Broadcast and Trace with N^ϵ Ciphertext Size from Standard Assumptions

Rishab Goyal[1] ⋆, Willy Quach[2] ⋆⋆, Brent Waters[1,3] ⋆⋆⋆, and Daniel Wichs[2] †

[1] University of Texas at Austin
[2] Northeastern University
[3] NTT Research

Abstract. We construct a *broadcast and trace* scheme (also known as *trace and revoke* or *broadcast, trace and revoke*) with N users, where the ciphertext size can be made as low as $O(N^\varepsilon)$, for any arbitrarily small constant $\varepsilon > 0$. This improves on the prior best construction of broadcast and trace under standard assumptions by Boneh and Waters (CCS '06), which had ciphertext size $O(N^{1/2})$. While that construction relied on bilinear maps, ours uses a combination of the learning with errors (LWE) assumption and bilinear maps.

Recall that, in both *broadcast encryption* and *traitor-tracing* schemes, there is a collection of N users, each of which gets a different secret key sk_i. In broadcast encryption, it is possible to create ciphertexts targeted to a subset $S \subseteq [N]$ of the users such that only those users can decrypt it correctly. In a traitor tracing scheme, if a subset of users gets together and creates a decoder box D that is capable of decrypting ciphertexts, then it is possible to trace at least one of the users responsible for creating D. A broadcast and trace scheme intertwines the two properties, in a way that results in more than just their union. In particular, it ensures that if a decoder D is able to decrypt ciphertexts targeted toward a set S of users, then it should be possible to trace one of the users in the set S responsible for creating D, even if other users outside of S also participated. As of recently, we have essentially optimal broadcast encryption (Boneh, Gentry, Waters CRYPTO '05) under bilinear maps and traitor tracing (Goyal, Koppula, Waters STOC '18) under LWE, where the ciphertext size is at most poly-logarithmic in N. The main contribution of our paper is to carefully combine LWE and bilinear-map based components, and get them to interact with each other, to achieve broadcast and trace.

⋆ Email: goyal@utexas.edu. Supported by IBM PhD Fellowship.
⋆⋆ Email: quach.w@husky.neu.edu
⋆⋆⋆ Email: bwaters@cs.utexas.edu. Supported by NSF CNS-1908611, CNS-1414082, DARPA SafeWare and Packard Foundation Fellowship.
† Email: wichs@ccs.neu.edu. Research supported by NSF grants CNS-1314722, CNS-1413964, CNS-1750795 and the Alfred P. Sloan Research Fellowship.

1 Introduction

Broadcast Encryption. In *broadcast encryption*, as introduced by Fiat and Naor [FN94], a broadcaster can encrypt a message m to an arbitrary subset $S \subseteq [N]$ of indexed users, which results in a ciphertext ct. The i-th user is given a secret key sk_i and can decrypt the ciphertext ct iff $i \in S$. When designing broadcast encryption systems, a primary goal is to achieve short ciphertexts, ideally independent of the number of users N. (In order to decrypt, one must also know the description of S, but we count this separately from the ciphertext size.) Almost all of the earliest proposed solutions were not collusion resistant [FN94, Sti97, SVT98, GSW00, HS02, DF02, GST04], but in 2005 Boneh, Gentry and Waters [BGW05] gave a collusion-resistant system from bilinear maps with ciphertext size that is independent of N; in particular, ciphertexts consist of just three group elements.[4]

Traitor Tracing. A closely related primitive called *traitor tracing* was introduced by Chor, Fiat and Naor [CFN94]. Here, a broadcaster encrypts messages to the entire set of N users, where the i-th user is given a secret key sk_i that always decrypts the broadcaster's ciphertexts. If some subset $T \subseteq [N]$ of users ("traitors") gets together and pools their secret keys to produce a decoder algorithm D that can decrypt the broadcaster's ciphertexts, then there is a tracing procedure that can identify at least one of the users in the set T.[5] While earlier tracing systems [CFN94, SW98, CFNP00, SSW01, PST06] were not collusion resistant, Boneh, Sahai and Waters [BSW06] showed how to leverage bilinear maps to provide collusion resistant systems with $N^{\frac{1}{2}}$ sized ciphertexts. Very recently, Goyal, Koppula and Waters [GKW18] constructed a traitor tracing scheme with essentially optimal ciphertext size, which only scales poly-logarithmically in the number of users N, under the Learning with Errors (LWE) assumption.

Broadcast and Trace. The concepts of broadcast encryption and traitor tracing are naturally intertwined to form a *broadcast and trace* system [NP00, NNL01] (also known as a "trace and revoke" or "broadcast,trace and revoke" system). Here we want the ability to broadcast to an arbitrary set of users *and* the ability to trace any rogue decoding algorithm or box. However, the combination of broadcast and tracing security is more than just the sum of the parts – the two requirements interact with each other in a non-trivial way. In particular, the tracing property now also incorporates the broadcast set S as follows. If some subset T of users get together and construct a decoder algorithm D that can decrypt ciphertexts targeted to a certain set S, then there is a tracing procedure that can identify at least one of the users in $T \cap S$ that contributed to constructing

[4] In a collusion-resistant system, there is no a-priori bound on the number of secret keys the adversary can see. Our discussion and comparisons will be in the collusion resistant setting.

[5] For both broadcast and traitor tracing, we require that the encryption procedure is public key. In traitor tracing, while some prior works also require that the tracing procedure is public key, here we consider secret-key tracing.

D, even if some other users outside of S also participated. At that point one might take certain punitive actions against such a user and most likely remove them from the broadcast set S used in future encryptions.

The requirement that the tracing procedure identifies a user in the set $T \cap S$ rather than just any user in T is important here. For example, consider a scenario where a broadcast encryption scheme is used to encrypt messages to various subgroups within a company, and one of the board members colludes with an intern to publish a decoder that decrypts ciphertexts targeted to the set S of all board members. In this case, we want to trace the responsible board member and *not* just the intern. Alternately, even in setting involving a flat hierarchy where with no distinctions between different types of users (e.g., broadcasting cable TV), this requirement is important. Assume some user i publishes an illegal decoder D online, and then gets identified and revoked from the broadcast set S, causing D to stop working. But then a new traitor j colludes with i to publish a new decoder D' that is able to decrypt newly created ciphertexts for the new broadcast set S. In this case, we need to identify the *new* traitor j (and not just the old traitor i who is already known) so that we can also revoke j them from the broadcast set, and eventually revoke all misbehaving users through this process.

The requirement that the tracing procedure identifies a user in $T \cap S$ and not just T is also what makes the problem of achieving broadcast and trace more technically challenging than just tackling the problems of broadcast encryption and traitor tracing separately. Otherwise, one could trivially construct a broadcast and trace cryptosystem with a basic combination of a broadcast encryption and a traitor tracing, by secret sharing the message across the two systems.

Historically, progress on broadcast and trace has followed progress on the two problems separately. For example, soon after the construction of the first broadcast with optimally succinct ciphertexts [BGW05] and the first traitor tracing scheme with $N^{\frac{1}{2}}$ sized ciphertexts [BSW06], the work of Boneh and Waters [BW06] built upon these works to give a broadcast and trace system with $N^{\frac{1}{2}}$ sized ciphertexts by carefully combining techniques from the two bilinear map-based schemes. We also have essentially optimal constructions of broadcast and trace using (positional) witness encryption [GVW19], but we don't currently have any construction that beats the $N^{\frac{1}{2}}$ barrier under any standard assumptions. Very recently, we finally reached the point where we have essentially optimal ciphertext size in both broadcast and traitor tracing separately, and therefore the time is ripe to revisit the problem of constructing an optimal broadcast and trace system under standard assumptions. However, the optimal broadcast scheme [BGW05] is based on bilinear maps and the optimal traitor tracing scheme [GKW18] is based on LWE. [6] Can we still come up with a way to combine these different techniques to get an optimal broadcast and trace scheme? In particular, can we meaningfully combine bilinear-map and LWE based com-

[6] There are actually no known collusion resistant broadcast encryption schemes from LWE other than the trivial one with N-sized ciphertexts.

ponents and get them to interact with each other to get something beyond just the sum of the parts?

Our Results. In this work, we show how to combine bilinear-map and LWE based techniques to construct broadcast and trace.

Theorem 1.1 (informal). *Under the Decisional Bilinear Diffie-Hellman Exponent (DBDHE) assumption and the Learning with Errors (LWE) assumptions, for any constant $\varepsilon > 0$, there exists a broadcast and trace scheme with ciphertext size $\widetilde{O}(N^{\varepsilon})\mathsf{poly}(\lambda)$, where N is the number of users and λ is the security parameter.*

As a tool in our construction, we rely on a black-box use of *attribute-based encryption (ABE)* with *succinct ciphertexts*, whose size is essentially independent of the attribute size (the attribute is assumed to be known by the decryption procedure but is not counted in the ciphertext size). This can be seen as a generalization of broadcast encryption, which is a special case of succinct ABE where the attribute is S and keys sk_i are associated with policies that allow decryption iff $i \in S$. Currently, we can instantiate such succinct ABE schemes for $\mathbf{NC}^1$ circuits using bilinear maps [HLR10,ALDP11,AHL+12,YAHK14]. However we note that: (1) while the best current construction of succinct ABE relies on the DBDHE assumption, it is very conceivable that this could be improved to milder bilinear assumptions in future work, and (2) while current constructions only work for $\mathbf{NC}^1$ circuits, if we had a succinct ABE for even the slightly larger class of $\mathbf{TC}^1$ circuits, we could leverage it to get essentially optimal broadcast and trace with only a poly-logairthmic dependence on N. Therefore, we state the following more general result of our work, which shows that future advances in succinct ABE will also lead to advances in broadcast and trace:

Theorem 1.2 (informal). *Assuming the existence of ABE with succinct ciphertexts for $\mathbf{NC}^1$ and the LWE assumption, for any constant $\varepsilon > 0$, there exists a broadcast and trace scheme with ciphertext size $\widetilde{O}(N^{\varepsilon})\mathsf{poly}(\lambda)$. Assuming the existence of ABE with succinct ciphertexts for $\mathbf{TC}^1$ and the LWE assumption, there exists a broadcast and trace scheme with ciphertext size $\mathsf{poly}(\log N, \lambda)$.*

Overall, picking a smaller constant ε yields shorter ciphertexts, at the cost of making both the secret keys bigger and the decryption time longer, with the exact tradeoff depending on the parameters of the underlying ABE.

Our main technique is to use a bilinear-based succinct ABE scheme for $\mathbf{NC}^1$ and use it to evaluate an LWE-based scheme, which we carefully engineer to be in $\mathbf{NC}^1$. This allows us to meaningfully combine the cryptographic properties of both schemes and achieve more than just their union. We provide a detailed technical overview below.

1.1 Technical Overview

We now give a technical overview of our result. We start by giving a high-level description of the state of the art construction of traitor tracing based on the

works of [BSW06, GKW18, CVW+18a]. Then we discuss our approach to incorporate broadcast and get a broadcast and trace system. Concretely, we describe a 3-step construction of traitor tracing and then show how to augment each of the steps to also accommodate broadcast. Finally, we discuss the complications that arise in realizing the augmented steps and our solutions.

Traitor Tracing in Three Steps The following is a high-level description of a 3-step approach to construct traitor-tracing based on the works of [BSW06, GKW18, CVW+18a].

Step 1: Traitor Tracing from PLBE. The first step is to construct traitor tracing from a conceptually simpler primitive called *private linear broadcast encryption* (PLBE) [BSW06]. A PLBE scheme is initialized with a master public key pk, a master secret key msk, and N user secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$. There is a "public encryption" procedure which encrypts a message m under pk and guarantees that every user secret key sk_i will decrypt it correctly. There is also a "secret encryption" procedure which encrypts a message m under msk with respect to some index $\mathsf{ind} \in [N+1]$ and guarantees that a user secret key sk_i will decrypt m correctly iff $i \geq \mathsf{ind}$. Moreover, one cannot distinguish a public encryption from a secret encryption or a secret encryption with one index ind versus another index ind' unless one has a secret key sk_i that correctly decrypts in one case but not the other. Lastly, a secret encryption with the index $\mathsf{ind} = N + 1$ should hide the message m even given all the secret keys. An important subtlety, discovered by [GKW18], is that these indistinguishability properties must hold even if the adversary is given a single arbitrary query to the secret encryption oracle, in addition to getting the challenge ciphertext.

A PLBE scheme can directly be used as a traitor tracing scheme, where the "secret encryption" procedure is used to implement the tracing algorithm. Assume some subset of users get together and create a decoder D that can correctly decrypt ciphertexts produced by the public encryption procedure. Then D should also correctly decrypt ciphertexts produced by the secret encryption procedure with index $\mathsf{ind} = 1$ (since these are indistinguishable even given all the user secret keys). On the other hand the decoder cannot correctly decrypt ciphertexts produced by the secret encryption procedure with index $\mathsf{ind} = N+1$ (since these are undecryptable even given all the user secret keys). Therefore there must be at least one index ind^* where the decoder's probability of successful decryption drops significantly between being given secret encryptions with index ind^* and $\mathsf{ind}^* + 1$. But this can only be the case if the decoder was created with knowledge of $\mathsf{sk}_{\mathsf{ind}^*}$ (since otherwise the two cases are indistinguishable). Therefore, this allows the tracing algorithm to finger user ind^* as a traitor.[7]

[7] The above argument implicitly assumes that, if an adversary can create a decoder D that can distinguish between certain types of ciphertexts, then the adversary himself can also distinguish. As observed by [GKW18], this is more subtle than it appears and not true in general. The issue arises from a discrepancy between the decoder's advantage, which is calculated only over the choice of the encryption

Step 2: PLBE from ABE and mixed FE. The work of [GKW18] showed how to construct PLBE from two simpler primitives. The first primitive is a *(key-policy) attribute-based encryption (ABE)* [SW05] for circuits, which is already known from LWE [GVW13]. The second primitive is a restricted form of functional encryption for the comparison function, called *mixed functional encryption (Mixed FE)*.

In Mixed FE, private keys sk_i are associated with values i and the adversary can collect an unrestricted number of such keys. There is a "secret encryption" algorithm which requires the master secret key and is used to encrypt an index ind. If a user with a secret key for input i decrypts a ciphertext encrypting an index ind, the output is 1 if $i \geq \mathsf{ind}$ and 0 otherwise. Security says that, given an encryption of ind and many secret keys $\{\mathsf{sk}_i\}_{i \in T}$, the adversary does not learn anything about ind beyond the decryptions. Security must hold even if the attacker is also allowed to make 1 query to the secret encryption oracle, in addition to getting the challenge ciphertext. So far, the above can be thought of as a secret-key FE scheme for the comparison functions with security for unbounded number of keys and two ciphertexts, which can actually be constructed based only on one-way functions via garbled circuits [GVW12, KMUW18]. The additional property that makes mixed FE different, is that it also requires a public encryption algorithm, which only uses a public key and generates ciphertexts ct that always decrypt to 1 under all private keys. Such an algorithm is a bit unusual in that there is no further choice in the index. The security of the system requires that an attacker who makes a single query to the "secret encryption" oracle cannot distinguish a public encryption versus a secret encryption or a secret encryption with one index ind versus another index ind' unless he has a secret key sk_i that decrypts to 0 in one case and 1 in the other. The name "Mixed FE" is derived from the fact that the scheme has both a public and secret encryption procedure.

The semantics of mixed FE scheme are already very close a PLBE; in both cases there is a "public encryption" and "secret encryption" algorithm and one should not be able to distinguish different types of ciphertexts without having a secret key that decrypts differently in one case versus the other. The one important difference is that, in PLBE, the ciphertext also incorporates a message m, while in mixed FE there is no message. The work of [GKW18] showed how to use ABE on top of a mixed FE to incorporate a message into the ciphertext and get PLBE. Essentially, the PLBE scheme uses a mixed FE ciphertext as an attribute and then encrypts the message m under this attribute via an ABE scheme. In more detail, to implement public PLBE encryption (resp. secret PLBE encryption for index ind), first create public mixed-FE ciphertext (resp. secret mixed-FE ciphertext for the index ind) denoted $\mathsf{ct}_{\mathrm{mfe}}$ and then use

randomness after the keys have been fixed, and the advantage of the adversary, which is calculated also over the choice of the keys and randomness simultaneously. To make this step work, [GKW18] showed that one needs to start with a stronger form of PLBE security, where the adversary also gets one query to the secret encryption oracle.

the ABE scheme to encrypt the message m under the attribute $\mathsf{ct}_{\mathrm{mfe}}$. To create a PLBE secret key sk_i for index i, first create a mixed-FE secret key $\mathsf{sk}_{\mathrm{mfe},i}$ for the index i and then set sk_i to be an ABE secret key for the function $f_{\mathsf{sk}_{\mathrm{mfe},i}}$ which takes as input $\mathsf{ct}_{\mathrm{mfe}}$ and decrypts it with $\mathsf{sk}_{\mathrm{mfe},i}$. This incorporates the message m into the PLBE scheme, while having the mixed FE dictate whether or not the message is decryptable and preserving the mixed FE security properties.

Step 3: Constructing mixed FE. The work of [GKW18] gave a self-contained albeit somewhat complex construction of mixed FE from the LWE assumption. Later, the work of [CVW+18a] gave two simple and modular constructions of mixed FE from previously studied primitives: one from lockable (AKA, compute-and-compare) obfuscation [WZ17, GKW17] and one from (key-homomorphic) private constrained PRFs (PCPRFs) [CC17, BTVW17, CVW18b]. Since either of these can be instantiated under LWE, so can the final mixed FE and traitor-tracing schemes.

We recall the PCPRF-based construction of mixed FE from [CVW+18a], which we will later rely on for our results. A PCPRF consists of a pseudorandom function (PRF) family $F_K(\cdot)$ with a key K. The constrained property states that given K, there is a way to generate a constrained key K_P for some program P such that $F_K(x) = F_{K_P}(x)$ if $P(x) = 0$. In addition, the constraints are private in that, one cannot distinguish between seeing the constrained key K_P, along the evaluations of $y_i = F_K(x_i)$ on various inputs x_i for which $P(x_i) = 1$, versus being given a "dummy key" that does not depend on P along with uniformly random values y_i.

Given a PCPRF for the comparison functions $P_{\mathsf{ind}}(i) = 1$ iff $i \geq \mathsf{ind}$, one can construct a simple mixed FE scheme as follows. The master secret key is a PRF key K and the secret key for an input i is the value $y = F_K(i)$. An encryption is a PRF key K^* and the decryption algorithm outputs 1 iff $y \neq F_{K^*}(x)$. A public encryption consists of a "dummy key" K^*. A secret encryption of some index ind consists of the constrained key $K^* = K_{P_{\mathsf{ind}}}$. It's relatively easy to see that the above gives a mixed FE scheme that is secure with $q = 0$ queries to the secret encryption oracle. In particular, the only way to distinguish different types of PRF keys is to have an evaluation on some i for which one is constrained and the other is not.

To get a mixed FE scheme with security for $q = 1$ queries to the secret encryption oracle, which is needed for traitor tracing, we rely on a PCPRF with an additional key homomorphic property saying that $F_K(x) + F_{K'}(x) = F_{K+K'}(x)$. The construction is only slightly more complex. Now the master secret key consists of 2λ PRF keys $\{K_{j,b}\}_{j\in\lambda,b\in\{0,1\}}$ and the secret key for an input i consists of the values $\{y_{j,b} = F_{K_{j,b}}(i)\}_{j\in\lambda,b\in\{0,1\}}$. An encryption is a PRF key K^* and some "tag" value $z \in \{0,1\}^\lambda$ and the decryption algorithm outputs 1 iff $\sum_{j=1}^{\lambda} y_{j,z_j} \neq F_{K^*}(i)$. A public encryption consists of a random z and a "dummy key" K^*. A secret encryption for some index ind consists of a random z along with the constrained key $K'_{P_{\mathsf{ind}}}$ where $K' = \sum_{j=1}^{\lambda} K_{j,z_j}$. The above gives a mixed FE scheme which is secure with $q = 1$ queries to the secret encryption oracle. With overwhelming probability, the z value used in the

challenge ciphertext differs from the one used by the oracle in answering the encryption query in some position j, and therefore we can rely on the security of the PRF $F_{K_{j,z_j}}$ in essentially the same way as was done in the $q = 0$ query case.

Adding Broadcast to Traitor Tracing We now discuss how to "upgrade" the above ideas to construct a broadcast and trace scheme.

Perhaps the first approach one would try is to combine broadcast and traitor-tracing directly; e.g., secret-share the message and encrypt one share via a broadcast scheme and the other share via a traitor-tracing scheme. Indeed, we can use the broadcast scheme to restrict the set S of users that can recover the first share and therefore the encrypted message. Also, any decoder D that decrypts the full ciphertext correctly must also necessarily decrypt the second share, and therefore we can use the traitor-tracing scheme to trace at least one user $i \in [N]$ that participated in constructing D. However, even if the decoder D can decrypt ciphertexts targeted toward some restricted set S of users, the traitor tracing procedure might find a user $i \notin S$, which is not good enough for a broadcast and trace scheme, as explained earlier. To fix this, we need to incorporate the broadcast set S into the tracing procedure itself. We revisit the 3-step approach outlined above and show how to upgrade it to get a broadcast and trace scheme.

Updated Step 1: Broadcast and Trace from AugBE. We previously saw how traitor-tracing can be constructed from "private linear broadcast encryption" (PLBE). The work of [BW06] showed that broadcast and trace can analogously be constructed from an augmented version of PLBE, called "augmented broadcast encryption" (AugBE), which can be thought of as combining PLBE and broadcast encryption. In particular, an AugBE scheme has a master public key pk, a master secret key msk, and N user secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$. There is a "public encryption" procedure using pk, which encrypts a message m to a target set S, and guarantees that a secret key sk_i will decrypt correctly iff $i \in S$. There is also a "secret encryption" procedure using msk, which encrypts a message m to a target set S with respect to some index $\mathsf{ind} \in [N+1]$, and guarantees that a secret key sk_i will decrypt correctly iff $i \in S \wedge i \geq \mathsf{ind}$. Moreover, one cannot distinguish a public encryption from a secret encryption or a secret encryption with one index ind versus another index ind' (all with the same set S) unless one has a secret key sk_i that correctly decrypts in one case but not the other. A secret encryption with the index $\mathsf{ind} = N + 1$ should hide the message even given all the secret keys. As before, these indistinguishability properties must hold even if the adversary is given a single query to the secret encryption oracle. We want the ciphertext size to be small, much smaller than N. As in broadcast encryption, the decryption algorithm is also given the set S separately, but we do not count it as part of the ciphertext size.

The notion of AugBE already incorporates the broadcast encryption requirements directly in the definition. To see that it also allows us to trace a traitor in the set S, one can adapt the previous argument that PLBE implies tracing.

The tracing algorithm tests the decoder's success probability on secret encryptions with the fixed broadcast set S and all possible values of $\mathsf{ind} \in [N+1]$. As before, the decoder must be successful when $\mathsf{ind} = 1$ (since it is successful with public encryptions and the two are indistinguishable) but cannot be successful when $\mathsf{ind} = N+1$ (since such encryptions hide the message by definition) and so there must be some value ind^* such that success probability drops significantly between ind^* and $\mathsf{ind}^* + 1$. But this means that the decoder can distinguish between these two types of ciphertexts and, in order for that to happen, the decoder must have been created using knowledge of $\mathsf{sk}_{\mathsf{ind}^*}$ with $\mathsf{ind}^* \in S$. Thus the tracing algorithm can finger the user $\mathsf{ind}^* \in S$ as a traitor.

Updated Step 2: AugBE from Succinct ABE and BMFE. Recall that the work of [GKW18] constructed of PLBE from ABE and mixed FE. As our first contribution, we given an analogous result showing how to construct AugBE (the augmented form of PLBE) from two simpler primitives: a (succinct) ABE scheme and an augmented variant of mixed FE that we call "broadcast mixed FE" (BMFE). At a high level, we incorporate the set S into the ABE to ensure that only users $i \in S$ can decrypt correctly. But we also incorporate the set S into the mixed FE to ensure that the keys of users $i \notin S$ cannot help to distinguish between ciphertexts with different values of the index ind. We now go into more detail on how this is done.

A BMFE scheme can be thought of as an augmented form of mixed FE that includes the set S. In particular, a BMFE has master public key pk, a master secret key msk and allows us to create user secret keys sk_i for values $i \in [N]$. There is a "public encryption" procedure using pk, which takes as input a set $S \subseteq [N]$ and outputs a ciphertext ct that decrypts to 1 under *all* secret keys sk_i. There is also a "secret encryption" procedure using msk, which takes as input a set S and an index ind and outputs a ciphertext ct that decrypts to 1 under sk_i if $i \notin S \vee i \geq \mathsf{ind}$ and decrypts to 0 otherwise. The security of the system requires that an attacker with $q = 1$ queries to the "secret encryption" oracle cannot distinguish a public encryption versus a secret encryption or a secret encryption with one index ind versus another index ind' (all with the same set S) unless he has a secret key sk_i that decrypts to 0 in one case and 1 in the other.

Note that the decryptability conditions of AugBE ($i \in S \wedge i \geq \mathsf{ind}$) and of BMFE ($i \notin S \vee i \geq \mathsf{ind}$) differ from each other. However, these decryptability conditions match up to ensure that the only way to distinguish between ciphertexts with some index ind versus ones with index $\mathsf{ind}' > \mathsf{ind}$ is to have a key sk_i for some $i \in S \cap [\mathsf{ind}, \mathsf{ind}')$.

We can construct AugBE by combining together ABE with BMFE. In particular, the ABE scheme allows us to simultaneously add a message m to the BMFE and also to ensure that only the users in S can decrypt correctly. In more detail, the AugBE encryption consists of creating a BMFE ciphertext $\mathsf{ct}_{\mathrm{bmfe}}$ with some set S and index ind and then using the ABE to encrypt the message m under attribute $a = (S, \mathsf{ct}_{\mathrm{bmfe}})$. The AugBE secret key sk_i is an ABE secret key for a function $f_{i,\mathsf{sk}_{\mathrm{bmfe},i}}$ which has the BMFE secret key $\mathsf{sk}_{\mathrm{bmfe},i}$ inside it and checks that $i \in S$ and that $\mathsf{ct}_{\mathrm{bmfe}}$ decrypts to 1 under $\mathsf{sk}_{\mathrm{bmfe},i}$. It is easy to see that

the above construction ensures that the set S and the index ind correctly determine whether an AugBE ciphetext is decryptable while preserving the BMFE indistinguishability properties.

Up until now we have completely ignored efficiency and, in particular, the requirement that ciphertexts are small. To ensure this we need the following:

- Firstly, we need a succinct ABE where the ciphertext size is essentially independent of the attribute size, since the attribute includes the set S (the decryption algorithm gets the attribute, but we don't count it as part of the ciphertext). Succinct ABE can be thought of as generalizing broadcast encryption, where the latter is a special case of succinct ABE in which attributes are sets S, and keys are associated with policies of the form $f_i(S) = 1$ iff $i \in S$. Unfortunately, the current ABE systems from the LWE assumption [GVW13, BGG$^+$14] do not satisfy this form of succinctness, and we do not know how to achieve even broadcast encryption from LWE. On the positive side, we do have constructions of succinct ABE from bilinear maps [HLR10, ALDP11, AHL$^+$12, YAHK14]; however, these constructions can only support policies for circuits in $\mathbf{NC}^1$, unlike the LWE-based ones that can support circuits of arbitrary depth. Recall that, in our case, the ABE policy checks that $i \in S$ and that a BMFE ciphertext decrypts to 1. The first part is in $\mathbf{NC}^1$ and therefore we need to ensure that the BMFE decryption is in $\mathbf{NC}^1$.
- Secondly, we need a succinct BMFE scheme, where decryption is in $\mathbf{NC}^1$ and the ciphertext size is much smaller than N (the decryption procedure gets S but we do not count it in the ciphertext size). We next show how to construct this primitive under LWE.

Note that we are using a bilinear-based succinct ABE to evaluate the decryption of an LWE-based BMFE scheme, which will be in $\mathbf{NC}^1$. This allows us to meaningfully combine the security properties of a bilinear-based scheme and an LWE-based scheme to achieve more than just the union of their capabilities.

Updated Step 3: Constructing BMFE in $\mathbf{NC}^1$. Our goal now is to construct a succinct BMFE with decryption in $\mathbf{NC}^1$. Recall that BMFE is an augmented form of mixed FE for which we have constructions from LWE [GKW18, CVW$^+$18a]. We face two challenges:

- We need to incorporate the set S into mixed FE to get BMFE.
- We need to ensure that BMFE decryption is in $\mathbf{NC}^1$.

Let's start by showing how to augment mixed FE to get BMFE. Recall that we previously outlined the [CVW$^+$18a] construction of mixed FE from (key-homomorphic) private constrained PRFs (PCPRFs) for comparison constraints: $P_{\mathsf{ind}}(i) = 1$ iff $i \geq \mathsf{ind}$. We now outline how to upgrade this construction to get a BMFE scheme. For simplicity, we describe how to get BMFE with security against $q = 0$ queries to the secret encryption oracle; to get security for $q = 1$ queries, as is needed for broadcast and trace, we then employ the same trick as

in the mixed FE case. The master secret key of the BMFE scheme now consists of N PCPRF keys $\{K_j\}_{j\in N}$. The secret key of user i consists of the values $\{y_{i,j} = F_{K_j}(i)\}_{j\neq i}$ for $i, j \in [N]$. To create a "secret encryption" to a set S with respect to an index ind, the encryptor computes a key $K^+ = \sum_{j\notin S} K_j$ and then constrains it on the program P_{ind} to get $K^* = K^+_{P_{\mathsf{ind}}}$. To create a "public encryption" to a set S, the encryptor chooses a dummy constrained key K^*. The decryption procedure takes a ciphertext K^* and outputs 1 iff $F_{K^*}(i) \neq \sum_{j\notin S} y_{i,j}$. We rely on the fact that, the only way to distinguish different types of BMFE ciphertexts (i.e., PRF keys), is to have a complete set of values $\{F_{K_j}(i)\}_{j\notin S}$ for some i which is constrained in one case but not the other, which requires having the BMFE key of some user i such that $i \in S$ (as no secret key contain the value $F_{K_i}(i)$), and where i is constrained in one case but not the other.

In our BMFE scheme, the decryption procedure is in $\mathbf{NC}^1$ if the underlying PCPRF evaluation $F_{K^*}(i)$ with a constrained key K^* is in $\mathbf{NC}^1$. If we go under the hood, and look at the PCPRF construction of [CVW18b], the constrained keys consist of $\log N$ tuples of square matrices $\{\mathbf{D}_{j,0}, \mathbf{D}_{j,1}\}_{j\in[\log N]}$ of dimension $\mathsf{poly}(\lambda)$, and the evaluation on some input $i = (b_1, \ldots, b_{\log N})$ computes a subset-product $\prod_{j=1}^{\log N} \mathbf{D}_{j,b_j}$ followed by rounding. While the product of a constant number of matrices and the rounding are in $\mathbf{NC}^1$, multiplying $\log N$ matrices is only known to be in $\mathbf{TC}^1$, which is not good enough for us.

We solve this problem by "pre-processing" the key which makes it longer but allows us to evaluate in $\mathbf{NC}^1$. In particular, we first group the $\log N$ matrix tuples into c groups of $(\log N)/c$ tuples each. Next, we pre-compute all possible $2^{(\log N)/c} = N^{1/c}$ subset-products within each group. This increases the key size from $2\log N$ original matrices to $c \cdot N^{1/c}$ pre-processed matrices, but now the evaluation only needs to multiply together c of the pre-processed matrices; as long as c is a constant (which can be arbitrarily large), this can be done in $\mathbf{NC}^1$. In other words, for any constant $\varepsilon > 0$ there is a PCPRF with key size $O(N^\varepsilon)$ (ignoring factors $\mathsf{poly}(\lambda)$ independent of ε) and evaluation in $\mathbf{NC}^1$. This translates into a BMFE with ciphetext size $O(N^\varepsilon)$ and decryption in $\mathbf{NC}^1$. Combining with succinct ABE for $\mathbf{NC}^1$, this in turn leads to an a AugBE scheme and eventually a Broadcast and Trace scheme with ciphertext size $O(N^\varepsilon)$. Note that if we instead had a succinct ABE for $\mathbf{TC}^1$ then we could avoid the pre-processing step and that would lead to the ciphertext size only $\mathsf{poly}\log N$.

2 Preliminaries

Notations. Let PPT denote probabilistic polynomial-time. We denote the set of all positive integers upto n as $[n] := \{1, 2, \ldots, n\}$. Throughout this paper, unless specified, all polynomials we consider are positive polynomials. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. The distribution $\mathcal{D}^n$ is used to represent a distribution over vectors of n components, where each component is drawn independently from the distribution $\mathcal{D}$.

2.1 Broadcast and Trace Systems

Here we recall the framework of broadcast and trace systems[8] and describe its security properties. In this work, we study broadcast and trace systems with secret key tracing. A broadcast and trace scheme BT, for message spaces $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, consists of four polytime algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pk}, \mathsf{tk}, \{\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N\})$. The setup algorithm takes as input a security parameter λ and number of users N. It outputs a public key pk, tracing key tk, and secret keys for N users $\{\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N\}$ respectively.
- $\mathsf{Enc}(\mathsf{pk}, S, m) \to \mathsf{ct}$. The encryption algorithm takes as input public key pk, a set $S \subseteq [N]$ of users, a message m and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}_i, S, \mathsf{ct}) \to m$ or $\perp$. The decryption algorithm takes as input a user secret key, a set of users $S \subseteq [N]$, a ciphertext ct, and outputs either a message m or special reject symbol $\perp$.
- $\mathsf{Trace}^D(\mathsf{tk}, S_D, m_0, m_1, 1^{1/\epsilon}) \to S^*$. The tracing algorithm takes as input a tracing key tk, a set of users S_D, two messages m_0, m_1 and parameter $\epsilon < 1$. The algorithm has a black-box access to the decoder D and outputs a set of indices $S^* \subseteq [N]$.
  Intuitively, the goal of the tracing algorithm is that when the decoder D can distinguish between encryptions of messages m_0 and m_1 encrypted to the set S_D with probability more than ϵ, the tracing algorithm should output a set S^* which is a subset of traitors (i.e., keys used to build decoder D). Here we consider the notion of secret key tracing, that is the algorithm takes as input a *private* tracing key to carry out the tracing procedure.

Correctness. A broadcast and trace system is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, any number of users $N \in \mathbb{N}$, every subset of users $S \subseteq [N]$, every message $m \in \mathcal{M}_\lambda$, every user $i \in S$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_i, S, \mathsf{ct}) = m : \begin{array}{c}(\mathsf{pk}, \mathsf{tk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N);\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, S, m)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

where the probability is taken over the random coins used during setup and encryption.

Security. Intuitively, the system is said to be secure if it is IND-CPA secure as well as if no poly-time adversary can produce a decoder that can fool the tracing algorithm. We formally define both of these properties below.

Definition 2.1 (Selective IND-CPA security). *We say that a broadcast and trace scheme is selective* IND-CPA *secure if for every stateful PPT adversary $\mathcal{A}$,*

[8] Prior works [NP00, NNL01, BW06] referred to such systems as Trace and Revoke.

there exists a negligible function $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds*

$$\Pr\left[\mathcal{A}(\mathsf{ct}) = b \ : \ \begin{array}{c}(1^N, S^*) \leftarrow \mathcal{A}(1^\lambda);\\ (\mathsf{pk}, \mathsf{tk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N);\\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]\setminus S^*});\\ b \leftarrow \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, S^*, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Next, we describe the secure tracing definition and experiment. Intuitively, it states that if an adversary $\mathcal{A}$ outputs a decoding box D such that D can distinguish between encryptions of messages m_0 and m_1 encrypted to the set $S_D \subseteq [N]$ with some non-negligible probability ϵ, then the tracing algorithm Trace, given oracle access to D, outputs (with all but negligible probability) a non-empty set of user indices such that all of them were corrupted by $\mathcal{A}$. Formally, it is described below.

Definition 2.2 (Selective Secure Tracing). *Let* $\mathsf{BT} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ *be a broadcast and trace scheme. For any non-negligible function* $\epsilon(\cdot)$ *and stateful PPT adversary* $\mathcal{A}$*, consider the experiment* $\mathsf{Expt\text{-}BT}_{\mathcal{A},\epsilon}(\lambda)$ *defined as follows.*

Experiment $\mathsf{Expt\text{-}BT}_{\mathcal{A},\epsilon}(\lambda)$

- $(1^N, S_D) \leftarrow \mathcal{A}(1^\lambda)$.
- $(\mathsf{pk}, \mathsf{tk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N)) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$.
- $(D, m_0, m_1) \leftarrow \mathcal{A}^{O(\cdot)}(\mathsf{pk})$.
- $S^* \leftarrow \mathsf{Trace}^D(\mathsf{tk}, S_D, m_0, m_1, 1^{1/\epsilon(\lambda)})$.

Here, $O(\cdot)$ is an oracle that has keys $\{\mathsf{sk}_i\}_{i\in[N]}$ hardwired, takes as input an index $i \in [N]$ and outputs i^{th} key sk_i. Let S be the set of indices queried by $\mathcal{A}$.

Fig. 1. Experiment $\mathsf{Expt\text{-}BT}$

Based on the above experiment, we now define the following (probabilistic) events and the corresponding probabilities (which are a functions of λ*, parameterized by* $\mathcal{A}, \epsilon$*):*

- $\mathsf{Good\text{-}Decoder} : \Pr[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, S_D, m_b)] \geq 1/2 + \epsilon(\lambda)$
  $\Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{Good\text{-}Decoder}]$
- $\mathsf{Cor\text{-}Tr} : |S^*| > 0, S^* \subseteq S \cap S_D$
  $\Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{Cor\text{-}Tr}]$
- $\mathsf{Fal\text{-}Tr} : S^* \not\subseteq S \cap S_D$
  $\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{Fal\text{-}Tr}]$

A broadcast and trace scheme BT *is said to satisfy selective secure tracing property if for every PPT adversary* $\mathcal{A}$*, polynomial* $q(\cdot)$ *and non-negligible function* $\epsilon(\cdot)$*, there exists negligible functions* $\mathsf{negl}_1(\cdot)$*,* $\mathsf{negl}_2(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$ *satisfying* $\epsilon(\lambda) > 1/q(\lambda)$*, the following holds*

$$\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}_1(\lambda), \quad \Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \geq \Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}_2(\lambda).$$

2.2 Augmented Broadcast Encryption

In this section, we define Augmented Broadcast Encryption (AugBE) and its security properties. The notion of AugBE was introduced by Boneh and Waters [BW06] as a building block towards realizing broadcast and trace systems. The original definition was described such that it could be used to build broadcast and trace scheme with public traceability. Here we relax the original definition since we only target secret key traceability. Specifically, the index encryption algorithm will now be a secret key algorithm, instead of being a public key algorithm. Below we describe the syntax.

- $\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\})$. The setup algorithm takes as input security parameter λ and number of users N. It outputs a public key pk, a master secret key msk and user secret keys $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}$, where sk_i is the secret key for user i.
- $\mathsf{Enc}(\mathsf{pk}, S, m) \to \mathsf{ct}$. The encryption algorithm takes as input public key pk, a set of users $S \subseteq [N]$, and a message m. It outputs a ciphertext ct.
- $\mathsf{Enc\text{-}index}(\mathsf{msk}, S, m, \mathsf{ind}) \to \mathsf{ct}$. The index encryption algorithm takes as input master secret key msk, a set of users $S \subseteq [N]$, a message m, and an index $\mathsf{ind} \in [N+1]$. It outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}_i, S, \mathsf{ct}) \to m$ or $\perp$. The decryption algorithm takes as input a secret key for i^{th} user sk_i, a set of users $S \subseteq [N]$, a ciphertext ct, and outputs a message m or $\perp$.

Correctness. An AugBE system is said to be correct if there exists a negligible function $\mathsf{negl}_1(\cdot), \mathsf{negl}_2(\cdot)$ such that for every $\lambda \in \mathbb{N}$, any number of users $N \in \mathbb{N}$, every subset of users $S \subseteq [N]$, any index $\mathsf{ind} \in [N+1]$, every message $m \in \mathcal{M}_\lambda$, every user $i \in S$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_i, S, \mathsf{ct}) = m : \begin{array}{c}(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N);\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, S, m)\end{array}\right] \geq 1 - \mathsf{negl}_1(\lambda),$$

$$i \geq \mathsf{ind} \Rightarrow \Pr\left[\mathsf{Dec}(\mathsf{sk}_i, S, \mathsf{ct}) = m : \begin{array}{c}(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N);\\ \mathsf{ct} \leftarrow \mathsf{Enc\text{-}index}(\mathsf{msk}, S, m, \mathsf{ind})\end{array}\right] \geq 1 - \mathsf{negl}_2(\lambda).$$

where the probabilities are taken over the random coins used during setup and encryption.

Security. Below we describe the security properties required from an AugBE scheme. The definitions are modelled after the bounded-ciphertext-query PLBE definitions [GKW18].

Definition 2.3 (q-query Selective Normal Hiding Security). *Let $q(\cdot)$ be any fixed polynomial. An AugBE scheme is said to satisfy q-query selective normal hiding security if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible*

function $\mathsf{negl}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, the following holds:*

$$\Pr\left[\mathcal{A}^{\mathsf{Enc\text{-}index}(\mathsf{msk},\cdot,\cdot,1)}(\mathsf{ct}_b) = b \ : \begin{array}{c}(1^N, S^*) \leftarrow \mathcal{A}(1^\lambda);\\ \left(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}\right) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)\\ m \leftarrow \mathcal{A}^{\mathsf{Enc\text{-}index}(\mathsf{msk},\cdot,\cdot,1)}\left(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]}\right)\\ b \leftarrow \{0,1\};\ \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, S^*, m)\\ \mathsf{ct}_1 \leftarrow \mathsf{Enc\text{-}index}(\mathsf{msk}, S^*, m, 1)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ *can make at most* $q(\lambda)$ *queries to* $\mathsf{Enc\text{-}index}(\mathsf{msk},\cdot,\cdot,1)$ *oracle. Note that here* $\mathcal{A}$ *is only allowed to query for ciphertexts corresponding to index* 1*.*

Definition 2.4 (q-query Selective Index Hiding Security). *Let* $q(\cdot)$ *be any fixed polynomial. An AugBE scheme is said to satisfy* q*-query selective index hiding security if for every (admissible) stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, the following holds:*

$$\Pr\left[\mathcal{A}^{O(\cdot),\mathsf{Enc\text{-}index}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{ct}) = b \ : \begin{array}{c}(1^N, \mathsf{ind} \in [N], S^*) \leftarrow \mathcal{A}(1^\lambda)\\ \left(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}\right) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)\\ m \leftarrow \mathcal{A}^{O(\cdot),\mathsf{Enc\text{-}index}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{pk})\\ b \leftarrow \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc\text{-}index}(\mathsf{msk}, S^*, m, \mathsf{ind}+b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ *can make at most* $q(\lambda)$ *queries to* $\mathsf{Enc\text{-}index}(\mathsf{msk},\cdot,\cdot,\cdot)$ *oracle. Here* $O(\cdot)$ *is an oracle that has keys* $\{\mathsf{sk}_i\}_{i\in[N]}$ *hardwired, takes as input an index* $i \in [N]$ *and outputs* sk_i*. Let the set of keys queried by the adversary be* S*. The adversary is* admissible *if and only if the challenge index* ind *it chooses satisfies* $\mathsf{ind} \notin (S^* \cap S)$*.*

Definition 2.5 (q-bounded Selective Message Hiding Security). *Let* $q(\cdot)$ *be any fixed polynomial. An AugBE scheme is said to satisfy* q*-query selective message hiding security if for every stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, the following holds:*

$$\Pr\left[\mathcal{A}^{\mathsf{Enc\text{-}index}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{ct}) = b \ : \begin{array}{c}(1^N, S^*) \leftarrow \mathcal{A}(1^\lambda); \left(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}\right) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)\\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{Enc\text{-}index}(\mathsf{msk},\cdot,\cdot,\cdot)}\left(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]}\right)\\ b \leftarrow \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc\text{-}index}(\mathsf{msk}, S^*, m_b, N+1)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ *can make at most* $q(\lambda)$ *queries to* $\mathsf{Enc\text{-}index}(\mathsf{msk},\cdot,\cdot,\cdot)$ *oracle.*

We refer for the full version of the paper for a construction of a broadcast and trace system from an AugBE scheme. The formal theorem is provided later.

2.3 Key-Policy Attribute Based Encryption with Short Ciphertexts

In this work we require a key-policy attribute based encryption (KP-ABE) scheme with short ciphertexts for obtaining our final result. Here we recall the definition of KP-ABE with short ciphertexts, and state the prior results with explicit succinctness guarantees.

A KP-ABE scheme ABE, for set of attribute spaces $\mathcal{X} = \{\mathcal{X}_\kappa\}_\kappa$, predicate classes $\mathcal{C} = \{\mathcal{C}_\kappa\}_\kappa$ and message spaces $\mathcal{M} = \{\mathcal{M}_\kappa\}_\kappa$, consists of four polytime algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^\kappa) \to (\mathsf{pp}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and a functionality index κ, and outputs the public parameters pp and master secret key msk.
- $\mathsf{Enc}(\mathsf{pp}, x, m) \to \mathsf{ct}$. The encryption algorithm takes as input public parameters pp, an attribute $x \in \mathcal{X}_\kappa$ and a message $m \in \mathcal{M}_\kappa$. It outputs a ciphertext ct.
- $\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm takes as input master secret key msk and a predicate $C \in \mathcal{C}_\kappa$. It outputs a secret key sk_C.
- $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}, x) \to m$ or $\perp$. The decryption algorithm takes as input a secret key sk_C, a ciphertext ct and an attribute x. It outputs either a message $m \in \mathcal{M}_\kappa$ or a special symbol $\perp$.

We point out that in our syntax the decryption algorithm takes the attribute x as explicit input. This is done so to simplify stating the succinctness requirement. Below we describe the correctness and security requirements, and later state the results achieving the requisite notion.

Correctness. A key-policy attribute based encryption scheme is said to be correct if there exists negligible functions $\mathsf{negl}(\cdot)$ such that for all $\lambda, \kappa \in \mathbb{N}$, for all $x \in \mathcal{X}_\kappa$, $C \in \mathcal{C}_\kappa$, $m \in \mathcal{M}_\kappa$, such that $C(x) = 1$ the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}, x) = m : \begin{array}{c}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa);\\ \mathsf{sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C);\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

where $\mathsf{negl}(\cdot)$ is a negligible function, and the probabilities are taken over the random coins used during setup, key generation, and encryption procedures.

Security. The standard notion of security for a KP-ABE scheme is that of IND-CPA security. It is formally defined as follows.

Definition 2.6. *A key-policy attribute based encryption scheme* $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *is said to be selectively secure if for every stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that for every* $\lambda \in \mathbb{N}$ *the following holds:*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c}(1^\kappa, x) \leftarrow \mathcal{A}(1^\lambda);\\ (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)\\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{pp})\\ b \leftarrow \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m_b)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where every predicate query C, made by adversary $\mathcal{A}$ to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ *oracle, must satisfy the condition that* $C(x) = 0$.

Below we state the result proved in [AHL+12] about a KP-ABE scheme with short ciphertexts from assumptions over bilinear maps. Concretely, they relied on the n-DBDHE assumption studied in [BGW05, BBG05]. Below we state the formal theorem.

Theorem 2.7 ([AHL+12, Theorem 4, Paraphrased]). *Assuming κ-DBDHE assumption holds, there exists a selectively-secure (Definition 2.6) KP-ABE scheme for non-monotonic access structures with length κ attributes (/number of parties). Additionally, the size of public parameters, secret keys, ciphertexts grow with λ and κ as follows — $|\mathsf{pp}| = O(\kappa \cdot \lambda)$, $|\mathsf{sk}_C| = O(\kappa \cdot \lambda \cdot |C|)$, and $|\mathsf{ct}| = O(\lambda)$.*

We point out that the size of the ciphertext does not depend on the length of the attributes, that is the KP-ABE scheme has short ciphertexts.

2.4 Key-Homomorphic Private Constrained PRFs

In this section, we recall the notion of almost-key-homomorphic private constrained PRFs (PCPRFs) from [CVW+18a]. As in [CVW+18a], we also work with PCPRFs that satisfy simulation-based security given one constrained key and many input queries. The existence of a simulator will be useful for the purpose of this paper. Below we describe the syntax and definition of PCPRFs.

A constrained PRF consists of five PPT algorithms (PPGen, SKGen, $\mathsf{Constrain}$, Eval, $\mathsf{Constrain.Eval}$) along with a domain family $\{D_\lambda\}_{\lambda\in\mathbb{N}}$, a range family $\{R_\lambda\}_{\lambda\in\mathbb{N}}$, and a constraint family $\mathcal{C} = \{\mathcal{C}_\lambda = \{C : D_\lambda \to \{0,1\}\}\}_{\lambda\in\mathbb{N}}$.

- $\mathsf{PPGen}(1^\lambda) \to \mathsf{PP}$. The public parameter generation algorithm takes the security parameter λ and generates the public parameters PP.
- $\mathsf{SKGen}(1^\lambda, \mathsf{PP}) \to \mathsf{SK}$. The secret key generation algorithm takes the security parameter λ, and the public parameters PP, and generates a secret key SK.
- $\mathsf{Eval}(\mathsf{SK}, x) \to y$. The evaluation algorithm takes SK, an input $x \in D_\lambda$, and deterministically outputs $y \in R_\lambda$. We will also use the alternative notation $y = F_{\mathsf{SK}}(x)$.
- $\mathsf{Constrain}(1^\lambda, \mathsf{PP}, \mathsf{SK}, C) \to \mathsf{CK}_C$. The constraining algorithm takes SK, a constraint $C \in \mathcal{C}_\lambda$, outputs the constrained key CK_C.
- $\mathsf{Constrain.Eval}(\mathsf{CK}_C, x) \to y$. The constrained evaluation algorithm takes a constrained key CK_C, an input x, outputs $y = F_{\mathsf{CK}_C}(x)$.

Definition 2.8 (Key-homomorphic private constrained PRF). *A constrained PRF (*PPGen, SKGen, $\mathsf{Constrain}$, Eval, $\mathsf{Constrain.Eval}$*) is a family of* almost-key-homomorphic private constrained PRF *for $\mathcal{C}$ if it satisfies the following properties:*

Functionality preservation for $C(x) = 0$. *For any constraint $C \in \mathcal{C}_\lambda$, any input $x \in D_\lambda$ s.t. $C(x) = 0$,*

$$\Pr[\mathsf{Eval}(\mathsf{SK}, x) = \mathsf{Constrain.Eval}(\mathsf{CK}_C, x)] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over the randomness used in algorithms PPGen*,* SKGen *and* Constrain*.*

Pseudorandomness and constraint-hiding. *There exists a polynomial time algorithm* Sim *such that for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:*

$$\Pr\left[\mathcal{A}^{\mathsf{Eval}(\mathsf{SK},\cdot)}(\mathsf{PP}, \mathsf{CK}_C) = 1 \ : \begin{array}{c} C \leftarrow \mathcal{A}(1^\lambda); \mathsf{PP} \leftarrow \mathsf{PPGen}(1^\lambda) \\ \mathsf{SK} \leftarrow \mathsf{SKGen}(1^\lambda, \mathsf{PP}) \\ \mathsf{CK}_C \leftarrow \mathsf{Constrain}(1^\lambda, \mathsf{PP}, \mathsf{SK}, C) \end{array}\right]$$
$$-\Pr\left[\mathcal{A}^{O(\cdot)}(\mathsf{PP}, \mathsf{CK}_C) = 1 \ : \begin{array}{c} C \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{PP}, \mathsf{CK}_C) \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C|}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

where the oracle $O(\cdot)$ is defined as follows. On each query x made by the adversary, if $C(x) = 0$ then it responds with $y = \mathsf{Constrain.Eval}(\mathsf{CK}_C, x)$, otherwise it responds with $y \leftarrow R_\lambda$.

Distribution requirement on the secret keys. *The space of keys K_λ is a group for all $\lambda \in \mathbb{N}$. Let $+$ denote the group operation over K_λ. We additionally require that for $\mathsf{PP} \leftarrow \mathsf{PPGen}(1^\lambda)$, for $\mathsf{SK}_1, \mathsf{SK}_2, \mathsf{SK}'$ sampled from $\mathsf{SKGen}(1^\lambda, \mathsf{PP})$ with uniform and independent randomness, $\mathsf{SK}_1 + \mathsf{SK}_2$, $\mathsf{SK}_1 + (-\mathsf{SK}_2)$, and SK' are identically distributed.*

Almost-key-homomorphism. *Let $B \in \mathbb{N}$, and suppose R_λ is endowed with a norm $\|\cdot\|$ and a group operation $+$ (by abuse of notation; whether we are considering addition over R_λ or over K_λ will be clear from the context) for all $\lambda \in N$. A constrained PRF (*PPGen*,* SKGen*,* Constrain*,* Eval*,* Constrain.Eval*) with domain D_λ and range R_λ is called B-almost-key-homomorphic if for $\mathsf{PP} \leftarrow \mathsf{PPGen}(1^\lambda)$, $\mathsf{SK}_1, \mathsf{SK}_2 \leftarrow \mathsf{SKGen}(1^\lambda, \mathsf{PP})$, and any input $x \in D_\lambda$:*

$$\|\mathsf{Eval}(\mathsf{SK}_1, x) + \mathsf{Eval}(\mathsf{SK}_2, x) - \mathsf{Eval}(\mathsf{SK}_1 + \mathsf{SK}_2, x)\| \leq B.$$

To instantiate the definition above, we will use PCPRFs from LWE [CC17, CVW18b], which happen to satisfy 1-almost-key homomorphism. We defer a more detailed exposition of the parameters and the efficiency of those PCPRFs to Section 6.1.

3 Broadcast Mixed FE for Comparison

The notion of mixed functional encryption was introduced in [GKW18] towards building efficient collusion-resistant Traitor Tracing systems. In this work, we adapt the notion of Mixed FE to additionally provide broadcast capability. We call this new primitive to Broadcast Mixed FE. This new notion is a central component of our approach to building Broadcast and Trace schemes. Let us first

recall the notion of Mixed FE scheme for comparisons. In such a scheme, both the secrets keys as well as ciphertexts are associated with a message string (say all natural numbers for instance) with the comparison predicate being implemented. In a Mixed FE system, there are two modes of encryption — secret-key and public-key. In the public-key (or normal) encryption mode, the algorithm takes as input only the public parameters and outputs a encryption of 'one' (i.e., inherently it encrypts a "canonical" *always-accepting* function '≥ 1'). Whereas in the secret-key mode, it takes as input the master secret key and a string x, and encrypts x. Now the functional secret keys are associated with a unique string as well. The decryption algorithm in a Mixed FE system works similar to that in standard FE, that is decrypting an encryption of message x using secret key for string i outputs 1 iff '$i \geq x$' (i.e., decryption evaluates the comparison function).

Here we extend this to provide a broadcast functionality as well. This means that now in both the public-key and secret-key modes, the encryption algorithms also take as input a set $S \subseteq [N]$. And, now the decryption functionality is altered as follows — decrypting an encryption of message x for set S using secret key for string i outputs 1 iff '$i \notin S \vee i \geq x$'. In other words, the decryption algorithm evaluates the comparison function *only if* $i \in S$, so that users outside of the broadcast set S cannot infer any information about x from their secret key. Next, we formally describe it.

A broadcast mixed functional encryption scheme BMFE consists of four polytime algorithms (Setup, Enc, $\mathsf{SK\text{-}Enc}$, Dec) with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pp}, \mathsf{msk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\})$. The setup algorithm takes as input the security parameter λ and number of users N, and outputs the public parameters pp, the master secret key msk and N user keys $\{\mathsf{sk}_i\}_{i\in[N]}$.
- $\mathsf{Enc}(\mathsf{pp}, S) \to \mathsf{ct}$. The normal encryption algorithm takes as input public parameters pp and a set $S \subseteq [N]$, and outputs a ciphertext ct.
- $\mathsf{SK\text{-}Enc}(\mathsf{msk}, S, j) \to \mathsf{ct}$. The secret key encryption algorithm takes as input master secret key msk, set $S \subseteq [N]$, and an index $j \in [N+1]$. It outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}_i, S, \mathsf{ct}) \to \{0,1\}$. The decryption algorithm takes as input a secret key sk_i, set $S \subseteq [N]$ and a ciphertext ct, and it outputs a single bit.

Correctness. A broadcast mixed functional encryption scheme is said to be correct if there exists negligible functions $\mathsf{negl}_1(\cdot), \mathsf{negl}_2(\cdot), \mathsf{negl}_3(\cdot)$ such that for all $\lambda, N \in \mathbb{N}$, for every set $S \subseteq [N]$, and for all user indices $i \in [N]$ and $j \in [N+1]$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_i, S, \mathsf{ct}) = 1 : \begin{array}{c}(\mathsf{pp}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N);\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, S)\end{array}\right] \geq 1 - \mathsf{negl}_1(\lambda),$$

$$(i \in S \wedge i < j) \Rightarrow \Pr\left[\mathsf{Dec}(\mathsf{sk}_i, S, \mathsf{ct}) = 0 : \begin{array}{c}(\mathsf{pp}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N);\\ \mathsf{ct} \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, S, j)\end{array}\right] \geq 1 - \mathsf{negl}_2(\lambda).$$

where the probabilities are taken over the random coins used during setup and encryption.

Security. The security notions are derived from the mixed FE security notions of function indistinguishability and accept indistinguishability as follows. Informally, the idea is that no PPT adversary should be able to distinguish between a normal ciphertext and a secret-key ciphertext encrypting index 1. Additionally, it should be hard to distinguish between two secret-key ciphertexts unless the adversary can trivially distinguish between using the keys given to it. As in prior works, we are only interested in broadcast mixed FE schemes that guarantee security against adversaries which make a bounded number of secret key encryption queries. Below we formally define it.

Definition 3.1 (q-query Selective Index Indistinguishability). *Let $q(\cdot)$ be any fixed polynomial. A broadcast mixed functional encryption scheme $\mathsf{BMFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{SK\text{-}Enc}, \mathsf{Dec})$ is said to satisfy q-query selective index indistinguishability security if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:*

$$\Pr\left[\mathcal{A}^{\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)}(\mathsf{pp},\mathsf{ct},\mathsf{Keys}) = b : \begin{array}{c}(1^N, \mathsf{ind} \in [N], S^*) \leftarrow \mathcal{A}(1^\lambda) \\ \left(\mathsf{pp}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}\right) \leftarrow \mathsf{Setup}(1^\lambda, 1^N) \\ b \leftarrow \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, S^*, \mathsf{ind}+b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to $\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)$ oracle. And, Keys is the following set of secret keys — $\mathsf{Keys} = \{\mathsf{sk}_i\}_{i\in[N]\setminus\{\mathsf{ind}\}}$ if $\mathsf{ind} \in S^*$, otherwise $\mathsf{Keys} = \{\mathsf{sk}_i\}_{i\in[N]}$.

Definition 3.2 (q-query Selective Mode Indistinguishability). *Let $q(\cdot)$ be any fixed polynomial. A broadcast mixed functional encryption scheme $\mathsf{BMFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{SK\text{-}Enc}, \mathsf{Dec})$ is said to satisfy q-query selective mode indistinguishability security if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:*

$$\Pr\left[\mathcal{A}^{\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,1)}(\mathsf{pp},\mathsf{ct}_b,\{\mathsf{sk}_i\}_{i\in[N]}) = b : \begin{array}{c}(1^N, S^*) \leftarrow \mathcal{A}(1^\lambda); \\ \left(\mathsf{pp}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}\right) \leftarrow \mathsf{Setup}(1^\lambda, 1^N) \\ b \leftarrow \{0,1\};\ \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pp}, S^*) \\ \mathsf{ct}_1 \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, S^*, 1)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to $\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,1)$ oracle.

4 Building Augmented BE from Broadcast Mixed FE and Key-Policy ABE with Short Ciphertexts

In this section we provide our construction for augmented BE from broadcast mixed FE and KP-ABE with short ciphertexts.

Let $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Dec})$ be a key-policy attribute based encryption scheme for set of attribute spaces $\{\mathcal{X}_\kappa\}_\kappa$, predicate classes $\{\mathcal{C}_\kappa\}_\kappa$ and message spaces $\{\mathcal{M}_\kappa\}_\kappa$, and $\mathsf{BMFE} = (\mathsf{BMFE.Setup}, \mathsf{BMFE.Enc}, \mathsf{BMFE.SK\text{-}Enc}, \mathsf{BMFE.Dec})$ be a broadcast mixed functional encryption scheme for comparison with ciphertexts of length $\ell = \ell(\lambda, N)$. Also, let $\kappa = \kappa(\lambda, N)$ be the lexicographically smallest functionality index such that every string of length ℓ can be uniquely represented in attribute class $\mathcal{X}_\kappa$ (i.e., $\{0,1\}^\ell \subseteq \mathcal{X}_\kappa$). We will suppose that for all $i \in [N]$ and $\mathsf{bmfe.sk}$ generated by $\mathsf{BMFE.Setup}$, $\mathcal{C}_\kappa$ contains the circuit $C_{i,\mathsf{bmfe.sk}}$ defined as:

$$C_{i,\mathsf{bmfe.sk}}(\mathsf{bmfe.ct}, S) := (i \in S) \wedge (\mathsf{BMFE.Dec}(\mathsf{bmfe.sk}, S, \mathsf{bmfe.ct}) = 1),$$

which composes a BMFE decryption with testing membership in $S \subseteq [N]$.

Below we describe our construction.

$\mathsf{Setup}(1^\lambda, 1^N) \to \left(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[N]}\right)$. The setup algorithm runs $\mathsf{ABE.Setup}$ and $\mathsf{BMFE.Setup}$ to generate ABE and broadcast mixed FE public parameters and master secret key as $(\mathsf{abe.pp}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^\kappa)$ and $(\mathsf{bmfe.pp}, \mathsf{bmfe.msk}, \{\mathsf{bmfe.sk}_i\}_{i\in[N]}) \leftarrow \mathsf{BMFE.Setup}(1^\lambda, 1^N)$.
Now let $C_{i,\mathsf{bmfe.sk}_i} : \{0,1\}^\ell \times [N] \to \{0,1\}$ denote the following circuit:

$$C_{i,\mathsf{bmfe.sk}_i}(\mathsf{bmfe.ct}, S) := (i \in S) \wedge (\mathsf{BMFE.Dec}(\mathsf{bmfe.sk}_i, S, \mathsf{bmfe.ct}) = 1).$$

That is, it corresponds to BMFE decryption circuit with key $\mathsf{bmfe.sk}_i$ hardwired along with a set membership check for index i. Next, it computes N ABE secret keys $\mathsf{abe.sk}_i$ as

$$\forall\ i \in [N], \quad \mathsf{abe.sk}_i \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, C_{i,\mathsf{bmfe.sk}_i})$$

Finally, it sets $\mathsf{pk} = (\mathsf{abe.pp}, \mathsf{bmfe.pp})$, $\mathsf{msk} = (\mathsf{abe.msk}, \mathsf{bmfe.msk})$ and $\mathsf{sk}_i = \mathsf{abe.sk}_i$ for $i \in [N]$.

$\mathsf{Enc}(\mathsf{pk}, S, m) \to \mathsf{ct}$. Let $\mathsf{pp} = (\mathsf{abe.pp}, \mathsf{bmfe.pp})$. The encryption algorithm first computes $\mathsf{ct}_{\mathsf{attr}} \leftarrow \mathsf{BMFE.Enc}(\mathsf{bmfe.pp}, S)$. Next, it encrypts message m as $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, \mathsf{attr} = (\mathsf{ct}_{\mathsf{attr}}, S), m)$, and outputs ciphertext $(\mathsf{ct}, \mathsf{ct}_{\mathsf{attr}})$.

$\mathsf{Enc\text{-}index}(\mathsf{msk}, S, m, \mathsf{ind}) \to \mathsf{ct}$. Let $\mathsf{msk} = (\mathsf{abe.msk}, \mathsf{bmfe.msk})$. The index-encryption algorithm first computes $\mathsf{ct}_{\mathsf{attr}} \leftarrow \mathsf{BMFE.SK\text{-}Enc}(\mathsf{bmfe.msk}, S, \mathsf{ind})$. Next, it encrypts message m as $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, \mathsf{attr} = (\mathsf{ct}_{\mathsf{attr}}, S), m)$, and outputs ciphertext $(\mathsf{ct}, \mathsf{ct}_{\mathsf{attr}})$.

$\mathsf{Dec}(\mathsf{sk}, S, (\mathsf{ct}, \mathsf{ct}_{\mathsf{attr}})) \to m$ or $\perp$. The decryption algorithm runs $\mathsf{ABE.Dec}$ on ct using key sk as $y = \mathsf{ABE.Dec}(\mathsf{sk}, \mathsf{ct}, (\mathsf{ct}_{\mathsf{attr}}, S))$, and sets y as the output of decryption.

We now state the correctness and security of the above construction. Their proofs are included in the full version of the paper..

Theorem 4.1. *Suppose* $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Dec})$ *is a correct attribute based encryption for set of attribute spaces* $\{\mathcal{X}_\kappa\}_\kappa$*, predicate classes* $\{\mathcal{C}_\kappa\}_\kappa$ *and message spaces* $\{\mathcal{M}_\kappa\}_\kappa$*, and* $\mathsf{BMFE} = (\mathsf{BMFE.Setup}, \mathsf{BMFE.Enc}, \mathsf{BMFE.SK\text{-}Enc}, \mathsf{BMFE.Dec})$ *is a correct broadcast mixed functional encryption scheme for comparison, then the above construction satisfies correctness.*

Theorem 4.2. *If* $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Dec})$ *is a selectively-secure attribute based encryption for set of attribute spaces* $\{\mathcal{X}_\kappa\}_\kappa$*, predicate classes* $\{\mathcal{C}_\kappa\}_\kappa$ *and message spaces* $\{\mathcal{M}_\kappa\}_\kappa$ *satisfying Definition 2.6, and* $\mathsf{BMFE} = (\mathsf{BMFE.Setup}, \mathsf{BMFE.Enc}, \mathsf{BMFE.SK\text{-}Enc}, \mathsf{BMFE.Dec})$ *is a broadcast mixed functional encryption scheme satisfying* 1*-query selective mode indistinguishability (Definition 3.2) and* 1*-query selective index indistinguishability (Definition 3.1) properties, then the above construction is a secure augmented broadcast encryption scheme, for messages spaces* $\{\mathcal{M}_\kappa\}_\kappa$*, satisfying* 1*-query selective normal, index and message hiding security properties as per Definitions 2.3 to 2.5. Additionally, the size of ciphertexts in the AugBE system is* $\ell + \widetilde{\ell}$*, where* $\ell = \ell(\lambda, N)$ *and* $\widetilde{\ell} = \widetilde{\ell}(\lambda, \kappa)$ *are the sizes of broadcast mixed FE and ABE ciphertexts, respectively.*

5 Building Broadcast Mixed FE for Comparison from PCPRFs

In this section we present our construction of a broadcast mixed FE for comparison with 1-query security based on almost-key-homomorphic private constrained PRFs.

In the following, if we let $N \in \mathbb{N}$ (which is the number of users), we will consider $N+1$ tuples of PCPRF keys indexed by $\{0, \ldots, N\}$. This can be viewed as adding a dummy user "0" who is never authorized to decrypt, so that no sums are empty (and in particular our scheme makes sense even if the set $S \subseteq [N]$ is $[N]$). As a result, in this whole section, whenever we consider a sum, unless specified otherwise, the set of indices live in $\{0, \ldots, N\}$; for instance, for $S \subseteq [N]$, $j \notin S$ will stand for $j \in \{0, \ldots, N\} \setminus S$.

Let $\mathsf{PCPRF} = (\mathsf{PPGen}, \mathsf{SKGen}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{Constrain.Eval})$ along with a family of constraints $\mathcal{C}$ be a PCPRF (Definition 2.8) satisfying B-almost-key homomorphism. For all $j \in D_\lambda$, let $C_j : i \mapsto [i \geq j]$ be a circuit that outputs 1 if $i \geq j$ and 0 otherwise. We will suppose that for all $j \in D_\lambda$, $C_j \in \mathcal{C}_\lambda$, that is C_j are valid constraints for the PCPRF. Let $|C_\lambda| = \mathsf{poly}(\lambda)$ be a common size for such circuits.

We define our broadcast mixed FE scheme as follows:

$\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pp}, \mathsf{msk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\})$: The setup algorithm first samples $\mathsf{PP} \leftarrow \mathsf{PPGen}(1^\lambda, \mathcal{F}_\lambda)$. It then generates for all $0 \leq i \leq N$, $t \in [\lambda]$ and $b \in \{0,1\}$: $\mathsf{SK}_{i,t,b} \leftarrow \mathsf{SKGen}(1^\lambda, \mathsf{PP})$.
It then sets $\mathsf{pp} = \mathsf{PP}$, $\mathsf{msk} = \{\mathsf{SK}_{i,t,b}\}_{0 \leq i \leq N, t \in [\lambda], b \in \{0,1\}}$, and for all $i \in [N]$: $\mathsf{sk}_i = \{i, \mathsf{Eval}(\mathsf{SK}_{j,t,b}, i)\}_{j \neq i, t \in [\lambda], b \in \{0,1\}}$.

$\mathsf{Enc}(\mathsf{pp}, S) \to \mathsf{ct}$. The normal encryption algorithm first picks a random tag $\mathbf{z} \leftarrow \{0,1\}^\lambda$. It then runs the PCPRF simulator: $\mathsf{CK} \leftarrow \mathsf{Sim}(1^\lambda, 1^{|\mathcal{C}_\lambda|})$, and sets $\mathsf{ct} = (\mathbf{z}, \mathsf{CK})$.

$\mathsf{SK\text{-}Enc}(\mathsf{msk}, S, j) \to \mathsf{ct}$. The secret key encryption algorithm first samples $\mathbf{z} \leftarrow \{0,1\}^\lambda$. It computes:

$$\mathsf{SK}_{S,\mathbf{z}} = \sum_{i \notin S, t \in [\lambda]} \mathsf{SK}_{i,t,\mathbf{z}_t},$$

(where the sum denotes the group operation over PCPRF keys). Note that this sum is never empty (as $i \notin S$ stands here for $i \in \{0, \ldots, N\} \setminus S$, so that it always contains the secret keys $\mathsf{SK}_{0,t,b}$ for all $t \in [\lambda], b \in \{0,1\}$). The algorithm computes the constrained key

$$\mathsf{CK}_{S,\mathbf{z},j} \leftarrow \mathsf{Constrain}(1^\lambda, \mathsf{PP}, \mathsf{SK}_{S,\mathbf{z}}, C_j),$$

where C_j is defined above. It finally sets $\mathsf{ct} = (\mathbf{z}, \mathsf{CK}_{S,\mathbf{z},j})$.

$\mathsf{Dec}(\mathsf{sk}_i, S, \mathsf{ct}) \to \{0,1\}$. The decryption algorithm parses ct as $(\mathbf{z}, \mathsf{CK})$. If $i \notin S$ where i is the secret key index, the decryption algorithm outputs 1. Otherwise, it computes $\mathsf{Constrain.Eval}(\mathsf{CK}, i)$, and outputs:

$$\begin{cases} 0 & \text{if } \|\mathsf{Constrain.Eval}(\mathsf{CK}, i) - \sum_{j \notin S, t \in [\lambda]} \mathsf{Eval}(\mathsf{SK}_{j,t,\mathbf{z}_t}, i)\| \leq (N+1) \cdot \lambda \cdot B \\ 1 & \text{otherwise.} \end{cases}$$

We now state the correctness and security of the above construction. The proofs are included in the full version of the paper.

Theorem 5.1. *Suppose* $\mathsf{PCPRF} = (\mathsf{PPGen}, \mathsf{SKGen}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{Constrain.Eval})$ *along with a constraint family* $\mathcal{C}$ *and range* R_λ *is a PCPRF (Definition 2.8) satisfying* B*-almost-key homomorphism for a norm* $\|\cdot\|$*. Suppose furthermore that* $\Pr_{x \leftarrow R_\lambda}[\,\|x\| \leq (N+1)\lambda B\,] \leq \mathsf{negl}(\lambda)$*, that is, random elements in the range of the PCPRF have large norm. Then the above construction satisfies correctness.*

Theorem 5.2. *If* $\mathsf{PCPRF} = (\mathsf{PPGen}, \mathsf{SKGen}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{Constrain.Eval})$ *along with a constraint family* $\mathcal{C}$ *is a PCPRF (Definition 2.8) satisfying* B*-almost-key homomorphism, then the above construction is a secure BMFE for comparison satisfying 1-query selective index indistinguishability and 1-query selective mode indistinguishability, as per Definitions 3.1 and 3.2.*

6 Efficiency

In this section we analyze the efficiency of our different constructions, in order to evaluate the efficiency of our broadcast and trace scheme.

6.1 Efficient PCPRF for Comparison Constraints

We first focus on the PCPRF used in Section 5. Looking ahead, it will be crucial that the resulting BMFE has *short ciphertext* and *efficient decryption*. More precisely, we will require to have the BMFE to have decryption in $\mathbf{NC}^1$ while having as short ciphertexts as possible.

Looking at our construction in Section 5, we first need to analyze the complexity of evaluating a PCPRF constrained evaluation for comparison constraints (which is performed during BMFE decryption, and therefore required be in $\mathbf{NC}^1$), as well as the size of the constrained keys (which are the BMFE ciphertexts). We do so by analyzing and tailoring the PCPRFs from the literature ([CC17, CVW18b]) for our needs.

Almost-key-homomorphic PCPRFs from LWE. For our constructions, we will focus on constructions of PCPRFs from LWE supporting (polynomial length) branching program constraints [CC17,CVW18b], where the range is $R_\lambda = \mathbb{Z}_p^{m\times m}$ where p is the output modulus of the PRF, and $m = \mathsf{poly}(n)$ where n is the lattice dimension in the underlying learning with errors assumption. They additionally satisfy 1-almost-key-homomorphism with the infinity norm $\|\cdot\|_\infty$ [CVW+18a]. For more details on the parameters, we refer the reader t the relevant sections of [CC17, CVW18b].

Again, we will be most interested in both the *size of the constrained keys* and the complexity of computing a *constrained evaluation*. In the constructions of [CC17, CVW18b], if we consider branching programs of constant width and length $h \in \mathbb{N}$, then constrained keys consist of a set of $2h$ matrices in $\mathbb{Z}_q^{m\times m}$ and a single matrix in $\mathbb{Z}_q^{n\times m}$, where $m = \mathsf{poly}(n)$ and n and q are respectively the lattice dimension and modulus of the underlying learning with errors assumption. In other words, the constrained keys are of the form:

$$\mathsf{CK} = (\mathbf{A}, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}),$$

where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{D}_{i,b} \in \mathbb{Z}_q^{m\times m}$ for all $i \in [h], b \in \{0,1\}$, and where $m = \mathsf{poly}(n)$, and q is exponential in h (for correctness). Constrained evaluation is performed by multiplying elements in the constrained key, namely the matrix $\mathbf{A}$, and a subset of h matrices determined by the input to the evaluation. For an input $x \in \{0,1\}^\ell$, we have:

$$\mathsf{Constrain.Eval}(\mathsf{CK}, x) = \left\lfloor \mathbf{A} \cdot \prod_{i\in[h]} \mathbf{D}_{i,x_{(i \bmod \ell)}} \right\rceil_p ,^9$$

where, for $q > p \geq 2$, $\lfloor\cdot\rceil_p : \mathbb{Z}_q \mapsto \mathbb{Z}_p$ rounds element in $\mathbb{Z}_q$ to $\mathbb{Z}_p$, that is, $\lfloor x\rceil_p = \lfloor x \cdot p/q\rceil$ where $\lfloor\cdot\rceil$ denotes the usual rounding to the nearest integer; and $\lfloor\cdot\rceil_p$

[9] Later, we will need the index-to-input map ι of the branching program to be independent of the program; we consider here $\iota : i \mapsto (i \bmod \ell)$ for simplicity. This is without loss of generality up to a blow-up in the branching program length by a factor ℓ.

extends over matrices by applying the rounding pointwise. In particular, for $m = \mathsf{poly}(n)$ and $q \leq 2^{\mathsf{poly}(n)}$, such a computation can be implemented by a circuit of depth $O(\log h \cdot \log n)$ (by computing the h matrix products using a binary tree). Actually, as both matrix multiplication and rounding (which is computable using integer multiplication, division and rounding) can be performed in $\mathbf{TC}^0$ in this regime (e.g. [RT92]), constrained evaluation can be performed in $\mathbf{TC}^1$.

Theorem 6.1 (PCPRFs from LWE [CC17, CVW18b]). *Assuming the hardness of LWE (with appropriate parameters), there exists PCPRFs satisfying 1-almost-key-homomorphism supporting branching program constraints. Additionally for any class of branching program constraints of width $O(1)$ and length $h \leq \mathsf{poly}(n)$, the constrained keys have size $O(h \cdot \mathsf{poly}(n) \cdot \log q)$, and constrained evaluation can be computed in $\mathbf{TC}^1$, where n and q are respectively the lattice dimension and modulus of the underlying LWE assumption.*

Pre-processing the constrained evaluation. As noted earlier, we will crucially need to be able to compute constrained evaluations in $\mathbf{NC}^1$. We note here that in the constructions of [CC17,CVW18b] of PCPRFs for branching program constraints (with index-to-input map independent of the program), we can improve the complexity of computing a constrained evaluation by pre-process the constrained keys. Recall that constrained keys contains matrices $\{\mathbf{D}_i^b\}_{i\in[h],b\in\{0,1\}}$, where $h \in \mathbb{N}$ is the length of the branching program. Let $0 < \varepsilon < 1$ be a fixed constant, such that $1/\varepsilon \in \mathbb{N}$, and that $\varepsilon h \in \mathbb{N}$ (this is without loss of generality up to padding the branching program with a constant number $\leq 1/\varepsilon$ of dummy levels). To pre-process the constrained keys, we pre-compute all the products of blocks of εh matrices.[10] In other words, for all $y \in \{0,1\}^{\varepsilon h}$ and all $j \in \{0, \ldots, 1/\varepsilon - 1\}$, the pre-processing phase computes:

$$\mathbf{M}_{j,y} = \prod_{i=1}^{\varepsilon h} \mathbf{D}_{j\varepsilon h+i\,,\, y_{i \bmod \ell}}.$$

For $x \in \{0,1\}^\ell$, $j \in \{0, \ldots, 1/\varepsilon - 1\}$, let $y^{(j)} = (x_{j\varepsilon h+1 \bmod \ell}, \ldots, x_{(j+1)\varepsilon h \bmod \ell})$ be the j-th block of εh consecutive coordinates of x, ranging from $j\varepsilon h+1 \mod \ell$ to $(j+1)\varepsilon h \bmod \ell$. Then, given those $2^{\varepsilon h} \cdot 1/\varepsilon$ matrices $\{\mathbf{M}_{j,y}\}_{0\leq j<1/\varepsilon,\, y\in\{0,1\}^{\varepsilon h}}$, and the original matrix $\mathbf{A}$, one can compute for all $x \in \{0,1\}^\ell$:

$$\mathsf{Constrain.Eval}(\mathsf{CK}, x) = \lfloor \mathbf{A} \cdot \prod_{j=0}^{1/\varepsilon-1} \mathbf{M}_{j,y^{(j)}} \rceil_p.$$

In other words, given the pre-processed constrained key, constrained evaluation can be performed by multiplying the appropriate $(1/\varepsilon)$ pre-computed block products together with $\mathbf{A}$ (and rounding). In particular, this only requires a *constant* number of matrix muliplications (as opposed to h originally). This is at the cost of using a pre-processed constrained key consisting of $2^{\varepsilon h} \times 1/\varepsilon$ matrices (which can seen as pre-processed constrained keys).

[10] We rely here on the fact that the index-to-input ι is independent of the branching program.

Efficient construction for comparison constraints. We note now that the BMFE of Section 5 does not need to support general constraints, but only *comparison* functions. Recall that for a parameter $N \in \mathbb{N}$ and for $\mathsf{ind} \in [N]$, the function P_{ind}, on input $i \in [N]$, outputs 1 if $i \geq \mathsf{ind}$ and 0 otherwise.

However, naively invoking Barrington theorem [Bar86] to obtain a generic branching program computing P_{ind}, only yields a branching program of length $\log^2(N)$, which makes the pre-processing described above output *super-polynomially* many matrices. Instead, we directly build a branching program for comparison constraints, with constant width and length $O(\log N)$, which will be good enough for our purposes.

Lemma 6.2. *Let $N \in \mathbb{N}$ be an integer. Then for all $\mathsf{ind} \in [N]$, there exists a (non-permutation) branching program of width 3 and length $\log N + 2$ computing P_{ind} (defined as $P_{\mathsf{ind}}(i) = 1$ if $i \geq \mathsf{ind}$ and 0 otherwise), with index-to-input map ι is independent of ind.*

We exhibit such a branching program in the full version of the paper. Note that this particular branching program is *not* a permutation branching program, which excludes the PCPRF of [CC17]. Fortunately [CVW18b] does support general (non-permutation) branching program constraints. Now, for $0 < \varepsilon < 1$ being a fixed constant, pre-processing the constrained keys results in N^{ε} matrices of size $\mathsf{poly}(n) \log q$ (where n and q are respectively the lattice dimension and the modulus of the underlying LWE assumption), while now multiplying $1/\varepsilon$ matrices can be performed using a circuit of depth $O(\log(1/\varepsilon) \log(n))$. The following Lemma follows by the fact that rounding can be computed in $\mathbf{TC}^0$ ([RT92]).

Lemma 6.3. *Let $N \in \mathbb{N}$ be an integer and $0 < \varepsilon < 1$ be a constant. Assuming the hardness of LWE (with appropriate parameters), there exists a PCPRF for comparison constraints (as defined above) satisfying 1-almost-key homomorphism. Furthermore, for $\mathcal{C}_\lambda = \{P_{\mathsf{ind}}\}_{\mathsf{ind} \in [N]}$ (defined above), that is if the constraints compare integers in $[N]$, then the constrained keys have size $N^{\varepsilon} \cdot \mathsf{poly}(n)$ (where n is the lattice dimension in the underlying LWE assumption) and constrained evaluation is in $\mathbf{NC}^1$.*

6.2 Wrapping-up

Efficiency and parameters of the BMFE. We are here most interested in the *size* of a BMFE ciphertext and its *decryption complexity*. First, adding polynomially many $\mathsf{poly}(n)$-bit numbers, and comparing $\mathsf{poly}(n)$-bit numbers can be done in $\mathbf{TC}^0$, and therefore in $\mathbf{NC}^1$. Therefore, combined with Lemma 6.3, we obtain that BMFE decryption from Section 5 can be evaluated in $\mathbf{NC}^1$, as summing PCPRF evaluations, taking their infinity norm and comparing them to the threshold are in $\mathbf{NC}^1$as well.

Alternatively, we can directly use the PCPRFs of [CVW18b] (without pre-processing the constrained keys). Combined our branching program for comparison (Lemma 6.2), this gives a BMFE with ciphertext size $\log N \cdot \mathsf{poly}(\lambda)$ with decryption in $\mathbf{TC}^1$.

Lemma 6.4. *Suppose $N = \mathsf{poly}(\lambda)$, and let ε be a constant such that $0 < \varepsilon < 1$. Assuming the hardness of LWE with (sufficiently large) quasi-polynomial modulus-to-noise ratio, there exists:*

- *a BMFE for comparison with ciphertext size $N^{\varepsilon} \cdot \mathsf{poly}(\lambda)$ and decryption in $\mathbf{NC}^1$;*
- *a BMFE for comparison with ciphertext size $\log(N) \cdot \mathsf{poly}(\lambda)$ and decryption in $\mathbf{TC}^1$.*

For the parameters of the LWE assumption, we can take those of [CVW18b, Remark 7.2] for branching programs of width $w = 3$ and length $h = \log N + 2$, with the additional requirement that $p \geq C \cdot N\lambda$ for some fixed constant $C > 1$ (e.g. $C = 1.1$), which we use to argue correctness of the BMFE. In particular, for $N = \mathsf{poly}(\lambda)$, this corresponds to assuming the hardness of LWE with a quasi-polynomial modulus to noise ratio. Looking ahead, this will be parameters of the LWE assumption of our final broadcast and trace scheme.

Efficiency of the broadcast and trace. The final broadcast and trace system directly inherits the ciphertext size from the augmented BE. Using the construction from Section 4, the resulting augmented BE scheme inherits its ciphertext size from its underlying ABE, assuming the ABE support the class of predicates $C_{i,\mathsf{bmfe.sk}_i}(\mathsf{bmfe.ct}, S) := (i \in S) \wedge (\mathsf{BMFE.Dec}(\mathsf{bmfe.sk}_i, S, \mathsf{bmfe.ct}) = 1)$ defined by the BMFE decryption procedure.

In conclusion, assuming the ABE has *succinct* ciphertexts of size independent of their attribute, then our broadcast and trace system has ciphertext size dominated by the size of the BMFE ciphertexts. Overall, Combining Lemma 6.4, and Theorem 2.7, we get the desired result:

Theorem 6.5. *Let $N = \mathsf{poly}(\lambda)$, and let ε be a constant such that $0 < \varepsilon < 1$. Assuming the hardness of LWE with (sufficiently large) quasi-polynomial modulus-to-noise ratio, and:*

- *assuming that the N-DBDHE assumption holds, there exists a broadcast and trace scheme with ciphertext size $N^{\varepsilon} \cdot \mathsf{poly}(\lambda)$.*
- *assuming the existence of an ABE for $\mathbf{TC}^1$ predicates with ciphertext size polylogarithmic in its attribute length, there exists a broadcast and trace scheme with ciphertext size $\mathsf{poly}(\log N, \lambda)$.*

References

AHL+12. Nuttapong Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie De Panafieu, and Carla Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical computer science*, 422:15–38, 2012.

ALDP11. Nuttapong Attrapadung, Benoît Libert, and Elie De Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *International Workshop on Public Key Cryptography*, pages 90–108. Springer, 2011.

Bar86. D A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, STOC '86, 1986.

BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT 2005*, volume 3494 of LNCS of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

BGG+14. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 533–556, 2014.

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

BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, pages 573–592, 2006.

BTVW17. Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained prfs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 264–302. Springer, 2017.

BW06. Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 211–220, 2006.

CC17. Ran Canetti and Yilei Chen. Constraint-hiding constrained prfs for nc1 from lwe. In *EUROCRYPT*, 2017.

CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, pages 257–270, 1994.

CFNP00. Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Trans. Information Theory*, 46(3):893–910, 2000.

CVW+18a. Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from lwe made simple and attribute-based. In *TCC*, 2018.

CVW18b. Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 577–607. Springer, 2018.

DF02. Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *ACM Workshop on Digital Rights Management*, pages 61–80. Springer, 2002.

FN94. Amos Fiat and Moni Naor. Broadcast encryption. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 480–491, 1994.

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 612–621, 2017.

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *STOC*, 2018.

GST04. Michael T Goodrich, Jonathan Z Sun, and Roberto Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Annual International Cryptology Conference*, pages 511–527. Springer, 2004.

GSW00. Juan A Garay, Jessica Staddon, and Avishai Wool. Long-lived broadcast encryption. In *Annual International Cryptology Conference*, pages 333–352. Springer, 2000.

GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, 2012.

GVW13. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

GVW19. Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. Collusion resistant trace and revoke from positional witness encryption. In *PKC*, 2019.

HLR10. Javier Herranz, Fabien Laguillaumie, and Carla Ràfols. Constant size ciphertexts in threshold attribute-based encryption. In *International Workshop on Public Key Cryptography*, pages 19–34. Springer, 2010.

HS02. Dani Halevy and Adi Shamir. The lsd broadcast encryption scheme. In *Annual International Cryptology Conference*, pages 47–60. Springer, 2002.

KMUW18. Lucas Kowalczyk, Tal Malkin, Jonathan Ullman, and Daniel Wichs. Hardness of non-interactive differential privacy from one-way functions, 2018.

NNL01. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology - CRYPTO 2001*, 2001.

NP00. Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography, 4th International Conference, FC 2000*, 2000.

PST06. Duong Hieu Phan, Reihaneh Safavi-Naini, and Dongvu Tonien. Generic construction of hybrid public key traitor tracing with full-public-traceability. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 264–275, 2006.

RT92. J. Reif and S. Tate. On threshold circuits and polynomial computation. *SIAM Journal on Computing*, 21(5):896–908, 1992.

SSW01. Jessica Staddon, Douglas R. Stinson, and Ruizhong Wei. Combinatorial properties of frameproof and traceability codes. *IEEE Trans. Information Theory*, 47(3):1042–1049, 2001.

Sti97. Doug R Stinson. On some methods for unconditionally secure key distribution and broadcast encryption. In *Selected Areas in Cryptography*, pages 3–31. Springer, 1997.

SVT98. Doug R Stinson and Tran Van Trung. Some new results on key distribution patterns and broadcast encryption. *Designs, Codes and Cryptography*, 14(3):261–279, 1998.

SW98. Douglas R. Stinson and Ruizhong Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM J. Discrete Math.*, 11(1):41–53, 1998.

SW05. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 600–611, 2017.

YAHK14. Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. In *International Workshop on Public Key Cryptography*, pages 275–292. Springer, 2014.