Low Overhead Broadcast Encryption from Multilinear Maps

Dan Boneh
Stanford University
dabo@cs.stanford.edu

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

Mark Zhandry
Stanford University
zhandry@cs.stanford.edu

Abstract

We use multilinear maps to provide a solution to the long-standing problem of public-key broadcast encryption where all parameters in the system are small. In our constructions, ciphertext overhead, private key size, and public key size are all poly-logarithmic in the total number of users. The systems are fully collusion-resistant against any number of colluders. All our systems are based on an $O(\log N)$-way multilinear map to support a broadcast system for N users. We present three constructions based on different types of multilinear maps and providing different security guarantees. Our systems naturally give identity-based broadcast systems with short parameters.

1 Introduction

Broadcast encryption [FN94] is an important generalization of public-key encryption to the multi-user setting. In a broadcast encryption scheme, a broadcaster encrypts a message for a subset S of users who are listening on a broadcast channel. The broadcaster can encrypt to any set S of its choice, and any user in S can decrypt the broadcast using its secret key. The system is said to be *fully collusion resistant* if even a coalition of all users outside of S learns nothing about the plaintext. Broadcast systems are regularly used in TV and radio subscription services where broadcasts are encrypted for currently active subscribers. They are also used in encrypted file systems where a file is encrypted so that only users who have access to the file can decrypt it.

The efficiency of a broadcast system is measured in the ciphertext overhead: the number of bits in the ciphertext beyond what is needed for the description of the recipient set S and the symmetric encryption of the plaintext payload. The shorter the overhead, the better. We say that the system has *low overhead* if the ciphertext overhead depends at most logarithmically on the number of users N in the system.

Existing constructions with low ciphertext overhead. Several broadcast systems are fully collusion resistant with low ciphertext overhead. The first such system by Boneh, Gentry, and Waters [BGW05] is built from bilinear maps. It has constant ciphertext overhead and short secret keys, but the public encryption key size is *linear* in the number of users N. Other systems using bilinear-maps achieve adaptive security [GW09, DPP07] and some are even identity-based [GW09, Del07, SF07], but the public encryption key is always large.

Multilinear maps give *secret-key* broadcast systems with optimal ciphertext overhead [BS03, GGH13a, FHPS13, CLT13, BW13]. However, in these systems the broadcaster's key must be kept secret, and they require an N-way multilinear map to support N users. Current constructions of N-linear maps [GGH13a, CLT13] have group elements of size $O(N^2)$ bits, resulting in large space requirements. While these broadcast systems can be made public-key by including a few group elements in the ciphertext, their dependence on N-linear maps leads to an $O(N^2)$ ciphertext overhead, which is worse than the trivial broadcast system. Until this work, it has not been known how to use multilinear maps to construct low overhead broadcast systems with a short public encryption key.

A third class of constructions employs the powerful candidates for indistinguishability obfuscation (iO) [BGI+01, GGH+13b]. Using iO it is possible to build a *public-key* broadcast system with optimal ciphertext overhead and short private keys, though public keys are large [BZ13]. The resulting systems have several other remarkable properties. However, current iO candidates add considerable complexity on top of multilinear maps. Our goal here is to construct broadcast systems using only simple assumptions on multilinear maps, namely, without relying on iO.

Our results. We describe three broadcast systems for N users that use an $O(\log N)$-way multilinear map. The systems have ciphertext overhead and decryption key of only $O(1)$ group elements which is $O(\log^2 N)$ bits using the current multilinear map candidates. The public encryption key contains $O(\log N)$ group elements which is $O(\log^3 N)$ bits. The first system uses an *asymmetric* multilinear map and follows the [BGW05] construction closely. It uses the $O(\log N)$-way multilinear map to compress the public key of that system from $O(N)$ group elements to $O(\log N)$ elements while keeping the ciphertext overhead and secret key short. We prove static security under a multivariate equivalent of the [BGW05] assumption.

The second system uses a general *symmetric* $O(\log N)$-way multilinear map to similarly compress the public key in [BGW05]. The added flexibility of a symmetric map has both positive and negative consequences. On the negative side, this flexibility allows the adversary to combine extra elements together. To maintain security we must ensure that all user indexes $u \in [N]$ are mapped to integers $\hat{u} \in [O(N \log N)]$ where all $\hat{u}$ have the same Hamming weight. This mapping does not affect ciphertext or private key size. On the positive side, this flexibility allows us to obtain slightly better parameters and base static security on a slightly simpler, though similar, complexity assumption.

The third system is built from a symmetric $O(\log N)$-way map, but we can prove *adaptive* security of the scheme in generic multilinear groups. This system has secret keys of length $O(\log^3 N)$ bits, which is longer than the previous two schemes, but has a tighter security proof in generic groups.

Because the parameters of these systems are logarithmic in N, we can let N be exponential, and in particular be as large as the range of a collision resistant hash function (e.g., $N = 2^{256}$). This, in effect, turns all our broadcast systems into efficient identity-based schemes. A user with identity $\mathsf{id} \in \{0,1\}^*$ is given the secret key associated with index number $H(\mathsf{id}) \in [N]$ where H is a collision resistant hash whose range is $[N]$. A broadcaster can then transmit to a set of recipients simply by hashing their public identities. For this reason, we describe all our broadcast systems as identity-based broadcast schemes.

Additional related work. Collusion resistant broadcast encryption has been widely studied. Revocation systems (e.g., [NNL01, HS02, GST04, DF02, LSW10]) can encrypt to $N - r$ users

with ciphertext size of $O(r)$. Further combinatorial solutions (e.g., [NP00, DF03]) achieve similar parameters. A broadcast encryption system is said to be recipient-private if broadcast ciphertexts reveal nothing about the intended set of recipients [BBW06, LPQ12, FP12]. Our broadcast systems are not recipient private, and it is a long-standing open problem to build a low-overhead recipient-private broadcast system. Such a system was recently built using indistinguishability obfuscation [BZ13], but constructing such systems under weaker assumptions remains open.

2 Preliminaries

2.1 Broadcast Encryption

We begin by defining broadcast encryption. A (public key) identity-based broadcast encryption scheme consists of three randomized algorithms:

$\mathsf{Setup}(\mathcal{ID})$: Sets up a broadcast scheme for identity space $\mathcal{ID}$. It outputs public parameters params as well as a master secret key msk

$\mathsf{KeyGen}(\mathsf{msk}, u)$: Takes the master secret key and a user $u \in \mathcal{ID}$ and outputs a secret key sk_u for user u.

$\mathsf{Enc}(\mathsf{params}, S)$: The encryption algorithm takes the public parameters and a polynomial sized set $S \subseteq \mathcal{ID}$ of recipients, and produces a pair (Hdr, K). We refer to Hdr as the header, and K as the message encryption key.

The message is encrypted using a symmetric encryption scheme with the key K to obtain a ciphertext c. The overall ciphertext is (Hdr, c).

$\mathsf{Dec}(\mathsf{params}, u, \mathsf{sk}_u, S, \mathsf{Hdr})$: The decryption algorithm takes the header Hdr and the secret key for user u, and if $u \in S$, outputs the message encryption key K. If $u \notin S$, the decryption algorithm outputs $\perp$.

To actually decrypt the overall ciphertext (Hdr, c), user u runs Dec to obtain K, and then decryption c using K to obtain the message.

For correctness, we require that the decryption algorithm always succeeds when it is supposed to. That is, for every $(\mathsf{params}, \mathsf{msk})$ output by $\mathsf{Setup}(\mathcal{ID})$, every set $S \subseteq \mathcal{ID}$, every sk_u output by $\mathsf{KeyGen}(\mathsf{msk}, u)$, and (Hdr, K) outputted by $\mathsf{Enc}(\mathsf{params}, S)$ where $u \in S$, that $\mathsf{Dec}(\mathsf{params}, u, \mathsf{sk}_u, S, \mathsf{Hdr}) = K$.

For security, several notions of security are possible. We start by defining active chosen ciphertext security. For any adversary $\mathcal{A}$, let $\mathtt{EXP}(b)$ denote the following experiment on $\mathcal{A}$:

Setup: The challenger runs $(\mathsf{params}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathcal{ID})$, and gives $\mathcal{A}$ the public key params.

Secret Key Queries: $\mathcal{A}$ may adaptively make secret key queries for user u. In response, the challenger runs $\mathsf{sk}_u \leftarrow \mathsf{KeyGen}(\mathsf{msk}, u)$ and gives sk_u to $\mathcal{A}$.

CCA Queries: $\mathcal{A}$ may make chosen ciphertext queries on tuples (u, S, Hdr). The challenger responds with $\mathsf{Dec}(\mathsf{params}, u, \mathsf{sk}_u, S, \mathsf{Hdr})$ where $\mathsf{sk}_u \leftarrow \mathsf{KeyGen}(\mathsf{msk}, u)$[1].

[1] Another variation is to have the challenger maintain a table of (u, sk_u) pairs, and only run KeyGen once for a particular user, using a single sk_u to answer multiple secret key and CCA queries. Note that the correctness of a broadcast scheme implies that this does not affect CCA queries.

Challenge: $\mathcal{A}$ submits a set $S^* \subset \mathcal{ID}$, subject to the restriction that $u \notin S^*$ for any user u requested in a secret key query. The challenger lets $(\mathsf{Hdr}^*, K_0^*) \leftarrow \mathsf{Enc}(\mathsf{params}, S^*)$. If $b = 0$, the challenger gives (Hdr^*, K_0^*) to the adversary. If $b = 1$, the challenger computes a random key K_1^* and gives (Hdr^*, K_1^*) to the adversary.

More Secret Key Queries: $\mathcal{A}$ may continue making secret key queries for users $u \notin S^*$

More CCA Queries: $\mathcal{A}$ may continue making CCA queries on headers $\mathsf{Hdr} \neq \mathsf{Hdr}^*$[2].

Guess: $\mathcal{A}$ produces a guess b' for b.

Using a simple hybrid argument, we can assume the adversary makes only a single challenge query. Let W_b be the event that $\mathcal{A}$ outputs 1 in $\mathtt{EXP}(b)$. We define the adaptive CCA advantage of $\mathcal{A}$, as

$$\mathtt{BE}^{(\mathrm{adv})}{}_{\mathcal{A}} = |\Pr[W_0] - \Pr[W_1]|$$

Definition 2.1. A broadcast encryption scheme is adaptively secure under a chosen ciphertext attack (adaptively CCA-secure) if, for all polynomial time adversaries $\mathcal{A}$, $\mathtt{BE}^{(\mathrm{adv})}{}_{\mathcal{A}}$ is negligible.

We will also consider several weaker notions of security. For example, we get static security if we require $\mathcal{A}$ to commit to the challenge set S^* before seeing the public parameters. We also get CPA security if we do not allow chosen ciphertext queries. In this paper, we will be focusing on the following notion of static CPA security, but will also discuss the other variants:

Definition 2.2. A broadcast encryption scheme is statically secure under a chosen plaintext attack (statically CPA-secure) if, for all polynomial time adversaries $\mathcal{A}$ that must commit to S^* before seeing the public parameters and cannot make CCA queries, $\mathtt{BE}^{(\mathrm{adv})}{}_{\mathcal{A}}$ is negligible.

2.2 Multilinear Maps

We now review multilinear maps [BS03, GGH13a, CLT13]. A multilinear map consists of two algorithms:

$\mathsf{Setup}(n)$: Sets up an n-linear map. It outputs n groups $\mathbb{G}_1, \ldots, \mathbb{G}_n$ of prime order p, along with generators $g_i \in \mathbb{G}_i$. We call $\mathbb{G}_1$ the source group, $\mathbb{G}_n$ the target group, and $\mathbb{G}_2, \ldots, \mathbb{G}_{n-1}$ intermediate groups.

$e_{i,j}(g, h)$: Takes in two elements $g \in \mathbb{G}_i$ and $h \in \mathbb{G}_j$ with $i + j \leq n$, and outputs an element of $\mathbb{G}_{i+j}$ such that

$$e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$$

We often omit the subscripts and just write e. We can also generalize e to multiple inputs as $e(h^{(1)}, \ldots, h^{(k)}) = e(h^{(1)}, e(h^{(2)}, \ldots, h^{(k)}))$.

We sometimes call g_i^a as a level-i *encoding* of a. The scalar a itself could be referred to as a level-0 encoding of a. Then the map e combines a level i encoding and a level j encoding, and produces a level $i + j$ encoding of the product.

We will make use of *asymmetric* multilinear maps. In such maps, groups are indexed by integer vectors rather than integers. The pairing operations maps $\mathbb{G}_{\mathbf{v}_1} \times \mathbb{G}_{\mathbf{v}_2}$ into $\mathbb{G}_{\mathbf{v}_1+\mathbf{v}_2}$. More precisely, we have the following algorithms:

[2] Another potentially stronger variation is to require $(S, \mathsf{Hdr}) \neq (S^*, \mathsf{Hdr}^*)$

$\mathsf{Setup}(\mathbf{n})$ Sets up an $\mathbf{n}$-linear map, where $\mathbf{n} \in \mathbb{Z}^\ell$ is some positive integer vector. It outputs a description of groups $\mathbb{G}_\mathbf{v}$ of prime order p where $\mathbf{v}$ are non-negative integer vectors and $\mathbf{v} \leq \mathbf{n}$ (that is, the comparison must hold component-wise). It also outputs a description of generators $g_\mathbf{v} \in \mathbb{G}_\mathbf{v}$[3]. Let $\mathbf{e}_i$ be the ith standard basis vector, with a 1 at position i and a 0 elsewhere. We call $\mathbb{G}_{\mathbf{e}_i}$ the ith source group, $\mathbb{G}_\mathbf{n}$ the target group, and the rest of the $\mathbb{G}_\mathbf{v}$ groups are intermediate groups.

$e_{\mathbf{v}_1,\mathbf{v}_2}(g,h)$ Takes in two elements $g \in \mathbb{G}_{\mathbf{v}_1}$ and $h \in \mathbb{G}_{\mathbf{v}_2}$ with $\mathbf{v}_1 + \mathbf{v}_2 \leq \mathbf{n}$, and outputs an element of $\mathbb{G}_{\mathbf{v}_1+\mathbf{v}_2}$ such that

$$e_{\mathbf{v}_1,\mathbf{v}_2}(g^a_{\mathbf{v}_1}, g^b_{\mathbf{v}_2}) = g^{ab}_{\mathbf{v}_1+\mathbf{v}_2}$$

We often omit the subscripts and just write e. We can also generalize e to multiple inputs as $e(h^{(1)}, \ldots, h^{(k)}) = e(h^{(1)}, e(h^{(2)}, \ldots, h^{(k)}))$.

We note that current candidates of multilinear maps [GGH13a, CLT13] depart from the ideal notions of multilinear maps described above. In particular, in these candidates, representations of group elements are not unique and contain a noise term that can cause errors during group and multilinear operations. While we present our constructions using ideal multilinear maps for simplicity, we stress that our constructions can easily be instantiated using current non-ideal candidates. We need multilinear maps with the following properties:

- A way to hide the group and multilinear operations that lead to a particular element. In current multilinear maps, this is obtained by performing a re-randomization procedure which makes the representation of an element statistically independent of the operations that lead to that element.

- A way, given any representation of an element in the target group, to "extract" a canonical representation of that element. This is handled by a "zero-test parameter" in current maps.

- The ability for the person who sets up the map to compute elements of the form g^{α^x} for exponentially-large x. In ideal multilinear maps, this would be accomplished by computing $z = \alpha^x$ in $\mathbb{Z}_p$, and then computing g^z. However, with current multilinear maps, it is not possible for normal users to compute g^z for a specific z of their choice[4]. However, the person who sets up the map knows a trapdoor that *does* allow computing g^z for any $z \in \mathbb{Z}_p$.

- The ability to generate asymmetric multilinear maps for any positive integer vector $\mathbf{n} \in \mathbb{Z}^\ell$. Section 4.3 of [GGH13a] shows how to do this.

- A way to make sure the noise growth does not cause any errors during normal execution of our protocols. Since there is no circular dependence between the parameters of the multilinear map and the number of operations our protocols require, we can set the parameters so the noise stays small enough to avoid errors.

[3] There may be an exponential number of groups and generators. The setup algorithm outputs a set of parameters from which the groups $\mathbb{G}_\mathbf{v}$ and generators $g_\mathbf{v}$ can be derived. In particular, each $g_\mathbf{v}$ can be derived from the pairing operation and $\{g_{\mathbf{e}_i}\}$, where $\mathbf{e}_i$ is the ith standard basis vector

[4] Instead, users can compute g^z for a random, but unknown, z.

3 Our Asymmetric Multilinear Map Construction

In this section, we give our first construction of identity-based broadcast encryption from multilinear maps. Our starting point is the scheme of Boneh, Gentry, and Waters [BGW05], henceforth referred to as the BGW scheme. Recall in their scheme, the public parameters consist of $O(N)$ source group elements (where N is the number of users), secret keys and headers are a constant number of source group elements, and the message encryption key is a group element in the target group. Our goal is to shrink the public key size to $O(\log N)$ group elements. We accomplish this by embedding the BGW scheme in a multilinear map, where the BGW parameters lie in an intermediate group. The BGW public parameters can then be derived from a small number of elements in the source group of the map — these few source group elements are the new public key.

In more detail, the significant component of the BGW public key are the elements $Z_1 = g_1^{\alpha}, Z_2 = g_1^{\alpha^2}, \ldots, Z_N = g_1^{\alpha^N}, Z_{N+2} = g_1^{\alpha^{N+2}}, \ldots, Z_{2N} = g_1^{\alpha^{2N}}$. The rest of the BGW public keys, secret keys, and header components are also element in $\mathbb{G}_1$, whereas the message encryption key is an element in the group $\mathbb{G}_2$.

Let $N = 2^n - 1$ for some integer n, and let $\mathbf{n} = (\overbrace{1, \ldots, 1}^{n+1 \text{ 1s}})$ be the vector of $n+1$ 1s. Our idea is to use an asymmetric multilinear map, where the target group is $\mathbb{G}_{2\mathbf{n}}$. We note that pairing two elements in $\mathbb{G}_{\mathbf{n}}$ gives an element in $\mathbb{G}_{2\mathbf{n}}$. Thus, while the entire multilinear map is asymmetric, the pairing operation acts symmetrically on the group $\mathbb{G}_{\mathbf{n}}$. Now we replace the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ in the BGW scheme with $\mathbb{G}_{\mathbf{n}}$ and $\mathbb{G}_{2\mathbf{n}}$. Thus $Z_u = g_{\mathbf{n}}^{\alpha^u}$. Rather than explicitly include the Z_u in the public parameters, we give a few group elements in the groups $\mathbb{G}_{\mathbf{e}_i}$ where $\mathbf{e}_i$ are the standard basis vectors. Specifically, we provide the parameters $X_i = g_{\mathbf{e}_i}^{\alpha^{(2^i)}}$ for $i = 0, \ldots, n-1$. By pairing various subsets of these X_i together, we can build all of the Z_u for $u \leq 2^n - 1 = N$. In particular, if $u = \sum_{i=0}^{n-1} u_i 2^i$ is the binary representation of u, then

$$Z_u = e(\ X_0^{u_0} g_{\mathbf{e}_0}^{1-u_0}\ ,\ X_1^{u_1} g_{\mathbf{e}_1}^{1-u_1}\ ,\ \ldots, X_{n-1}^{u_{n-1}} g_{\mathbf{e}_{n-1}}^{1-u_{n-1}}\ ,\ g_{\mathbf{e}_n}\ )$$

where $X_i^0 g_{\mathbf{e}_i}^1 = g_{\mathbf{e}_i}$ and $X_i^1 g_{\mathbf{e}_i}^0 = X_i$

To allow computation of Z_u for $u \geq 2^n + 1 = N + 2$, we might decide to publish $g_{\mathbf{e}_n}^{\alpha^{(2^n)}}$. However, this would allow computation of Z_{N+1}, which will break the security of the BGW scheme. Therefore, we instead publish $X_n = g_{\mathbf{e}_n}^{\alpha^{(2^n+1)}}$. Then, for $u \in [N+2, 2N]$, let $u' = u - (2^n + 1) = \sum_{i=0}^{n-1} u'_i 2^i$. Then we can write

$$Z_u = e(\ X_0^{u'_0} g_{\mathbf{e}_0}^{1-u'_0}\ ,\ X_1^{u'_1} g_{\mathbf{e}_1}^{1-u'_1}\ ,\ \ldots, X_{n-1}^{u'_{n-1}} g_{\mathbf{e}_{n-1}}^{1-u'_{n-1}}\ ,\ X_n\ )$$

Now we make the observation that $O(\log N)$ graded encodings remain efficient even up to exponential N. Therefore, we can actually make our scheme identity-based, where identities are bit strings of length n with the caveat that the 0^n is not a valid identity. Now we give our entire construction:

Construction 3.1. *Let* Setup' *be the setup algorithm for a multilinear map, where groups have order* p*. Our first identity-based broadcast scheme consists of the following algorithms:*

$\mathsf{Setup}(n)$*: Takes as input the length* n *of identities. Let* $\mathcal{ID} = \{0,1\}^n \setminus \{0^n\}$ *be the identity space. Let* $\mathbf{n}$ *be the all-ones vector of length* $n+1$*. Run* Setup' *on* $2\mathbf{n}$*, obtaining the public parameters* params' *for a multilinear map with target group* $\mathbb{G}_{2\mathbf{n}}$*.*

Choose a random $\alpha \in \mathbb{Z}_p$ *and let* $X_i = g_{\mathbf{e}_i}^{\alpha^{(2^i)}}$ *for* $i = 0, \ldots, n-1$ *and let* $X_n = g_{\mathbf{e}_n}^{\alpha^{(2^n+1)}}$. *Also choose a random* $\gamma \in \mathbb{Z}_p$ *and let* $Y = g_{\mathbf{n}}^{\gamma}$. *Lastly, let* $W = g_{2\mathbf{n}}^{\alpha^{(2^n)}}$. *The public key is*

$$\mathsf{params} = (\mathsf{params}', W, \{X_i\}_{i \in \{0,\ldots,n\}}, Y)$$

The master secret key is (α, γ).

$\mathsf{KeyGen}(\mathsf{params}, \alpha, \gamma, u)$*: The secret key for identity* $u \in [1, 2^n - 1]$ *is* $\mathsf{sk}_u = g_{\mathbf{n}}^{\gamma\alpha^u}$.

$\mathsf{Enc}(\mathsf{params}, S)$*: Recall that we can compute* Z_j *for* $j \in [1, 2^n - 1]$ *from the public parameters* $\{X_i\}_{i \in \{0,\ldots,n-1\}}$. *Pick a random* $t \in \mathbb{Z}_p$ *and compute the key and header as*

$$K = W^t = g_{2\mathbf{n}}^{t\alpha^{(2^n)}} \quad \text{and}$$
$$\mathsf{Hdr} = \left(g_{\mathbf{n}}^t \ , \ (Y \cdot \prod_{u \in S} Z_{2^n - u})^t \right) = \left(g_{\mathbf{n}}^t \ , \ g_{\mathbf{n}}^{t\left(\gamma + \sum_{u \in S} \alpha^{(2^n - u)}\right)} \right)$$

$\mathsf{Dec}(\mathsf{params}, u, \mathsf{sk}_u, S, \mathsf{Hdr})$*: If* $u \notin S$, *output* $\perp$. *Otherwise, write* Hdr *as* (C_0, C_1). *Recall that we can compute* Z_j *for* $j \in [2^n + 1, 2^{n+1}]$. *Output*

$$K = \frac{e(Z_u \ , \ C_1)}{e\left((\mathsf{sk}_u \cdot \prod_{j \in S, j \neq u} Z_{2^n - j + u}) \ , \ C_0 \right)}$$

If C_0 *and* C_1 *are as above, then we can write* $K = g_{2\mathbf{n}}^c$ *where*

$$c = \alpha^u t \left(\gamma + \sum_{j \in S} \alpha^{(2^n - j)} \right) - \left(\gamma\alpha^u + \sum_{j \in S, j \neq u} \alpha^{(2^n - j + u)} \right) t$$

Most of the terms cancel, leaving $c = t\alpha^{2^n}$ *as desired.*

Implementation details. As mentioned in Section 2, there are some minor complications with implementing our scheme using current multilinear map constructions [GGH13a, CLT13], but we stress that these complications do not affect the semantics of our scheme. First, during normal operations, computing g_1^{α} for a random α involves computing a "level-0" encoding of a random (unknown) α, and then pairing with g_1. In order to compute $g_1^{\alpha^2}$, we would pair g_1 with the level-0 encoding twice. However, the noise growth with repeated pairing operations would prevent us from computing $g_1^{\alpha^{(2^i)}}$ for sufficiently high powers of i. Instead, the setup algorithm must choose an explicit (known) $\alpha \in \mathbb{Z}_p$, compute the various α^{2^i}, encode these powers as level-0 encodings, and only then pair with g_1. This requires knowing the secrets used to set up the multilinear map, meaning the broadcaster must set up the map himself and cannot rely on maps generated by trusted parties. Note, however, that this exponentiation is only required during setup, and not encryption or decryption, meaning the secrets can be discarded immediately after setup, and anyone can still broadcast using the public parameters.

To make sure the header does not leak any important information, we also need to re-randomize the header components. This means re-randomization parameters need to be included for the group $\mathbb{G}_{\mathbf{n}}$. No other re-randomization parameters are necessary.

Before discussing security, we must discuss our new security assumption, which is closely related to the bilinear Diffe-Hellman Exponent assumption (BDHE) as used in BGW.

3.1 The Hybrid Diffie-Hellman Exponent Assumption (HDHE) Assumption

We define the (computational) n-Hybrid Diffie-Hellman Exponent problem as follows: Let $\mathsf{params}' \leftarrow \mathsf{Setup}'(2\mathbf{n})$ where $\mathbf{n}$ is the all-ones vector of length $n+1$. Choose $\alpha \in \mathbb{Z}_p$ at random, and let $X_i = g_{\mathbf{e}_i}^{\alpha^{(2^i)}}$ for $i = 0, \ldots, n-1$ and $X_n = g_{\mathbf{e}_n}^{\alpha^{(2^n+1)}}$. Choose a random $t \in \mathbb{Z}_p$ and let $V = g_{\mathbf{n}}^t$. Given $(\{X_i\}_{i\in\{0,\ldots,n-1\}}, V)$, the goal is to compute $K = g_{2\mathbf{n}}^{t\alpha^{(2^n)}}$.

We now define the decisional n-Hybrid Diffie-Hellman Expoent problem as, given the tuple $(\{X_i\}_{i\in\{0,\ldots,n-1\}}, V, K)$ where K is either $g_{2\mathbf{n}}^{t\alpha^{(2^n)}}$ or a random element of $\mathbb{G}_{2\mathbf{n}}$, to distinguish the two cases.

Definition 3.2. We say the decisional n-Hybrid Diffie-Hellman Exponent assumption holds for Setup' if, for any polynomial n and probabilistic polynomial time algorithm $\mathcal{A}$, $\mathcal{A}$ has negligible advantage in solving the n-Hybrid Diffie-Hellman Exponent problem.

Given the X_i for $i = 0, \ldots, n-1$, it is straightforward to compute $g_{\mathbf{n}}^{\alpha^j}$ for any $j \in [0, 2^n - 1]$. Moreover, including X_n, it is straightforward to extend this to $j \in [2^n+1, 2^{n+1}]$. However, computing $K = g_{2\mathbf{n}}^{t\alpha^{(2^n)}}$ from the X_i and V appears difficult. The reason is that we only have one term that depends on t, namely V. So to compute K, we would need to pair V with some combination of the X_i. In other words, we need to be able to compute $g_{\mathbf{n}}^{\alpha^{(2^n)}}$ from the X_i. However, since $\mathbf{n}$ has a one in each component, we can never pair any of the X_i with itself. This means we can only compute products of terms of the form $e(X_0^{s_0}, X_1^{s_1}, \ldots, X_n^{s_n})$ for $s_i \in \{0, 1\}$, where we take $X_i^0 = g_{\mathbf{e}_1}$. Notice that we can never include an X_n, since then we would already exceed the desired degree of 2^n. Put another way, we can only compute products of terms of the form

$$g_{\mathbf{n}}^{\prod_{i\in S} \alpha^{(2^i)}}$$

where $S \subseteq [0, n-1]$. However, $\prod_{i\in S} \alpha^{2^i} = \alpha^{\sum_{i\in S} 2^i}$, and $\sum_{i\in S} 2^i < 2^n$ for all subsets $S \subseteq [0, n-1]$. This is the basis for our assumption that the n-HDHE assumption is hard. In Appendix B, we discuss the difficulty of our assumption in the generic multilinear map model.

3.2 Security Of Our Construction

With our assumption defined, we can now state and prove the security of our scheme:

Theorem 3.3. *Let Setup' be the setup algorithm for a multilinear map, and suppose that the decisional n-Hybrid Diffie-Hellman Exponent assumption holds for Setup'. Then the identity-based broadcast encryption scheme in Construction 3.1 is a statically CPA-secure.*

Proof. Our proof closely follows the security proof for BGW [BGW05]. Suppose we have an adversary $\mathcal{A}$ that breaks the security of the scheme. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ that breaks the decisional n-HDHE problem for Setup'. $\mathcal{B}$ works as follows:

- $\mathcal{B}$ obtains a challenge tuple $(\mathsf{params}', \{X_i\}_{i\in[0,n]}, V, K)$ where:
  - $\mathsf{params}' \leftarrow \mathsf{Setup}'(2\mathbf{n})$ where $\mathbf{n}$ is the all-ones vector of length $n+1$.
  - $X_i = g_{\mathbf{e}_i}^{\alpha^{2^i}}$ for $i = 0, \ldots, n-1$ for a random $\alpha \in \mathbb{Z}_p$.
  - $X_n = g_{\mathbf{e}_n}^{\alpha^{2^n+1}}$

- $V = g_{\mathbf{n}}^t$ for a random $t \in \mathbb{Z}_p$
- $K = g_{2\mathbf{n}}^{t\alpha^{2^n}}$ or K is a random group element in $\mathbb{G}_{2\mathbf{n}}$.

- $\mathcal{B}$ simulates $\mathcal{A}$ until $\mathcal{A}$ submits a subset $S \subseteq [1, 2^n - 1]$ of users that $\mathcal{A}$ will challenge.
- $\mathcal{B}$ chooses a random $r \in \mathbb{Z}_p$. It sets

$$Y = \frac{g_{\mathbf{n}}^r}{\prod_{u \in S} Z_{2^n - u}}$$

where the Z_j are calculated from the X_i as before. This amounts to setting

$$\gamma = r - \sum_{u \in S} \alpha^{2^n - u}$$

Since r is uniform in $\mathbb{Z}_p$ and independent of α, so is γ. $\mathcal{B}$ also computes

$$W' = e(g_{\mathbf{e}_0}, g_{\mathbf{e}_1}, \ldots, g_{\mathbf{e}_{n-2}}, X_{n-1}, g_{\mathbf{e}_n})$$

and

$$W = e(W', W')$$

Observe that $W = g_{2\mathbf{n}}^{\alpha^{2^n}}$.

- $\mathcal{B}$ gives $\mathcal{A}$ the public parameters $(W, \{X_i\}_{i \in [0,n]}, Y)$
- Now $\mathcal{A}$ is allowed to ask for private keys for users $u \notin S$. $\mathcal{B}$ computes

$$\mathsf{sk}_u = \frac{Z_u^r}{\prod_{j \in S} Z_{2^n - j + u}}$$

Observe that

$$\mathsf{sk}_u = g_{\mathbf{n}}^{r\alpha^u - \sum_{j \in S} \alpha^{(2^n - j + u)}} = g_{\mathbf{n}}^{\gamma\alpha^u + \sum_{j \in S} \alpha^{(2^n - j + u)} - \sum_{j \in S} \alpha^{(2^n - j + u)}} = g_{\mathbf{n}}^{\gamma\alpha^u}$$

as desired.

- When $\mathcal{A}$ asks for the challenge, $\mathcal{B}$ lets $\mathsf{Hdr} = (V, V^r)$ and responds with (Hdr, K). Observe that

$$V^r = g_{\mathbf{n}}^{rt} = g_{\mathbf{n}}^{t\left(\gamma + \sum_{u \in S} \alpha^{(2^n - u)}\right)}$$

which means (V, V^r) is a valid header for the set S. Also observe that if $K = g_{2\mathbf{n}}^{t\alpha^{(2^n)}}$, then K is the correct key for this header.

- When $\mathcal{A}$ returns a guess b for which K it is given, $\mathcal{B}$ returns b as its guess.

As shown above, $\mathcal{B}$ perfectly simulates the view of $\mathcal{A}$ in the broadcast encryption security game. Therefore, $\mathcal{B}$ has the same advantage as $\mathcal{A}$, which must therefore be negligible, as desired.

□

4 Our Symmetric Multilinear Map Construction

In this section, we give our second construction of broadcast encryption, this time from traditional symmetric multilinear maps. That is, we do not require the more complicated asymmetric structure of Construction 3.1, but can use a basic multilinear map. The idea, however, is very similar. We implement BGW [BGW05] in middle levels of the multilinear map, and use elements in the bottom level to generate the BGW public parameters. Similar to the graded encoding scheme, the BGW parameters will have the form $Z_u = g_n^{\alpha^u}$, which can be computed from the public parameters $X_i = g_1^{\alpha^{(2^i)}}$.

However, we run into a problem. With asymmetric maps, we could enforce that X_i could not be paired with itself. This was used to ensure that Z_{2^n} was not computable given X_i for $i = 0, \ldots, n$. However, in the symmetric multilinear map setting, X_{n-1} could be paired with itself, giving Z_{2^n}. Instead, we create a hole by limiting the total number of X_i that can be paired together. If we allow only $n-1$ of them to be paired together, the first hole occurs at Z_{2^n-1}. We therefore set $N = 2^n - 2$ so that the hole is at $N+1$ as in BGW.

Notice that a second hole occurs at $Z_{2^n+2^{n-1}-1}$, and since $2^n + 2^{n-1} - 1 < 2(2^n - 2) = 2N$, we can not yet compute all the Z_u needed by BGW. One possible fix is to include extra X_i that can be used to fill in the unwanted holes. Instead, we opt to restrict the bit representations of all identities in the system to having the same Hamming weight. We show that this allows the computation of all the necessary Z_u.

We now describe our scheme:

Construction 4.1. *Let* Setup' *be the setup algorithm for a multilinear map, where groups have order* p*. Our second identity-based broadcast scheme consists of the following algorithms:*

$\mathsf{Setup}(n, \ell)$ *Sets up a broadcast scheme for* n*-bit identities with Hamming weight* ℓ*. Run* Setup' *on* $n + \ell - 1$*, obtaining the public parameters* params' *for a multilinear map with target group* $\mathbb{G}_{n+\ell-1}$*. Let* $\alpha, \gamma \in \mathbb{Z}_p$ *be chosen at random. Let* $W = g_{n+\ell-1}^{\alpha^{(2^n-1)}}$*. Compute* $X_i = g_1^{\alpha^{(2^i)}}$ *for* $i = 0, \ldots, n$*. Lastly, let* $Y = g_{n-1}^{\gamma}$*. Output*

$$\mathsf{params} = (\mathsf{params}', W, \{X_i\}_{i \in [0,n]}, Y)$$

$\mathsf{KeyGen}(\mathsf{params}, \alpha, \gamma, u)$ *The secret key for an identity* $u \in \{0,1\}^n$ *of Hamming weight* ℓ *is*

$$\mathsf{sk}_u = g_{n-1}^{\gamma \alpha^u}$$

$\mathsf{Enc}(\mathsf{params}, S)$ *Let* $Z_j = g_{n-1}^{\alpha^j}$*. We will show shortly that we can compute all of the necessary* Z_j *from the* X_i*. Pick a random* $t \in \mathbb{Z}_p$ *and compute the key and header as*

$$K = W^t = g_{n+\ell-1}^{t\alpha^{(2^n-1)}} \quad \text{and}$$
$$\mathsf{Hdr} = \left(g_\ell^t \ , \ (Y \prod_{u \in S} Z_{2^n-1-u})^t \right) = \left(g_\ell^t \ , \ g_{n-1}^{t\left(\gamma + \sum_{u \in S} \alpha^{(2^n-1-u)}\right)} \right)$$

$\mathsf{Dec}(\mathsf{params}, u, \mathsf{sk}_u, S, \mathsf{Hdr})$ *If* $u \notin S$*, output* $\perp$*. Otherwise, write* $\mathsf{Hdr} = (C_0, C_1)$*. Also let* $Z'_u = g_\ell^{\alpha^u}$*. We will shortly show that* Z'_u *can be computed from the* X_i*. Compute*

$$K = \frac{e(Z'_u, C_1)}{e(\mathsf{sk}_u \cdot \prod_{j \in S, j \neq u} Z_{2^n-1-j+u} \ , \ C_0)}$$

If C_0, C_1 are as above, notice that we can write $K = g_{n+\ell-1}^c$ where

$$c = \alpha^u t(\gamma + \sum_{j \in S} \alpha^{2^n-1-j}) - (\gamma\alpha^u + \sum_{j \in S, j \neq u} \alpha^{2^n-1-j+u})t = t\alpha^{2^n-1}$$

as desired.

We need to show how to compute Z_j and Z'_j.

Claim 4.2. *Let $Z_j = g_{n-1}^{\alpha^j}$ and $Z'_j = g_\ell^{\alpha^j}$. Let $X_i = g_1^{\alpha^{(2^i)}}$ for $i = 0, \ldots, n$. Then, using group multiplications and paring operations on the X_i, it is possible to compute:*

- *Z'_j for $j \in [1, 2^n - 2]$ of weight exactly ℓ.*
- *Z_{2^n-1-j} for $j \in [1, 2^n - 2]$ of weight exactly ℓ.*
- *$Z_{2^n-1-j+u}$ for $j, u \in [1, 2^n - 2], j \neq u$ of weight exactly ℓ.*

Proof. Let $h(j)$ denote the Hamming weight of j. First, observe that we can easily compute $g_{h(j)}^{\alpha^j}$ for $j \in [2^n - 1]$ by paring together X_i where the ith bit of j is 1. This allows us to compute the Z'_j. We can also compute $g_{n-\ell}^{\alpha^{(2^n-1-j)}}$ for any j of weight exactly ℓ. Thus, we can pair with $g_{\ell-1}$ to get Z_{2^n-1-j}.

Now we show how to compute $Z_{2^n-1-j+u}$. $2^n - 1 - j$, written as a bit string, has Hamming weight $n - \ell$. Therefore, write $2^n - 1 - j = \sum_{i \in T} 2^i$ for some subset $T \subseteq [0, n-1]$ of size $n - \ell$. Similarly, write $u = \sum_{i \in U} 2^i$ for some subset $U \subseteq [0, n-1]$ of size ℓ. Notice that U and T are only disjoint if $2^n - 1 - j + u = 2^n - 1$, in which case $j = u$. Since we do not allow this case, there must be some $\hat{i} \in [0, n-1]$ inside U and T. Then we can write

$$2^n - 1 - j + u = \left(\sum_{i \in T \setminus \{\hat{i}\}} 2^i\right) + \left(\sum_{i \in U \setminus \{\hat{i}\}} 2^i\right) + 2^{\hat{i}+1}$$

which is the sum of $n + \ell - 1$ powers of two. This means we can write

$$Z_{2^n-1-j+u} = e\left(\{X_i\}_{i \in T \setminus \{\hat{i}\}}\ ,\ \{X_i\}_{i \in U \setminus \{\hat{i}\}}\ ,\ X_{\hat{i}+1}\right)$$

which is the pairing of $n + \ell - 1$ of the X_i, as desired. □

Setting n and ℓ Suppose we want to handle λ-bit identities. We would map those identities to bit strings of length n and weight ℓ. Therefore, we need

$$\lambda \geq \log_2 \binom{n}{\ell}$$

A simple solution which minimizes n (and hence the number of elements in the public parameters) is to set $n = \lambda + \lceil(\log_2 \lambda)/2\rceil + 1$ and $\ell = \lfloor n/2 \rfloor$. However, for existing multilinear map constructions, the multilinearity itself is expensive, so we might try to minimize the total multilinearity $n + \ell - 1$. When $\ell = \lfloor n/2 \rfloor$, the total multilinearity is roughly $1.5(\lambda + (\log_2 \lambda)/2)$. However, setting $n \approx 1.042(\lambda + (\log_2 \lambda)/2)$ and $\ell \approx 0.398(\lambda + (\log \lambda)/2)$ gives us roughly 2^λ identities with total multilinearity about $1.440(\lambda + (\log \lambda)/2)$, slightly beating the trivial construction. The following table gives the settings of n and ℓ which minimize the total multilinearity for common identity lengths:

Length of identities (λ)	n	ℓ	Total Multilinearity ($n + \ell - 1$)
128	138	52	189
160	175	62	236
256	272	103	374
512	545	200	744

Implementation As with Construction 3.1, we must take advantage of the secrets used to construct the multilinear map to compute the X_i. We also need to re-randomize the header components. This time, however, there are two groups that need re-randomization terms: $\mathbb{G}_\ell$ and $\mathbb{G}_{n-1}$. No other re-randomization parameters are necessary.

4.1 The Multilinear Diffie-Hellman Exponent Assumption

We define the computational (n, ℓ)-multilinear Diffie-Hellman Exponent ((n, ℓ)-MDHE) Problem as follows: Let $\mathsf{params} \leftarrow \mathsf{Setup}'(n + \ell - 1)$. Choose random $\alpha, t \in \mathbb{Z}_p$, and let $X_i = g_1^{\alpha^{(2^i)}}$ for $i = 0, \ldots, n$. Let $V = g_\ell^t$. Given $(\{X_i\}_{i\in[0,n]}, V)$, the goal is to compute $K = g_{n+\ell-1}^{t\alpha^{(2^n-1)}}$.

As before, we define the decisional version as the problem of distinguishing K from a random element in $G_{n+\ell-1}$.

Definition 4.3. We say the decisional (n, ℓ)-multilinear Diffie-Hellman Exponent assumption holds for Setup' if, for any polynomial n and probabilistic polynomial time algorithm $\mathcal{A}$, $\mathcal{A}$ has negligible advantage in solving the (n, ℓ)-multilinear Diffie-Hellman Exponent problem.

This problem appears difficult for the same reasons as the n-HDHE assumption from Section 3. Computing $K = g_{n+\ell-1}^{t\alpha^{(2^n-1)}}$ requires pairing $V = g_\ell^t$ with a term $g_{n-1}^{\alpha^{(2^n-1)}}$, which must in turn be computed from the X_i. However, there is no way to pair at most $n - 1$ of the X_i to create the desired exponent $2^n - 1$. In Appendix B, we discuss the difficulty of the (n, ℓ)-MDHE problem in the generic multilinear map model.

4.2 Security of Our Construction

With our assumption defined, we can now state the security of our scheme:

Theorem 4.4. *Let Setup' be the setup algorithm for a multilinear map, and suppose that the decisional (n, ℓ)-multilinear Diffie-Hellman Exponent assumption holds for Setup'. Then the identity-based broadcast encryption scheme in Construction 4.1 is a statically CPA-secure.*

Proof. Again, our proof follows BGW [BGW05]. Suppose we have an adversary $\mathcal{A}$ that breaks the security of the scheme. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ that breaks the decisional MDHE problem for Setup'. $\mathcal{B}$ works as follows:

- $\mathcal{B}$ obtains a challenge tuple $(\mathsf{params}', \{X_i\}_{i\in[0,n+1]}, V, K)$ where:
  - $\mathsf{params}' \leftarrow \mathsf{Setup}'(n + \ell - 1)$
  - $X_i = g_1^{\alpha^{(2^i)}}$ for $i = 0, \ldots, n$ for a random $\alpha \in \mathbb{Z}_p$
  - $V = g_\ell^t$ for a random $t \in \mathbb{Z}_p$
  - $K = g_{n+\ell-1}^{t\alpha^{(2^n-1)}}$ or K is a random element in $\mathbb{G}_{n+\ell-1}$.

- $\mathcal{B}$ simulates $\mathcal{A}$ until $\mathcal{A}$ submits a subset $S \subseteq [1, 2^n - 2]$ of users that all have Hamming weight ℓ.
- $\mathcal{B}$ chooses a random $r \in \mathbb{Z}_p$. It sets

$$Y = g_{n-1}^r / \prod_{u \in S} Z_{2^n - 1 - u}$$

  where the Z_j are calculated from the X_i as before. This amounts to setting

$$\gamma = r - \sum_{u \in S} \alpha^{2^n - 1 - u}$$

  Since r is uniform in $\mathbb{Z}_p$ and independent of α, so is γ. $\mathcal{B}$ computes

$$W = e(X_0, X_1, \ldots, X_{n-1}, g_{\ell-1})$$

  Observe that $W = g_{n+\ell-1}^{2^n - 1}$.
- $\mathcal{B}$ gives α the public parameters $(W, \{X_i\}_{i \in [0, n+1]}, Y)$
- Now $\mathcal{A}$ is allowed to ask for private keys for users $u \notin S$ of Hamming weight ℓ. $\mathcal{B}$ computes

$$\mathsf{sk}_u = Z_u^r / \prod_{j \in S} Z_{2^n - 1 - j + u}$$

  Observe that

$$\mathsf{sk}_u = g_{n-1}^{r\alpha^u - \sum_{j \in S} \alpha^{(2^n - 1 - j + u)}} = g_{n-1}^{\left(r - \sum_{j \in S} \alpha^{(2^n - 1 - j)}\right)\alpha^u} = g_{n-1}^{\gamma \alpha^u}$$

  as desired.
- When $\mathcal{A}$ asks for the challenge, $\mathcal{B}$ lets $\mathsf{Hdr} = (V, e(V, g_{n-1-\ell})^r)$ and responds with (Hdr, K). Observe that

$$e(V, g_{n-1-\ell})^r = g_{n-1}^{rt} = g_{n-1}^{t\left(\gamma + \sum_{u \in S} \alpha^{(2^n - 1 - u)}\right)}$$

  which means $(V, e(V, g_{n-1-\ell})^r)$ is a valid header for the set S. Also, observe that if $K = g_{n+\ell-1}^{t\alpha^{(2^n - 1)}}$, then K is the correct key for this header.
- When $\mathcal{A}$ returns a guess b for which K it is given, $\mathcal{B}$ returns b as its guess.

As shown above, $\mathcal{B}$ perfectly simulates the view of $\mathcal{A}$ in the broadcast encryption security game. Therefore, $\mathcal{B}$ has the same advantage as $\mathcal{A}$, which must therefore be negligible, as desired. □

5 Our Third Construction

In this section, we give our third and final broadcast scheme. This scheme is based on the basic broadcast scheme of Gentry and Waters [GW09], henceforth called the GW scheme. Like the BGW scheme, the GW scheme has public keys consisting of $O(N)$ elements, where N is the number of

users. Our idea is to, similar to Constructions 3.1 and 4.1, run the GW scheme in the higher levels of a multilinear map, and derive the public key elements from $O(\log N)$ low-level elements.

However, unlike the BGW public parameters, which are all derived from a single scalar $\alpha \in \mathbb{Z}_p$, each of the GW public key elements are derived from a separate random scalar. Therefore, we cannot possibly hope to simulate the GW public key elements exactly. Instead, we we generate them using a Naor-Reingold-style PRF [NR97].

Also, unlike the BGW scheme, the secret keys in the GW scheme have $O(N)$ group elements. To make our scheme more efficient, and more importantly to make our scheme identity-based, we need to shrink the secret keys to $O(\log N)$ elements. To accomplish this, we observe that the secret key components are actually some of the outputs of another Naor-Reingold-style PRF, and we can allow the secret key holder to compute just those outputs by puncturing the PRF, similar to Boneh and Waters [BW13].

We now present out scheme:

Construction 5.1. *Let* Setup' *be the setup algorithm for a multilinear map, where groups have order* p*. Our final identity-based broadcast scheme consists of the following algorithms:*

$\mathsf{Setup}(n)$ *Takes as input the length* n *of identities. Run the setup algorithm for a multilinear map,* Setup'*, to construct an* $n+1$*-linear map with parameters* params'*. Draw a random* $\alpha \in \mathbb{Z}_p$*. For* $i = 0, \ldots, n-1$ *and* $b = 0, 1$*, draw random* $\beta_{i,b} \in \mathbb{Z}_p$*. The public key is*

$$\mathsf{pk} = (\mathsf{params}', \{X_{i,b} = g_1^{\beta_{i,b}}\}_{i \in [0,n-1], b \in \{0,1\}}, W = g_{n+1}^{\alpha})$$

For any user $\mathbf{u} \in \{0,1\}^n$*, note that we can compute*

$$Z_{\mathbf{u}} \equiv g_n^{\prod_{i=1}^n \beta_{i,u_i}} = e(X_{1,u_1}, X_{2,u_2}, \ldots, X_{n,u_n})$$

$\mathsf{KeyGen}(\mathsf{params}, \alpha, \{\beta_{i,b}\}, \mathbf{u})$ *Pick a random* $r_{\mathbf{u}} \in \mathbb{Z}_p$*. Let*

$$\begin{aligned} U_0^{(\mathbf{u})} &= g_1^{r_{\mathbf{u}}} \\ U_i^{(\mathbf{u})} &= X_{i,1-u_i}^{r_{\mathbf{u}}} = g_1^{r_{\mathbf{u}} \beta_{i,1-u_i}} \quad \text{for } i = 1, \ldots, n \\ U_{n+1}^{(\mathbf{u})} &= g_n^{\alpha} Z_{\mathbf{u}}^{r_{\mathbf{u}}} = g_n^{\alpha + r_{\mathbf{u}} \cdot \prod_{i=1}^n \beta_{i,u_i}} \end{aligned}$$

The secret key for user $\mathbf{u}$ *is* $\mathsf{sk}_{\mathbf{u}} = \{U_i^{(\mathbf{u})}\}_{i \in [0,n+1]}$*.*

Observe that for $\mathbf{v} \neq \mathbf{u}$*, we can compute* $Z_{\mathbf{v}}^{r_{\mathbf{u}}}$ *by finding an* i^* *where* $v_{i^*} = 1 - u_{i^*}$*, and computing*

$$\begin{aligned} e(X_{1,v_1}, \ldots, X_{i^*-1,v_{i^*-1}}, U_{i^*}^{(\mathbf{u})}, X_{i^*+1,v_{i^*+1}}, \ldots, X_{n,v_n}) &= g_n^{r_{\mathbf{u}} \beta_{i^*,v_{i^*}} \cdot \prod_{i \neq i^*} \beta_{i,v_i}} \\ &= g_n^{r_{\mathbf{u}} \cdot \prod_{i=1}^n \beta_{i,v_i}} = Z_{\mathbf{v}}^{r_{\mathbf{u}}} \end{aligned}$$

$\mathsf{Enc}(\mathsf{params}, S)$ *Choose a random* $t \in \mathbb{Z}_p$ *and compute the key and header as*

$$K = W^t = g_{n+1}^{t\alpha} \quad \text{and} \quad \mathsf{Hdr} = \left(g_1^t \, , \, \left(\prod_{\mathbf{u} \in S} Z_{\mathbf{u}} \right)^t \right) = \left(g_1^t \, , \, g_n^{t \sum_{\mathbf{u} \in S} \prod_{i=1}^n \beta_{i,u_i}} \right)$$

where $Z_{\mathbf{u}}$ *are computed as above.*

$\mathsf{Dec}(\mathsf{params}, \mathbf{u}, \mathsf{sk}_{\mathbf{u}}, S, \mathsf{Hdr})$ *If* $\mathbf{u} \notin S$*, output* $\perp$*. Otherwise, write* $\mathsf{Hdr} = (C_0, C_1)$*. Compute*

$$k = \frac{e(U_{n+1}^{(\mathbf{u})} \cdot \prod_{\mathbf{v} \in S, \mathbf{v} \neq \mathbf{u}} Z_{\mathbf{v}}^{r_{\mathbf{u}}}\ ,\ C_0)}{e(U_0^{(\mathbf{u})}\ ,\ C_1)}$$

Observe that if (C_0, C_1) *are as above, we can write* k *as* g_{n+1}^c *where*

$$c = (\alpha + r_{\mathbf{u}} \prod \beta_{i,u_i} + \sum_{\mathbf{v} \in S, \mathbf{v} \neq \mathbf{u}} r_{\mathbf{u}} \prod \beta_{i,v_i}) \cdot t - r_{\mathbf{u}} \cdot (t \sum_{\mathbf{v} \in S} \prod \beta_{i,v_i}) = \alpha t$$

as desired.

Correctness follows from the comments above.

Differences from GW. In the Gentry and Waters scheme [GW09], the $Z_{\mathbf{u}}$ are generated independently and given explicitly in the public parameters (as elements of the source group $\mathbb{G}_1$). In our scheme, the $Z_{\mathbf{u}}$ are generated pseudorandomly by means of a Naor-Reingold PRF. Similarly, in the GW scheme, the $Z_{\mathbf{v}}^{r_{\mathbf{u}}}$ for $\mathbf{v} \neq \mathbf{u}$ are also given explicitly to user $\mathbf{u}$. In our scheme, we note that the $Z_{\mathbf{v}}^{r_{\mathbf{u}}}$ for fixed $\mathbf{u}$ actually form another Naor-Reignold PRF, which we puncture at the point $\mathbf{u}$ to allow user $\mathbf{u}$ to compute the necessary values without learning $Z_{\mathbf{u}}^{r_{\mathbf{u}}}$. Our puncturing follows the puncturing used by Boneh and Waters [BW13].

Comparison to Constructions 3.1 and 4.1. Construction 5.1 has a couple advantages and disadvantages over our previous schemes:

- Unlike the BGW-based schemes, there are no high-degree terms being generated. This means we do not need the secret parameters for the multilinear map to set up our scheme. Therefore, we can use a map from some trusted third party. We do, however, need to make sure re-randomization parameters are available in the groups $\mathbb{G}_1$ and $\mathbb{G}_n$ to re-randomize the header elements. If we are using a map that we did not set up, we also need to re-randomize the user secret keys.
- To handle identities of length λ, the total multilinearity of Construction 5.1 is $\lambda + 1$. Compare this to 2λ and $1.440(\lambda + (\log_2 \lambda)/2)$ from the previous constructions.
- On the negative side, secret keys in Construction 5.1 consist of $O(\log N)$ group elements, compared to the single element secret keys of the previous schemes.
- For security, we unfortunately are unable to prove security relative to a non-interactive assumption. In the original GW scheme, the security proof involved manipulating the $Z_{\mathbf{u}}$ for $\mathbf{u} \notin S$. Since each of the $Z_{\mathbf{u}}$ are independent in the GW scheme, this is achievable. For our scheme, however, the $Z_{\mathbf{u}}$ are generated from $O(\log N)$ parameters, meaning we cannot modify them independently. Instead, in Appendix B, we opt to prove security in the generic multilinear map model. We note, however, that we obtain a better generic security theorem than is possible for Constructions 3.1 and 4.1.

Acknowledgments

Dan Boneh and Mark Zhandry are supported by NSF, the DARPA PROCEED program, a grant from ONR, an IARPA project provided via DoI/NBC, and by a Google faculty scholarship.

Brent Waters is supported by NSF CNS-0915361 and CNS-0952692, CNS-1228599 DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or IARPA.

References

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, pages 440–456, 2005.

[BBW06] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography*, pages 52–64, 2006.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (Im)possibility of obfuscating programs. In *Advances in Cryptology — CRYPTO 2001*, number Im, 2001.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology — CRYPTO*, pages 258–275, 2005.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Asiacrypt*, pages 280–300, 2013.

[BZ13] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/642, 2013.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology — CRYPTO*, pages 476–493, 2013.

[Del07] Cécile Delerablée. Identity-Based Broadcast Encryption with Constant Size Ciphertexts and Private Keys. 2:200–215, 2007.

[DF02] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Proceedings of the Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.

[DF03] Y. Dodis and N. Fazio. Public key broadcast encryption secure against adaptive chosen ciphertext attack. In *Workshop on Public Key Cryptography (PKC)*, 2003.

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. *PAIRING 2007*, (July), 2007.

[FHPS13] Eduarda S.V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In *CRYPTO 2103*, pages 513–530, 2013.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. *Advances in Cryptology — CRYPTO 1993*, 773:480–491, 1994.

[FP12] Nelly Fazio and IrippugeMilinda Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In *Public Key Cryptography — PKC 2012*, volume 7293 of *LNCS*, pages 225–242, 2012.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology — EUROCRYPT*, pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *Proc. of FOCS 2013*, 2013.

[GST04] M. T. Goodrich, J. Z. Sun, , and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Proceedings of Crypto '04*, volume 2204 of *LNCS*, 2004.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *Advances in Cryptology — EUROCRYPT*, pages 171–188, 2009.

[HS02] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In *Proceedings of Crypto '02*, volume 2442 of *LNCS*, pages 47–60, 2002.

[LPQ12] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In *Public Key Cryptography*, pages 206–224, 2012.

[LSW10] Allison B. Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *IEEE Symposium on Security and Privacy*, pages 273–285, 2010.

[NNL01] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Proceedings of Crypto '01*, volume 2139 of *LNCS*, pages 41–62, 2001.

[NP00] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial cryptography 2000*, volume 1962 of *LNCS*, pages 1–20. Springer, 2000.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, pages 458–467, 1997.

[SF07] Ryuichi Sakai and Jun Furukawa. Identity-Based Broadcast Encryption. *IACR Cryptology ePrint Archive*, 2007.

A Extensions and Variations

A.1 Parameter Trade-offs

To handle N identities, our symmetric multilinear map scheme (Section 4) requires a total multilinearity of roughly $1.5 \log_2 N$, and roughly $\log_2 N$ group elements in the public key. Contrast this to the BGW scheme, which only requires multilinearity 2, but needs roughly $2N$ public key elements. Since multilinearity is expensive, here we discuss a generalization of both the BGW scheme and Construction 4.1 which allows interpolating between the two. By instantiating the scheme with the right parameters, it may be possible to obtain better performance.

Observe in our scheme that the main reason for the multilinearity is so that we can compute many different Z_u, Z'_u from relatively few X_i. For a set $\mathcal{ID}$ of users, the Z_u, Z'_u we need to compute are:

- Z'_u for $u \in \mathcal{ID}$
- Z_{h-u} and Z_{h-j+u} for $j, u \in \mathcal{ID}, j \neq u$, for some "hole" h.

We can generalize the requirements using the following definition:

Definition A.1. Let $\mathcal{ID}$ be a finite set of positive integers. We say a set T of positive integers (h, n, ℓ)-covers $\mathcal{ID}$ if:

- $h > \max_{u \in \mathcal{ID}} u$.
- For every $u \in \mathcal{ID}$, u can be represented as a sum of at most ℓ (possibly repeating) integers in T.
- For every $u, j \in \mathcal{ID}$, $h - u$ and $h - j + u$ can be represented as a sum of at most $n - 1$ (possibly repeating) integers in T.
- h can be represented as a sum of at most $n + \ell - 1$ (possibly repeating) integers in T.
- h cannot be represented as a sum of fewer than n (possibly repeating) integers in T.

Then the public key will consist of $X_i = g_1^{\alpha^i}$ for $i \in T$ (along with the value V). The requirements for T show that the necessary values of Z_u, Z'_u (as well as W) can be derived from the X_i. The security assumption the scheme will be based on the following problem. Let $\mathsf{params} \leftarrow \mathsf{Setup}'(n + \ell - 1)$, and choose random $\alpha, t \in \mathbb{Z}_p$. Let $X_i = g_1^{\alpha^i}$ for $i \in T$ and $V = g_\ell^t$. Given $(\{X_i\}_{i \in T}, V)$, the goal is to compute $K = g_{n+\ell-1}^{t\alpha^h}$. We call this the (T, h, n, ℓ)-(computational) generalized multilinear Diffie-Hellman Exponent (gMDHE) assumption. The decisional variant is the problem of distinguishing K from a random element in $\mathbb{G}_{n+\ell-1}$. The requirements for the hole h ensure that the (T, h, n, ℓ)-gMDHE problem is not trivially solvable.

Now we give some examples:

- $\mathcal{ID} = [1, N]$, $T = [1, N] \bigcup [N + 2, 2N]$, $h = N + 1$, $n = 2$, and $\ell = 1$. Here we recover the original BGW construction.
- $\mathcal{ID} = \{u \in [1, 2^n - 2] : u \text{ has weight exactly } \ell\}$, $T = \{2^0, 2^1, 2^2, \ldots, 2^n\}$, $h = 2^n - 1$. Here we recover the scheme in Construction 4.1.

- $\mathcal{ID} = \{u \in [1, 2^n - 2] : u \text{ has weight } \textit{at most } \ell\}$, $T = \{2^0, 2^1, 2^2, \ldots, 2^n, 2^n + 1\}$, $h = 2^n - 1$. Here we get a variant of the scheme in Construction 4.1 where identities do not all need the same weight. This construction shaves of logarithmic additive factors in the total multilinearity $n + \ell - 1$, at the expense of a more complicated complexity assumption.
- $\mathcal{ID} = [1, \frac{b^n-1}{b-1} - 1]$, $T = \{1, 2, 3, \ldots, b-1, b, 2b, \ldots, (b-1)b, b^2, \ldots, b^{n-1}, \frac{b^n-1}{b-1} + 1, \frac{b^n-1}{b-1} + 2, \ldots, \frac{b^n-1}{b-1} + b\}$, $h = \frac{b^n-1}{b-1}$, $\ell = n - 1$. Here we get a variant of the scheme that uses a base other than 2. The result is a somewhat larger identity space (or reduced multilinearity) at the expense of a significantly larger public key. Note that when $b = N$,$n = 2$, we again get the BGW scheme.

A.2 CCA Security

Similar to the BGW construction, we can also obtain CCA security. The construction utilizes a one-time signature scheme $(\mathsf{G}, \mathsf{Sign}, \mathsf{Ver})$. The main difference is that verification keys for the signature scheme cannot directly be hashed into the necessary group $\mathbb{G}_{n-1}$, as described by BGW. The reason is that, in current multilinear map constructions, users cannot sample elements from intermediate groups directly, but must instead combine elements of the public parameters together to arrive at group elements. Note that in the multilinear map construction of Garg, Gentry, and Halevi, users can sample the source group $\mathbb{G}_1$ directly. Therefore, we will hash verification keys into $\mathbb{G}_1$, and then lift the element to $\mathbb{G}_{n-1}$ by pairing with g_{n-2}.

B Security Using Generic Multilinear Maps

In this section, we discuss the security of our schemes in the generic multilinear map model. In particular, explain why our two assumptions, the n-HDHE and (n, ℓ)-MDHE assumptions, are secure in the generic model, provided p is sufficiently large. This shows Constructions 3.1 and 4.1 are statically secure in this model for sufficiently large p. We also directly show that Construction 5.1 is adaptively secure in this model for much smaller p. We note that adaptive proofs of security can also be obtained for Constructions 3.1 and 4.1, though for much larger p.

Generic Multilinear Maps Generic multilinear maps are a generalization of the generic group model. Let $\mathbf{n} \in \mathbb{Z}^\ell$ be the target integer vector. We represent the groups $\mathbb{G}_\mathbf{v}$ for $\mathbf{v} \in \mathbb{Z}^\ell$ using a random injective function $\xi : \mathbb{Z}_p \times \mathbb{Z}^\ell \to \{0,1\}^m$ mapping elements of the additive group $\mathbb{Z}_p$ and vectors $\mathbf{v}$ into strings of length m. We are given oracles Mult and Pair to compute the induced multiplication and pairing operation. More precisely, any algorithm in the generic multilinear map model interacts with the multilinear map using the following queries:

$\mathsf{Encode}(x, \mathbf{v})$ If $\mathbf{v} \in \mathbb{Z}^\ell$ is a non-negative integer vector satisfying $\mathbf{v} \leq \mathbf{n}$, then the response is $\xi(x, \mathbf{v})$. Otherwise return $\perp$. Note that we can recover the generator $g_\mathbf{v}$ for the group $\mathbb{G}_\mathbf{v}$ as $\mathsf{Encode}(1, \mathbf{v})$.

$\mathsf{Mult}(\xi_1, \xi_2, b)$ If $\xi_1 = \xi(x_1, \mathbf{v}_1)$ and $\xi_2 = \xi(x_2, \mathbf{v}_2)$ where $\mathbf{v}_1 = \mathbf{v}_2 = \mathbf{v}$, then return $\xi(x_1 + (-1)^b x_2, \mathbf{v})$. Otherwise, return $\perp$.

$\mathsf{Pair}(\xi_1, \xi_2)$ If $\xi_1 = \xi(x_1, \mathbf{v}_1)$ and $\xi_2 = \xi(x_2, \mathbf{v}_2)$ where $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v} \leq \mathbf{n}$, then return $\xi(x_1 \cdot x_2, \mathbf{v})$. Otherwise, return $\perp$.

Generic security of our assumptions. Using the techniques of Boneh, Boyen, and Goh [BBG05], it is straightforward to prove hardness results for the n-HDHE and (n, ℓ)-MDHE assumptions in the generic multilinear map model. However, these assumptions involve high degree exponents (as high as α^{2^n}), meaning the adversary can construct high degree polynomials (namely, degree $n2^n$) in the secrets of the assumption. As a result, we can only bound the generic adversary's advantage to $\approx t^2 2^n/p$, where t is the number of queries the adversary makes. This means we must set p to be somewhat large: if $n = \lambda$, λ-bit security would require $p \approx 2^{3\lambda}$, rather than the usual $p \approx 2^\lambda$.

The generic security of our assumptions, together with Theorems 3.3 and 4.4 also shows the generic static security of Constructions 3.1 and 4.1. We note that it is also possible to show generic *adaptive* security for these schemes. However, these generic security results still require $p \geq 2^{3\lambda}$. Next, we show that, for Construction 5.1, we can actually obtain adpative generic security for $p \approx 2^\lambda$.

Theorem B.1. *For any generic adversary $\mathcal{A}$ whose total number of queries to* Encode, Mult, Pair *is polynomial, $\mathcal{A}$ has negligible advantage in breaking the adaptive security of Construction 5.1, provided $1/p$ is negligible.*

Proof. Let $\mathcal{A}$ be a generic adaptive attacker. $\mathcal{A}$ plays the following game:

- The challenger choose random $\beta_{i,b}$ from $\mathbb{Z}_p$ for $i \in [1, n]$ and $b \in \{0, 1\}$, random $\alpha, t \in \mathbb{Z}_p$, and a random bit $c \in \{0, 1\}$. The challenger sets $k_c = \alpha t$ and k_{1-c} to be a random element in $\mathbb{Z}_p$.
- $\mathcal{A}$ receives $\{X_{i,b} = \xi(\beta_{i,b}, 1)\}_{i\in[1,n],b\in\{0,1\}}$, as well as $W = \xi(\alpha, n + 1)$.
- $\mathcal{A}$ can adaptively make secret key queries for identities $\mathbf{u} \in \{0, 1\}^n$. In response, $\mathcal{A}$ receives $\{U_i^{(\mathbf{u})}\}_{i\in[0,n+1]}$ where

$$\begin{aligned} U_0^{(\mathbf{u})} &= \xi(r_{\mathbf{u}}, 1) \text{ for a randomly chosen } r_{\mathbf{u}} \in \mathbb{Z}_p \\ U_i^{(\mathbf{u})} &= \xi(r_{\mathbf{u}} \cdot \beta_{i,1-u_i}, 1) \text{ for } i = 1, \ldots, n \\ U_{n+1}^{(\mathbf{u})} &= \xi(\alpha + r_{\mathbf{u}} \cdot \prod_{i=1}^{n} \beta_{i,u_i}, n) \end{aligned}$$

  We require that for a particular identity $\mathbf{u}$, $\mathcal{A}$ can only make a single key query for $\mathbf{u}$.
- $\mathcal{A}$ can also adaptively make queries to Encode, Mult, Pair.
- $\mathcal{A}$ makes a challenge query on a set S, subject to the restriction that $\mathbf{u} \notin S$ for any $\mathbf{u}$ queried in a secret key query. In response, $\mathcal{A}$ receives

$$\mathsf{Hdr} = (\xi(t, 1), \xi(t \cdot \sum_{\mathbf{v}\in S} \prod_{i=1}^{n} \beta_{i,v_i}, n))$$

  In addition, $\mathcal{A}$ receives $K_0 = \xi(k_0, n + 1)$ and $K_1 = \xi(k_1, n + 1)$.
- $\mathcal{A}$ can continue making secret key queries for identities $\mathbf{u} \notin S$.
- $\mathcal{A}$ produces a guess c' for c.

Now consider an algorithm $\mathcal{B}$ that plays the above game with $\mathcal{A}$. Rather than choose values for $\beta_{i,b}, \alpha, t, r_{\mathbf{u}}, k_0, k_1$, algorithm $\mathcal{B}$ treats them as formal variables. $\mathcal{B}$ maintains a list

$$L = \{(p_j, i_j, \xi_j)\}$$

where p_j is a polynomial in the variables $\{\beta_{i,b}\}_{i\in[1,n],b\in\{0,1\}}, \alpha, t, k_0, k_1, \{r_{\mathbf{u}}\}$, the integer i_j indexes the groups, and ξ_j is a string in $\{0,1\}^m$. The list is initialized with the tuples $(\beta_{i,b}, 1, \xi_{2i+b-1})$ for randomly generated strings $\xi_{2i+b-1} \in \{0,1\}^m$, as well as $(\alpha, n+1, \xi_{2n+1})$ for a random string $\xi_{2n+1} \in \{0,1\}^m$. Initially, L contains $2n+1$ entries.

The game starts with $\mathcal{B}$ giving $\mathcal{A}$ the tuple of strings $\{\xi_i\}_{i\in[1,2n+1]}$. Now, $\mathcal{A}$ is allowed to make the following queries:

$\mathsf{Encode}(x, i)$: If $x \in \mathbb{Z}_p$ and $1 \le i \le n+1$, then $\mathcal{B}$ looks for a tuple $(p, i, \xi) \in L$, where p is the constant polynomial equal to x. If such a tuple exists, then $\mathcal{B}$ responds with ξ. Otherwise, $\mathcal{B}$ generates a random string $\xi \in \{0,1\}^m$, adds the tuple (p, i, ξ) (again, where p is a constant polynomial equal to x) to L,and responds with ξ.

$\mathsf{Mult}(\xi_k, \xi_\ell, b)$: $\mathcal{B}$ looks for tuples $(p_k, i_k, \xi_k), (p_\ell, i_\ell, \xi_\ell) \in L$. If one or both tuples do not exist, then $\mathcal{B}$ responds with $\perp$. If both tuples are found, but $i_k \neq i_\ell$, then $\mathcal{B}$ responds with $\perp$. Otherwise, $\mathcal{B}$ lets $i \equiv i_k = i_\ell$, computes the polynomial $p = p_k + (-1)^b p_\ell$, and looks for a tuple $(p, i, \xi) \in L$. If the tuple is found, then $\mathcal{B}$ responds with ξ. Otherwise, $\mathcal{B}$ generates a random string ξ, adds the tuple (p, i, ξ) to L, and resonds with ξ.

$\mathsf{Pair}(\xi_k, \xi_\ell)$: $\mathcal{B}$ looks for tuples $(p_k, i_k, \xi_k), (p_\ell, i_\ell, \xi_\ell) \in L$. If one or both tuples do not exist, then $\mathcal{B}$ responds with $\perp$. If both tuples are found, but $i \equiv i_k + i_\ell > n+1$, then $\mathcal{B}$ responds with $\perp$. Otherwise, $\mathcal{B}$ computes the polynomial $p = p_k \cdot p_\ell$, and looks for a tuple $(p, i, \xi) \in L$. If the tuple is found, then $\mathcal{B}$ responds with ξ. Otherwise, $\mathcal{B}$ generates a random string ξ, adds the tuple (p, i, ξ) to L, and responds with ξ.

$\mathsf{KeyGen}(\mathbf{u})$: $\mathcal{B}$ creates a new formal variable $r_{\mathbf{u}}$. It adds the tuple $(r_{\mathbf{u}}, 1, \xi)$ to L for a randomly generated $\xi \in \{0,1\}^m$. It also adds tuples $(r_{\mathbf{u}}\beta_{i,1-u_i}, 1, \xi_i)$ for $i = 1, \ldots, n$, where the ξ_i are generated at random in $\{0,1\}^m$. Finally, it adds the tuple $(\alpha + r_{\mathbf{u}} \cdot \prod_{i=1}^n \beta_{i,u_i}, n, \xi)$ for another randomly generated $\xi \in \{0,1\}^m$. $\mathcal{B}$ responds with the list of ξ values generated in this step.

$\mathsf{Enc}(S)$: $\mathcal{B}$ creates new formal variables t, k_0, k_1. It adds several tuples to L:

$$(t, 1, \xi_1)\ ,\ (t \cdot \sum_{\mathbf{v}\in S} \prod_{i=1}^{n} \beta_{i,v_i}\ ,\ n\ ,\ \xi_2)\ ,\ (k_0\ ,\ n+1\ ,\ \xi_3)\ ,\ (k_1\ ,\ n+1\ ,\ \xi_4)$$

Where the ξ_i are randomly generated. $\mathcal{B}$ gives $\mathcal{A}$ the strings $\xi_1, \xi_2, \xi_3, \xi_4$.

$\mathcal{B}$ can increase m arbitrarily, thus making strings ξ hard to guess. Therefore, we can assume without loss of generality that $\mathcal{A}$ only makes Mult and Pair queries on strings obtained from $\mathcal{B}$.

After a polynomial number of queries, $\mathcal{A}$ returns a guess $c' \in \{0,1\}$. Now, $\mathcal{B}$ chooses a random $c \in \{0,1\}$. If also chooses random values for $\beta_{i,b}, \alpha, t \in \mathbb{Z}_p, r_{\mathbf{u}}$. It also chooses a random $k \in \mathbb{Z}_p$. $\mathcal{B}$ sets $k_c = \alpha t$ and $k_{1-c} = k$.

The simulation provided by $\mathcal{B}$ is perfect unless out choices for the variables $\beta_{i,b}, \alpha, t, k_0, k_1$ results in an equality between values for two values p_k, p_ℓ that is not an equality for polynomials. More precisely the simulation is perfect unless for some k, ℓ the following hold:

- $i_k = i_\ell$
- $p_k(\beta_{i,b}, \dots) - p_\ell(\beta_{i,b}, \dots) = 0$, yet the polynomials p_k, p_ℓ are not equal.

Let Fail be the event that these conditions hold for some k, ℓ. We need to bound the probability that Fail occurs. First, prior to choosing values for all the variables, consider setting $k_b = \alpha t$ *as polynomials*. We claim that this does not create any new polynomial equalities.

Claim B.2. *Substituting the formal variable k_b with the polynomial αt does not create any new polynomial equalities. That is, if $p_k \neq p_\ell$ before the substitution, the same is true after the substitution*

Before proving Claim B.2, we use it to finish the proof of Theorem B.1. Notice that each of the polynomials has degree at most $2n + 2$. For any pair k, ℓ, the Swartz-Zippel lemma then shows that, for $p_k \neq p_\ell$, the probability $(p_k - p_\ell)(\beta_{i,b}, \dots) = 0$ is at most $(2n + 2)/p$.

Let q_e, q_m, q_p, q_k be the total number of encode, Mult, Pair, and KeyGen queries made by $\mathcal{A}$. Then the total length of L is at most

$$|L| \leq q_e + q_m + q_p + (n + 2)q_k + (2n + 5)$$

Therefore, the number of pairs is at most

$$\binom{|L|}{2} \leq (q_e + q_m + q_p + (n + 2)q_k + (2n + 5))^2/2$$

Therefore, Fail happens with probability at most

$$(q_e + q_m + q_p + (n + 2)q_k + (2n + 5))^2(2n + 2)/2p$$

If Fail does not occur, $\mathcal{B}$'s simulation is perfect, and in this case c is independent from $\mathcal{A}$'s view (in particular, c was chosen *after* the simulation). It is straightforward to show that the $\mathcal{A}$'s advantage in winning the broadcast encryption experiment is at most

$$(q_e + q_m + q_p + (n + 2)q_k + (2n + 5))^2(n + 1)/2p$$

For polynomial q_e, q_m, q_p, q_k, n, this is negligible provided $1/p$ is negligible, as desired.

It remains to prove Claim B.2. Suppose there are two polynomials $p_k \neq p_\ell$ such that, when we replace the variable k_c with αt, $p_k = p_\ell$. This means $p_k - p_\ell = 0$. Moreover, $p_k - p_\ell$ must have contained a k_c term, and this term cannot have been multiplied by any other variables. Therefore, we can write $p_k - p_\ell$ as

$$p_k - p_\ell = C_0 k_c + C_1 k_{1-c} \tag{B.1}$$

$$+ C_2\alpha + t\sum_{\mathbf{u}\in S}\prod_{i=1}^{n}\beta_{i,u_i} \cdot \mathsf{poly}_0(t, \{r_\mathbf{v}\}_{\mathbf{v}\notin S}, \{r_\mathbf{v}\beta_{i,1-v_i}\}_{\mathbf{v}\notin S, i\in[1,n]}, \{\beta_{i,b}\}_{i\in[1,n],b\in\{0,1\}}) \tag{B.2}$$

$$+ \sum_{\mathbf{u}\notin S}(\alpha + r_\mathbf{u}\prod_{i=1}^{n}\beta_{i,u_i})(C_\mathbf{u} t + \mathsf{poly}_\mathbf{u}(\{r_\mathbf{v}\}_{\mathbf{v}\notin S}, \{r_\mathbf{v}\beta_{i,1-v_i}\}_{\mathbf{v}\notin S, i\in[1,n]}, \{\beta_{i,b}\}_{i\in[1,n],b\in\{0,1\}}) \tag{B.3}$$

$$+ \mathsf{poly}_1(t, \{r_\mathbf{v}\}_{\mathbf{v}\notin S}, \{r_\mathbf{v}\beta_{i,1-v_i}\}_{\mathbf{v}\notin S, i\in[1,n]}, \{\beta_{i,b}\}_{i\in[1,n],b\in\{0,1\}}) \tag{B.4}$$

Where poly_0 has degree 1, each of the $\mathsf{poly}_{\mathbf{u}}$ has degree 1, and poly_1 has degree $n+1$, and $C_0, C_1, C_2, C_{\mathbf{u}}$ are constants. If $p_k - p_\ell$ is non-zero, but substituting k_c as αt makes the difference zero, we can conclude the following:

- $C_0 \neq 0$, $C_1 = 0$
- $\sum_{\mathbf{u} \neq S} C_{\mathbf{u}} = -C_0$. In particular, there is a $\mathbf{u} \notin S$ with $C_{\mathbf{u}} \neq 0$.

Now pick some $\mathbf{u} \notin S$ with $C_{\mathbf{u}} \neq 0$ and expand out all of the polynomials, looking for monomials $M = t r_{\mathbf{u}} \prod_{i=1}^{n} \beta_{i,u_i}$. Clearly, Line B.1 gives no such monomials. All monomials in Line B.2 involving t contain a product $\prod_{i=1}^{n} \beta_{i,v_i}$ for some $\mathbf{v} \in S$ — in particular, $\mathbf{v} \neq \mathbf{u}$. Therefore, Line B.2 gives no such monomials either. Line B.3 gives exactly one, with a coefficient of $C_{\mathbf{u}}$. Now, suppose Line B.4 (that is, poly_1) contained the monomial M. This means we can build M by taking the product of a subset of the terms $t, \{r_{\mathbf{v}}\}_{\mathbf{v} \notin S}, \{r_{\mathbf{v}} \beta_{i,1-v_i}\}_{\mathbf{v} \notin S, i \in [1,n]}, \{\beta_{i,b}\}_{i \in [1,n], b \in \{0,1\}}$. We can infer the following:

- t must be included exactly once in the product
- $r_{\mathbf{v}} \beta_{i,1-v_i}$ for any $\mathbf{v} \notin S, i \in \{0,1\}$ cannot be included in the product. Otherwise, if $\mathbf{v} = \mathbf{u}$, there will be some $\beta_{i,1-u_i}$ with positive exponent, and if $\mathbf{v} \neq \mathbf{u}$, then $r_{\mathbf{v}}$ will have a positive exponent.
- $r_{\mathbf{u}}$ must therefore be included exactly once in the product. $r_{\mathbf{v}}$ for $\mathbf{v} \neq \mathbf{u}$ cannot be included.
- β_{i,u_i} must be included for each i.

This means we must multiply $n+2$ of the terms together, exceeding the maximum degree of poly_1. Therefore, We conclude that Line B.4 does not have any monomial M. This means that the total coefficient for M is $C_{\mathbf{u}} \neq 0$. This is true even *after* substituting k_b with αt, contradicting the assertion that $p_k - p_\ell = 0$. This completes the proof of Theorem B.1. □