryption scheme of Boneh, Goh, and Nissim [6]. This scheme has the feature that given two ciphertexts, one can create a new ciphertext that encrypts either the sum or the product of the corresponding plaintexts. The product operation can only be carried out once; the system is thus "partially doubly homomorphic."

Step 1 of the conversion process is to write the scheme in the abstract framework and transfer it to asymmetric groups. In the original BGN protocol any ciphertext may be paired with any other ciphertext, so in the asymmetric setting each computation in G must be duplicated in H. We must use a projecting pairing, as the decryption algorithm requires projection away from a certain subgroup. (Indeed, we defined the projecting property by using the BGN scheme as our model.)

$\mathsf{KeyGen}(\lambda)$: Let $\mathcal{G}$ be a projecting bilinear group generator (Definition 3.1). Compute $(G, G_1, H, H_1, G_t, G'_t, e, \pi_1, \pi_2, \pi_t) \leftarrow \mathcal{G}(\lambda)$. Choose $g \overset{\text{R}}{\leftarrow} G$, $h \overset{\text{R}}{\leftarrow} H$, and output the public key $PK = (G, G_1, H, H_1, G_t, e, \ g, h)$ and the secret key $SK = (\pi_1, \pi_2, \pi_t)$.

$\mathsf{Encrypt}(PK, M)$: Choose $g_1 \overset{\text{R}}{\leftarrow} G_1$ and $h_1 \overset{\text{R}}{\leftarrow} H_1$. (Recall that the output of $\mathcal{G}$ allows random sampling from G_1 and H_1.) Output the ciphertext $(C_A, C_B) = (g^M \cdot g_1,\ h^M \cdot h_1) \in G \times H$.

$\mathsf{Multiply}(PK, C_A, C_B)$: This algorithm takes as input two ciphertexts $C_A \in G$ and $C_B \in H$. Choose $g_1 \overset{\text{R}}{\leftarrow} G_1$ and $h_1 \overset{\text{R}}{\leftarrow} H_1$, and output $C = e(C_A, C_B) \cdot e(g, h_1) \cdot e(g_1, h) \in G_t$.

$\mathsf{Add}(PK, C, C')$: This algorithm takes as input two ciphertexts C, C' in one of G, H, or G_t. Choose $g_1 \overset{\text{R}}{\leftarrow} G_1$ and $h_1 \overset{\text{R}}{\leftarrow} H_1$, and do the following:

1. If $C, C' \in G$, output $C \cdot C' \cdot g_1 \in G$.
2. If $C, C' \in H$, output $C \cdot C' \cdot h_1 \in H$.
3. If $C, C' \in G_t$, output $C \cdot C' \cdot e(g, h_1) \cdot e(g_1, h) \in G_t$.

$\mathsf{Decrypt}(SK, C)$: The input ciphertext C can be an element of G, H, or G_t.

1. If $C \in G$, output $M \leftarrow \log_{\pi_1(g)}(\pi_1(C))$.
2. If $C \in H$, output $M \leftarrow \log_{\pi_2(h)}(\pi_2(C))$.
3. If $C \in G_t$, output $M \leftarrow \log_{\pi_t(e(g,h))}(\pi_t(C))$.

It is clear that if C, C' are encryptions of M, M' respectively, then the Add algorithm gives a correctly distributed encryption of $M + M'$. Furthermore, it follows from the bilinear property of the pairing that if $C_A \in G$, $C_B \in H$ are the left and right halves of encryptions of M, M' respectively, then the $\mathsf{Multiply}$ algorithm gives a correctly distributed encryption of $M \cdot M'$. Since there is no pairing on G_t we can only perform the multiplication once.

Correctness of decryption of ciphertexts in G and H follows from the fact that G_1, H_1 are in the kernels of π_1, π_2, respectively. Correctness of decryption of ciphertexts in G_t follows from both "projecting" properties of $\mathcal{G}$; for example, we have $\pi_t(e(g, h_1)) = e(\pi_1(g), \pi_2(h_1)) = e(\pi_1(g), 1) = 1$.

Step 2 of the conversion process is to translate the security assumptions to asymmetric bilinear groups. In this case, semantic security of ciphertexts in G follows from the subgroup decision assumption on the left for $\mathcal{G}$. Intuitively, if $\mathcal{G}$ satisfies the subgroup decision assumption on the left, then an adversary cannot distinguish the real system from a "fake" system in which $g \in G_1$. Semantic security then follows from the fact that in the fake system the ciphertext element C_A will be a uniformly random element of G_1 and thus will contain no information about the message M. The same argument holds for ciphertexts in H, and semantic security of ciphertexts in G_t follows from semantic security in G and H. For further details see [6, Theorem 3.1].

Step 3 is to translate the assumption to prime-order groups. Since the security proof uses no intrinsic properties of the groups G and H, it carries over to our more general setting. This is the main result of our construction, so we record it as a theorem; the proof follows directly from [6, Theorem 3.1] and Proposition 3.4.

Theorem 5.1. *Let $\mathcal{P}$ be a prime-order bilinear group generator, and let $\mathcal{G}_P$ be the projecting bilinear group generator constructed from $\mathcal{P}$ in Example 3.3. If $\mathcal{P}$ satisfies the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, then the BGN cryptosystem instantiated with $\mathcal{G} = \mathcal{G}_P$ is semantically secure.*

When instantiated with either $\mathcal{G}_{BGN}$ or $\mathcal{G}_P$, decryption in the BGN system requires taking discrete logarithms in a group of large prime order. Thus to achieve efficient decryption the message space must be small (i.e., logarithmic in the group size). It is an open problem to find a bilinear group generator $\mathcal{G}$ for which the subgroup decision assumption holds and for which discrete logarithms can be computed in a subset of $\pi_1(G)$ whose size is a constant fraction of the full group order.

If we carry out the tensor product construction described in Section 3.1 for any k and $n \geq k+1$, we obtain an instantiation of the BGN cryptosystem whose security depends on the k-Linear assumption. Since ciphertexts will consist of n elements of $\mathbb{G}_1$ or $\mathbb{G}_2$ or n^2 elements of $\mathbb{G}_t$, these systems will in general be less efficient than the system constructed using $\mathcal{G}_P$, which has $(n, k) = (2, 1)$. We do note, however, that if $k \geq 2$ we can use a group with a symmetric pairing, in which case the Encrypt algorithm needs only to output the ciphertext C_A.

Okamoto and Takashima [27, Section 5] also suggest instantiating the BGN system using products of cyclic groups, but their system does not provide the product functionality on ciphertexts.

6 Application 2: Traitor Tracing

Traitor tracing systems, introduced by Chor, Fiat, and Naor [12], allow content distributors to identify pirates. In such a system, the distributor broadcasts encrypted content to N legitimate users, each of whom has a secret key that allows the user to decrypt. If an unauthorized user creates a "pirate decoder" that can decrypt content, the distributor can identify (or "trace") which user's key was compromised simply by attempting to decrypt certain messages with the pirate decoder.

Boneh, Sahai, and Waters [7] devised a traitor tracing system that is fully collusion resistant (i.e., even a pirate with all but one secret key can be traced) and has ciphertext length $O(\sqrt{N})$, where N is the number of users in the system. They build the system from a new primitive called *private linear broadcast encryption* system (or PLBE). Formal definitions of the syntax and security of a PLBE scheme appear in Appendix B. After reducing the construction of a traitor tracing scheme to the construction of a PLBE scheme, Boneh et al. devise a PLBE scheme using bilinear groups of composite order and show it secure under three assumptions in bilinear groups.

In this section we apply our framework to convert the Boneh et al. PLBE scheme to prime-order groups. When we instantiate the scheme using the bilinear group generator $\mathcal{G}_{2C}$ of Example 3.8, we obtain a system that is secure under the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$ and an assumption called the "3-party Diffie-Hellman assumption" in $\mathbb{G}_1$.

6.1 Construction

Step 1 is to write the original system in a general context. In the system users are indexed as entries in an $m \times m$ matrix. In the symmetric setting ciphertexts contain three group elements $R_x, \tilde{R}_x, A_x \in G$ and $B_x \in G_t$ for each row x in the matrix, and two group elements $C_y, \tilde{C}_y \in G$ for each column y in the matrix. User (x, y) has a key $K_{x,y} \in G$ and decrypts by computing

$$M = B_x \cdot e(C_y, R_x) \cdot e(K_{x,y}, A_x)^{-1} \cdot e(\tilde{C}_y, \tilde{R}_x)^{-1}.$$

When we convert to the asymmetric situation, we must take one argument of each of the three pairings to be in G and the other argument to be in H. We choose to assign the first argument of each pairing to G and the second argument to H. We note that the correctness properties of the system depend on cancellation between different subgroups when paired, so we must use a cancelling pairing. The scheme is as follows.

$\mathsf{Setup}(\lambda, n)$**:** The Setup algorithm takes as input a security parameter λ and a positive integer $n = m^2$ that is the number of users in the system. Let $\mathcal{G}$ be a 2-cancelling bilinear group generator (Definition 3.5) such that each output group G_j, H_j is cyclic of prime order p_j for $j = 1, 2$. Do the following.

1. Compute $(G, G_1, G_2, H, H_1, H_2, G_t, e) \xleftarrow{\text{R}} \mathcal{G}(\lambda)$. Let $N = \operatorname{lcm}(p_1, p_2)$.
2. Choose $g_1 \xleftarrow{\text{R}} G_1$, $f_2, g_2 \xleftarrow{\text{R}} G_2$, $h_1 \xleftarrow{\text{R}} H_1$, and $h_2, k_2 \xleftarrow{\text{R}} H_2$ for $i = 1, 2$.
3. Choose $r_1, \ldots, r_m, c_1, \ldots, c_m, \alpha_1, \ldots, \alpha_m, \beta, \gamma \xleftarrow{\text{R}} \mathbb{Z}_N$.
4. Set $f_1 = g_1^\gamma$, $k_1 = h_1^\gamma$, $h = h_1 h_2$, $k = k_1 k_2$.
5. Output the public parameters (G, H, G_t, e, N), along with

$$PK = \begin{bmatrix} f = f_1 f_2,\ g = g_1 g_2,\ E = h_1^\beta,\ E_1 = h_1^{\beta r_1}, \ldots, E_m = h_1^{\beta r_m},\ F_1 = k_1^{\beta r_1}, \ldots, F_m = k_1^{\beta r_m}, \\ P_1 = e(g_1, h_1)^{\beta \alpha_1}, \ldots, P_m = e(g_1, h_1)^{\beta \alpha_m}, Q_1 = g^{c_1}, \ldots, Q_m = g^{c_m} \end{bmatrix}.$$

The private key for user (x, y) is $K_{x,y} = g^{\alpha_x + r_x c_y} \in G$. The master secret key SK consists of the elements chosen in Step (2) and the exponents chosen in Step (3) above.

$\mathsf{TrEncrypt}(SK, M, (i, j)$**:** This algorithm encrypts the message $M \in G_t$ to the subset of receivers (x, y) with either $x > i$ or $x = i$ and $y \geq j$. Choose $t, w_1, \ldots, w_m, s_1, \ldots, s_m, z_1, \ldots, z_j \xleftarrow{\text{R}} \mathbb{Z}_N$ and $(v_{1,1}, v_{1,2}, v_{1,3}), \ldots, (v_{m,1}, v_{m,2}, v_{m,3}) \xleftarrow{\text{R}} \mathbb{Z}_N^3$.

For each row x output four ciphertext components $R_x, R_x, A_x \in H$ and $B_x \in G_t$ as follows:

$$\begin{array}{lllll} \text{if } x > i: & R_x = h_1^{s_x r_x}, & \tilde{R}_x = k_1^{s_x r_x}, & A_x = h_1^{s_x t}, & B_x = M e(g_1, h)^{\alpha_x s_x t} \\ \text{if } x = i: & R_x = h^{s_x r_x}, & \tilde{R}_x = k^{s_x r_x}, & A_x = h^{s_x t}, & B_x = M e(g, h)^{\alpha_x s_x t} \\ \text{if } x < i: & R_x = h^{v_{x,1}}, & \tilde{R}_x = k^{v_{x,1}}, & A_x = h^{v_{x,2}}, & B_x = M e(g, h)^{v_{x,3}} \end{array}$$

For each column y output two ciphertext components $C_y, \tilde{C}_y \in G$ as follows:

$$\begin{array}{lll} \text{if } y \geq j: & C_y = g^{c_y t} f^{w_y}, & \tilde{C}_y = g^{w_y} \\ \text{if } y < j: & C_y = g^{c_y t} g_1^{z_y} f^{w_y}, & \tilde{C}_y = g^{w_y} \end{array}$$

$\mathsf{Encrypt}(PK, M)$**:** This algorithm encrypts message $M \in G_t$ to all recipients. Choose $t, w_1, \ldots, w_m$, $s_1, \ldots, s_m \xleftarrow{\text{R}} \mathbb{Z}_N$. For each row x output four ciphertext components $R_x, R_x, A_x \in H$ and $B_x \in G_t$ as follows:

$$R_x = E_x^{s_x}, \quad \tilde{R}_x = F_x^{s_x}, \quad A_x = E^{s_x t}, \quad B_x = M P_x^{s_x t}.$$

For each column y output two ciphertext components $C_y, \tilde{C}_y \in G$ as follows:

$$C_y = Q_y^t f^{w_y}, \quad \tilde{C}_y = g^{w_y}.$$

$\mathsf{Decrypt}((x, y), K_{x,y}, C, PK)$**:** Output

$$B_x \cdot \left(\frac{e(C_y, R_x)}{e(K_{x,y}, A_x) e(\tilde{C}_y, \tilde{R}_x)} \right).$$

The cancelling property of the pairing implies that if the ciphertext was created from the tracing algorithm $\mathsf{TrEncrypt}$ with parameters (i, j) then the result of decryption with key $K_{x,y}$ is M if $x > 1$ or $x = 1$ and $y \geq j$. If the ciphertext was created using $\mathsf{Encrypt}$ then all parties can decrypt with their secret key to recover M.

6.2 Security Assumptions

Step 2 is to convert the assumptions used in proving security of the Boneh et al. PLBE scheme to our more general framework. There are three assumptions: the subgroup decision assumption, the *3-party Diffie-Hellman assumption*, and the *bilinear subgroup decision assumption.*

The subgroup decision assumption we have discussed previously (see Section 2). In examining Boneh et al.'s security proof, we see that the challenge element T in the subgroup decision problem is used only to construct ciphertext elements $R_x, \tilde{R}_x, A_x \in H$. Thus the assumption needs to hold only in H.

In the symmetric version of the 3-party Diffie-Hellman assumption the adversary is given g, g^a, g^b, g^c, T in a certain subgroup of G and asked to determine whether $T = g^{abc}$ or T is random. The fixed challenge elements g, g^a, g^b, g^c are used in the simulations to form all ciphertext components, so we need to duplicate them in G and H. The "variable" element T is only used to construct one component, but the proof involves a number of hybrid games and T may appear in different groups in different games. Thus we define a "left" and a "right" version of the assumption in the asymmetric setting.

Definition 6.1 (3-party decision Diffie-Hellman assumption). Let $\mathcal{G}$ be a 2-cancelling bilinear group generator such that for $i = 1, 2$ the output groups G_i and H_i are of prime order p_i. We define the following distribution:

$$\mathbb{G} = (G, H, G_t, e, N := \operatorname{lcm}(p_1, p_2)) \xleftarrow{\text{R}} \mathcal{G}(\lambda),\ g_2 \xleftarrow{\text{R}} G_2,\ h_2 \xleftarrow{\text{R}} H_2, a, b, c \xleftarrow{\text{R}} \mathbb{Z}_{p_2},$$
$$Z \leftarrow \big(g_2,\ g_2^a,\ g_2^b,\ g_2^c,\ h_2,\ h_2^a,\ h_2^b,\ h_2^c\big),$$
$$T_0 \leftarrow g_2^{abc},\ T_1 \xleftarrow{\text{R}} G_2.$$

We define the *advantage* of an algorithm $\mathcal{A}$ in solving the *3-party decision Diffie-Hellman assumption on the left* to be

$$\text{3DDH}_\text{L}\text{-Adv}[\mathcal{A}, \mathcal{G}] = \Big| \Pr[\mathcal{A}(\mathbb{G}, Z, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_1) = 1] \Big|.$$

We say that $\mathcal{G}$ *satisfies the 3-party decision Diffie-Hellman assumption on the left* if $\text{3DDH}_\text{L}\text{-Adv}[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$.

We define the *3-party decision Diffie-Hellman assumption on the right* and $\text{3DDH}_\text{R}\text{-Adv}[\mathcal{A}, \mathcal{G}]$ analogously, with $T_0 \leftarrow h_2^{abc}$ and $T_1 \xleftarrow{\text{R}} H_2$.

In the symmetric setting the bilinear subgroup decision problem is to distinguish an element of the subgroup $e(G_2, G)$ from a random element in G_t when given generators for G_1 and G_2. Since we require G_i and H_i to have the same order in the asymmetric case, the assumption translates directly to the asymmetric setting.

Definition 6.2 (Bilinear subgroup decision assumption). Let $\mathcal{G}$ be a 2-cancelling bilinear group generator such that for $i = 1, 2$ the output groups G_i and H_i are of prime order p_i. We define the following distribution:

$$\mathbb{G} = (G, H, G_t, e, N := \operatorname{lcm}(p_1, p_2)) \xleftarrow{\text{R}} \mathcal{G}(\lambda),\ f_1, g_1 \xleftarrow{\text{R}} G_1,\ f_2, g_2 \xleftarrow{\text{R}} G_2,\ h_1 \xleftarrow{\text{R}} H_1,\ h_2 \xleftarrow{\text{R}} H_2,$$
$$Z \leftarrow \big(g_1, g_2, h_1, h_2\big),$$
$$T_0 \leftarrow e(f_2, h_1 h_2),\ T_1 \leftarrow e(f_1 f_2, h_1 h_2).$$

We define the *advantage* of an algorithm $\mathcal{A}$ in solving the *bilinear subgroup decision assumption* to be

$$\text{BSD-Adv}[\mathcal{A}, \mathcal{G}] = \Big| \Pr[\mathcal{A}(\mathbb{G}, Z, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_1) = 1] \Big|.$$

We say that $\mathcal{G}$ *satisfies the bilinear subgroup decision assumption* if BSD-Adv$[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$.

The reason the BSD assumption does not reduce to the ordinary subgroup decision assumption (with G_1, G_2 switched) is that in the latter the challenger is not given generators of both G_1 and G_2.

With these definitions, we can now state the security theorem for the generalized Boneh-Sahai-Waters scheme. The original proof applies in our more general context.

Theorem 6.3. *Suppose that $\mathcal{G}$ satisfies the subgroup decision assumption on the right, the bilinear subgroup decision assumption, and the 3-party Diffie-Hellman assumptions on the left and right. Then the generalized Boneh-Sahai-Waters PLBE scheme is secure.*

6.3 Security in Prime-Order Groups

Step 3 in the conversion procedure is to translate the security assumptions to prime-order groups; here we use the 2-cancelling bilinear group generator $\mathcal{G}_{2C}$ of Example 3.8. The subgroup decision assumption holds for $\mathcal{G}_{2C}$ by Proposition 3.10. The challenge element for the bilinear subgroup decision assumption is $e(g, h) \in G_t$ with $h \overset{\text{R}}{\leftarrow} H$ and either $g \overset{\text{R}}{\leftarrow} G$ or $g \overset{\text{R}}{\leftarrow} G_1$. Since the group G_t produced by $\mathcal{G}_{2C}$ is cyclic, this assumption holds unconditionally for $\mathcal{G}_{2C}$. The 3-party DDH assumption has a natural analogue in prime-order groups, which we record here.

Definition 6.4. Let $\mathcal{P}$ be a prime-order bilinear group generator that outputs three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ and a pairing $\hat{e}$. If $\mathcal{A}$ is an algorithm that takes as input five elements of $\mathbb{G}_1$, we define the *advantage* of $\mathcal{A}$ in solving the *3-party decision Diffie-Hellman problem in* $\mathbb{G}_1$ to be

$$\begin{aligned} \text{3DDH}_{\mathbb{G}_1}\text{-Adv}[\mathcal{A}, \mathcal{P}] = \Big| & \Pr[\mathcal{A}(\hat{\mathbb{G}}, g, g^a, g^b, g^c, g^{abc}) = 1 : \hat{\mathbb{G}} \overset{\text{R}}{\leftarrow} \mathcal{P}, g \overset{\text{R}}{\leftarrow} \mathbb{G}_1, a, b, c \overset{\text{R}}{\leftarrow} \mathbb{F}_p] \\ & - \Pr[\mathcal{A}(\hat{G}, g, g^a, g^b, g^c, g^d) = 1 : \hat{\mathbb{G}} \overset{\text{R}}{\leftarrow} \mathcal{P}, g \overset{\text{R}}{\leftarrow} \mathbb{G}_1, a, b, c, d \overset{\text{R}}{\leftarrow} \mathbb{F}_p] \Big|, \end{aligned}$$

and similarly for $\text{3DDH}_{\mathbb{G}_2}$-Adv$[\mathcal{A}, \mathcal{P}]$. We say that $\mathcal{G}$ *satisfies the 3-party DDH assumption in* $\mathbb{G}_1$ if $\text{3DDH}_{\mathbb{G}_1}$-Adv$[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$ (and similarly for $\mathbb{G}_2$).

In Appendix D we show that the 3-party DDH assumption holds in the generic group model. The security theorem for PLBE in prime-order groups is as follows.

Theorem 6.5. *Let $\mathcal{P}$ be a prime-order bilinear group generator, and let $\mathcal{G}_{2C}$ be the 2-cancelling bilinear group generator constructed from $\mathcal{P}$ as in Example 3.8. If $\mathcal{P}$ satisfies the DDH assumption in $\mathbb{G}_2$ and the 3-party DDH assumptions in $\mathbb{G}_1$ and $\mathbb{G}_2$, then the Boneh-Sahai-Waters PLBE system is secure when instantiated with $\mathcal{G} = \mathcal{G}_{2C}$.*

Proof. It suffices to show that the three assumptions listed in Theorem 6.3 hold for $\mathcal{G}_{2C}$. First, by Proposition 3.10, since $\mathcal{P}$ satisfies the DDH assumption in $\mathbb{G}_2$, the generator $\mathcal{G}_{2C}$ satisfies the SDP assumption on the right. Next, we observe that the bilinear subgroup decision assumption (Definition 6.2) holds unconditionally. To see this, we write the f_1, f_2, h_1, h_2 generated in the assumption as

$$f_1 = (\alpha_1, \alpha_1^x), \quad f_2 = (\alpha_2, \alpha_2^y), \quad h_1 = (\beta_1^{-y}, \beta_1), \quad h_2 = (\beta_2^{-x}, \beta_2)$$

for $\alpha_1, \alpha_2 \xleftarrow{R} \mathbb{G}_1$ and $\beta_1, \beta_2 \xleftarrow{R} \mathbb{G}_2$. Then $e(f_2, h_1h_2) = \hat{e}(\alpha_2, \beta_2)^{(y-x)}$, while $e(f_1f_2, h_1h_2) = \hat{e}(\alpha_1, \beta_1)^{(x-y)}\hat{e}(\alpha_2, \beta_2)^{(y-x)}$. Since the α_i, β_i are chosen uniformly at random, both $e(f_2, h_1h_2)$ and $e(f_1f_2, h_1h_2)$ are uniformly distributed in $G_t = \mathbb{G}_t$, so no adversary $\mathcal{A}$ can have a nonzero advantage in solving the BSD problem.

Now suppose we are given a 3-party DDH challenge for $\mathcal{P}$; that is, a tuple $(g, g^a, g^b, g^c, h, h^a, h^b, h^c, T)$ with $g, T \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. We wish to create a properly distributed 3-party DDH challenge for $\mathbb{G}_{2C}$. We choose $g_1 \xleftarrow{R} \mathbb{G}_1$, $h_1 \xleftarrow{R} \mathbb{G}_2$, and $x, y \xleftarrow{R} \mathbb{F}_p$. If we set

$$G_1 = \langle(g_1, g_1^x)\rangle, \quad G_2 = \langle(g, g^y)\rangle, \quad H_1 = \langle(h_1^{-y}, h_1)\rangle, \quad H_2 = \langle(h^{-x}, h)\rangle,$$

then these groups are distributed exactly as in the construction of $\mathcal{G}_{2C}$. Furthermore, if $\gamma = (g, g^y)$ then we can construct $\gamma^a, \gamma^b, \gamma^c \in G$, and similarly for $\eta = (h^{-x}, h), \eta^a, \eta^b, \eta^c \in H$. Finally, the challenge element $\tau \in G$ is constructed as (T, T^y). Thus we have constructed a properly distributed 3-party DDH challenge on the left for $\mathcal{G}_{2C}$, so any adversary $\mathcal{A}$ that solves the 3-party DDH problem on the left for $\mathcal{G}_{2C}$ can be used to solve the 3-party DDH problem in $\mathbb{G}_1$ for $\mathcal{P}$. An identical analysis holds on the right. $\square$

7 Application 3: Predicate Encryption

In a *predicate encryption scheme*, a ciphertext C_I is associated with an *attribute* I and each user has one or more keys K_f corresponding to *predicates* f. The key K_f can be used to decrypt the ciphertext C_I if and only if $f(I) = 1$. The scheme is secure if an adversary holding key K_f and ciphertext C_I with $f(I) = 0$ learns nothing about either the plaintext corresponding to C_I or the attribute I. (Formal definitions of syntax and security are in Appendix C.)

Katz, Sahai, and Waters [23] construct a predicate encryption scheme using bilinear groups whose order is a product N of three distinct primes. In the system both predicates and attributes correspond to vectors of length n over $\mathbb{Z}_N$. Predicates are evaluated using dot products: the predicate vector $\vec{v}$ evaluates to 1 on the attribute vector $\vec{x}$ if and only if $\vec{x} \bullet \vec{v} = 0$.[3] The security of the system is based on two complex (though constant-size) assumptions in composite-order bilinear groups. To support their assumptions, Katz et al. show that breaking either assumption in the generic group model reveals the factorization of the group order N.

In this section, we translate the predicate encryption scheme of Katz et al. into our more general framework. The security assumptions translate into similarly complex assumptions; however, when instantiated using prime-order bilinear groups they also hold in the generic group model.

7.1 Construction.

Step 1 is to write the procedure using a bilinear group generator with an asymmetric pairing. We note that the decryption algorithm requires pairing ciphertexts with keys. Thus we may streamline the setup by computing ciphertexts in G and keys in H. Since public key elements are used to form ciphertexts, they will also be in G. Here the correctness properties depend on cancellation between various subgroups; specifically, we need to choose a bilinear group generator $\mathcal{G}$ that is 3-cancelling.

[3] Here we use a large $\bullet$ to indicate vector dot product and a small $\cdot$ to indicate multiplication in a group.

We now give a detailed description of the "full-fledged" predicate encryption scheme of Katz et al. [23, Appendix C] in our framework.

$\mathsf{Setup}(\lambda, n)$**:** The Setup algorithm takes as input a security parameter λ and a positive integer n that is the length of vectors that represent predicates and attributes. Let $\mathcal{G}$ be a 3-cancelling bilinear group generator (Definition 3.5) such that each output group G_j, H_j is cyclic of prime order p_j for $j = 1$ to 3. Do the following.

1. Compute $(G, G_1, G_2, G_3, H, H_1, H_2, H_3, G_t, e) \overset{\text{R}}{\leftarrow} \mathcal{G}(\lambda)$.
2. Choose $g_i \overset{\text{R}}{\leftarrow} G_i$ and $h_i \overset{\text{R}}{\leftarrow} H_i$ for $i = 1, 2, 3$.
3. Choose $R_0 \overset{\text{R}}{\leftarrow} G_3$, $\gamma \overset{\text{R}}{\leftarrow} \mathbb{Z}_{p_1}$, $h_0 \overset{\text{R}}{\leftarrow} H_1$.
4. For $i = 1, \ldots, n$, choose $u_{1,i}, u_{2,i} \overset{\text{R}}{\leftarrow} \mathbb{Z}_{p_1}$ and $R_{1,i}, R_{2,i} \overset{\text{R}}{\leftarrow} G_3$.
5. Output the public parameters $(G, H, G_t, e, N := \text{lcm}(p_1, p_2, p_3))$, along with

$$PK = \big(g_1, \quad g_3, \quad Q = g_2 R_0, \quad P = e(g_1, h_0)^\gamma, \quad \{L_{1,i} = g_1^{u_{1,i}} R_{1,i}, \quad L_{2,i} = g_1^{u_{2,i}} R_{2,i}\}_{i=1}^n\big).$$

The master secret key is $SK = (h_1, h_2, h_3, h_0^{-\gamma}, \{u_{1,i}, u_{2,i}\}_{i=1}^n)$.

$\mathsf{KeyGen}(SK, \vec{v})$**:** Let $\vec{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_N^n$ be a predicate. Choose $r_{1,i}, r_{2,i} \overset{\text{R}}{\leftarrow} \mathbb{Z}_N$ for $i = 1$ to n, choose $f_1, f_2 \overset{\text{R}}{\leftarrow} \mathbb{Z}_N$, and choose $S \overset{\text{R}}{\leftarrow} H_2 \times H_3$. Output the secret key corresponding to predicate $\vec{v}$ as

$$SK_{\vec{v}} = \left(K_0 = S \cdot h_0^{-\gamma} \cdot \prod_{i=1}^n h_1^{-r_{1,i}u_{1,i} - r_{2,i}u_{2,i}}, \quad \left\{ K_{1,i} = h_1^{r_{1,i}} \cdot h_2^{f_1 \cdot v_i}, \quad K_{2,i} = h_1^{r_{2,i}} \cdot h_2^{f_2 \cdot v_i} \right\}_{i=1}^n \right).$$

$\mathsf{Encrypt}(PK, \vec{x}, M)$**:** Let $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_N^n$ be an attribute, and $M \in G_t$ be a message. Choose $s, \alpha, \beta \overset{\text{R}}{\leftarrow} \mathbb{Z}_N$ and random $t_{1,i}, t_{2,i} \overset{\text{R}}{\leftarrow} \mathbb{Z}_N$ for $i = 1, \ldots, n$. Output the ciphertext

$$C = \left(C' = M \cdot P^s, \quad C_0 = g_1^s, \quad \left\{ C_{1,i} = L_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot g_3^{t_{1,i}}, \quad C_{2,i} = L_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot g_3^{t_{2,i}} \right\}_{i=1}^n \right).$$

$\mathsf{Decrypt}(SK_{\vec{v}}, C)$**:** Output

$$C' \cdot e(C_0, K_0) \cdot \prod_{i=1}^n e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}).$$

The cancelling property of the pairing implies that when the ciphertexts and secret keys are formed correctly, the decryption algorithm outputs $M \cdot e(g_1, h_1)^{(\alpha f_1 + \beta f_2)(\vec{x} \bullet \vec{v})}$. If $\vec{x} \bullet \vec{v} = 0$ then we obtain M as required; otherwise we obtain M multiplied by $e(g_1, h_1)$ raised to a random power. As in the original scheme, we can restrict the message space to some efficiently recognizable set of negligible density in G_t and output an error when the decryption does not lie in this space.

7.2 Security

Step 2 is to translate the assumptions used to prove the scheme secure into our more general context. There are two assumptions, which we call Assumptions 1 and 2. Both can be seen as variants of the subgroup decision problem: the adversary is given a set of elements $Z \subset G$ with some specified relationship, and is asked to determine whether a challenge element T is in a proper subgroup of G.

The following two assumptions, stated in symmetric groups, are used to prove the security of the Katz-Sahai-Waters predicate encryption scheme.

Assumption 1. Let $\mathcal{G}$ be a 3-cancelling bilinear group generator such that for $i = 1, 2, 3$ the output groups G_i and H_i are equal to each other and of prime order p_i. We define the following distribution:

$$\begin{gathered}
\mathbb{G} = (G, G_t, e, N := \operatorname{lcm}(p_1, p_2, p_3)) \xleftarrow{\text{R}} \mathcal{G}(\lambda),\ g_1 \xleftarrow{\text{R}} G_1,\ g_2 \xleftarrow{\text{R}} G_2,\ g_3 \xleftarrow{\text{R}} G_3, \\
Q_1, Q_2, Q_3 \xleftarrow{\text{R}} G_2,\ R_1, R_2, R_3 \xleftarrow{\text{R}} G_3,\ a, b, s \xleftarrow{\text{R}} \mathbb{Z}_{p_1}, \\
Z \leftarrow \big(g_1,\ g_3,\ g_2R_1,\ g_1^b,\ g_1^{b^2},\ g_1^a g_2,\ g_1^{ab}Q_1,\ g_1^s,\ g_1^{bs}Q_2R_2\big), \\
T_0 \leftarrow g_1^{b^2s}R_3,\ T_1 \leftarrow g_1^{b^2s}Q_3R_3.
\end{gathered}$$

We define the *advantage* of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be

$$\text{A1-Adv}[\mathcal{A}, \mathcal{G}] = \Big|\Pr[\mathcal{A}(\mathbb{G}, Z, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_1) = 1]\Big|.$$

We say that $\mathcal{G}$ *satisfies Assumption 1* if A1-Adv$[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$.

Assumption 2. Let $\mathcal{G}$ be as in Assumption 1. We define the following distribution:

$$\begin{gathered}
\mathbb{G} = (G, G_t, e, N := \operatorname{lcm}(p_1, p_2, p_3)) \xleftarrow{\text{R}} \mathcal{G}(\lambda),\ g_1 \xleftarrow{\text{R}} G_1,\ g_2 \xleftarrow{\text{R}} G_2,\ g_3 \xleftarrow{\text{R}} G_3, \\
P \xleftarrow{\text{R}} G_1,\ Q_1, Q_2 \xleftarrow{\text{R}} G_2,\ \gamma \xleftarrow{\text{R}} \mathbb{Z}_{p_1}, s \xleftarrow{\text{R}} \mathbb{Z}_{p_2}, \\
Z \leftarrow \big(g_1,\ g_2,\ g_3,\ P,\ g_1^s,\ P^sQ_1,\ g_1^\gamma Q_2,\ e(g_1, P)^\gamma\big), \\
T_0 \leftarrow e(g_1, P)^{\gamma s},\ T_1 \xleftarrow{\text{R}} G_t.
\end{gathered}$$

We define the *advantage* of an algorithm $\mathcal{A}$ in breaking Assumption 2 to be

$$\text{A2-Adv}[\mathcal{A}, \mathcal{G}] = \Big|\Pr[\mathcal{A}(\mathbb{G}, Z, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_1) = 1]\Big|.$$

We say that $\mathcal{G}$ *satisfies Assumption 2* if A2-Adv$[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$.

To translate these assumptions to the asymmetric setting with a pairing $e : G \times H \to G_t$, we must look into the "guts" of the reductions in the original security proof. In particular, we must determine whether each group element in a challenge (Z, T) is used by the simulator to produce a key element in G, a ciphertext element in H, or both. We do not repeat the details of the simulations, but give our conclusions in Table 2 below.

Table 2. Allocation of group elements in asymmetric versions of Assumptions 1 and 2.

	used to produce ciphertext elements in G	used to produce key elements in H	in G_t
Assumption 1	$g_1,\ g_3,\ g_2R_1,\ g_1^b,\ g_1^{b^2},\ g_1^s,\ g_1^{bs}Q_2R_2,\ T$	$g_1,\ g_3,\ g_2R_1,\ g_1^b,\ g_1^{b^2},\ g_1^a g_2,\ g_1^{ab}Q_1$	
Assumption 2	$g_1,\ g_2,\ g_3,\ P,\ g_1^s,\ P^sQ_1$	$g_1,\ g_2, g_3,\ g_1^\gamma Q_2$	$e(g_1, P)^\gamma,\ T$

Thus to modify Assumption 1 for the asymmetric setting, we must also choose $h_i \xleftarrow{\text{R}} H_i$, $Q_1' \xleftarrow{\text{R}} H_2$, and $R_1' \xleftarrow{\text{R}} H_3$, and set

$$Z \leftarrow \big(g_1,\ g_3,\ g_2R_1,\ g_1^b,\ g_1^{b^2},\ g_1^s,\ g_1^{bs}Q_2R_2, h_1,\ h_3,\ h_2R_1',\ h_1^b,\ h_1^{b^2},\ h_1^a h_2,\ h_1^{ab}Q_1'\big). \tag{7.1}$$

To modify Assumption 2 for the asymmetric setting, we choose $h_i \xleftarrow{\text{R}} H_i$ and $Q_2' \xleftarrow{\text{R}} H_2$, and set

$$Z \leftarrow \big(g_1,\ g_2,\ g_3,\ P,\ g_1^s,\ P^sQ_1,\ h_1,\ h_2,\ h_3,\ h_1^\gamma Q_2',\ e(P, h_1)^\gamma\big). \tag{7.2}$$

7.3 Security in Prime-Order Groups

Step 3 of the conversion procedure is to instantiate the scheme using prime-order groups. We use the 3-cancelling bilinear group generator $\mathcal{G}_{3C}$ of Example 3.7. Let $\mathcal{P}$ be the prime-order group generator from which we construct $\mathcal{G}_{3C}$. We now translate the asymmetric versions of Assumptions 1 and 2 explicitly to this setting, obtaining two (constant-size) assumptions for $\mathcal{P}$. For the first assumption, the set of challenge elements in the general setting is given by (7.1).

Assumption 3. Let $\mathcal{P}$ be a prime-order bilinear group generator. We define the following distribution:

$$\begin{gathered}
\mathbb{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e}) \xleftarrow{\text{R}} \mathcal{P}(\lambda),\ g_1, g_2, g_3 \xleftarrow{\text{R}} \mathbb{G}_1,\ h_1, h_2, h_3 \xleftarrow{\text{R}} \mathbb{G}_2, \\
x, y, z, u, v, w, a, b, s, c_1, c_2, c_3, d_1, d_2, d_3 \xleftarrow{\text{R}} \mathbb{F}_p, \\
Z \leftarrow \big((g_1, g_1^x, g_1^u),\ (g_3, g_3^z, g_3^w),\ (g_2 g_3^{d_1}, g_2^y g_3^{zd_1}, g_2^v g_3^{wd_1}),\ (g_1^b, g_1^{xb}, g_1^{ub}),\ (g_1^{b^2}, g_1^{xb^2}, g_1^{ub^2}),\ (g_1^s, g_1^{xs}, g_1^{us}), \\
(g_1^{bs} g_2^{c_2} g_3^{d_2}, g_1^{xbs} g_2^{yc_2} g_3^{zd_2}, g_1^{ubs} g_2^{vc_2} g_3^{wd_2}),\quad (h_1^{zv-yw}, h_1^{w-v}, h_1^{y-z}),\ (h_3^{yu-xv}, h_3^{v-u}, h_3^{x-y}), \\
(h_2^{zu-xv} h_3^{(yu-xv)d_1}, h_2^{w-u} h_3^{(v-u)d_1}, h_2^{zu-xw} h_3^{(yu-xv)d_1}),\ (h_1^{(zv-yw)b}, h_1^{(w-v)b}, h_1^{(y-z)b}), \\
(h_1^{(zv-yw)b^2}, h_1^{(w-v)b^2}, h_1^{(y-z)b^2}),\ (h_1^{(zv-yw)a} h_2^{zu-xw}, h_1^{(w-v)a} h_2^{w-u}, h_1^{(y-z)a} h_2^{z-x}), \\
(h_1^{(zv-yw)ab} h_2^{(zu-xw)c_1}, h_1^{(w-v)ab} h_2^{(w-u)c_1}, h_1^{(y-z)ab} h_2^{(z-x)c_1})\big). \\
T_0 \leftarrow (g_1^{b^2 s} g_3^{d_3}, g_1^{xb^2 s} g_3^{zd_3}, g_1^{ub^2 s} g_3^{wd_3}),\ T_1 \leftarrow (g_1^{b^2 s} g_2^{c_3} g_3^{d_3}, g_1^{xb^2 s} g_2^{yc_3} g_3^{zd_3}, g_1^{ub^2 s} g_2^{vc_3} g_3^{wd_3}).
\end{gathered}$$

We define the *advantage* of an algorithm $\mathcal{A}$ in breaking Assumption 3 to be

$$\text{A3-Adv}[\mathcal{A}, \mathcal{G}] = \Big| \Pr[\mathcal{A}(\mathbb{G}, Z, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_1) = 1] \Big|.$$

We say that $\mathcal{G}$ *satisfies Assumption 3* if A3-Adv$[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$.

For the second assumption, the set of challenge elements in the general setting is given by (7.2).

Assumption 4. Let $\mathcal{P}$ be a prime-order bilinear group generator. We define the following distribution:

$$\begin{gathered}
\mathbb{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e}) \xleftarrow{\text{R}} \mathcal{P}(\lambda),\ g_1, g_2, g_3 \xleftarrow{\text{R}} \mathbb{G}_1,\ h_1, h_2, h_3 \xleftarrow{\text{R}} \mathbb{G}_2,\ x, y, z, u, v, w, a, s, \gamma, c_1, c_2 \xleftarrow{\text{R}} \mathbb{F}_p, \\
Z \leftarrow \big((g_1, g_1^x, g_1^u),\ (g_2, g_2^y, g_2^v),\ (g_3, g_3^z, g_3^w),\ (g_1^a, g_1^{xa}, g_1^{ua}),\ (g_1^s, g_1^{xs}, g_1^{us}),\ (g_1^{as} g_2^{c_1}, g_1^{xas} g_2^{yc_1}, g_1^{uas} g_2^{vc_1}), \\
(h_1^{zv-yw}, h_1^{w-v}, h_1^{y-z}),\ (h_2^{zu-xw}, h_2^{w-u}, h_2^{z-x}),\ (h_3^{yu-xv}, h_3^{v-u}, h_3^{x-y}), \\
(h_1^{(zv-yw)\gamma} h_2^{(zu-xw)c_2}, h_1^{(w-v)\gamma} h_2^{(w-u)c_2}, h_1^{(y-z)\gamma} h_2^{(z-x)c_2}),\ e(g_1, h_1)^{(xw-xv+yu-yw-zu+zv)a\gamma}\big). \\
T_0 \leftarrow \hat{e}(g_1, h_1)^{(xw-xv+yu-yw-zu+zv)a\gamma s},\ T_1 \xleftarrow{\text{R}} \mathbb{G}_t.
\end{gathered}$$

We define the *advantage* of an algorithm $\mathcal{A}$ in breaking Assumption 4 to be

$$\text{A4-Adv}[\mathcal{A}, \mathcal{G}] = \Big| \Pr[\mathcal{A}(\mathbb{G}, Z, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_1) = 1] \Big|.$$

We say that $\mathcal{G}$ *satisfies Assumption 4* if A4-Adv$[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$.

We obtain the following security theorem.

Theorem 7.1. *Let $\mathcal{P}$ be a prime-order bilinear group generator, and let $\mathcal{G}_{3C}$ be the 3-cancelling bilinear group generator constructed from $\mathcal{P}$ in Example 3.7. If $\mathcal{P}$ satisfies Assumptions 3 and 4, then the Katz-Sahai-Waters predicate encryption scheme is secure when instantiated with $\mathcal{G} = \mathcal{G}_{2C}$.*

In Appendix D we show that Assumptions 3 and 4 hold in the generic group model. This result does not necessarily mean that the assumptions hold when instantiated with any specific group; however, it does suggest that we should favor these assumptions neither more nor less than the assumptions in composite-order groups from which they are derived.

8 Further Work

We expect that our framework can be used to create prime-order group instantiations of other cryptosystems that use composite-order bilinear groups. However, since our construction is not black box, the security proof of each cryptosystem will need to be checked to ensure that it is still valid in our more general framework. For example, the zero-knowledge proofs of Groth, Ostrovsky, and Sahai [19] use in an essential manner the fact that a group element can be paired with itself, so we cannot instantiate this system using an asymmetric pairing. However, we do expect that the system can be instantiated with a symmetric pairing under the decision linear assumption using the bilinear group generator $\mathcal{G}_L$ of Example 3.9.

As another example, the proof of the Lewko-Waters identity-based encryption system [24] uses in an essential way the fact that G has two subgroups with relatively prime order; thus our prime-order construction does not apply in this case. Lewko and Waters do give a version of their system in prime-order groups, with a different security proof under new assumptions. It remains an open problem to find a framework that incorporates the security proofs of both versions of the system.

Acknowledgments

The author thanks Dennis Hofheinz, Eike Kiltz, and Brent Waters for helpful discussions.

References

1. L. Ballard, M. Green, B. de Medeiros, and F. Monrose. "Correlation-resistant storage via keyword-searchable encryption." Cryptology ePrint Archive, Report 2005/417 (2005). Available at `http://eprint.iacr.org/2005/417`.
2. E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. "Recommendation for key management — Part 1: General (revised)." National Institute of Standards and Technology (2007). Available at `http://csrc.nist.gov/groups/ST/toolkit/documents/SP800-57Part1_3-8-07.pdf`.
3. P. Barreto and M. Naehrig. "Pairing-friendly elliptic curves of prime order." In *Selected Areas in Cryptography — SAC 2005*, Springer LNCS **3897** (2006), 319–331.
4. D. Boneh, X. Boyen, and E.-J. Goh. "Hierarchical identity based encryption with constant size ciphertext." In *Advances in Cryptology — Eurocrypt 2005*, Springer LNCS **3494** (2005), 440–456.
5. D. Boneh, X. Boyen, and H. Shacham. "Short group signatures." In *Advances in Cryptology — CRYPTO 2004*, Springer LNCS **3152** (2004), 41–55.
6. D. Boneh, E.-J. Goh, and K. Nissim. "Evaluating 2-DNF formulas on ciphertexts." In *Theory of Cryptography — TCC 2005*, Springer LNCS **3378** (2005), 325–341.
7. D. Boneh, A. Sahai, and B. Waters. "Fully collusion resistant traitor tracing with short ciphertexts and private keys." In *Advances in Cryptology — Eurocrypt 2006*, Springer LNCS **4004** (2006), 573–592. Full version available at `http://eprint.iacr.org/2006/045`.
8. D. Boneh and B. Waters. "Conjunctive, subset, and range queries on encrypted data." In *Theory of Cryptography — TCC 2007*, Springer LNCS **4392** (2007), 535–554.
9. W. Bosma, J. Cannon, and C. Playoust. "The Magma algebra system. I. The user language." *Journal of Symbolic Computation* **24** (1997), 235–265.

10. X. Boyen and B. Waters. "Full-domain subgroup hiding and constant-size group signatures." In *Public Key Cryptography — PKC 2007*, Springer LNCS **4450** (2007), 1–15.
11. F. Brezing and A. Weng. "Elliptic curves suitable for pairing based cryptography." *Designs, Codes and Cryptography* **37** (2005), 133–141.
12. B. Chor, A. Fiat, and M. Naor. "Tracing traitors." In *Advances in Cryptology — CRYPTO 1994*, Springer LNCS **839** (1994), 257–270.
13. R. Cramer and V. Shoup. "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack." *SIAM Journal on Computing* **33** (2003), 167–226.
14. S. Duquesne and T. Lange. "Pairing-based cryptography." In *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC, Boca Raton, FL (2006), 573–590.
15. D. Freeman, M. Scott, and E. Teske. "A taxonomy of pairing-friendly elliptic curves." To appear in *Journal of Cryptology* (2009). Available at `http://eprint.iacr.org/2006/372`.
16. S. Galbraith. "Pairings." In *Advances in Elliptic Curve Cryptography*, London Math. Soc. Lecture Note Ser. **317**. Cambridge University Press, Cambridge (2005), 183–213.
17. S. Galbraith and E. Verheul. "An analysis of the vector decomposition problem." In *Public Key Cryptography — PKC 2008*, Springer LNCS **4939** (2008), 308–327.
18. K. Gjøsteen. *Subgroup membership problems and public key cryptosystems.* Ph.D. dissertation, Norwegian University of Science and Technology (2004). Available at `http://ntnu.diva-portal.org/smash/get/diva2:121977/FULLTEXT01`.
19. J. Groth, R. Ostrovsky, and A. Sahai. "Perfect non-interactive zero knowledge for NP." In *Advances in Cryptology — Eurocrypt 2006*, Springer LNCS **4004** (2006), 339–358.
20. J. Groth and A. Sahai. "Efficient non-interactive proof systems for bilinear groups." In *Advances in Cryptology — Eurocrypt 2008*, Springer LNCS **4965** (2008), 415–432.
21. D. Hofheinz and E. Kiltz. "Secure hybrid encryption from weakened key encapsulation." In *Advances in Cryptology — CRYPTO 2007*, Springer LNCS **4622** (2007), 553–571.
22. R. A. Horn and C. R. Johnson. *Topics in matrix analysis.* Cambridge University Press, Cambridge (1991).
23. J. Katz, A. Sahai, and B. Waters. "Predicate encryption supporting disjunctions, polynomial equations, and inner products." In *Advances in Cryptology — Eurocrypt 2008*, Springer LNCS **4965** (2008), 146–162. Full version available at `http://eprint.iacr.org/2007/404`.
24. A. B. Lewko and B. Waters. "Fully secure HIBE with short ciphertexts." Cryptology ePrint Archive, Report 2009/482 (2009). Available at `http://eprint.iacr.org`.
25. A. Menezes, T. Okamoto, and S. Vanstone. "Reducing elliptic curve logarithms to logarithms in a finite field." *IEEE Transactions on Information Theory* **39** (1993), 1639–1646.
26. A. Miyaji, M. Nakabayashi, and S. Takano. "Characterization of elliptic curve traces under FR-reduction." In *Information Security and Cryptology — ICISC 2000*, Springer LNCS **2015** (2001), 90–108.
27. T. Okamoto and K. Takashima. "Homomorphic encryption and signatures from vector decomposition." In *Pairing-Based Cryptography — Pairing 2008*, Springer LNCS **5209** (2008), 57–74.
28. K. Rubin and A. Silverberg. "Using abelian varieties to improve pairing-based cryptography." *Journal of Cryptology* **22** (2009), 330–364.
29. M. Scott. Personal communication (17 February 2009).
30. H. Shacham. "A Cramer-Shoup encryption scheme from the Linear assumption and from progressively weaker Linear variants." Cryptology ePrint Archive, Report 2007/074 (2007). Available at `http://eprint.iacr.org/2007/074`.
31. H. Shacham and B. Waters. "Efficient ring signatures without random oracles." In *Public Key Cryptography — PKC 2007*, Springer LNCS **4450** (2007), 166–180.
32. V. Shoup. "Lower bounds for discrete logarithms and related problems." In *Advances in Cryptology — Eurocrypt 1997*, Springer LNCS **1233** (1997), 256–266.
33. B. Waters. "Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions." In *Advances in Cryptology — Crypto 2009*, Springer LNCS **5677** (2009), 619–636.
34. M. Yoshida. "Inseparable multiplex transmission using the pairing on elliptic curves and its application to watermarking." Proc. Fifth Conference on Algebraic Geometry, Number Theory, Coding Theory and Cryptography, University of Tokyo, 2003 (2003). Available at `http://www.math.uiuc.edu/~duursma/pub/yoshida_paper.pdf`.
35. M. Yoshida, S. Mitsunari, and T. Fujiwara. "Vector decomposition problem and the trapdoor inseparable multiplex transmission scheme based the problem." In *Proceedings of the 2003 Symposium on Cryptography and Information Security (SCIS)* (2003), 491–496.

A Pairing-Friendly Elliptic Curves

To describe pairing-friendly elliptic curves we first review some notation. If E is an elliptic curve over a finite field $\mathbb{F}_q$, then $E(\mathbb{F}_q)$ denotes the $\mathbb{F}_q$-rational points of E. We let $E[n]$ denote all points of E with order dividing n, defined over any extension of $\mathbb{F}_q$; if n is prime to q then $E[n] \cong \mathbb{Z}_n^2$. If r is a prime dividing $\#E(\mathbb{F}_q)$ and not dividing q, the *embedding degree of E with respect to r* is the smallest integer such that $q^k \equiv 1 \pmod{r}$. If E has embedding degree k with respect to r then $E(\mathbb{F}_{q^k})$ has two linearly independent subgroups of order r and the modified Tate pairing on $E[p]$ takes values in $\mathbb{F}_{q^k}^*$ [14, Section 2.1]. A "pairing-friendly" curve is one that has a small embedding degree (e.g., $k \leq 50$) with respect to a large prime-order subgroup (e.g., $p > 2^{160}$).

If E is has embedding degree $k > 1$ with respect to r then there are two distinguished order-r subgroups of $E(\mathbb{F}_{q^k})$: namely, the group $\mathbb{G}_1$ of rational points of order r, i.e., $E(\mathbb{F}_q) \cap E[r]$, and the group $\mathbb{G}_2$ of points of order r in the *trace-zero subgroup* of $E(\mathbb{F}_{q^k})$., i.e.,

$$E[r] \cap \left\{ (x, y) \in E(\mathbb{F}_{q^k}) : \textstyle\sum_{i=1}^{k} (x^{q^i}, y^{q^i}) = O \right\},$$

where $\sum$ indicates addition on the elliptic curve and O is the identity element.

Equivalently, we can define $\mathbb{G}_1$ and $\mathbb{G}_2$ as the subgroups of $E[r]$ on which the q-power Frobenius endomorphism acts as multiplication by 1 and q, respectively.

If E is *ordinary* (i.e., $\#E(\mathbb{F}_q) \neq q + 1$), then there is no known efficient algorithm for solving the DDH problem in either $\mathbb{G}_1$ or $\mathbb{G}_2$. In particular, the Weil and Tate pairings are trivial when restricted to either $\mathbb{G}_1$ or $\mathbb{G}_2$ [16, Lemma IX.16], so we cannot use the pairing to solve DDH in either group. If we let $\mathbb{G}_t$ be the order-p subgroup of $\mathbb{F}_{q^k}^*$ and $\hat{e}$ be the modified Tate pairing, then we set the output of the bilinear group generator $\mathcal{P}$ to be $(r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e})$. (Of course, in an application the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ could be switched in order to make the group of $\mathbb{F}_q$-rational points the one in which the most computation is done.)

An elliptic curve E as above not only provides an example of a bilinear groups in which DDH is hard, but also admits arbitrary embedding degree k, which allows us to scale the sizes of the groups $E(\mathbb{F}_q) \supset \mathbb{G}_1$ and $\mathbb{F}_{q^k}^* \supset \mathbb{G}_t$ independently, providing maximum efficiency for any desired security level (see e.g. [15]). By contrast, symmetric pairings are restricted to embedding degree $k = 2$ over prime fields and to $k \leq 6$ in general.

The three ordinary elliptic curves we choose to match our specified levels of AES security are as follows:

80-bit security: We use a Miyaji-Nakabayashi-Takano elliptic curve E over a 170-bit field $\mathbb{F}_q$ with a prime group order $p = \#E(\mathbb{F}_q)$ and embedding degree 6 [26]. We can represent elements of $\mathbb{G}_2$ as points in the trace-zero subgroup of the quadratic twist of E over $\mathbb{F}_q$. This group has order roughly q^2, so if we use the compression method of Rubin and Silverberg [28, Section 10], we can represent elements in $\mathbb{G}_2$ using 340 bits.

128-bit security: We use a Barreto-Naehrig elliptic curve E over a 256-bit field $\mathbb{F}_q$ with a prime group order $p = \#E(\mathbb{F}_q)$ and embedding degree 12 [3]. We can represent elements of $\mathbb{G}_2$ as points on the sextic twist of E over $\mathbb{F}_{p^2}$. This group has order roughly q^2, so we can represent elements in $\mathbb{G}_2$ using 512 bits.

256-bit security: We use a Brezing-Weng elliptic curve E over a 640-bit field $\mathbb{F}_q$ that has a 512-bit prime-order subgroup and embedding degree 24 [11]. We can represent elements of $\mathbb{G}_2$ as points on the sextic twist of E over $\mathbb{F}_{q^4}$. This group has order roughly q^4, so we can represent elements in $\mathbb{G}_2$ using 2560 bits.

B Private Linear Broadcast Encryption: Definitions and Construction

In this section we give the formal definition of a private linear broadcast encryption system and define what it means for the system to be secure. This material appears in [7, Section 2] and is duplicated here for ease of reference.

B.1 Syntax

Formally, a PLBE system is comprised of four probabilistic polynomial-time algorithms:

$\mathsf{Setup}(\lambda, n)$**:** Takes as input a security parameter λ and a positive integer n that is the number of users in the system. The algorithm outputs a public key PK, a secret key SK, and private keys $K_1, \ldots, K_n$, where key K_i is given to user i.

$\mathsf{Encrypt}(PK, M)$**:** Takes as input a public key PK and a message M and outputs a ciphertext C. This algorithm is used to encrypt a message to all N users.

$\mathsf{TrEncrypt}(SK, i, M)$**:** Takes as input a secret key SK, an integer i with $1 \leq i \leq n+1$, and a message M, and outputs a ciphertext C. This algorithm encrypts a message to a set $\{i, \ldots, N\}$ and is primarily used for traitor tracing. We will require below that $\mathsf{TrEncrypt}(SK, 1, M)$ outputs a distribution on ciphertexts that is indistinguishable from the distribution generated by $\mathsf{Encrypt}(PK, M)$.

$\mathsf{Decrypt}(j, K_j, C, PK)$**:** Takes as input a private key K_j for user j, a ciphertext C, and the public key PK. The algorithm outputs a message M or the symbol $\perp$.

The system must satisfy the following correctness property for all $i, j \in \{1, \ldots, N+1\}$, with $j \leq N$, and all messages M: Let $(PK, SK, (K_1, \ldots, K_n) \xleftarrow{\text{R}} \mathsf{Setup}(\lambda, n)$, and let $C \xleftarrow{\text{R}} \mathsf{TrEncrypt}(SK, i, M)$. If $j \geq i$ then $\mathsf{Decrypt}(j, K_j, C, PK) = M$.

B.2 Security

Security of a PLBE system is defined using three games. The first game captures a consistency property which says that $\mathsf{TrEncrypt}(TK, 1, M)$ outputs a distribution on ciphertexts that is indistinguishable from the distribution generated by $\mathsf{Encrypt}(PK, M)$. The second game is a "message hiding" game and says that a ciphertext created using index $i = n+1$ is unreadable by anyone. The third game is an "index hiding" game and captures the intuition that a broadcast ciphertext created using index i reveals no non-trivial information about i. We consider all these games for a fixed number of users n.

Game 1 — Indistinguishability. The first game says that the output of $\mathsf{TrEncrypt}(SK, 1, M)$ is indistinguishable from $\mathsf{Encrypt}(PK, M)$. The game proceeds as follows:

- **Setup:** The challenger runs the Setup algorithm and gives the adversary PK and the set of all private keys $\{K_1, \ldots, K_n\}$.
- **Challenge:** The adversary gives the challenger a message M. The challenger chooses $\beta \xleftarrow{\text{R}} \{0, 1\}$ and computes
$$C \xleftarrow{\text{R}} \begin{cases} \mathsf{TrEncrypt}(TK, 1, M) & \text{if } \beta = 0, \\ \mathsf{Encrypt}(PK, M) & \text{if } \beta = 1. \end{cases}$$
It gives C to the adversary.
- **Guess:** The adversary returns a guess $\beta' \in \{0, 1\}$ for β.

We define the advantage of adversary $\mathcal{A}$ as CG-Adv$[\mathcal{A}] = |\Pr[\beta' = \beta] - 1/2|$.

Game 2 — Message Hiding. The second game says that an adversary cannot break semantic security when encrypting using index $i = N + 1$. The game proceeds as follows:

- **Setup:** The challenger runs the Setup algorithm and gives the adversary PK and the set of all private keys $\{K_1, \ldots, K_n\}$.
- **Challenge:** The adversary outputs two equal-length messages M_0, M_1. The challenger chooses $\beta \overset{\text{R}}{\leftarrow} \{0, 1\}$ and sets $C \overset{\text{R}}{\leftarrow} \mathsf{TrEncrypt}(TK, n+1, M_\beta)$. The challenger gives C to the adversary.
- **Guess:** The adversary returns a guess $\beta' \in \{0, 1\}$ for β.

We define the advantage of adversary $\mathcal{A}$ as $\text{MH-Adv}[\mathcal{A}] = |\Pr[\beta' = \beta] - 1/2|$.

Game 3 — Index Hiding. The third game says that an adversary cannot distinguish between an encryption to index i and one to index $i + 1$ without the key K_i. The game takes as input a parameter $i \in \{1, \ldots, N\}$ which is given to both the challenger and the adversary. The game proceeds as follows:

- **Setup:** The challenger runs the Setup algorithm and gives the adversary PK and the set of all private keys $\{K_j : j \neq i\}$.
- **Challenge:** The adversary gives the challenger a message M. The challenger chooses $\beta \overset{\text{R}}{\leftarrow} \{0, 1\}$ and computes $C \overset{\text{R}}{\leftarrow} \mathsf{TrEncrypt}(TK, i + \beta, M)$. The challenger returns C to the adversary.
- **Guess:** The adversary returns a guess $\beta' \in \{0, 1\}$ for β.

We define the advantage of adversary $\mathcal{A}$ as $\text{IH-Adv}[\mathcal{A}, i] = |\Pr[\beta' = \beta] - 1/2|$.

Now that the three games are established we are ready to define secure PLBE.

Definition B.1. We say that an n-user Private Linear Broadcast Encryption (PLBE) system is *secure* if for all polynomial-time adversaries $\mathcal{A}$ we have that $\text{CG-Adv}[\mathcal{A}]$, $\text{MH-Adv}[\mathcal{A}]$, and $\text{IH-Adv}[\mathcal{A}, i]$ for $i = 1, \ldots, n$ are all negligible functions of λ.

C Predicate Encryption: Definitions and Construction

In this section we give the formal definition of a predicate encryption scheme and define what it means for the system to be secure. This material appears in [23, Section 2] and is duplicated here for ease of reference.

C.1 Syntax.

Formally, a predicate encryption scheme for the class of predicates $\mathcal{F}$ over the set of attributes Σ consists of four probabilistic polynomial-time algorithms:

$\mathsf{Setup}(\lambda)$**:** Takes as input a security parameter λ and outputs a (master) public key PK and a (master) secret key SK.

$\mathsf{KeyGen}(SK, f)$**:** Takes as input the master secret key SK and a (description of a) predicate $f \in \mathcal{F}$. It outputs a key SK_f.

$\mathsf{Encrypt}(PK, I, M)$**:** Takes as input the public key PK, an attribute $I \in \Sigma$, and a message M in some associated message space. It returns a ciphertext C.

$\mathsf{Decrypt}(SK_f, C)$**:** Takes as input a secret key SK_f and a ciphertext C. It outputs either a message M or the distinguished symbol $\perp$.

For correctness, we require that for all λ, all (PK, SK) generated by $\mathsf{Setup}(\lambda)$, all $f \in \mathcal{F}$, any key $SK_f \leftarrow \mathsf{KeyGen}(SK, f)$, and all $I \in \Sigma$, the following hold:

- If $f(I) = 1$ then $\mathsf{Decrypt}(SK_f, \mathsf{Encrypt}(PK, I, M)) = M$.
- If $f(I) = 0$ then $\mathsf{Decrypt}(SK_f, \mathsf{Encrypt}(PK, I, M)) = \perp$ with all but negligible probability.

C.2 Security.

The Katz et al. construction produces a predicate encryption scheme where the set of attributes in $\Sigma = \mathbb{Z}_N^n$ and the class of predicates is $\mathcal{F} = \{f_{\vec{x}} : \vec{x} \in \mathbb{Z}_N^n\}$. Security of such a system is defined using the following game. (Compare with [23, Definition 2.2 and Definition B.1].) As before, we use a large $\bullet$ to denote the dot product of vectors over $\mathbb{Z}_N$.

1. $\mathsf{Setup}(\lambda, n)$ is run to generate keys PK, SK. This defines a value N which is given to $\mathcal{A}$.
2. $\mathcal{A}$ outputs $\vec{x}, \vec{y} \in \mathbb{Z}_N^n$, and is then given PK.
3. $\mathcal{A}$ may adaptively request keys corresponding to the vectors $\vec{v}_1, \ldots, \vec{v}_\ell \in \mathbb{Z}_N^n$, subject to the restriction that for all i, we have $\vec{v}_i \bullet \vec{x} = 0$ if and only if $\vec{v}_i \bullet \vec{y} = 0$. In response, $\mathcal{A}$ is given the corresponding keys $SK_{\vec{v}_i} \leftarrow \mathsf{KeyGen}(SK, \vec{v}_i)$.
4. $\mathcal{A}$ outputs two equal-length messages M_0, M_1. If there is an i for which $\vec{v}_i \bullet \vec{x} = \vec{v}_i \bullet \vec{y} = 0$, then it is required that $M_0 = M_1$.
5. A random bit β is chosen. If $\beta = 0$ then $\mathcal{A}$ is given $C \leftarrow \mathsf{Encrypt}(PK, \vec{x}, M_0)$, and if $\beta = 1$ then $\mathcal{A}$ is given $C \leftarrow \mathsf{Encrypt}(PK, \vec{y}, M_1)$.
6. The adversary may continue to request keys for additional predicates, subject to the same restrictions as before.
7. $\mathcal{A}$ outputs a bit β'., and succeeds if $\beta' = \beta$.

We define the advantage of adversary $\mathcal{A}$ as PE-Adv$[\mathcal{A}] = |\Pr[\beta' = \beta] - 1/2|$.

Definition C.1. A predicate encryption scheme with respect to $\mathcal{F}$ and Σ is *attribute hiding* or *secure* if for all polynomial-time adversaries $\mathcal{A}$, we have that PE-Adv$[\mathcal{A}]$ is negligible in the security parameter λ.

D Security of Predicate Encryption in Generic Prime-Order Groups

The security of the Katz-Sahai-Waters predicate encryption scheme using the bilinear generator $\mathcal{G}_{3C}$ depends on the complex (though constant-size) Assumptions 3 and 4. To determine whether these assumptions are reasonable, we wish to determine whether they hold in the generic group model [32]. To do this, we use the "master theorem" of Boneh, Boyen, and Goh [4, Theorem A.5].

In a group of prime order, we represent each randomly chosen group element g as a random variable X, which indicates the exponent of g relative to some fixed group generator g_0 (represented by 1). Interdependencies of group elements are made explicit by reusing the same variables; for example, a generic Diffie-Hellman tuple can be represented by the expression $(1, X, Y, XY)$ in the variables X and Y. The key concept we will use is *dependence* of variables, which we now define.

Definition D.1. Let $P = (u_1, \ldots, u_r)$, $Q = (v_1, \ldots, v_s)$, $R = (w_1, \ldots, w_t)$, $S = (\chi_1, \ldots, \chi_m)$ be tuples of polynomials in $\mathbb{Z}[X_1, \ldots, X_n]$. Let f be a polynomial in $\mathbb{Z}[X_1, \ldots, X_n]$. We say that $f \cdot S$ is *dependent on* (P, Q, R) if there exist integers $a_{i,j}$ for $1 \le i \le r$ and $1 \le j \le s$, integers b_k for $1 \le k \le t$, and integers c_l with $1 \le l \le m$, such that

$$\sum_{i=1}^{r} \sum_{j=1}^{s} a_{i,j} u_i v_j + \sum_{k=1}^{t} b_k w_k + \sum_{\ell=1}^{m} c_\ell \chi_\ell Y \tag{D.1}$$

is nonzero in $\mathbb{Z}[X_1, \ldots, X_n, Y]$ but becomes zero when we set $Y = f$.

We say that $f \cdot S$ is *independent of* (P, Q, R) if $f \cdot S$ is not dependent on (P, Q, R).

We say that f is independent of (P, Q, R) if $f \cdot \{1\}$ is independent of (P, Q, R).

In the definition, the polynomials u_i represent elements of $\mathbb{G}_1$, polynomials v_j represent elements of $\mathbb{G}_2$, and polynomials w_k represent elements of $\mathbb{G}_t$, while the polynomial f represents the challenge element in the assumption. Our definition generalizes that of Boneh et al. by allowing the challenge element to be in any of the three groups; the polynomials χ_ℓ represent the elements with which the challenge element can be paired.

We note that Boneh et al. use polynomials over $\mathbb{F}_p$ to represent group elements; however, this choice does not take into account the fact that in our assumptions the relations between the variables remain the same while the group order p may vary. Thus we define our variables in terms of polynomials over $\mathbb{Z}$ and observe that if independence holds over $\mathbb{Z}$ then it holds over $\mathbb{F}_p$ for all sufficiently large p.

Definition D.2. Let $\mathcal{P}$ be a prime-order bilinear group generator, and let P, Q, R, f be as in Definition D.1. Define the following distribution:

$$\mathbb{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e) \xleftarrow{\mathrm{R}} \mathcal{P}(\lambda),\ g \xleftarrow{\mathrm{R}} \mathbb{G}_1,\ h \xleftarrow{\mathrm{R}} \mathbb{G}_2,\ g_t \leftarrow \hat{e}(g, h),\ \vec{x} \xleftarrow{\mathrm{R}} \mathbb{F}_p^\ell$$
$$Z \leftarrow \left(g^{u_1(\vec{x})}, \ldots, g^{u_r(\vec{x})}, h^{v_1(\vec{x})}, \ldots, h^{v_s(\vec{x})}, g_t^{w_1(\vec{x})}, \ldots, g_t^{v_r(\vec{x})}\right),$$
$$T_0 \leftarrow g_1^{f(\vec{x})}, \quad T_1 \xleftarrow{\mathrm{R}} \mathbb{G}_1.$$

We define the advantage of algorithm $\mathcal{A}$ that outputs $b \in \{0,1\}$ in solving the (P, Q, R, f)*-decision Diffie-Hellman problem in* $\mathbb{G}_1$ to be

$$(P, Q, R, f) - \text{DDH-Adv}[\mathcal{A}, \mathcal{P}] = \Big| \Pr[\mathcal{A}(\mathbb{G}, Z, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_1) = 1] \Big|.$$

We define the analogous problem $\mathbb{G}_t$ by taking $T_0 \leftarrow g_t^{f(\vec{x})},\ T_1 \xleftarrow{\mathrm{R}} \mathbb{G}_t$.

The Boneh-Boyen-Goh “master theorem” is as follows. Boneh et al. prove the theorem for the (P, Q, R, f)-DDH problem in $\mathbb{G}_t$. The same argument carries over to the statement in $\mathbb{G}_1$, using our generalized definition of independence.

Theorem D.3 ([4, Theorem A.5]). *Let P, Q, R be as in Definition D.1, let $f \in \mathbb{Z}[X_1, \ldots, X_n]$, and let p be a prime. Let $d = 2 \cdot \max\{\deg \alpha : \alpha \in P \cup Q \cup R \cup \{f\}\}$. If f is independent of (P, Q, R), then any algorithm that solves the (P, Q, R, f)-DDH problem in $\mathbb{G}_t$ with advantage $1/2$ in a generic bilinear group of order p must take time at least $\Omega(\sqrt{p/d} - n)$, asymptotically as $p \to \infty$.*

If $f \cdot Q$ is independent of (P, Q, R), then the same statement holds for the (P, Q, R, f)-DDH problem in $\mathbb{G}_1$.

We now apply this theorem to Assumptions 3 and 4. The assumptions are too complex for us to check the independence conditions by hand, so we turn to a computer. To do this, we first note that the independence condition of Definition D.1 is equivalent to the two $\mathbb{Z}$-modules $\mathbb{Z}[\{u_i v_j\}, \{w_k\}, \{\chi_\ell Y\}]$ and $\mathbb{Z}[\{u_i v_j\}, \{w_k\}, \{\chi_\ell f\}]$ having the same rank. Thus we can determine independence by the following procedure:

1. Let V be the free $\mathbb{Z}$-module with a basis consisting of all the distinct monomials in the polynomials $u_i v_j, w_k, \chi_\ell Y, \chi_\ell f$, listed in some fixed order.
2. Represent each polynomial p in $\{u_i v_j\}, \{w_k\}, \{\chi_\ell Y\}, \{\chi_\ell f\}$ as an integer vector $\vec{x}_p \in V$ with respect to the above basis.
3. Let M_1 be the matrix whose rows are the vectors $\vec{x}_{u_i v_j}, \vec{x}_{w_k}, \vec{x}_{\chi_\ell Y}$.
4. Let M_2 be the matrix whose rows are the vectors $\vec{x}_{u_i v_j}, \vec{x}_{w_k}, \vec{x}_{\chi_\ell f}$.

5. Compute rank(M_1) and rank(M_2). If they are equal, then $f \cdot S$ is independent of (P, Q, R); otherwise there is a dependence.

We begin with Assumption 4 since it fits exactly into this framework. If we use $1, A, B$ to represent g_1, g_2, g_3 and $1, E, F$ to represent h_1, h_2, h_3, then we are working in the polynomial ring $\mathbb{Z}[x, y, z, u, v, w, a, s, \gamma, c_1, c_2, A, B, E, F]$ in 15 variables over $\mathbb{Z}$. We set

$$\begin{aligned}
P &= \big(1, x, u, A, Ay, Av, B, Bz, Bw, a, xa, ua, s, xs, us, as + Ac_1, xas + Ayc_1, uas + Avc_1\big),\\
Q &= \big(zv - yw, w - v, y - z, E(zu - xw), E(w - u), E(z - x), F(yu - xv), F(v - u), F(x - y),\\
&\qquad (zv - yw)\gamma + E(zu - xw)c_2, (w - v)\gamma + E(w - u)c_2, (y - z)\gamma + E(z - x)c_2\big),\\
R &= ((xw - xv + yu - yw - zu + zv)a\gamma),\\
f &= (xw - xv + yu - yw - zu + zv)a\gamma s
\end{aligned}$$

We use Magma [9] to apply the above procedure with $S = \{1\}$. The space V has dimension 565 over $\mathbb{Z}$, and the matrices M_1 and M_2 each have 218 rows (corresponding to polynomials). We find that the ranks of M_1 and M_2 are both 207, so f is independent of (P, Q, R). Thus we have proven the following.

Proposition D.4. *Any algorithm that breaks Assumption 4 with advantage* $1/2$ *in a generic bilinear group of order* p *must take time at least* $\Omega(\sqrt{p})$*, asymptotically as* $p \to \infty$

On the other hand, Assumption 3 does not quite fit into the framework. First of all, in Assumption 3 the "variable" part of the challenge (i.e., T_0 or T_1) consists of three group elements, whereas Theorem D.3 allows only one element to vary. In addition, both T_0 and T_1 have a specified form, whereas in Theorem D.3 one of these elements should be uniformly random. However, a simple hybrid argument shows that Assumption 3 follows from a number of related assumptions that do fit the model of Theorem D.3. Specifically, we define new variables T_{0i}, T_{1i} for $i = 1, 2, 3$, such that $T_0 = (T_{01}, T_{02}, T_{03})$ and $T_1 = (T_{11}, T_{12}, T_{13})$. We then have

$$\begin{aligned}
\text{A3-Adv}[\mathcal{A}, \mathcal{G}] \le &\Big|\Pr[\mathcal{A}(\mathbb{G}, Z, T_{01}, T_{02}, T_{03}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, R, T_{02}, T_{03}) = 1]\Big| &\text{(D.2)}\\
+&\Big|\Pr[\mathcal{A}(\mathbb{G}, Z, R, T_{02}, T_{03}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_{11}, T_{02}, T_{03}) = 1]\Big| &\text{(D.3)}\\
+&\Big|\Pr[\mathcal{A}(\mathbb{G}, Z, T_{11}, T_{02}, T_{03}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_{11}, R, T_{03}) = 1]\Big| &\text{(D.4)}\\
+&\Big|\Pr[\mathcal{A}(\mathbb{G}, Z, T_{11}, R, T_{03}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_{11}, T_{12}, T_{03}) = 1]\Big| &\text{(D.5)}\\
+&\Big|\Pr[\mathcal{A}(\mathbb{G}, Z, T_{11}, T_{12}, T_{03}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_{11}, T_{12}, R) = 1]\Big| &\text{(D.6)}\\
+&\Big|\Pr[\mathcal{A}(\mathbb{G}, Z, T_{11}, T_{12}, R) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_{11}, T_{12}, T_{13}) = 1]\Big| &\text{(D.7)}
\end{aligned}$$

where in each case we choose $R \overset{\text{R}}{\leftarrow} \mathbb{G}_1$.

Each of the terms (D.2)–(D.7) on the right hand side corresponds to an assumption that can be modeled in the generic group framework. If all six of the terms are negligible, then $\mathcal{G}$ satisfies Assumption 3. We note that the term (D.3) is equal to zero since T_{11} is chosen randomly and independently of $Z \cup \{T_{02}, T_{03}\}$. To address the other five terms, we set the variables in the generic group framework as follows. If we again use $1, A, B$ to represent g_1, g_2, g_3 and $1, E, F$ to represent h_1, h_2, h_3, then we are working in the polynomial ring $\mathbb{Z}[x, y, z, u, v, w, a, b, s, c_1, c_2, c_3, d_1, d_2, d_3, A, B, E, F]$ in

19 variables over $\mathbb{Z}$. We define the sets

$$\begin{aligned}
P' &= \big(1, x, u, B, Bz, Bw, A + Bd_1, Ay + Bzd_1, Av + Bwd_1, b, xb, ub, b^2, xb^2, ub^2, s, xs, us,\\
&\quad bs + Ac_2 + Bd_2, xbs + Ayc_2 + Bzd_2, ubs + Avc_2 + Bwd_2\big)\\
Q &= \big(zv - yw, w - v, y - z, F(yu - xv), F(v - u), F(x - y),\\
&\quad E(zu - xv) + F(yu - xv)d_1, E(w - u) + F(v - u)d_1, E(zu - xw) + F(yu - xv)d_1,\\
&\quad (zv - yw)b, (w - v)b, (y - z)b, (zv - yw)b^2, (w - v)b^2, (y - z)b^2,\\
&\quad (zv - yw)a + E(zu - xw), (w - v)a + E(w - u), (y - z)a + E(z - x),\\
&\quad (zv - yw)ab + E(zu - xw)c_1, (w - v)ab + E(w - u)c_1, (y - z)ab + E(z - x)c_1\big)\\
R &= \varnothing\\
(f_{01}, f_{02}, f_{03}) &= (b^2s + Bd_3, xb^2s + Bzd_3, ub^2s + Bwd_3)\\
(f_{11}, f_{12}, f_{13}) &= (b^2s + Ac_3 + Bd_3, xb^2s + Ayc_3 + Bzd_3, ub^2s + Avc_3 + Bwd_3)
\end{aligned}$$

We construct the matrices M_1 and M_2 for each of the five assumptions corresponding to equations (D.2),(D.4)–(D.7) and determine their ranks. The set P and the polynomial f depend on the equation we are considering; in each case we use Q as above and set $S = Q$. The results are summarized in Table 3.

Table 3. Construction of matrices M_1 and M_2 for hybrid security assumptions (D.2),(D.4)–(D.7).

Equation	P	f	matrix columns (monomials)	matrix rows (polynomials)	rank M_1	rank M_2
(D.2)	$P' \cup (f_{02}, f_{03})$	f_{01}	1432	484	434	434
(D.4)	$P' \cup (f_{11}, f_{03})$	f_{02}	1484	484	434	434
(D.5)	$P' \cup (f_{11}, f_{03})$	f_{12}	1484	484	434	434
(D.6)	$P' \cup (f_{11}, f_{12})$	f_{03}	1529	484	434	434
(D.7)	$P' \cup (f_{11}, f_{12})$	f_{13}	1529	484	434	434

Since the ranks of M_1 and M_2 are equal in each case, the required independence holds, and we have proven the following.

Proposition D.5. *Any algorithm that breaks Assumption 3 with advantage* $1/2$ *in a generic bilinear group of order* p *must take time at least* $\Omega(\sqrt{p})$*, asymptotically as* $p \to \infty$

Finally, we consider the 3-party Diffie-Hellman assumption in prime-order groups (Definition 6.4). This assumption is simple enough that we can do the necessary computations by hand. We work in the polynomial ring $\mathbb{Z}[a, b, c]$, and define

$$P = Q = (1, a, b, c),\ \ R = \varnothing,\ \ f = abc.$$

Suppose $f \cdot Q$ is dependent on (P, Q, R). Then there is a nonzero expression of the form (D.1) that is equal to zero when we set $Y = f$. Thus we can write

$$\sum_{u \in P} \sum_{v \in Q} a_{u,v} u \cdot v = \sum_{\chi \in Q} -c_\chi \chi \cdot f.$$

Since each term on the left hand side has degree at most 2 and each term on the right hand side has degree at least 3, the sums on each side must both be equal to zero. Furthermore, since the four elements of Q are linearly independent over $\mathbb{Z}$, the coefficients c_χ must all be zero. But then when we replace f with Y we get an expression that is also zero, a contradiction. Thus $f \cdot Q$ is independent of (P, Q, R), and we have proven the following.

Proposition D.6. *Any algorithm that breaks the 3-party decision Diffie-Hellman assumption with advantage $1/2$ in a generic bilinear group of order p must take time at least $\Omega(\sqrt{p})$, asymptotically as $p \to \infty$*