Broadcast and Trace from Lattices: Collusion-Resistant and Publicly Traceable

Jeffrey Champion
Bocconi University
jeffrey.champion@unibocconi.it

Abstract

A broadcast and trace scheme combines broadcast encryption with accountability. Specifically, broadcast encryption is augmented with an algorithm for tracing traitors within a specified set, even if the given pirate decoder is created by an arbitrary collusion of users. As in broadcast encryption, the size of the ciphertext should be sublinear in the total number of users N.

In this work, we generically construct collusion-resistant broadcast and trace that supports public tracing from attribute-based encryption with succinct ciphertexts and lockable obfuscation, both of which are known from the decomposed LWE assumption. Our scheme has ciphertext size $\sqrt{N} \cdot \mathsf{poly}(\lambda, \log N)$, while the master public key and secret keys have size $\mathsf{poly}(\lambda, \log N)$. Previously, collusion-resistant broadcast and trace with public tracing required pairing-based assumptions, positional witness encryption, or indistinguishability obfuscation. In particular, we provide the first construction that relies on a simple and instance-independent lattice assumption.

At the heart of our construction is a novel way to use nested lockable obfuscations, taking inspiration from the recent work of Nassar, Waters, and Wu that constructs positional witness encryption from indistinguishability obfuscation for null programs.

1 Introduction

A traitor tracing scheme [CFN94] allows a content provider to send encrypted content to N legitimate users, such that the source of any illegitimate decryption algorithm can be traced. Specifically, each legitimate user $i \in [N]$ is given a secret key sk_i that allows them to decrypt ciphertexts that are encrypted under some master public key mpk. However, a collusion of adversarial users $C \subseteq [N]$ may give their keys to illegitimate users or carefully craft a (possibly obfuscated) pirate decoder box that uses sk_i for $i \in C$. To prevent such misbehavior, there is a tracing algorithm that can recover an active traitor $i^* \in C$ given black-box access to any successful pirate decoder. Such a tracing procedure should not falsely accuse an honest user. We say a traitor tracing scheme is *collusion-resistant* if $C \subseteq [N]$ can be an arbitrary set with no bounds on its size. The primary efficiency property of traitor tracing schemes is that the size of the ciphertext is $o(N)$.

Broadcast and trace. The basic notion of traitor tracing does not have the capability to revoke a traitor's ability to decrypt. Thus, a desirable property is to allow the encrypter to choose an arbitrary set $S \subseteq [N]$ to encrypt to such that users outside the set cannot decrypt while still keeping the ciphertexts succinct, as is required in broadcast encryption [FN93]. This would allow the content provider to remove traitors from the encryption set S as they are found, by additionally strengthening the tracing requirement to always accuse

users $i^* \in C \cap S$. We refer to such a scheme as a broadcast and trace scheme [NP00, NNL01] (also called a trace and revoke scheme). In this work, our focus is on collusion-resistant broadcast and trace schemes.

Public tracing. Another particularly useful property of a traitor tracing scheme is *public traceability*, meaning that tracing is a public procedure. This removes a central point of failure, since there are no tracing secrets that can compromise security if leaked. Unauthorized users being able to trace also aids in finding a given pirate decoder and deters traitors from freely giving out their pirate decoder. Moreover, public tracing provides a natural verification mechanism for tracing by having the verifier (e.g. a judge) simply rerun the tracing algorithm. However, achieving public tracing and collusion resistance in tandem has been difficult.

Prior work. Currently, there are only a handful of constructions of collusion-resistant broadcast and trace with public tracing. There is a scheme with $\sqrt{N}$-size ciphertexts based on bilinear maps [BW06], schemes with optimal parameters from indistinguishability obfuscation (iO) [NWZ16], optimal positional witness encryption [GLW14, GVW19], and compact functional encryption [AKYY23], and a scheme with N^{ε}-size ciphertexts for $\varepsilon > 0$ from iO for null programs [NWW26]. Notably, all known schemes live in (or close to) obfustopia or can be broken efficiently by quantum computers.

Our results. In this work, we construct collusion-resistant broadcast and trace with public tracing generically from attribute-based encryption [SW05, GPSW06] with succinct ciphertexts and lockable obfuscation [GKW17, WZ17]. Both of these primitives are known from the decomposed learning with errors (LWE) assumption[1] [AMR25], a simple and instance-independent lattice assumption that can be reduced to the worst-case hardness of lattice problems [AAC+26]. Before this work, no collusion-resistant traitor tracing schemes with public tracing were known from lattice assumptions that did not also imply null-iO. We summarize our main instantiation with the following informal theorem:

Theorem 1.1 (Broadcast and Trace from Decomposed LWE, Informal)**.** *Let λ be a security parameter. Under the polynomial hardness of decomposed LWE, there exists a collusion-resistant broadcast and trace scheme for N users with public/secret key size $\mathrm{poly}(\lambda, \log N)$ and ciphertext size $\sqrt{N} \cdot \mathrm{poly}(\lambda, \log N)$.[2] The scheme supports public tracing.*

The above is a corollary of the following more general statement:

Theorem 1.2 (Generic Broadcast and Trace, Informal)**.** *Let λ be a security parameter and d be a policy depth bound. Assuming the existence of attribute-based encryption for depth-d circuits where the master public key size, secret key size, ciphertext size, and decryption depth are all $\mathrm{poly}(\lambda, d)$ (ignoring polylog terms), lockable obfuscation for circuits, and weakly succinct lockable obfuscation for Turing machines, there exists a collusion-resistant broadcast and trace scheme for N users with public/secret key size $\mathrm{poly}(\lambda, \log N)$ and ciphertext size $\sqrt{N} \cdot \mathrm{poly}(\lambda, \log N)$. The scheme supports public tracing.*

1.1 Technical Overview

In this section, we give an overview of our approach for constructing collusion-resistant broadcast and trace with public tracing. As with prior work on broadcast and trace, we focus on building a simpler primitive called augmented broadcast encryption [BW06], which can be generically transformed into a collusion-resistant broadcast and trace scheme that supports public tracing [BW06, GVW19].

[1] We note that lockable obfuscation follows from plain LWE, which is implied by decomposed LWE.

[2] As noted in Remark 3.13, we believe this can likely be improved to match the N^{ϵ} ciphertext size in [NWW26].

Augmented broadcast encryption. Similar to traditional broadcast encryption, an augmented broadcast encryption scheme consists of a tuple of efficient algorithms (Setup, Encrypt, Decrypt). The setup algorithm outputs a master public key mpk and secret keys sk_i for $i \in [N]$. The encryption algorithm takes as input mpk, a set $S \subseteq [N]$, a message m and a cutoff $t \in [0, N] = \{0, \ldots, N\}$ and generates a ciphertext ct. Decryption takes in $\mathsf{mpk}, \mathsf{sk}_i, S, \mathsf{ct}$ and only recovers m for users $i \in S$ such that $i > t$. The public/secret key and ciphertext sizes are preserved in the resulting broadcast and trace scheme, so the succinctness still requires ciphertexts of size $o(N)$. The security requirements are message hiding and index hiding. Message hiding states that even given every secret key the message is hidden when the cutoff is N. Index hiding states that only a secret key sk_t such that $t \in S$ can distinguish between encryptions to S with cutoff t or $t-1$, so if such a key is not given the encryptions should be indistinguishable. Intuitively, index hiding allows a tracer to test whether a pirate decoder includes a given user's key by encrypting to the cutoffs $t \in [N]$ in order and seeing where there is a gap in decryption success probability. For ease of exposition, we will focus on a simplified version of augmented broadcast encryption that assumes the set is always $[N]$, and then discuss how to extend to arbitrary sets at the end of this overview.

Starting point: a trivial construction. We start by describing a trivial construction of augmented broadcast encryption for $S = [N]$ from plain public-key encryption (PKE) with ciphertexts and a master public key of size $O(N)$. The construction is as follows:

- **Setup:** For $i \in [N]$, the setup algorithm generates a key pair for a PKE scheme $(\mathsf{pk}_i, \mathsf{sk}_i)$. The master public key mpk is $\{\mathsf{pk}_i\}_{i \in [N]}$ and user i is given sk_i as their secret key.
- **Encryption:** To encrypt a message m to a fixed set $[N]$ and cutoff $t \in [0, N]$, the encrypter computes $\mathsf{ct}_i \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}_i, 0)$ for $i \leq t$ and $\mathsf{ct}_i \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}_i, m)$ for $i > t$. The ciphertext is $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_N)$.
- **Decryption:** User $i \in [N]$ simply decrypts ct_i with sk_i, which recovers the message when $i > t$.

When $t = N$ the message no longer appears, so message hiding is immediate. For index hiding, the only difference between encrypting to cutoff $t-1$ compared to cutoff t is that ct_t goes from encrypting m to encrypting 0. Such a change is indistinguishable by PKE security when sk_t is not given to the adversary.

Reusing the trivial construction. To get $\sqrt{N}$-size ciphertexts, we follow the approach of [NWW26] restricted specifically to the case of augmented broadcast encryption. At a high level, [NWW26] shows how to reuse a trivial construction of positional witness encryption for $n = \sqrt{N}$ statements by using a separate null-iO program to manage each block of n statements. More precisely, the set $[N]$ is treated as an $n \times n$ grid where each $i \in [N]$ is written as a row-column pair $(r, c) \in [n]^2$ such that $i = (r-1)n + c$ and each null-iO program manages a row r. Our aim is to reuse the trivial augmented broadcast encryption scheme from PKE described above by utilizing a lockable obfuscation (LO) [GKW17, WZ17] program P_r to manage each row r instead of a null-iO program. For a cutoff $t = (t_1 - 1)n + t_2$, we use the trivial PKE construction with cutoff t_2 on the columns of the "active" row t_1 that t lies in. Thus, we want the programs managing each row to roughly act as follows:

- For $r < t_1$, the row program P_r outputs $\perp$.
- Program P_{t_1} "defers" to the trivial construction for users in row t_1 by running PKE decryption.
- For $r > t_1$, program P_r outputs m for users in row r.

The ciphertext would consist of the programs P_r for $r \in [n]$ and ciphertexts $\mathsf{ct}_{\mathsf{PKE}}^{(c)}$ for $c \in [n]$, each encrypted under pk_j for $j = (t_1 - 1)n + c$ and encrypting m if $c > t_2$. If each P_r has size independent of N, then the total ciphertext size is $n = \sqrt{N}$ as desired. However, achieving a secure instantiation of the above sketch is challenging. The main missing step is a method to securely switch from row t_1 to $t_1 + 1$ in the index hiding game. To start, there is currently no clear way to switch the pk_j for one row to the pk_j for the next row when the index hiding adversary is given the PKE decryption keys corresponding to the next row. A straightforward solution is to generate a fresh symmetric encryption key $\mathsf{sk}_{\mathsf{SKE}}$ and use it to encrypt each column ciphertext $\mathsf{ct}_{\mathsf{PKE}}^{(c)}$ to get $\mathsf{ct}_{\mathsf{SKE}}^{(c)}$, and only publish $\mathsf{ct}_{\mathsf{SKE}}^{(c)}$ components in the ciphertext, similar to [NWW26]. The hope is that SKE security can allow us to switch the current PKE ciphertexts to PKE ciphertexts encrypted using the public keys for indices in the next row, which will ensure index hiding even if the adversary holds PKE decryption keys for the next row.

However, getting correctness with this approach means hard-wiring the symmetric key $\mathsf{sk}_{\mathsf{SKE}}$ in an LO program that decrypts with it, which in turn means we will need a way to simulate such a program without knowing $\mathsf{sk}_{\mathsf{SKE}}$ in the security proof if we want to employ the above idea. Recall that an LO scheme obfuscates a program P with respect to a fixed lock α, such that the resulting obfuscation outputs a message m if $P(x) = \alpha$ and $\perp$ otherwise. Security roughly states that the obfuscation can be simulated (by a null program) if the description of P and any auxiliary information are (computationally) independent of a randomly generated lock α. Thus, making use of LO security requires a method to include the lock to get functionality, but also a way to remove it in the security proof. With this in mind, a candidate underlying program for the target row t_1 is the following:

Hard-wired: A secret key $\mathsf{sk}_{\mathsf{SKE}}$.
Inputs: A PKE secret key $\mathsf{sk}_{\mathsf{PKE}}$ and an SKE ciphertext $\mathsf{ct}_{\mathsf{SKE}}$.

1. Compute $\mathsf{ct}_{\mathsf{PKE}} \leftarrow \mathsf{SKE.Decrypt}(\mathsf{sk}_{\mathsf{SKE}}, \mathsf{ct}_{\mathsf{SKE}})$.
2. Output $\mathsf{PKE.Decrypt}(\mathsf{sk}_{\mathsf{PKE}}, \mathsf{ct}_{\mathsf{PKE}})$.

Figure 1: Program $C[\mathsf{sk}_{\mathsf{SKE}}]$.

For $C[\mathsf{sk}_{\mathsf{SKE}}]$ to be a functional LO program that outputs the message to authorized users, each authorized column $c > t_2$ should generate $\mathsf{ct}_{\mathsf{PKE}}^{(c)}$ that encrypts the lock α rather than the message. Once we switch all PKE ciphertexts to 0 at the end of a row, the lock is no longer needed to generate the ciphertext, so the obfuscation can be simulated. This allows us to use SKE security to switch the encrypted PKE ciphertexts to PKE ciphertexts corresponding to the next row. However, the above SKE security step only works if the programs for all rows $r \neq t_1$ are simulated, since otherwise $\mathsf{sk}_{\mathsf{SKE}}$ would be needed to generate a row program. The issue is that we simultaneously want it to be the case that programs managing rows $r > t_1$ output the message to users i such that $\mathrm{row}(i) = r$, while the programs for rows $r < t_1$ do not contain message information. Thus, we need a way to interpret these simulated programs differently depending on how the row r compares to t_1.

Switching rows. A natural interpretation mechanism is to use an ABE scheme with succinct ciphertexts to encrypt either 0 or the message for each row depending on the cutoff: the decryption policy for user i checks that the row is correct and if so evaluates the obfuscation of $C[\mathsf{sk}_{\mathsf{SKE}}]$ with the appropriate $\mathsf{ct}_{\mathsf{SKE}}^{(c)}$ as input. A user's ABE policy should output 1 if P_r outputs $\perp$ (inactive) and output 0 if P_r reveals some fixed message, say $m = 1$ (active and unauthorized). Consequently, the target row PKE ciphertexts will encrypt

0 instead of α to denote that a user is authorized. This means at the end of a row we can appeal to ABE security to remove the message, since the PKE ciphertexts all encrypt the lock (causing keys for the active row to fail) and the other keys fail the row check. An additional benefit is that at the start of a new row, all PKE ciphertexts would encrypt 0 and thus there is no correlation with the lock of a newly activated P_r.

However, this is still not enough to switch rows in the index hiding game. The problem is that we can no longer make P_{t_1} a simulated obfuscation at the end of a row since all the PKE ciphertexts would now encrypt the lock, the adversary has all PKE decryption keys except sk_t, and the description of P_{t_1} contains $\mathsf{sk}_{\mathsf{SKE}}$. However, while appealing to LO security of P_{t_1} seems out of the question, we can still hope to *indirectly* nullify P_{t_1}. Indeed, our final ingredient will be to utilize the power of *nested* LO programs to gain additional flexibility similar to how [NWW26] uses null-iO programs. Specifically, if we are able to apply LO security to simulate an obfuscated program with another LO program hardwired inside of it, then the nested program will vanish and free up secrets that can allow us to continue the security proof, even if the auxiliary information and nested program's description would reveal the lock.

A construction with succinct ciphertexts. We now describe an augmented broadcast encryption scheme with an $O(N)$-size master public key, $\mathrm{polylog}(N)$-size secret keys, and $\sqrt{N}$-size ciphertexts that encrypts to the fixed set $[N]$. The scheme makes use of SKE, PKE, succinct ABE, LO for circuits, and weakly succinct LO for Turing machines. We switch the indexing convention for the end-of-row cutoff to be $(t_1 + 1, 0)$ in place of (t_1, n) for convenience.

- **Setup:** Generate a PKE key pair $(\mathsf{pk}_i, \mathsf{sk}'_i)$ for $i \in [N]$. Additionally, generate an ABE key pair $(\mathsf{mpk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}})$. For each $i \in [N]$, construct the policy $f[i, \mathsf{sk}'_i]$ which takes as input an attribute x consisting of a row $r \in [n]$, a program $\widehat{C}_{\mathsf{inner}}$, and ciphertexts $\{\mathsf{ct}^{(c)}_{\mathsf{SKE}}\}_{c\in[n]}$. The policy $f[i, \mathsf{sk}'_i]$ is satisfied (outputs 1) if
$$r = \mathsf{row}(i) \quad \text{and} \quad \perp = \widehat{C}_{\mathsf{inner}}\left(\mathsf{sk}'_i, \mathsf{ct}^{(\mathsf{col}(i))}_{\mathsf{SKE}}\right).$$
It outputs 0 otherwise. For each $i \in [N]$, set $f_i = f[i, \mathsf{sk}'_i]$ and sample $\mathsf{sk}_{f_i} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}_{\mathsf{ABE}}, f_i)$. The master public key is $\mathsf{mpk} = (\{\mathsf{pk}_i\}_{i\in[N]}, \mathsf{mpk}_{\mathsf{ABE}})$ and the secret key for user i is $\mathsf{sk}_i = (\mathsf{sk}'_i, \mathsf{sk}_{f_i})$.
- **Encryption:** To encrypt a message m to the set $[N]$ with cutoff $t \in [0, N]$ parsed as $t = (t_1 - 1)n + t_2$ where $t_1 \in [n + 1], t_2 \in [0, n - 1]$, proceed as follows:
  1. **Auxiliary ciphertexts.** Generate a trivial augmented broadcast encryption ciphertext with cutoff t_2 and encrypt the ciphertext under SKE. Specifically, sample a fresh lock α_{inner} and a fresh symmetric key $\mathsf{sk}_{\mathsf{SKE}}$. For each column $c \in [n]$, set $j = (t_1 - 1)n + c$ and compute
$$\mathsf{ct}^{(c)}_{\mathsf{PKE}} \leftarrow \begin{cases} \mathsf{PKE.Encrypt}(\mathsf{pk}_j, \alpha_{\mathsf{inner}}) & c \le t_2 \\ \mathsf{PKE.Encrypt}(\mathsf{pk}_j, 0) & c > t_2. \end{cases}$$
Compute $\mathsf{ct}^{(c)}_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Encrypt}(\mathsf{sk}_{\mathsf{SKE}}, \mathsf{ct}^{(c)}_{\mathsf{PKE}})$ and set $\mathsf{ct}_{\mathsf{aux}} = \{\mathsf{ct}^{(c)}_{\mathsf{SKE}}\}_{c\in[n]}$. When $t = N$, compute $\mathsf{ct}^{(c)}_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Encrypt}(\mathsf{sk}_{\mathsf{SKE}}, 0)$ for $c \in [n]$ instead of the above.
  2. **Inner row programs.** For each row $r \ge t_1$, compute the inner obfuscation
$$\widehat{C}^{(r)}_{\mathsf{inner}} \leftarrow \begin{cases} \mathsf{LO.Obf}(C[\mathsf{sk}_{\mathsf{SKE}}], \alpha_{\mathsf{inner}}, 1) & r = t_1 \\ \mathsf{LO.Sim}(\mathsf{pp}(C)) & r > t_1, \end{cases}$$
where $C[\mathsf{sk}_{\mathsf{SKE}}]$ is defined as in Fig. 1 and $\mathsf{pp}(C)$ is the public list of length parameters of the program $C[\mathsf{sk}_{\mathsf{SKE}}]$.

3. **Outer row programs.** Define the outer Turing machine $M[r, \widehat{C}_{\mathsf{inner}}, \mathsf{ct}_{\mathsf{ABE}}]$, which on input an ABE public key $\mathsf{mpk}_{\mathsf{ABE}}$ and secret key sk_f, a policy f, and ciphertexts $\mathsf{ct}_{\mathsf{aux}}$, outputs

$$\mathsf{ABE.Decrypt}(\mathsf{mpk}_{\mathsf{ABE}}, \mathsf{sk}_f, f, \mathsf{ct}_{\mathsf{ABE}}, x = (r, \widehat{C}_{\mathsf{inner}}, \mathsf{ct}_{\mathsf{aux}})).$$

For each row $r \in [n]$, generate the outer obfuscation $\widehat{M}^{(r)}_{\mathsf{outer}}$ as follows:

- If $r < t_1$, sample $\widehat{M}^{(r)}_{\mathsf{outer}} \leftarrow \mathsf{TMLO.Sim}(\mathsf{pp}(M))$ where $\mathsf{pp}(\cdot)$ is defined as above.
- If $r \geq t_1$, sample a fresh lock $\alpha^{(r)}_{\mathsf{outer}}$, set $x_r = (r, \widehat{C}^{(r)}_{\mathsf{inner}}, \mathsf{ct}_{\mathsf{aux}})$ and compute

$$\mathsf{ct}^{(r)}_{\mathsf{ABE}} \leftarrow \mathsf{ABE.Encrypt}(\mathsf{mpk}_{\mathsf{ABE}}, x_r, \alpha^{(r)}_{\mathsf{outer}}),$$
$$\widehat{M}^{(r)}_{\mathsf{outer}} \leftarrow \mathsf{TMLO.Obf}\big(M[r, \widehat{C}^{(r)}_{\mathsf{inner}}, \mathsf{ct}^{(r)}_{\mathsf{ABE}}], \alpha^{(r)}_{\mathsf{outer}}, m\big).$$

4. Output the ciphertext $\mathsf{ct} = (\mathsf{ct}_{\mathsf{aux}}, \widehat{M}^{(1)}_{\mathsf{outer}}, \ldots, \widehat{M}^{(n)}_{\mathsf{outer}})$.

- **Decryption:** User $i \in [N]$ with $\mathsf{sk}_i = (\mathsf{sk}'_i, \mathsf{sk}_{f_i})$ reconstructs $f_i = f[i, \mathsf{sk}'_i]$ and outputs

$$\widehat{M}^{(\mathsf{row}(i))}_{\mathsf{outer}}(\mathsf{mpk}_{\mathsf{ABE}}, \mathsf{sk}_{f_i}, f_i, \mathsf{ct}_{\mathsf{aux}}).$$

For any $i = (r-1)n + c > t$, the ABE policy is satisfied either by the simulated inner program outputting $\perp$ when $r > t_1$, or by the active inner program outputting $\perp$ when $r = t_1$ and $c > t_2$ because the PKE ciphertext encrypts 0. Correspondingly, evaluating $\widehat{M}^{(r)}_{\mathsf{outer}}$ will recover m.

Succinctness. To see why the scheme has $\sqrt{N}$-size ciphertexts, first note that $\mathsf{ct}_{\mathsf{ABE}}$ itself does not grow with the attribute length by ABE succinctness and the size of $C[\mathsf{sk}_{\mathsf{SKE}}]$ does not depend on N, meaning the description size of M is $\mathsf{poly}(\lambda, \log N)$. If the scheme TMLO is a weakly succinct LO for TMs, its obfuscation size grows polylogarithmically with the machine runtime T and polynomially with the description size and corresponding circuit depth for running the TM for T steps. The runtime of M is $\mathsf{poly}(\lambda, N)$ and the circuit depth is $\mathsf{poly}(\lambda, \log N)$, so the size of each $\widehat{M}^{(r)}_{\mathsf{outer}}$ is $\mathsf{poly}(\lambda, \log N)$ for $r \in [n]$. Each $\mathsf{ct}^{(c)}_{\mathsf{SKE}}$ for $c \in [n]$ has size $\mathsf{poly}(\lambda)$, so in total $|\mathsf{ct}| = \sqrt{N} \cdot \mathsf{poly}(\lambda, \log N)$.

Security analysis. Message hiding is immediate since all obfuscations are simulated and SKE ciphertexts encrypt 0 when $t = N$. We now sketch the argument for index hiding for all choices of $t \in [N]$. When t is not at the end of a row, index hiding follows by the PKE security since the only difference between ciphertexts for t and $t-1$ is a single PKE ciphertext $\mathsf{ct}^{(t_2)}_{\mathsf{PKE}}$ encrypting α_{inner} or 0 under pk_t, and sk'_t is not given to the adversary. This is just the argument from the trivial construction described earlier.

When $t = t_1 n$ is at the end of a row, its cutoff decomposes to $(t_1 + 1, 0)$ while $t - 1$ decomposes to $(t_1, n-1)$. This means that from a cutoff $t-1$ ciphertext the PKE ciphertexts need to be swapped to the next row, $\widehat{C}^{(t_1+1)}_{\mathsf{inner}}$ needs to be activated when $t_1 < n$, and $\widehat{M}^{(t_1)}_{\mathsf{outer}}$ needs to become a simulated obfuscation in order to reach a cutoff t ciphertext. These changes can be made with the following sequence of steps:

1. Appeal to ABE security to switch $\mathsf{ct}^{(t_1)}_{\mathsf{ABE}}$ to an encryption of 0 instead of $\alpha^{(t_1)}_{\mathsf{outer}}$. Notably, we do not need to generate the ABE key sk_{f_t} which would satisfy the policy since $\mathsf{ct}^{(n)}_{\mathsf{PKE}}$ still encrypts 0. Every other f_i evaluates to 0 either due to the row check or because the inner obfuscation outputs 1 due to $\mathsf{ct}^{(c)}_{\mathsf{PKE}}$ encrypting α_{inner} for $c < n$.

2. Switch $\widehat{M}^{(t_1)}_{\mathsf{outer}}$ to simulated since the ciphertext no longer depends on $\alpha^{(t_1)}_{\mathsf{outer}}$. Crucially, this step removes the need to generate $\widehat{C}^{(t_1)}_{\mathsf{inner}}$ during encryption, which then removes all appearances of $\mathsf{sk}_{\mathsf{SKE}}$ other than ciphertexts in $\mathsf{ct}_{\mathsf{aux}}$.

3. Apply SKE security to switch the encrypted PKE ciphertexts to encryptions of 0 under the public keys of the next row. When $t = N$, switch to SKE encryptions of 0 instead.

4. If $t_1 < n$, swap $\widehat{C}^{(t_1+1)}_{\mathsf{inner}}$ from simulated to an obfuscation of $C[\mathsf{sk}_{\mathsf{SKE}}]$ since the PKE ciphertexts encrypt 0 and thus there are no inner lock correlations.

We give the formal construction and analysis in Section 3.

Shrinking the public key and handling arbitrary sets. We can compress the master public key and keep essentially the same security proof by simply replacing the N PKE schemes with identity-based encryption [Sha84] that supports identity space $\{0,1\}^\lambda$, which will readily allow us to assign each user an identity $i \in [N]$ for all $N = \mathsf{poly}(\lambda)$. To additionally handle an arbitrary set $S \subseteq [N]$ at encryption time, we make the following changes:

- The ABE policy f_i now checks that $i \in S$ in addition to the row check and S becomes part of the attribute. The set S is also added as an input to the outer machine M which includes it in the attribute when running ABE decryption.

- At encryption time, $\mathsf{ct}^{(c)}_{\mathsf{PKE}}$ only encrypts α_{inner} if $c \le t_2$ and identity $j = (t_1 - 1)n + c \in S$.

The security proof is largely unchanged since cutoffs $t \notin S$ mean there is no change within a row and the adversary getting the key for t is admissible at the end of a row since the set check makes the ABE policy unsatisfied.

2 Preliminaries

We write λ to denote the security parameter. For an integer $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For a distribution $\mathcal{D}$ we write $x \leftarrow \mathcal{D}$ to denote that x is a random draw from $\mathcal{D}$. For a finite set S, we write $x \xleftarrow{R} S$ to denote a uniform random draw from S. By default, $S \subseteq [n]$ is represented as n bits indicating containment. We write $\mathsf{poly}(\lambda)$ to denote a fixed function that is $O(\lambda^c)$ for some $c \in \mathbb{N}$ and $\mathsf{negl}(\lambda)$ to denote a function that is $o(\lambda^{-c})$ for all $c \in \mathbb{N}$. We say an event occurs with overwhelming probability if its complement occurs with negligible probability. We say an algorithm is efficient if it runs in probabilistic polynomial time (in the length of its input). All message spaces $\mathcal{M}_\lambda$ are finite, efficiently uniformly sampleable, have at least two elements, and have fixed-length messages. Thus $m \xleftarrow{R} \mathcal{M}_\lambda$ always means a uniform sample, independent of the fresh encryption coins and all other fresh coins in the same context. All program, ciphertext, key, and message lengths are padded to their common bounds unless otherwise specified.

Encryption schemes. We recall the basic notions of CPA-secure secret-key encryption, identity-based encryption, and attribute-based encryption.

Definition 2.1 (Secret-Key Encryption)**.** A secret key encryption scheme with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of PPT algorithms $\Pi_{\mathsf{SKE}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{KeyGen}(1^\lambda) \to \mathsf{sk}$: On input the security parameter λ, the key generation algorithm outputs a secret key sk.
- $\mathsf{Encrypt}(\mathsf{sk}, m) \to \mathsf{ct}$: On input the secret key sk and a message $m \in \mathcal{M}_\lambda$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) \to m$: On input the secret key sk and a ciphertext ct, the decryption algorithm outputs a message $m \in \mathcal{M}_\lambda$.

We require that Π_{SKE} satisfy the following correctness property, along with at least one of the following security properties:

- **Correctness:** For all $\lambda \in \mathbb{N}$ and any message $m \in \mathcal{M}_\lambda$, we have

$$\Pr\left[\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) = m : \begin{array}{c} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, m) \end{array}\right] = 1$$

- **CPA security:** For a security parameter λ, a bit $b \in \{0,1\}$, and an adversary $\mathcal{A}$, we define the indistinguishability security game as follows:

  1. At the beginning of the game, the challenger samples $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
  2. Adversary $\mathcal{A}$ can make encryption queries on pairs of messages $m_0, m_1 \in \mathcal{M}_\lambda$. The challenger replies with $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, m_b)$
  3. At the end of the game, adversary $\mathcal{A}$ outputs $b' \in \{0,1\}$, which is the output of the experiment.

  We say an SKE scheme satisfies CPA security if for all efficient adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 | b = 0] - \Pr[b' = 1 | b = 1]| = \mathsf{negl}(\lambda)$$

  in the above indistinguishability security game.

Definition 2.2 (Identity-Based Encryption [Sha84, BF01, Coc01])**.** An identity-based encryption (IBE) scheme with identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda\in\mathbb{N}}$ and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is a tuple of efficient algorithms $\Pi_{\mathsf{IBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$: On input the security parameter λ, the setup algorithm outputs the set of public parameters pp and a master secret key msk.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$: On input the master secret key msk and an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, the key-generation algorithm outputs an identity key $\mathsf{sk}_{\mathsf{id}}$.
- $\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{id}, m) \to \mathsf{ct}$: On input the master public key mpk, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a message $m \in \mathcal{M}_\lambda$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}) \to m$: On input an identity secret key $\mathsf{sk}_{\mathsf{id}}$, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a ciphertext ct, the decryption algorithm outputs a message $m \in \mathcal{M}_\lambda$.

Moreover, Π_{IBE} should satisfy the following properties:

- **Correctness:** For all security parameters $\lambda \in \mathbb{N}$, all identities $\mathsf{id} \in \mathcal{ID}_\lambda$, all $(\mathsf{mpk}, \mathsf{msk})$ in the support of $\mathsf{Setup}(1^\lambda)$, and all messages $m \in \mathcal{M}_\lambda$,

$$\Pr\left[\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}) = m : \begin{array}{c} \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{id}, m) \end{array}\right] = 1.$$

- **Semantic security:** For a security parameter λ and a bit $b \in \{0, 1\}$, we define the (adaptive) semantic security game between an adversary $\mathcal{A}$ and a challenger as follows:
  - **Setup:** The challenger starts by sampling $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives mpk to $\mathcal{A}$.
  - **Pre-challenge queries:** Algorithm $\mathcal{A}$ can now issue key-generation queries to the challenger. On each key-generation query, adversary $\mathcal{A}$ specifies an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and the challenger replies with $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.
  - **Challenge:** Algorithm $\mathcal{A}$ outputs a challenge identity id^* and two messages $m_0, m_1 \in \mathcal{M}_\lambda$. The challenger replies with $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{id}^*, m_b)$.
  - **Post-challenge queries:** Algorithm $\mathcal{A}$ can continue to make key-generation queries as in the pre-challenge phase.
  - **Output:** At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$, which is the output of the experiment.

  An adversary $\mathcal{A}$ is admissible for the semantic security game if it does not issue a key-generation query on the challenge identity id^*. We say Π_{IBE} satisfies adaptive security if for all efficient and admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda)$$

  in the semantic security game.

Definition 2.3 (Attribute-Based Encryption [SW05, GPSW06])**.** A key-policy attribute-based encryption (ABE) scheme with policy space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$, attribute space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of efficient algorithms $\Pi_{\mathsf{ABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$: On input the security parameter λ, the setup algorithm outputs the master public key mpk and a master secret key msk. We assume that mpk and msk include the security parameter 1^λ.
- $\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$: On input the master secret key msk and a policy $f \in \mathcal{F}_\lambda$, the key-generation algorithm outputs a policy key sk_f.
- $\mathsf{Encrypt}(\mathsf{mpk}, x, m) \to \mathsf{ct}$: On input the master public key mpk, an attribute $x \in \mathcal{X}_\lambda$, and a message $m \in \mathcal{M}_\lambda$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}, x) \to m$: On input the master public key mpk, a policy key sk_f, its policy $f \in \mathcal{F}_\lambda$, a ciphertext ct, and an attribute $x \in \mathcal{X}_\lambda$, the decryption algorithm outputs a message $m \in \mathcal{M}_\lambda$.

Moreover, Π_{ABE} should satisfy the following properties:

- **Correctness:** For all security parameters $\lambda \in \mathbb{N}$, all attributes $x \in \mathcal{X}_\lambda$, all policies $f \in \mathcal{F}_\lambda$ such that $f(x) = 1$, all $(\mathsf{mpk}, \mathsf{msk})$ in the support of $\mathsf{Setup}(1^\lambda)$, and all messages $m \in \mathcal{M}_\lambda$,

$$\Pr\left[\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}, x) = m : \begin{array}{c} \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, x, m) \end{array}\right] = 1.$$

- **Attribute-selective security:** For a security parameter λ and a bit $b \in \{0,1\}$, we define the attribute-selective security game between an adversary $\mathcal{A}$ and a challenger as follows:
  - **Setup:** At the beginning of the game, algorithm $\mathcal{A}$ outputs a challenge attribute x^*. The challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives mpk to $\mathcal{A}$.
  - **Pre-challenge queries:** Algorithm $\mathcal{A}$ can now issue key-generation queries to the challenger. On each key-generation query, adversary $\mathcal{A}$ specifies a policy $f \in \mathcal{F}_\lambda$, and the challenger replies with $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$.
  - **Challenge:** Algorithm $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}_\lambda$. The challenger replies with $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, x^*, m_b)$.
  - **Post-challenge queries:** Algorithm $\mathcal{A}$ can continue to make key-generation queries as in the pre-challenge phase.
  - **Output:** At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

  An adversary $\mathcal{A}$ is admissible for the attribute-selective security game if it does not issue a key-generation query on a function f such that $f(x^*) = 1$. We say Π_{ABE} satisfies attribute-selective security if for all efficient and admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda)$$

  in the attribute-selective security game.

Fact 2.4 (Succinct ABE from Decomposed LWE [Wee25])**.** Let $\mathcal{F}$ be the set of all Boolean circuits with depth at most d and input length ℓ. Then, under the decomposed LWE assumption [AMR25] with a sub-exponential modulus-to-noise ratio, there exists attribute-selectively secure ABE with master public key, secret key, and ciphertext size $\mathsf{poly}(\lambda, d)$ (ignoring logarithmic terms and the public attribute). Moreover, decryption has depth $\mathsf{poly}(\lambda, d)$ for policies f of depth d. Such a scheme implies IBE and SKE.

Lockable obfuscation. We now recall the notion of lockable obfuscation [WZ17, GKW17] for both circuits and Turing machines. For a circuit C we use $\mathsf{pp}(C)$ to denote a unary tuple of input length, output length and circuit size. For a Turing machine M, the function $\mathsf{pp}(M)$ includes input length, output length, description size, and corresponding circuit depth in unary, and the runtime in binary.

Definition 2.5 (Lockable Obfuscation [WZ17, GKW17])**.** A lockable obfuscation scheme with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, output space $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, program class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$, and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is a tuple of efficient algorithms $\Pi_{\mathsf{LO}} = (\mathsf{Obf}, \mathsf{Eval})$ with the following syntax:

- $\mathsf{Obf}(1^\lambda, P, \alpha, m) \to \widehat{P}$: On input the security parameter λ, a program $P \in \mathcal{P}_\lambda$, a lock $\alpha \in \mathcal{Y}_\lambda$, and a message $m \in \mathcal{M}_\lambda$, the obfuscation algorithm outputs an obfuscated program $\widehat{P}$.

- $\mathsf{Eval}(\widehat{P}, x) \to m$: On input an obfuscated program $\widehat{P}$ and an input $x \in \mathcal{X}_\lambda$, the evaluation algorithm outputs a message $m \in \mathcal{M}_\lambda \cup \{\bot\}$.

Moreover, Π_{LO} should satisfy the following properties:

- **Correctness:** There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, programs $P \in \mathcal{P}_\lambda$, locks $\alpha \in \mathcal{Y}_\lambda$, and messages $m \in \mathcal{M}_\lambda$, we have

$$\Pr_{\widehat{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \alpha, m)} [\forall x \in \mathcal{X} \ : \ \mathsf{Eval}(\widehat{P}, x) = P_{\alpha,m}(x)] \geq 1 - \mathsf{negl}(\lambda),$$

  where $P_{\alpha,m}$ is the program that outputs m if $P(x) = \alpha$ and $\bot$ otherwise.

- **Simulation security:** For an adversary $\mathcal{A}$, a simulator Sim, and a bit $b \in \{0,1\}$, we define the simulation-security game as follows:

  1. At the start of the game, the adversary $\mathcal{A}$ outputs a program $P \in \mathcal{P}_\lambda$ and a message $m \in \mathcal{M}_\lambda$. The challenger samples $\alpha \xleftarrow{R} \mathcal{Y}_\lambda$, computes $\widehat{P}_0 \leftarrow \mathsf{Obf}(1^\lambda, P, \alpha, m)$, and computes $\widehat{P}_1 \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{pp}(P), 1^{|m|})$. The challenger replies with $\widehat{P}_b$.
  2. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

  We say Π_{LO} satisfies simulation security if there exists an efficient simulator Sim such that for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda)$$

  in the simulation security game. We require that with overwhelming probability, the simulated obfuscation always outputs $\bot$.

Fact 2.6 (Lockable Obfuscation from LWE [WZ17, GKW17])**.** Under the plain LWE assumption, there exists a lockable obfuscation scheme for the following program classes:

- Size-s circuits C with obfuscation size $\mathsf{poly}(s)$.
- Time-t Turing machines $\mathcal{TM}$ that have description length ℓ and can be written as circuits of depth d. The obfuscation size is $\mathsf{poly}(\ell, \log t, d)$.

Here, we exclude polynomials in the message length, lock length, and security parameter.

2.1 Broadcast and Trace

In this section, we recall the definition of broadcast and trace [NP00, NNL01], the main primitive we construct in this work. We additionally recall the notion of augmented broadcast encryption [BW06], a useful intermediate primitive that implies broadcast and trace.

Definition 2.7 (Broadcast and Trace [BW06, adapted])**.** A broadcast and trace scheme over a message space $\mathcal{M}_\lambda$ is a tuple of efficient algorithms $\Pi_{\mathsf{BT}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Trace})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{mpk}, \{\mathsf{sk}_i\}_{i \in [N]})$: On input the security parameter λ and a number of users N, the setup algorithm outputs a master public key mpk and a set of user secret keys $\{\mathsf{sk}_i\}_{i \in [N]}$.

- $\mathsf{Encrypt}(\mathsf{mpk}, S, m) \to \mathsf{ct}$: On input the master public key mpk, a set $S \subseteq [N]$, and a message $m \in \mathcal{M}_\lambda$, the (randomized) encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{mpk}, i, \mathsf{sk}_i, S, \mathsf{ct}) \to m$: On input the master public key mpk, a secret key sk_i, a set $S \subseteq [N]$, and a ciphertext ct, the (deterministic) decryption algorithm outputs a message m or $\perp$ to indicate failure.
- $\mathsf{Trace}^{\mathsf{D}}(\mathsf{mpk}, 1^K, S) \to i^*$: On input the master public key mpk, a tracing advantage 1^K, a set $S \subseteq [N]$, and oracle access to a decoding box D, the tracing algorithm outputs a traitor's index $i^* \in [N]$ or $\perp$ to indicate failure.

Moreover, Π_{BT} should satisfy the following properties:

- **Correctness:** For every polynomially bounded $N = N(\lambda)$, there exists a negligible function negl such that, for all λ, all $m \in \mathcal{M}_\lambda$, all $S \subseteq [N]$, and all $i \in S$,

$$\Pr\left[\mathsf{Decrypt}(\mathsf{mpk}, i, \mathsf{sk}_i, S, \mathsf{ct}) = m \;:\; \begin{array}{l}(\mathsf{mpk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Setup}(1^\lambda, 1^N); \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, S, m)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

- **Succinctness:** There exists a polynomial poly such that for all $\lambda, N \in \mathbb{N}$, all $(\mathsf{mpk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ in the support of $\mathsf{Setup}(1^\lambda, 1^N)$, all sets $S \subseteq [N]$, all messages $m \in \mathcal{M}_\lambda$, and all ciphertexts ct in the support of $\mathsf{Encrypt}(\mathsf{mpk}, S, m)$, we have that

$$|\mathsf{mpk}|, |\mathsf{sk}_i| \leq \mathsf{poly}(\lambda, \log N) \quad \text{and} \quad |\mathsf{ct}| \leq o(N) \cdot \mathsf{poly}(\lambda).$$

- **Set-selective semantic security:** For $b \in \{0, 1\}$ and an adversary $\mathcal{A}$, define the semantic security game as follows:

  1. The adversary $\mathcal{A}$ is initialized on input 1^λ, and it sends a user count 1^N and a set $S \subseteq [N]$ to the challenger who computes $(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i \in [N]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$ and returns the master public key mpk and sk_i for $i \in [N] \setminus S$ to $\mathcal{A}$.
  2. The adversary $\mathcal{A}$ outputs two equal-length messages $m_0, m_1 \in \mathcal{M}_\lambda$. The challenger computes $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, S, m_b)$ and gives ct to $\mathcal{A}$.
  3. The adversary outputs a bit $b' \in \{0, 1\}$, which is the output of the experiment.

  We say Π_{BT} satisfies set-selective semantic security if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda)$$

  in the above game.

- **Set-selective traceability:** Let $b \in \{0, 1\}$, $\mathcal{A}$ be an adversary, and $\varepsilon = \varepsilon(\lambda)$ be an inverse-polynomial function such that $0 < \varepsilon(\lambda) \leq 1$ for all $\lambda \in \mathbb{N}$. We define the selective traceability experiment as follows:

  – **Setup phase:** The adversary $\mathcal{A}$ is initialized on input 1^λ, and it sends a user count 1^N and a set $S \subseteq [N]$ to the challenger who computes $(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i \in [N]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$ and returns the master public key mpk to $\mathcal{A}$. The challenger additionally initializes an empty set C.

– **Key query phase:** In a secret key corruption query, $\mathcal{A}$ specifies a user index $i \in [N]$. In response, the challenger returns sk_i to the adversary and updates the set $C \leftarrow C \cup \{i\}$.

– **Tracing phase:** The adversary $\mathcal{A}$ outputs a pirate decoding box D^*. With this, the challenger sets $K = \lceil 1/\varepsilon(\lambda) \rceil$ and executes $i^* \leftarrow \mathsf{Trace}^{\mathsf{D}^*}(\mathsf{mpk}, 1^K, S)$, giving the tracing algorithm oracle access to D^*.

Let $p_{\mathsf{D}^*} \coloneqq \Pr[\mathsf{D}^*(\mathsf{ct}) = m \mid m \xleftarrow{\mathrm{R}} \mathcal{M}_\lambda, \mathsf{ct} \xleftarrow{\mathrm{R}} \mathsf{Encrypt}(\mathsf{mpk}, S, m)]$ and define the following events:

$$\mathsf{GoodDec} \coloneqq \left[p_{\mathsf{D}^*} \geq \frac{1}{|\mathcal{M}_\lambda|} + \varepsilon(\lambda) \right], \quad \begin{aligned} \mathsf{GoodTr} &\coloneqq [i^* \neq \bot], \\ \mathsf{BadTr} &\coloneqq [i^* \neq \bot \wedge i^* \notin S \cap C]. \end{aligned}$$

We say that Π_{BT} satisfies selective traceability if, for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, for all $\lambda \in \mathbb{N}$,

$$\left| \max \left\{ \Pr[\mathsf{GoodDec}] - \Pr[\mathsf{GoodTr}], \Pr[\mathsf{BadTr}] \right\} \right| \leq \mathsf{negl}(\lambda).$$

Augmented broadcast encryption. We now recall the notion of augmented broadcast encryption [BW06] with set-selective index hiding, which implies a broadcast and trace scheme with set-selective semantic security and tracing.

Definition 2.8 (Augmented Broadcast Encryption [BW06, adapted])**.** An augmented broadcast encryption scheme over a message space $\mathcal{M}_\lambda$ is a tuple of efficient algorithms $\Pi_{\mathsf{AugBE}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{mpk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$: On input the security parameter λ and a number of users N, the setup algorithm outputs a master public key mpk and secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$.
- $\mathsf{Encrypt}(\mathsf{mpk}, S, m, t) \to \mathsf{ct}$: On input the master public key mpk, a recipient set $S \subseteq [N]$, a message $m \in \mathcal{M}_\lambda$, and a cutoff $t \in \{0, \ldots, N\}$, the encryption algorithm output a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{mpk}, i, \mathsf{sk}_i, S, \mathsf{ct}) \to m$: On input the master public key mpk, a user index i, a secret key sk_i, a recipient set S, and a ciphertext ct, the decryption algorithm deterministically outputs a message $m \in \mathcal{M}_\lambda \cup \{\bot\}$.

Moreover, Π_{AugBE} should satisfy the following properties:

- **Correctness:** For every polynomially bounded $N = N(\lambda)$, there exists a negligible function negl such that, for all λ, all $m \in \mathcal{M}_\lambda$, all $S \subseteq [N]$, all $t \in \{0, \ldots, N\}$, and all $i \in S$ with $i > t$,

$$\Pr\left[\mathsf{Decrypt}(\mathsf{mpk}, i, \mathsf{sk}_i, S, \mathsf{ct}) = m \;:\; \begin{array}{l} (\mathsf{mpk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Setup}(1^\lambda, 1^N); \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, S, m, t) \end{array} \right] \geq 1 - \mathsf{negl}(\lambda).$$

- **Succinctness:** There exists a polynomial poly such that for all $\lambda, N \in \mathbb{N}$, all $(\mathsf{mpk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ in the support of $\mathsf{Setup}(1^\lambda, 1^N)$, all sets $S \subseteq [N]$, all messages $m \in \mathcal{M}_\lambda$, all indices $t \in \{0, \ldots, N\}$, and all ciphertexts ct in the support of $\mathsf{Encrypt}(\mathsf{mpk}, S, m, t)$, we have that

$$|\mathsf{mpk}|, |\mathsf{sk}_i| \leq \mathsf{poly}(\lambda, \log N) \quad \text{and} \quad |\mathsf{ct}| \leq o(N) \cdot \mathsf{poly}(\lambda).$$

- **Message hiding:** For $b \in \{0, 1\}$ and an adversary $\mathcal{A}$, define the message hiding game as follows:

1. At the beginning of the game, the adversary $\mathcal{A}$, outputs 1^N. The challenger samples

$$(\mathsf{mpk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$$

   and gives all of these values to $\mathcal{A}$.
2. The adversary outputs a set $S \subseteq [N]$ and equal-length messages $m_0, m_1 \in \mathcal{M}_\lambda$. The challenger samples $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, S, m_b, N)$ and gives ct to $\mathcal{A}$.
3. The adversary outputs a bit $b' \in \{0,1\}$, which is the experiment's output.

We say Π_{AugBE} satisfies message hiding if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda)$$

in the above message hiding game.

- **Set-selective index hiding:** For $b \in \{0,1\}$ and an adversary $\mathcal{A}$, define the set-selective index hiding experiment as follows:

  1. At the beginning of the game, the adversary $\mathcal{A}$ outputs 1^N, an index $t \in [N]$, and a set $S \subseteq [N]$. The challenger samples $(\mathsf{mpk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$ and gives $(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i \neq t})$ to $\mathcal{A}$ if $t \in S$ or $(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i \in [N]})$ to $\mathcal{A}$ otherwise.
  2. The adversary outputs a message $m \in \mathcal{M}_\lambda$. The challenger samples the challenge ciphertext $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, S, m, t - b)$ and gives ct to $\mathcal{A}$.
  3. The experiment outputs the adversary's final bit $b' \in \{0,1\}$.

  We say Π_{AugBE} satisfies index hiding if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda)$$

  in the above index hiding game.

Fact 2.9 (Broadcast and Trace from AugBE [BW06, GVW19])**.** Suppose there exists a secure augmented broadcast encryption scheme Π_{AugBE} satisfying message hiding and set-selective index hiding. Then, there exists a secure broadcast and trace scheme Π_{BT} that satisfies set-selective traceability and semantic security with the same parameters as Π_{AugBE}.

3 Broadcast and Trace from Decomposed LWE

We now construct collusion-resistant broadcast and trace with public tracing from the decomposed LWE assumption. We do so by first constructing augmented broadcast encryption with keys of size $\mathsf{poly}(\lambda, \log N)$ and ciphertexts of size $\sqrt{N} \cdot \mathsf{poly}(\lambda, \log N)$. Our construction generically combines lockable obfuscation, ABE with succinct ciphertexts, IBE, and SKE, all of which can be obtained from decomposed LWE by Facts 2.4 and 2.6.

Notation. Let $N = n^2$ for $n \in \mathbb{N}$. We arrange the set $[N]$ as an $n \times n$ grid. For each $i \in [N]$, define $\mathrm{row}(i), \mathrm{col}(i) \in [n]$ by

$$i = (\mathrm{row}(i) - 1)n + \mathrm{col}(i).$$

For a cutoff $t \in \{0, \ldots, N\}$, write

$$t = (t_1 - 1)n + t_2, \qquad \text{where } t_1 \in [n+1], \quad t_2 \in \{0, \ldots, n-1\}.$$

In particular, $t = N$ corresponds to $(t_1, t_2) = (n+1, 0)$.

Hard-wired: A secret key $\mathsf{sk}_{\mathsf{SKE}}$.
Inputs: A user index $i \in [N]$, a candidate IBE secret key $\mathsf{sk}_{\mathsf{IBE}}$, and an SKE ciphertext $\mathsf{ct}_{\mathsf{SKE}}$.

1. Compute $\mathsf{ct}_{\mathsf{IBE}} \leftarrow \mathsf{SKE.Decrypt}(\mathsf{sk}_{\mathsf{SKE}}, \mathsf{ct}_{\mathsf{SKE}})$.
2. Compute $m' \leftarrow \mathsf{IBE.Decrypt}(\mathsf{sk}_{\mathsf{IBE}}, i, \mathsf{ct}_{\mathsf{IBE}})$.
3. If $m' \in \{0,1\}^\lambda$, output m'. Otherwise, output 0^λ.

Figure 2: Circuit $C_{\mathsf{inner}}[\mathsf{sk}_{\mathsf{SKE}}]$.

Hard-wired: A user index $i \in [N]$ and its IBE secret key $\mathsf{sk}_{\mathsf{IBE}}^{(i)}$.
Input: An attribute $x = (S, r, \widehat{C}_{\mathsf{inner}}, \mathsf{ct}_{\mathsf{aux}})$, where $S \subseteq [N]$, $r \in [n]$, and $\mathsf{ct}_{\mathsf{aux}} = (\mathsf{ct}_{\mathsf{SKE}}^{(1)}, \ldots, \mathsf{ct}_{\mathsf{SKE}}^{(n)})$.

1. If $i \notin S$, $r \neq \mathrm{row}(i)$, or $\widehat{C}_{\mathsf{inner}}$ is not a well-formed inner program, output 0.
2. Compute
$$b' \leftarrow \mathsf{LO.Eval}\big(\widehat{C}_{\mathsf{inner}}, (i, \mathsf{sk}_{\mathsf{IBE}}^{(i)}, \mathsf{ct}_{\mathsf{SKE}}^{(\mathrm{col}(i))})\big).$$
3. If $b' = \bot$, output 1. Otherwise, output 0.

Figure 3: Circuit $f[i, \mathsf{sk}_{\mathsf{IBE}}^{(i)}]$.

Hard-wired: A row $r \in [n]$, an inner obfuscation $\widehat{C}_{\mathsf{inner}}$, and an ABE ciphertext $\mathsf{ct}_{\mathsf{ABE}}$.
Inputs: An ABE public key mpk and secret key sk_f, a function f, a set $S' \subseteq [N]$, and a list of ciphertexts $\mathsf{ct}'_{\mathsf{aux}}$.

1. Set $x' = (S', r, \widehat{C}_{\mathsf{inner}}, \mathsf{ct}'_{\mathsf{aux}})$ and compute $m' \leftarrow \mathsf{ABE.Decrypt}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}_{\mathsf{ABE}}, x')$.
2. If $m' \in \{0,1\}^\lambda$, output m'. Otherwise, output 0^λ.

Figure 4: Turing machine $M_{\mathsf{outer}}[r, \widehat{C}_{\mathsf{inner}}, \mathsf{ct}_{\mathsf{ABE}}]$.

Construction 3.1 (Augmented Broadcast Encryption)**.** Let $\lambda \in \mathbb{N}$ be a security parameter and $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ be a message space with message length $\ell = \ell(\lambda)$. Let $N = n^2$ for some $n \in \mathbb{N}$ without loss of generality (it is always possible to pad to the next square). Our construction relies on the following primitives:

- Let $\Pi_{\mathsf{IBE}} = (\mathsf{IBE.Setup}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Encrypt}, \mathsf{IBE.Decrypt})$ be an IBE scheme with identity and message space $\{0,1\}^\lambda$, which notably contains $[N+n]$ in binary for any poly-bounded N.
- Let $\Pi_{\mathsf{SKE}} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Encrypt}, \mathsf{SKE.Decrypt})$ be a CPA-secure SKE scheme with message space that contains (padded) Π_{IBE} ciphertexts.
- Let $\Pi_{\mathsf{ABE}} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Encrypt}, \mathsf{ABE.Decrypt})$ be an attribute-selectively secure ABE scheme for the attributes and predicates in Fig. 3, with message space $\{0,1\}^\lambda$.
- Let $\Pi_{\mathsf{LO}} = (\mathsf{LO.Obf}, \mathsf{LO.Eval})$ and $\Pi_{\mathsf{TMLO}} = (\mathsf{TMLO.Obf}, \mathsf{TMLO.Eval})$ be lockable obfuscation schemes for circuits and Turing machines, respectively. The message space for Π_{LO} is $\{0,1\}$, the message space for Π_{TMLO} is $\mathcal{M}_\lambda$, and the output space for both is $\{0,1\}^\lambda$. The input spaces are defined as in Figs. 2 and 4. Write LO.Sim and TMLO.Sim for their simulators.

We construct $\Pi_{\mathsf{AugBE}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^N)$: On input the security parameter λ and the number of users $N = n^2$, the setup algorithm proceeds as follows:

  1. Sample $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$ and $(\mathsf{mpk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$.
  2. For each $i \in [N]$, sample $\mathsf{sk}_{\mathsf{IBE}}^{(i)} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}_{\mathsf{IBE}}, i)$, form the predicate $f_i = f[i, \mathsf{sk}_{\mathsf{IBE}}^{(i)}]$ from Fig. 3, and sample $\mathsf{sk}_{f_i} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}_{\mathsf{ABE}}, f_i)$.

  Output
  $$\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{mpk}_{\mathsf{ABE}}) \quad \text{and} \quad \mathsf{sk}_i = (\mathsf{sk}_{\mathsf{IBE}}^{(i)}, \mathsf{sk}_{f_i}) \quad \text{for each } i \in [N].$$

- $\mathsf{Encrypt}(\mathsf{mpk}, S, m, t)$: On input a master public key $\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{mpk}_{\mathsf{ABE}})$, a set $S \subseteq [N]$, a message $m \in \mathcal{M}_\lambda$, and a cutoff $t \in \{0, \ldots, N\}$ parsed as $t = (t_1 - 1)n + t_2$, the encryption algorithm proceeds as follows:

  1. Sample $\alpha_{\mathsf{inner}} \xleftarrow{R} \{0,1\}^\lambda$ and $\mathsf{sk}_{\mathsf{SKE}} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$. For each $c \in [n]$, generate an IBE ciphertext for identity $j = (t_1 - 1)n + c$ as follows:
  $$\mathsf{ct}_{\mathsf{IBE}}^{(c)} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{mpk}_{\mathsf{IBE}}, j, z_c) \text{ where } z_c = \begin{cases} \alpha_{\mathsf{inner}} & j \in S,\ c \le t_2 \\ 0^\lambda & \text{otherwise.} \end{cases}$$
  For each $c \in [n]$, compute
  $$\mathsf{ct}_{\mathsf{SKE}}^{(c)} \leftarrow \mathsf{SKE.Encrypt}(\mathsf{sk}_{\mathsf{SKE}}, \mathsf{ct}_{\mathsf{IBE}}^{(c)}).$$
  Set $\mathsf{ct}_{\mathsf{aux}} = (\mathsf{ct}_{\mathsf{SKE}}^{(1)}, \ldots, \mathsf{ct}_{\mathsf{SKE}}^{(n)})$.
  2. For each $r \ge t_1$, generate an inner obfuscation by
  $$\widehat{C}_{\mathsf{inner}}^{(r)} \leftarrow \begin{cases} \mathsf{LO.Obf}(1^\lambda, C_{\mathsf{inner}}[\mathsf{sk}_{\mathsf{SKE}}], \alpha_{\mathsf{inner}}, 1), & r = t_1, \\ \mathsf{LO.Sim}(1^\lambda, \mathsf{pp}(C_{\mathsf{inner}}), 1), & r > t_1. \end{cases}$$
  3. For each $r \in [n]$, generate an outer obfuscation $\widehat{M}_{\mathsf{outer}}^{(r)}$ as follows:
     - If $r < t_1$, sample $\widehat{M}_{\mathsf{outer}}^{(r)} \leftarrow \mathsf{TMLO.Sim}(1^\lambda, \mathsf{pp}(M_{\mathsf{outer}}), 1^\ell)$.

– Otherwise, sample $\alpha_{\mathsf{outer}}^{(r)} \xleftarrow{R} \{0,1\}^\lambda$, set $x_r = (S, r, \widehat{C}_{\mathsf{inner}}^{(r)}, \mathsf{ct}_{\mathsf{aux}})$, and compute

$$\mathsf{ct}_{\mathsf{ABE}}^{(r)} \leftarrow \mathsf{ABE.Encrypt}(\mathsf{mpk}_{\mathsf{ABE}}, x_r, \alpha_{\mathsf{outer}}^{(r)}),$$
$$\widehat{M}_{\mathsf{outer}}^{(r)} \leftarrow \mathsf{TMLO.Obf}\big(1^\lambda, M_{\mathsf{outer}}[r, \widehat{C}_{\mathsf{inner}}^{(r)}, \mathsf{ct}_{\mathsf{ABE}}^{(r)}], \alpha_{\mathsf{outer}}^{(r)}, m\big).$$

Output the ciphertext $\mathsf{ct} = (\mathsf{ct}_{\mathsf{aux}}, \widehat{M}_{\mathsf{outer}}^{(1)}, \ldots, \widehat{M}_{\mathsf{outer}}^{(n)})$.

- $\mathsf{Decrypt}(\mathsf{mpk}, i, \mathsf{sk}_i, S, \mathsf{ct})$: On input a master public key $\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{mpk}_{\mathsf{ABE}})$, a user index i, its secret key $\mathsf{sk}_i = (\mathsf{sk}_{\mathsf{IBE}}^{(i)}, \mathsf{sk}_{f_i})$, a set $S \subseteq [N]$, and a ciphertext $\mathsf{ct} = (\mathsf{ct}_{\mathsf{aux}}, \widehat{M}_{\mathsf{outer}}^{(1)}, \ldots, \widehat{M}_{\mathsf{outer}}^{(n)})$, the decryption algorithm sets $f_i = f[i, \mathsf{sk}_{\mathsf{IBE}}^{(i)}]$ and outputs

$$\mathsf{TMLO.Eval}\big(\widehat{M}_{\mathsf{outer}}^{(\mathsf{row}(i))}, (\mathsf{mpk}_{\mathsf{ABE}}, \mathsf{sk}_{f_i}, f_i, S, \mathsf{ct}_{\mathsf{aux}})\big).$$

Theorem 3.2 (Correctness). *Suppose* Π_{IBE}, Π_{SKE}, Π_{ABE}, Π_{LO}, *and* Π_{TMLO} *are correct. Then, Construction 3.1 is correct.*

Proof. Take any security parameter λ, number of users N, recipient set $S \subseteq [N]$, message $m \in \mathcal{M}_\lambda$, cutoff t, and user $i \in S$ with $i > t$. Let $(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$ and $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, S, m, t)$. Necessarily $t < N$. Except with probability $2^{-\lambda}$, we have $\alpha_{\mathsf{inner}} \neq 0^\lambda$. Condition on this event and consider the output of $\mathsf{Decrypt}(\mathsf{mpk}, i, \mathsf{sk}_i, S, \mathsf{ct})$:

- If $\mathsf{row}(i) > t_1$, the inner obfuscation in its row is simulated, so it outputs $\perp$ and $f_i(x_{\mathsf{row}(i)}) = 1$.
- If $\mathsf{row}(i) = t_1$, then $\mathsf{col}(i) > t_2$. The corresponding IBE ciphertext encrypts 0^λ, so $C_{\mathsf{inner}}[\mathsf{sk}_{\mathsf{SKE}}]$ outputs 0^λ. Since $\alpha_{\mathsf{inner}} \neq 0^\lambda$, the inner obfuscation outputs $\perp$ and again $f_i(x_{\mathsf{row}(i)}) = 1$.
- In either case, correctness of ABE gives $\mathsf{ABE.Decrypt}(\mathsf{mpk}_{\mathsf{ABE}}, \mathsf{sk}_{f_i}, f_i, \mathsf{ct}_{\mathsf{ABE}}^{(\mathsf{row}(i))}, x_{\mathsf{row}(i)}) = \alpha_{\mathsf{outer}}^{(\mathsf{row}(i))}$. Correctness of TM-LO then gives the output m.

A union bound over the obfuscations used in this evaluation, together with the probability $2^{-\lambda}$ of a zero inner lock, bounds the failure probability by $\mathsf{negl}(\lambda)$, as desired. □

Theorem 3.3 (Succinctness). *Suppose* Π_{ABE} *has succinctness matching Fact 2.4 and* $\Pi_{\mathsf{LO}}, \Pi_{\mathsf{TMLO}}$ *have succinctness matching Fact 2.6. Then, Construction 3.1 is succinct.*

Proof. We show that there exists a fixed polynomial p such that

$$|\mathsf{mpk}|, |\mathsf{sk}_i| \le p(\lambda, \log N) \quad \text{and} \quad |\mathsf{ct}| \le \sqrt{N} \cdot p(\lambda, \log N).$$

Each IBE ciphertext has size $\mathsf{poly}(\lambda)$. The inner circuit decrypts and evaluates only one IBE ciphertext, so it also has size $\mathsf{poly}(\lambda, \log N)$.

The predicate f_i checks membership in S and that $r = \mathsf{row}(i)$, and then passes part of $\mathsf{ct}_{\mathsf{aux}}$ to the inner obfuscation. Thus, its gate count and depth d_f are $\mathsf{poly}(\lambda, \log N)$. By Fact 2.4, the ABE public key, policy key, and ciphertext have size $\mathsf{poly}(\lambda, \log N)$.

By using a sufficient uniform parallel implementation of ABE decryption, M_{outer} has circuit depth $d_{\mathsf{outer}} = \mathsf{poly}(\lambda, \log N)$, a uniform description of length $\mathsf{poly}(\lambda, \log N)$, and time bound $\mathsf{poly}(\lambda, N)$ on all fixed-length inputs. The TMLO bound in Fact 2.6 gives outer obfuscations of size $\mathsf{poly}(\lambda, \log N)$.

There are $\sqrt{N}$ SKE ciphertexts and $\sqrt{N}$ outer obfuscations. The public key contains one IBE master public key and one ABE master public key, and each user key contains one IBE identity key and one ABE policy key. Summing up the sizes proves the theorem. □

Theorem 3.4 (Message Hiding). *Construction 3.1 satisfies message hiding.*

Proof. At cutoff N, all outer obfuscations are simulated. Their distribution and the distribution of the auxiliary SKE ciphertexts are independent of the message. This remains true given every user secret key and for a recipient set chosen after setup, as desired. □

Theorem 3.5 (Set-Selective Index Hiding). *Suppose Π_{IBE} is semantically secure, Π_{SKE} is CPA-secure, Π_{ABE} is attribute-selectively secure, and $\Pi_{\mathsf{LO}}, \Pi_{\mathsf{TMLO}}$ satisfy simulation security. Then, Construction 3.1 satisfies set-selective index hiding.*

Proof. We divide adversaries into two categories: ones that choose an index t in the middle of a row, and ones that choose t at a row boundary (i.e. t is a multiple of n). We call these intra-row adversaries and inter-row adversaries respectively. We will give a proof of selective index hiding for each type individually.

Lemma 3.6 (Intra-Row Index Hiding). *Suppose Π_{IBE} is semantically-secure. Then, for all efficient adversaries $\mathcal{A}$ that send indices $t = (t_1 - 1)n + t_2$ where $t_1 \in [n]$ and $t_2 \in [n-1]$, Construction 3.1 satisfies set-selective index hiding.*

Proof. Let $\mathcal{A}$ be an efficient intra-row adversary for the set-selective index hiding game with non-negligible advantage ε. By definition, cutoffs $t-1$ and t have representations $(t_1, t_2 - 1)$ and (t_1, t_2), respectively. If $t \notin S$, the distributions are identical, so we are done. Assuming $\mathcal{A}$ chooses $t \in S$, we construct an adversary $\mathcal{B}$ that breaks security of Π_{IBE} as follows:

1. Algorithm $\mathcal{B}$ starts running algorithm $\mathcal{A}$ to get 1^N, $S \subseteq [N]$, and $t = (t_1 - 1)n + t_2 \in S$. Algorithm $\mathcal{B}$ samples $(\mathsf{mpk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and gets $\mathsf{mpk}_{\mathsf{IBE}}$ from the IBE challenger. For $i \in [N] \setminus \{t\}$, algorithm $\mathcal{B}$ queries the IBE challenger on identity $\mathsf{id} = i$ for $\mathsf{sk}^{(i)}_{\mathsf{IBE}}$, sets $f_i = f[i, \mathsf{sk}^{(i)}_{\mathsf{IBE}}]$, and samples $\mathsf{sk}_{f_i} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}_{\mathsf{ABE}}, f_i)$. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with $\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{mpk}_{\mathsf{ABE}})$ and $\mathsf{sk}_i = (\mathsf{sk}^{(i)}_{\mathsf{IBE}}, \mathsf{sk}_{f_i})$ for $i \in [N] \setminus \{t\}$.

2. When $\mathcal{A}$ outputs the challenge message $m \in \mathcal{M}_\lambda$, algorithm $\mathcal{B}$ constructs the challenge ciphertext as follows:

   (a) Sample $\alpha_{\mathsf{inner}} \xleftarrow{R} \{0,1\}^\lambda$ and $\mathsf{sk}_{\mathsf{SKE}} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$. For each $c \in [n]$ such that $c \neq t_2$, generate an IBE ciphertext for identity $j = (t_1 - 1)n + c$ as follows:

   $$\mathsf{ct}^{(c)}_{\mathsf{IBE}} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{mpk}_{\mathsf{IBE}}, j, z_c) \text{ where } z_c = \begin{cases} \alpha_{\mathsf{inner}} & j \in S,\ c < t_2 \\ 0^\lambda & \text{otherwise.} \end{cases}$$

   Generate $\mathsf{ct}^{(t_2)}_{\mathsf{IBE}}$ by making a challenge query on identity $\mathsf{id}^* = t$ with messages $0^\lambda, \alpha_{\mathsf{inner}}$. For $c \in [n]$, compute $\mathsf{ct}^{(c)}_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Encrypt}(\mathsf{sk}_{\mathsf{SKE}}, \mathsf{ct}^{(c)}_{\mathsf{IBE}})$. Set $\mathsf{ct}_{\mathsf{aux}} = (\mathsf{ct}^{(1)}_{\mathsf{SKE}}, \ldots, \mathsf{ct}^{(n)}_{\mathsf{SKE}})$.

   (b) For each $r \geq t_1$, generate the inner obfuscations by

   $$\widehat{C}^{(r)}_{\mathsf{inner}} \leftarrow \begin{cases} \mathsf{LO.Obf}(1^\lambda, C_{\mathsf{inner}}[\mathsf{sk}_{\mathsf{SKE}}], \alpha_{\mathsf{inner}}, 1), & r = t_1, \\ \mathsf{LO.Sim}(1^\lambda, \mathsf{pp}(C_{\mathsf{inner}}), 1), & r > t_1. \end{cases}$$

   (c) For each $r \in [n]$, generate an outer obfuscation $\widehat{M}^{(r)}_{\mathsf{outer}}$ as follows:

   - If $r < t_1$, sample $\widehat{M}^{(r)}_{\mathsf{outer}} \leftarrow \mathsf{TMLO.Sim}(1^\lambda, \mathsf{pp}(M_{\mathsf{outer}}), 1^\ell)$.

- Otherwise, sample $\alpha_{\mathsf{outer}}^{(r)} \xleftarrow{R} \{0,1\}^\lambda$, set $x_r = (S, r, \widehat{C}_{\mathsf{inner}}^{(r)}, \mathsf{ct}_{\mathsf{aux}})$, and compute

$$\mathsf{ct}_{\mathsf{ABE}}^{(r)} \leftarrow \mathsf{ABE.Encrypt}(\mathsf{mpk}_{\mathsf{ABE}}, x_r, \alpha_{\mathsf{outer}}^{(r)}),$$
$$\widehat{M}_{\mathsf{outer}}^{(r)} \leftarrow \mathsf{TMLO.Obf}\big(1^\lambda, M_{\mathsf{outer}}[r, \widehat{C}_{\mathsf{inner}}^{(r)}, \mathsf{ct}_{\mathsf{ABE}}^{(r)}], \alpha_{\mathsf{outer}}^{(r)}, m\big).$$

  Algorithm $\mathcal{B}$ gives the challenge ciphertext $\mathsf{ct} = (\mathsf{ct}_{\mathsf{aux}}, \widehat{M}_{\mathsf{outer}}^{(1)}, \ldots, \widehat{M}_{\mathsf{outer}}^{(n)})$ to $\mathcal{A}$.

3. At the end of the game, adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which algorithm $\mathcal{B}$ also outputs.

Clearly algorithm $\mathcal{B}$ is efficient if algorithm $\mathcal{A}$ is, and moreover algorithm $\mathcal{B}$ is admissible in the IBE security game since the challenge identity never has its key queried. If the IBE challenger outputs an IBE encryption of 0^λ, algorithm $\mathcal{B}$ perfectly simulates an execution of the index hiding game for challenge bit $b = 1$. If the semantic security challenger outputs an IBE encryption of α_{inner}, algorithm $\mathcal{B}$ perfectly simulates an execution of the index hiding game for challenge bit $b = 0$. Thus, algorithm $\mathcal{B}$ breaks semantic security with advantage ε.

□

Lemma 3.7 (Inter-Row Index Hiding). *Suppose Π_{ABE} is attribute-selectively secure, Π_{SKE} is CPA-secure, and $\Pi_{\mathsf{LO}}, \Pi_{\mathsf{TMLO}}$ are simulation secure. Suppose also that Π_{IBE}, Π_{LO}, and Π_{SKE} are correct. Then, for all efficient adversaries $\mathcal{A}$ that send indices $t = t_1 n$ where $t_1 \in [n]$, Construction 3.1 satisfies set-selective index hiding.*

Proof. Let $\mathcal{A}$ be an efficient inter-row adversary for the set-selective index hiding game with non-negligible advantage ε. For an index $t \in [N]$ and set $S \subseteq [N]$, we define the set $C_{t,S}$ as

$$C_{t,S} = \begin{cases} [N] \setminus \{t\}, & t \in S, \\ [N], & t \notin S. \end{cases}$$

By definition, cutoffs $t-1$ and t have representations $(t_1, n-1)$ and $(t_1+1, 0)$, respectively. We define the following hybrids:

- Hyb_0: This is the set-selective index hiding game with bit $b = 1$ which we recall below:

  1. At the beginning of the game, the adversary $\mathcal{A}$ outputs a user count 1^N, an index $t \in [N]$ that can be parsed as $t = t_1 n$ where $t_1 \in [n]$, and a set $S \subseteq [N]$. The challenger samples

$$(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda) \quad \text{and} \quad (\mathsf{mpk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.Setup}(1^\lambda).$$

     It sets $\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{mpk}_{\mathsf{ABE}})$. For $i \in [N]$, the challenger samples

$$\mathsf{sk}_{\mathsf{IBE}}^{(i)} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}_{\mathsf{IBE}}, i) \quad \text{and} \quad \mathsf{sk}_{f_i} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}_{\mathsf{ABE}}, f_i)$$

     where $f_i = f[i, \mathsf{sk}_{\mathsf{IBE}}^{(i)}]$ is defined as in Fig. 3. It sets $\mathsf{sk}_i = (\mathsf{sk}_{\mathsf{IBE}}^{(i)}, \mathsf{sk}_{f_i})$. The challenger gives mpk and sk_i for $i \in C_{t,S}$ to $\mathcal{A}$.

  2. When $\mathcal{A}$ outputs the challenge message $m \in \mathcal{M}_\lambda$, the challenger constructs the challenge ciphertext as follows:

(a) Sample $\alpha_{\mathsf{inner}} \xleftarrow{R} \{0,1\}^\lambda$ and $\mathsf{sk}_{\mathsf{SKE}} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$. For each $c \in [n]$, generate an IBE ciphertext for identity $j = (t_1 - 1)n + c$ as follows:

$$\mathsf{ct}_{\mathsf{IBE}}^{(c)} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{mpk}_{\mathsf{IBE}}, j, z_c) \text{ where } z_c = \begin{cases} \alpha_{\mathsf{inner}} & j \in S,\ c < n \\ 0^\lambda & \text{otherwise.} \end{cases}$$

For $c \in [n]$, compute $\mathsf{ct}_{\mathsf{SKE}}^{(c)} \leftarrow \mathsf{SKE.Encrypt}(\mathsf{sk}_{\mathsf{SKE}}, \mathsf{ct}_{\mathsf{IBE}}^{(c)})$. Set $\mathsf{ct}_{\mathsf{aux}} = (\mathsf{ct}_{\mathsf{SKE}}^{(1)}, \ldots, \mathsf{ct}_{\mathsf{SKE}}^{(n)})$.

(b) For each $r \geq t_1$, generate the inner obfuscations by

$$\widehat{C}_{\mathsf{inner}}^{(r)} \leftarrow \begin{cases} \mathsf{LO.Obf}(1^\lambda, C_{\mathsf{inner}}[\mathsf{sk}_{\mathsf{SKE}}], \alpha_{\mathsf{inner}}, 1), & r = t_1, \\ \mathsf{LO.Sim}(1^\lambda, \mathsf{pp}(C_{\mathsf{inner}}), 1), & r > t_1. \end{cases}$$

(c) For each $r \in [n]$, generate an outer obfuscation $\widehat{M}_{\mathsf{outer}}^{(r)}$ as follows:

- If $r < t_1$, sample $\widehat{M}_{\mathsf{outer}}^{(r)} \leftarrow \mathsf{TMLO.Sim}(1^\lambda, \mathsf{pp}(M_{\mathsf{outer}}), 1^\ell)$.
- Otherwise, sample $\alpha_{\mathsf{outer}}^{(r)} \xleftarrow{R} \{0,1\}^\lambda$, set $x_r = (S, r, \widehat{C}_{\mathsf{inner}}^{(r)}, \mathsf{ct}_{\mathsf{aux}})$, and compute

$$\begin{aligned} \mathsf{ct}_{\mathsf{ABE}}^{(r)} &\leftarrow \mathsf{ABE.Encrypt}(\mathsf{mpk}_{\mathsf{ABE}}, x_r, \alpha_{\mathsf{outer}}^{(r)}), \\ \widehat{M}_{\mathsf{outer}}^{(r)} &\leftarrow \mathsf{TMLO.Obf}\big(1^\lambda, M_{\mathsf{outer}}[r, \widehat{C}_{\mathsf{inner}}^{(r)}, \mathsf{ct}_{\mathsf{ABE}}^{(r)}], \alpha_{\mathsf{outer}}^{(r)}, m\big). \end{aligned}$$

The challenger gives the challenge ciphertext $\mathsf{ct} = (\mathsf{ct}_{\mathsf{aux}}, \widehat{M}_{\mathsf{outer}}^{(1)}, \ldots, \widehat{M}_{\mathsf{outer}}^{(n)})$ to $\mathcal{A}$.

3. At the end of the game, adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

- Hyb_1: Same as Hyb_0, except the challenger computes $\mathsf{ct}_{\mathsf{ABE}}^{(t_1)} \leftarrow \mathsf{ABE.Encrypt}(\mathsf{mpk}_{\mathsf{ABE}}, x_{t_1}, 0^\lambda)$ when constructing the challenge ciphertext.
- Hyb_2: Same as Hyb_1, except the challenger computes $\widehat{M}_{\mathsf{outer}}^{(t_1)} \leftarrow \mathsf{TMLO.Sim}(1^\lambda, \mathsf{pp}(M_{\mathsf{outer}}), 1^\ell)$ without constructing $\widehat{C}_{\mathsf{inner}}^{(t_1)}$ or $\mathsf{ct}_{\mathsf{ABE}}^{(t_1)}$ when constructing the challenge ciphertext.
- Hyb_3: Same as Hyb_2, except that, for each $c \in [n]$, the challenger sets $j = t_1 n + c$ and computes $\mathsf{ct}_{\mathsf{IBE}}^{(c)} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{mpk}_{\mathsf{IBE}}, j, 0^\lambda)$ when constructing the challenge ciphertext.
- Hyb_4: Same as Hyb_3, except that, if $t_1 < n$, the challenger computes

$$\widehat{C}_{\mathsf{inner}}^{(t_1+1)} \leftarrow \mathsf{LO.Obf}(1^\lambda, C_{\mathsf{inner}}[\mathsf{sk}_{\mathsf{SKE}}], \alpha_{\mathsf{inner}}, 1)$$

when constructing the challenge ciphertext.

We write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the random variable corresponding to the output of an execution of hybrid Hyb_i with adversary $\mathcal{A}$ (and an implicit security parameter λ). We now bound the difference between the output distributions of each adjacent pair of hybrid experiments.

Claim 3.8. *Suppose Π_{ABE} is attribute-selectively secure and $\Pi_{\mathsf{IBE}}, \Pi_{\mathsf{LO}}, \Pi_{\mathsf{SKE}}$ are correct. Then, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. Suppose $|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| = \delta$ for some non-negligible δ. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ that breaks attribute-selective security of Π_{ABE}:

1. Algorithm $\mathcal{B}$ starts running algorithm $\mathcal{A}$ to get 1^N, $S \subseteq [N]$, and $t = t_1 n$ for $t_1 \in [n]$. Algorithm $\mathcal{B}$ samples $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, $\alpha_{\mathsf{inner}} \leftarrow \{0,1\}^\lambda$, and $\mathsf{sk}_{\mathsf{SKE}} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$. For each $c \in [n]$ algorithm $\mathcal{B}$ sets $j = (t_1 - 1)n + c$ and computes

$$\mathsf{ct}_{\mathsf{IBE}}^{(c)} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{mpk}_{\mathsf{IBE}}, j, z_c) \text{ where } z_c = \begin{cases} \alpha_{\mathsf{inner}} & j \in S,\ c < n \\ 0^\lambda & \text{otherwise.} \end{cases}$$

   For $c \in [n]$, algorithm $\mathcal{B}$ computes $\mathsf{ct}_{\mathsf{SKE}}^{(c)} \leftarrow \mathsf{SKE.Encrypt}(\mathsf{sk}_{\mathsf{SKE}}, \mathsf{ct}_{\mathsf{IBE}}^{(c)})$. Algorithm $\mathcal{B}$ then sets $\mathsf{ct}_{\mathsf{aux}} = (\mathsf{ct}_{\mathsf{SKE}}^{(1)}, \ldots, \mathsf{ct}_{\mathsf{SKE}}^{(n)})$. For $r \geq t_1$, algorithm $\mathcal{B}$ computes

$$\widehat{C}_{\mathsf{inner}}^{(r)} \leftarrow \begin{cases} \mathsf{LO.Obf}(1^\lambda, C_{\mathsf{inner}}[\mathsf{sk}_{\mathsf{SKE}}], \alpha_{\mathsf{inner}}, 1) & r = t_1, \\ \mathsf{LO.Sim}(1^\lambda, \mathsf{pp}(C_{\mathsf{inner}}), 1) & r > t_1. \end{cases}$$

   Algorithm $\mathcal{B}$ then sets $x_r = (S, r, \widehat{C}_{\mathsf{inner}}^{(r)}, \mathsf{ct}_{\mathsf{aux}})$ for $r \geq t_1$ and gives the attribute x_{t_1} to the ABE challenger to get $\mathsf{mpk}_{\mathsf{ABE}}$. For $i \in C_{t,S}$, algorithm $\mathcal{B}$ samples $\mathsf{sk}_{\mathsf{IBE}}^{(i)} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}_{\mathsf{IBE}}, i)$, sets $f_i = f[i, \mathsf{sk}_{\mathsf{IBE}}^{(i)}]$, aborts with output 0 if $f_i(x_{t_1}) = 1$, queries the ABE challenger on policy f_i to get sk_{f_i}, and sets $\mathsf{sk}_i = (\mathsf{sk}_{\mathsf{IBE}}^{(i)}, \mathsf{sk}_{f_i})$. Algorithm $\mathcal{B}$ gives $\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{mpk}_{\mathsf{ABE}})$ and sk_i for $i \in C_{t,S}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs $m \in \mathcal{M}_\lambda$, algorithm $\mathcal{B}$ computes $\widehat{M}_{\mathsf{outer}}^{(r)} \leftarrow \mathsf{TMLO.Sim}(1^\lambda, \mathsf{pp}(M_{\mathsf{outer}}), 1^\ell)$ for $r < t_1$. For $r > t_1$, algorithm $\mathcal{B}$ samples $\alpha_{\mathsf{outer}}^{(r)} \leftarrow \{0,1\}^\lambda$ and computes

$$\begin{aligned} \mathsf{ct}_{\mathsf{ABE}}^{(r)} &\leftarrow \mathsf{ABE.Encrypt}(\mathsf{mpk}_{\mathsf{ABE}}, x_r, \alpha_{\mathsf{outer}}^{(r)}), \\ \widehat{M}_{\mathsf{outer}}^{(r)} &\leftarrow \mathsf{TMLO.Obf}\big(1^\lambda, M_{\mathsf{outer}}[r, \widehat{C}_{\mathsf{inner}}^{(r)}, \mathsf{ct}_{\mathsf{ABE}}^{(r)}], \alpha_{\mathsf{outer}}^{(r)}, m\big). \end{aligned}$$

   Algorithm $\mathcal{B}$ samples $\alpha_{\mathsf{outer}}^{(t_1)} \leftarrow \{0,1\}^\lambda$, submits messages $0^\lambda, \alpha_{\mathsf{outer}}^{(t_1)}$ to the ABE challenger to get $\mathsf{ct}_{\mathsf{ABE}}^{(t_1)}$, and computes $\widehat{M}_{\mathsf{outer}}^{(t_1)} \leftarrow \mathsf{TMLO.Obf}\big(1^\lambda, M_{\mathsf{outer}}[t_1, \widehat{C}_{\mathsf{inner}}^{(t_1)}, \mathsf{ct}_{\mathsf{ABE}}^{(t_1)}], \alpha_{\mathsf{outer}}^{(t_1)}, m\big)$. Algorithm $\mathcal{B}$ gives the challenge ciphertext $\mathsf{ct} = (\mathsf{ct}_{\mathsf{aux}}, \widehat{M}_{\mathsf{outer}}^{(1)}, \ldots, \widehat{M}_{\mathsf{outer}}^{(n)})$ to $\mathcal{A}$.

3. At the end of the game, adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which algorithm $\mathcal{B}$ also outputs.

Clearly algorithm $\mathcal{B}$ is efficient and admissible. We argue that with overwhelming probability, algorithm $\mathcal{B}$ does not abort:

- By construction of f_i, for $i \notin S$ or $\mathsf{row}(i) \neq t_1$ we have $f_i(x_{t_1}) = 0$.
- For $i \in S$ such that $t_1 = \mathsf{row}(i)$ and $i < t$, by correctness of $\Pi_{\mathsf{IBE}}, \Pi_{\mathsf{LO}}, \Pi_{\mathsf{SKE}}$, with overwhelming probability we have that

$$1 \leftarrow \mathsf{LO.Eval}\big(\widehat{C}_{\mathsf{inner}}, (i, \mathsf{sk}_{\mathsf{IBE}}^{(i)}, \mathsf{ct}_{\mathsf{SKE}}^{(\mathsf{col}(i))})\big),$$

  and thus $f_i(x_{t_1}) = 0$. Note that f_t is not queried if $t \in S$ by definition of $C_{t,S}$.

If the ABE challenger outputs an encryption of $\alpha_{\mathsf{outer}}^{(t_1)}$, algorithm $\mathcal{B}$ simulates an execution of Hyb_0 with overwhelming probability. If the ABE challenger outputs an encryption of 0^λ, algorithm $\mathcal{B}$ simulates an execution of Hyb_1 with overwhelming probability. Thus, algorithm $\mathcal{B}$ breaks attribute-selective security with advantage at least $\delta - \mathsf{negl}(\lambda)$, which is non-negligible. □

Claim 3.9. *Suppose* Π_{TMLO} *satisfies simulation security. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. Suppose $|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]| = \delta$ for some non-negligible δ. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ that breaks simulation security of Π_{TMLO}:

1. Algorithm $\mathcal{B}$ runs step 1 with $\mathcal{A}$ exactly as in Hyb_1.
2. When $\mathcal{A}$ outputs the challenge message $m \in \mathcal{M}_\lambda$, algorithm $\mathcal{B}$ constructs the challenge ciphertext exactly as in Hyb_1, except it does not sample $\alpha^{(t_1)}_{\mathsf{outer}}$ and instead constructs $\widehat{M}^{(t_1)}_{\mathsf{outer}}$ by submitting $M_{\mathsf{outer}}[t_1, \widehat{C}^{(t_1)}_{\mathsf{inner}}, \mathsf{ct}^{(t_1)}_{\mathsf{ABE}}]$ and m to the simulation security challenger.
3. At the end of the game, algorithm $\mathcal{B}$ outputs what algorithm $\mathcal{A}$ outputs.

Clearly algorithm $\mathcal{B}$ is efficient if algorithm $\mathcal{A}$ is. If the simulation security challenger outputs a simulated obfuscation, algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_2. If the simulation security challenger outputs a real obfuscation, algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_1. Thus, algorithm $\mathcal{B}$ breaks simulation security with advantage δ. □

Claim 3.10. *Suppose* Π_{SKE} *satisfies CPA security. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. Suppose $|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]| = \delta$ for some non-negligible δ. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ that breaks CPA security of Π_{SKE}:

1. Algorithm $\mathcal{B}$ runs step 1 with $\mathcal{A}$ exactly as in Hyb_2.
2. When $\mathcal{A}$ outputs the challenge message $m \in \mathcal{M}_\lambda$, algorithm $\mathcal{B}$ constructs the challenge ciphertext exactly as in Hyb_2, except for $c \in [n]$ algorithm $\mathcal{B}$ computes

$$\mathsf{ct}^{(c)}_{\mathsf{IBE}} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{mpk}_{\mathsf{IBE}}, (t_1 - 1)n + c, z_c)$$
$$\left(\mathsf{ct}^{(c)}_{\mathsf{IBE}}\right)' \leftarrow \mathsf{IBE.Encrypt}(\mathsf{mpk}_{\mathsf{IBE}}, t_1 n + c, 0^\lambda),$$

   where z_c is α_{inner} if $(t_1 - 1)n + c \in S, c < n$ and 0^λ otherwise. For $c \in [n]$, algorithm $\mathcal{B}$ submits messages $\mathsf{ct}^{(c)}_{\mathsf{IBE}}, \left(\mathsf{ct}^{(c)}_{\mathsf{IBE}}\right)'$ to the CPA challenger to get $\mathsf{ct}^{(c)}_{\mathsf{SKE}}$.
3. At the end of the game, algorithm $\mathcal{B}$ outputs what algorithm $\mathcal{A}$ outputs.

Clearly algorithm $\mathcal{B}$ is efficient if algorithm $\mathcal{A}$ is. If the CPA security challenger encrypts $\mathsf{ct}^{(c)}_{\mathsf{IBE}}$ in each ciphertext, algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_2. If the CPA security challenger encrypts $\left(\mathsf{ct}^{(c)}_{\mathsf{IBE}}\right)'$ in each ciphertext, algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_3. Thus, algorithm $\mathcal{B}$ breaks CPA security with advantage δ. □

Claim 3.11. *Suppose* Π_{LO} *satisfies simulation security. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_4(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. Suppose $|\Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_4(\mathcal{A}) = 1]| = \delta$ for some non-negligible δ and $\mathcal{A}$ outputs $t = t_1 n$ such that $t_1 < n$. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ that breaks simulation security of Π_{LO}:

1. Algorithm $\mathcal{B}$ runs step 1 with $\mathcal{A}$ exactly as in Hyb_3.
2. When $\mathcal{A}$ outputs the challenge message $m \in \mathcal{M}_\lambda$, algorithm $\mathcal{B}$ constructs the challenge ciphertext exactly as in Hyb_3, except it does not sample α_{inner}, and generates $\widehat{C}_{\mathsf{inner}}^{(t_1+1)}$ by querying the simulation security challenger with program $C_{\mathsf{inner}}[\mathsf{sk}_{\mathsf{SKE}}]$ and message 1.
3. At the end of the game, algorithm $\mathcal{B}$ outputs what algorithm $\mathcal{A}$ outputs.

Clearly algorithm $\mathcal{B}$ is efficient if algorithm $\mathcal{A}$ is. If the simulation security challenger outputs a simulated obfuscation, algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_3. If the simulation security challenger outputs a real obfuscation, algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_4. Thus, algorithm $\mathcal{B}$ breaks simulation security with advantage δ. □

Combining Claims 3.8 to 3.11 proves the lemma by a standard hybrid argument. □

Combining Lemmas 3.6 and 3.7 proves the theorem. □

Combining Theorems 3.2 to 3.5 with Facts 2.4, 2.6 and 2.9 yields the following instantiation:

Corollary 3.12 (Broadcast and Trace from Decomposed LWE)**.** *Let λ be a security parameter and $N = N(\lambda)$ be any polynomial. Under the polynomial hardness of the decomposed LWE assumption with a sub-exponential modulus-to-noise ratio, there exists a collusion-resistant broadcast and trace scheme satisfying set-selective traceability and semantic security. The master public key and user secret key size is* $\mathrm{poly}(\lambda, \log N)$*, and the ciphertext size is* $\sqrt{N} \cdot \mathrm{poly}(\lambda, \log N)$*. The scheme has a public tracing procedure.*

Remark 3.13 (Smaller Ciphertexts)**.** We believe it is possible to obtain ciphertext size N^ϵ for any $\epsilon > 0$ by using the recursive approach detailed in [NWW26] on Construction 3.1. Specifically, both Construction 3.1 and the cutoff-iO scheme in [NWW26] reuse a trivial base scheme to get $\sqrt{N}$-succinctness, which can be improved to $N^{1/3}$-succinctness by reusing the $\sqrt{N}$-succinct scheme; this process can be repeated a constant number of times to get N^ϵ-succinctness. However, for the sake of redundancy and ease of exposition, we do not explicitly give such a construction in this work.

Acknowledgments

Research supported by the European Research Council through an ERC Starting Grant (GA 101077455, ObfusQation).

Statement on AI use. The author organically generated the approach to solve this problem and used GPT-6 to check the details. GPT-6 was used to generate an initial draft of the construction and a sketch of the proof, which were then simplified and formalized by the author. The author used GPT-6 for copy-editing and proofreading. The author verified the correctness and originality of all content in the paper.

References

[AAC+26] Damiano Abram, Gal Arnon, Valerio Cini, Paul Lou, Giulio Malavolta, and Lawrence Roy. Worst-case hardness for succinct and decomposed learning with errors. *IACR Cryptol. ePrint Arch.*, 2026. Available at https://eprint.iacr.org/2026/2443.pdf.

[AKYY23] Shweta Agrawal, Simran Kumari, Anshu Yadav, and Shota Yamada. Broadcast, trace and revoke with optimal parameters from polynomial hardness. In *EUROCRYPT*, 2023.

[AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Key-homomorphic computations for RAM: Fully succinct randomised encodings and more. In *CRYPTO*, 2025.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, 2001.

[BW06] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *ACM CCS*, 2006.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, 1994.

[Coc01] Clifford C. Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding*, 2001.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, 1993.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, 2017.

[GLW14] Craig Gentry, Allison Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In *CRYPTO*, 2014.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, 2006.

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

[NNL01] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, 2001.

[NP00] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography*, 2000.

[NWW26] Shafik Nassar, Brent Waters, and David J. Wu. Obfuscating non-evasive programs: Making witness encryption positional. *IACR Cryptol. ePrint Arch.*, 2026. Available at https://eprint.iacr.org/2026/1932.pdf.

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In *EUROCRYPT*, 2016.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, 1984.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.

[Wee25] Hoeteck Wee. Almost optimal KP and CP-ABE for circuits from succinct LWE. In *EUROCRYPT*, 2025.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. In *FOCS*, 2017.