Ad Hoc (Decentralized) Broadcast, Trace, and Revoke

罗雈 (Ji Luo) ☞

Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, USA
luoji@cs.washington.edu

September 2022

Abstract

Traitor tracing schemes [Chor–Fiat–Naor, Crypto ’94] help content distributors fight against piracy and are defined with the content distributor as a trusted authority having access to the secret keys of all users. While the traditional model caters well to its original motivation, its centralized nature makes it unsuitable for many scenarios. For usage among mutually untrusted parties, a notion of ad hoc traitor tracing (naturally with the capability of broadcast and revocation) is proposed and studied in this work. Such a scheme allows users in the system to generate their own public/secret key pairs, without trusting any other entity. To encrypt, a list of public keys is used to identify the set of recipients, and decryption is possible with a secret key for any of the public keys in the list. In addition, there is a tracing algorithm that given a list of recipients’ public keys and a pirate decoder capable of decrypting ciphertexts encrypted to them, identifies at least one recipient whose secret key must have been used to construct the said decoder.

Two constructions are presented. The first is based on obfuscation and has constant-size ciphertext, yet its decryption time is linear in the number of recipients. The second is a generic transformation that reduces decryption time at the cost of increased ciphertext size. A lower bound on the trade-off between ciphertext size and decryption time is shown, indicating that the two constructions achieve all possible optimal trade-offs, i.e., fully demonstrate the Pareto front of efficiency. The lower bound also applies to general attribute-based encryption and may be of independent interest.

Keywords. traitor tracing, obfuscation, attribute-based encryption.

This research is “open-thoughts”. See https://github.com/GeeLaw/ahbtr.
## Contents

1 Introduction .................................................. 1
   1.1 Overview .................................................. 2

2 Preliminaries ................................................ 6

3 Ad Hoc Broadcast, Trace, and Revoke ....................... 10
   3.1 Simplified Security Notions ............................... 11

4 Ad Hoc Private Linear Broadcast Encryption ............... 13
   4.1 Construction ............................................ 14
   4.2 Message-Hiding Property ................................. 17
   4.3 Index-Hiding Property .................................. 17

5 AH-BTR from AH-PLBE ........................................ 19

6 Trading Ciphertext Size for Decryption Time in AH-BTR ...... 22

7 Lower Bound on Ciphertext Size and Decryption Time ........ 24

References ...................................................... 28
1 Introduction

Traitor tracing schemes [CFN94] enable content distributors to fight against piracy. A content distributor such as a media streaming service can generate a public key and many different secret keys for individual subscribers, all of which can decrypt the ciphertexts created using the public key. Given a pirate decoder capable of decrypting, which could have been created from the secret keys of multiple subscribers, the tracing algorithm can find at least one subscriber (a traitor) whose key was used to create the said decoder. A long line of subsequent works [BSW06,BW06,BN08,BZ14,NWZ16,GKW18,GKW19,GQWW19,Zha20a,Zha21] proposed the different security definitions, extended the functionality, and presented new constructions.

While the traditional model caters well to the needs of content distributors, its centralized nature makes it unsuitable for many scenarios, e.g., when a group of individuals want to communicate amongst themselves and trace traitors who provide decoders to outsiders. (See [Zha21] for a more concrete example.) This motivation naturally calls for a decentralized notion of traitor tracing, termed ad hoc traitor tracing in this work.

The first question is thus naturally the following:

What is the right notion of a secure ad hoc traitor tracing scheme?

Having formalized its syntax and security, we study its constructions:

How can such a scheme be constructed, from what assumptions and with what efficiency?

Efficiency improvement never ends until we reach the optimum, for which it is necessary to understand where the limit stands:

What bounds are there on the efficiency of such schemes?

Our Contributions. We provide answers to all three questions:

• Conceptually, we pose the question of ad hoc traitor tracing, develop from the ideas thereof, and eventually reach the definitions for ad hoc broadcast, trace, and revoke (AH-BTR). We prove the relation among the security notions considered in this work.

• Construction-wise, we present secure AH-BTR schemes based on functional encryption for general circuits [BSW11]. With polynomial factors in the security parameter ignored, they achieve

\[ T_{\text{Enc}} = O(N), \]
\[ |\text{ct}| = O(N^{1-\gamma}), \]
\[ T_{\text{Dec}} = O(N^\gamma), \]

for any constant \(0 \leq \gamma \leq 1\).

• Questing for the ultimate efficiency, we prove that for all secure AH-BTR,

\[ |\text{ct}| \cdot T_{\text{Dec}} = \Omega(N), \]
so our schemes offer all possible optimal trade-offs between |ct| and $T_{\text{Dec}}$, fully
demonstrating the Pareto front of AH-BTR efficiency. Better yet, the bound holds for a
restricted kind of weakly secure broadcast encryption [FN94], which is a
specific case of attribute-based encryption [SW05,GPSW06], thus also shedding new
insights into the efficiency of ABE and BE.\footnote{Previous works [BC95,LS98,KYDB98,AK08,KY09,GKW15,DLY21] show a few
efficiency lower bounds related to ABE and BE. Yet they only apply to information-theoretically
secure primitives and even specific construction techniques. Moreover, all of them are space (ciphertext or secret key size, or their trade-off) lower bounds. Indeed, based on obfuscation [BWZ14] or both LWE and pairing [AY20], BE with
|ct|, |sk| = O(1) can be constructed. Our result is the first time-space lower bound that applies to any
computationally secure broadcast encryption scheme.}

A final addition is that our scheme is compatible with the existing public-key encryption
schemes, i.e., the keys of such a scheme can be those of any secure public-key
encryption, and there is no need to regenerate keys to take advantage of our scheme.

**Open Questions.** The tracing model in this work is black-box and classical, and recent
works [Zha21,Zha20c] have studied white-box or quantum traitor tracing. Conceptually,
it is interesting to understand the ad hoc versions of those tracing models.

Another question for future investigation is whether (weakened versions of) AH-BTR
can be constructed from more lightweight assumptions such as factoring-related, group-based,
or lattice-based assumptions. Potential relaxations include making the scheme bounded,\footnote{A maximum of number of recipients per ciphertext is set when generating a key pair, and only
“compatible” public keys can be used to form a recipient set.} settling for static security, considering security against bounded collusion,
and only achieving threshold tracing [NP98].

### 1.1 Overview

**Developing Definitions.** We start with the first principles of ad hoc traitor tracing.
Syntactically, there should be a key generation algorithm that is run by each user of
the system.\footnote{We aim for a scheme without any trusted party, so there should be no global set-up.} To encrypt, a list of public keys is used to identify the set of recipients. Decryption should only require one secret key from the list of public keys. In addition, the decryptor gets random access to all the recipients’ public keys as well as the
ciphertext. The choice to give random access to these inputs is based on performance
concerns, as the decryptor might not have to read all of the public keys or the ciphertext.

It should be clear that such a scheme would automatically have the functionality of
broadcast encryption [FN94].\footnote{Decentralized versions of broadcast encryption were studied in [PPS12,DPP07] with interactive
management of recipient sets. Ad hoc (threshold) broadcast encryption was studied in [DHMR08,WQZD10]
with constructions for bounded schemes requiring global set-up.} There is no event prior to encryption that “binds” the
system to a specific, fixed set of possible recipients, and the encryptor is free to use
whatever public keys it sees fit. Similarly, the encryptor is free to remove any public key
when it encrypts a second ciphertext, i.e., the scheme automatically enjoys the capability
of revocation. Therefore, the object is named ad hoc broadcast, trace, and revoke (AH-
BTR).

As usual with broadcast encryption, we do not hide the list of recipients. Hiding
the recipients makes ciphertext grow at least linearly with the number of recipients,
diminishing the potential of efficiency. As we shall see, it is possible to construct AH-BTR with short ciphertexts.

Due to the decentralized nature of such systems, an adversary might indistinguishably generate malformed keys, which could potentially evade tracers that only take well-formed keys into account. To make it worse, a malformed key could be used to mount a denial-of-service attack against (other) honest users if it appears in the list of recipients’ public keys during encryption — the encryption algorithm might have been carelessly designed and the presence of certain malformed keys could make it impossible to decrypt for anyone, including the recipients with honestly generated public keys.

In order to protect against such attacks by definition, we require correctness be robust against malformed keys — however, for performance reasons, namely to be able to index into any particular public key in constant time, we reject blatantly malformed keys, e.g., those of incorrect lengths, in the definition of correctness. This restriction does not hamper the usefulness of such a scheme.

As for security, when attacking the traceability of the scheme, the adversary is free to supply an arbitrary list of recipients’ public keys, generated honestly by the challenger or (adversarially) by the adversary, so that the definition covers the scenario when (blatantly or not) malformed keys are present in the list of recipients’ public keys. The tracing algorithm is given oracle access to a stateless decoder. It must not accuse an honest user, defined as one whose public key is generated by the challenger without its secret key revealed to the adversary. It must find a traitor as long as the decoder has sufficient advantage (i.e., succeeds in decrypting with sufficient probability), where traitors are associated with public keys in the recipient list that are either generated by the challenger with their secret keys revealed to the adversary or crafted by the adversary in any manner (e.g., skewed distribution, or even without a well-defined secret key).

Once the issues above are identified and conceptually resolved (as done here), it is straightforward to define AH-BTR analogously to traditional broadcast, revoke, and trace schemes [NP01, NNL01, GQWW19].

Simplifying Security Notions. Traditionally [BSW06], traceability has been defined using one comprehensive interactive experiment, which is complicated to work with. Intuitively, the notion requires that i) a traitor should be found from a decoder with sufficient advantage and ii) no honest user should be identified as a traitor, regardless of the decoder advantage.

We thus define two security notions for AH-BTR capturing the requirements separately. The former is called completeness and the latter is called soundness. Their conjunction is equivalent to traceability. Since only one requirement is considered in each notion, both of them can be vastly simplified and the interactions in those notions are minimal. They are also more convenient for reductionist proofs.

Construction. Our first construction of AH-BTR follows the existing blueprint of traitor tracing schemes from private linear broadcast encryption (PLBE) schemes introduced in [BSW06]. We first define an ad hoc version of PLBE.  

---

5The general transformation [KY01, BSW06] to deal with stateful decoder applies to our definition of AH-BTR, mutatis mutandis.
6While some previous works [BF99, GKW18, Zha20a] separate traceability into multiple notions, those notions still share one single complicated security experiment.
7AH-PLBE can be cast as multi-authority attribute-based encryption [Cha07] for 1-local monotone functions without global set-up.
• Everyone generates their own public and secret key pair \((pk, sk)\).

• Encryption uses a list \(\{pk_j\}_{j \in [N]}\) of \(N\) public keys of the recipients as well as a cut-off index \(0 \leq i_\perp \leq N\).

• Decryption is possible with \(sk_j\) if \(j > i_\perp\).

There are two security requirements. Message-hiding requires that the plaintext is hidden if \(i_\perp = N\). Index-hiding requires that an adversary without \(sk_j\) for an honest \(pk_j\) cannot distinguish between cut-off index being \((j - 1)\) versus \(j\).

Colloquially, the cut-off index \(i_\perp\) disables \(sk_1, \ldots, sk_{i_\perp}\), and the only way to detect whether an index is disabled is to have control over the corresponding key pair (by knowing \(sk\) or generating a malformed \(pk\)). When \(i_\perp = N\), the plaintext should be hidden since all keys are disabled.

Given an AH-PLBE scheme, an AH-BTR scheme can be constructed by adapting [BSW06]. The AH-BTR inherits key generation and decryption algorithms from AH-PLBE. To perform AH-BTR encryption, simply encrypt using AH-PLBE with \(i_\perp = 0\), disabling no key so that every recipient can decrypt. Given a pirate decoder with advantage at least \(\epsilon\), the tracing algorithm computes its advantages with the cut-off index \(i_\perp\) being \(0, 1, 2, \ldots, N\), and identifies the recipient associated with \(pk_{i^*}\) as a traitor if the advantage changes by \(\Omega(\epsilon/N)\) when \(i_\perp\) increases from \((i^* - 1)\) to \(i^*\).

The message-hiding property translates to completeness, and index-hiding to soundness. It now remains to construct an AH-PLBE.

**Constructing AH-PLBE.** It is folklore that any public-key encryption (PKE) scheme can be used to construct a naïve PLBE by encrypting individually to each recipient. The individual ciphertext that corresponds to a disabled key encrypts garbage instead of the actual plaintext. This scheme is also ad hoc. The downside of it is that the ciphertext is of size \(\Omega(N)\).

Our scheme uses obfuscation to compress the naïve PLBE ciphertext. The ciphertext will contain an obfuscated program, which, when evaluated at \(j \in [N]\), allows us to recover the PKE ciphertext under \(pk_j\). However, the obfuscated program itself cannot simply compute each PKE ciphertext if we want AH-PLBE ciphertexts of size \(o(N)\), as there is no enough space in the program to encode all the public keys that have been independently generated.

Laconic oblivious transfer (OT) [CDG+17] comes to rescue. It allows compressing an arbitrarily long string \(D\) down to a fixed-length hash \(h\) with which one can efficiently perform oblivious transfer. The sender can encrypt messages \(L_0, L_1\) to a hash \(h\) and an index \(m\) into \(D\). The time to encrypt is independent of the length of \(D\). The receiver will be able to obtain \(L_{D[m]}\) by decrypting the laconic OT ciphertext.

During AH-PLBE encryption, we use laconic OT to compress the list of public keys. The obfuscated program in our AH-PLBE ciphertext, when evaluated at \(j \in [N]\), will output i) a garbled circuit whose input (resp. output) is a PKE public key (resp. ciphertext) and ii) a bunch of laconic OT ciphertexts that decrypts to the labels so that the garbled circuit is evaluated at \(pk_j\). Decryption proceeds in the obvious manner.

The obfuscated program size, thus the ciphertext size, can be made constant,\(^8\)

\(^8\)We ignore fixed polynomial factors in the security parameter. The point is that the size does not grow with \(N\) (the number of recipients).
because both the time to garble a PKE encryption circuit and the time of laconic OT encryptions are constant.

**You Can (Not) Optimize.** While our basic construction enjoys constant-size ciphertext, its decryption algorithm runs in time $\Omega(N)$. Concretely, the laconic OT hash is a Merkle tree, and before performing laconic OT decryption, it is necessary to reconstruct the tree as it is not stored in the ciphertext. In contrast, the decryption time of the scheme implied by the naïve PLBE is constant in the RAM model, as it only looks at the relevant piece of the underlying PKE ciphertext.

We can trade ciphertext size for decryption time by using the naïve PLBE on top of our construction. By grouping the recipients into $\Theta(N^{1-\gamma})$ sets of size $\Theta(N^\gamma)$ and using our basic construction over each set, we obtain a scheme with ciphertext size $\Theta(N^{1-\gamma})$ and decryption time $\Theta(N^\gamma)$. This transformation was formalized as the user expansion compiler [Zha20a] in the context of traditional traitor tracing.

All the constructions we now know have $|ct| \cdot T_{\text{Dec}} = \Omega(N)$, where $|ct|$ is the ciphertext length and $T_{\text{Dec}}$ is the decryption time. It turns out that this bound necessarily holds for all secure AH-BTR, and the blame is on the functionality of broadcast encryption (not traitor tracing). Indeed, it is possible to make both $|ct|$ and $T_{\text{Dec}}$ constant in a traditional traitor tracing scheme [BZ14]. In existing broadcast encryption (or revocation) schemes [BGW05,Del07,GW09,BZ14,AAY20,AZY20,BV20] for $N$ users, encrypting to arbitrary subsets of size $S$ or $(N-S)$ makes $|ct| \cdot T_{\text{Dec}} = \Omega(S)$. It is precisely the capability to encrypt to many $\Theta(N)$-subsets among $N$ users that is the deal breaker, as we shall see in the formal proof. Interestingly, the adversary used in the proof is simply the decryption algorithm, so the bound holds as long as the scheme is not blatantly insecure.

We explain the ideas of our proof based on a corollary$^9$ of a result [Unr07] dealing with random oracles in the presence of non-uniform advice. Let $S,T \geq 0$ be such that $ST \ll N$. The corollary says that for any adversary learning any $S$-bit function (advice) of a random string $R \leftarrow \{0,1\}^N$ and additionally (adaptively) querying at most $T$ bits in $R$, it is “indistinguishable” to flip a bit in $R$ at a random location after the advice is computed (using the non-flipped $R$) and before queries are answered, even if the index of the potentially flipped bit is known to the adversary.

Back to AH-BTR. Imagine that there are $2N$ users in the system, associated with key pairs $(pk_{j,s}, sk_{j,s})$ for $j \in [N]$ and $s \in \{0,1\}$. Consider a ciphertext $ct$ encrypting a random plaintext to $\{pk_{j,R(j)}\}_{j \in [N]}$ for a random string $R$ and regard $ct$ as the advice. Let’s try decrypting $ct$ using $sk_{i^*,R(i^*)}$ for a random $i^* \leftarrow [N]$. Each time the AH-BTR decryption algorithm queries $pk_{j}$, we probe $R[j]$ and respond with $pk_{j,R[j]}$. By way of contradiction, suppose $|ct| \cdot T_{\text{Dec}} \ll N$, which would translate to the setting of the corollary as $S = |ct|$, $T \leq T_{\text{Dec}}$, and $ST \ll N$.

By the correctness of AH-BTR, the attempted decryption should successfully recover the plaintext. From the corollary it follows that flipping $R[i^*]$ should also lead to successful recovery. But if $R[i^*]$ is flipped after $ct$ is computed, by the security of AH-BTR, the attempted decryption must fail to recover the plaintext except for negligible probability, yielding a contradiction.

---

$^9$This corollary is also a lower bound of a probabilistic variant of Yao’s box problem [Yao90] (generalized and studied in [CHK22]), on which our proof can be alternatively based.
2 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter, by $\text{poly}(\cdot)$ a polynomial function, and by $\text{negl}(\lambda)$ a negligible function of $\lambda$. Efficient algorithms are probabilistic random-access machines $M^w(x)$ of running time $\text{poly}(|x|, |w|)$. Efficient adversaries (in interactive experiments) are probabilistic Turing machines of (total) running time $\text{poly}(\lambda)$, with or without $\text{poly}(\lambda)$-long advices. (All of the proofs in this work are uniform.) The advantage of $A$ in distinguishing $\text{Exp}_0$ and $\text{Exp}_1$ is $\Pr[\text{Exp}^A_0(1^\lambda) = 1] - \Pr[\text{Exp}^A_1(1^\lambda) = 1]$. We write $\approx_n, \approx_s, \equiv$ for computational indistinguishability, statistical indistinguishability, and identity.

Under the standard assumption that a pseudorandom generator (with polynomial security) exists, we can assume, whenever convenient, that a randomized algorithm uses a uniformly random $\lambda$-bit string as its randomness (without losing polynomial security considered in this work or degrading its efficiency).

For $n, n' \in \mathbb{N}$, we write $[n..n']$ for the set $\{n, \ldots, n'\}$, and $[n]$ for $\{1..n\}$. For a bit-string $D$, we denote by $|D|$ its bit-length, and given an index $m \in [|D|]$, we denote by $D[m]$ the $m$th bit of $D$. For two bit-strings $D, D'$, their concatenation is $D \| D'$. Given a circuit $C : \{0,1\}^{n+M_0} \rightarrow \{0,1\}^{n'}$ and $w \in \{0,1\}^n$, we define $C[w]$ to be $C$ with $w$ hardwired as its first portion of input, so $C[w](x) = C(w||x)$. For an event $X$, its indicator random variable is $1_X$. For events $X, Y$ in the same probability space, “$X$ implies $Y$” means $X \subseteq Y$.

**Garbled Circuits.** The following version of partially hiding garbling [IW14] suffices for the purpose of this work.

**Definition 1** (garbled circuit [Yao86, LP09, BHR12, IW14]). A circuit garbling scheme consists of 2 efficient algorithms:

- $\text{Garble}(1^\lambda, C, w)$ takes as input a circuit $C : \{0,1\}^{n+M_0} \rightarrow \{0,1\}^{n'}$ and some hardwired input $w \in \{0,1\}^n$. It outputs a garbled circuit $\widehat{C}$ and $M_0$ pairs of labels $L_{m_0, b} \in \{0,1\}^\lambda$ for $m_0 \in [M_0]$, $b \in \{0,1\}$.

- $\text{Eval}(1^\lambda, \widehat{C}, x, \{L_{m_0,b}\}_{m_0\in[M_0]})$ takes as input a garbled circuit, a non-hardwired input, and $M_0$ labels. It outputs an $n'$-bit string.

The scheme must be correct, i.e., for all $\lambda \in \mathbb{N}$, $n, M_0, n' \in \mathbb{N}$, $C : \{0,1\}^{n+M_0} \rightarrow \{0,1\}^{n'}$, $w \in \{0,1\}^n$, $x \in \{0,1\}^{M_0}$,

$$\Pr \left[ \left( \widehat{C}, \{L_{m_0,b}\}_{m_0\in[M_0], b\in\{0,1\}} \right) \leftarrow \text{Garble}(1^\lambda, C, w) \right. \left. : \text{Eval}(1^\lambda, \widehat{C}, x, \{L_{m_0,x|m_0}\}_{m_0\in[M_0]}) = C[w](x) \right] = 1.$$

**Definition 2** (garbled circuit security [Yao86, LP09, BHR12, IW14]). Let $(\text{Garble}, \text{Eval})$ be a circuit garbling scheme (Definition 1). A simulator is an efficient algorithm

$$\text{SimGarble}(1^\lambda, C : \{0,1\}^{n+M_0} \rightarrow \{0,1\}^{n'}, x \in \{0,1\}^{M_0}, y \in \{0,1\}^{n'}) \rightarrow (\widehat{C}, \{L_{m_0}\}_{m_0\in[M_0]})$$

taking as input a circuit, a non-hardwired input, and a circuit output, and producing a simulated garbled circuit and $M_0$ simulated labels. The scheme is $w$-hiding (or secure for the purpose of this work) if there exists a simulator $\text{SimGarble}$ such that $\text{Exp}^0_{\text{GC}} \approx \text{Exp}^1_{\text{GC}}$, where $\text{Exp}^b_{\text{GC}}(1^\lambda)$ with adversary $A$ proceeds as follows:
• **Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive a circuit $C : \{0,1\}^{n+M_0} \rightarrow \{0,1\}^n$, a hardwired input $w \in \{0,1\}^n$, and a non-hardwired input $x \in \{0,1\}^{M_0}$ from it. Run

\[
\begin{align*}
&\text{if } b = 0, \quad (\tilde{C}, \{L_{m_0, b} \}_{m_0 \in [M_0], b \in \{0,1\}}) \leftarrow \text{Garble}(1^\lambda, C, w); \\
&\text{if } b = 1, \quad (\tilde{C}, \{L_{m_0, x[m_0]} \}_{m_0 \in [M_0]}) \leftarrow \text{SimGarble}(1^\lambda, C, x, C[w](x));
\end{align*}
\]

and send $(\tilde{C}, \{L_{m_0, x[m_0]} \}_{m_0 \in [M_0]})$ to $\mathcal{A}$.

• **Guess.** $\mathcal{A}$ outputs a bit $b'$, which is the output of the experiment.

**Puncturable Pseudorandom Function.** We rely on PPRF [BW13,KPTZ13,BGI14,SW14].

**Definition 3** (PPRF [BW13,KPTZ13,BGI14,SW14]). A puncturable pseudorandom function (PPRF) family (with key space, domain, and codomain $\{0,1\}^\lambda$) consists of 2 efficient algorithms:

- Puncture$(1^\lambda, k \in \{0,1\}^\lambda, x)$ takes as input a non-punctured key and a point. It outputs a punctured key $\hat{k}_x$.
- Eval$(1^\lambda, k, x \in \{0,1\}^\lambda)$ takes as input a (punctured or non-punctured) key and a point. It is deterministic and outputs a $\lambda$-bit string.

The scheme must be **correct**, i.e., for all $\lambda \in \mathbb{N}$, $x, x' \in \{0,1\}^\lambda$ such that $x \neq x'$,

\[
\Pr\left[ k \xleftarrow{\$} \{0,1\}^\lambda, \hat{k}_x \xleftarrow{\$} \text{Puncture}(1^\lambda, k, x) : \text{Eval}(1^\lambda, k, x') = \text{Eval}(1^\lambda, \hat{k}_x, x') \right] = 1.
\]

**Definition 4** (PPRF security [BW13,KPTZ13,BGI14,SW14]). A PPRF (Puncture, Eval) per Definition 3 is **pseudorandom at the punctured point** (or secure for the purpose of this work) if $\text{Exp}_\mathcal{A}^0 = \text{Exp}_\mathcal{A}^1$, where $\text{Exp}_\mathcal{A}^b(1^\lambda)$ with adversary $\mathcal{A}$ proceeds as follows:

• **Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive from it a point $x \in \{0,1\}^\lambda$. Run

\[
\begin{align*}
&k \xleftarrow{\$} \{0,1\}^\lambda, \quad \hat{k}_x \xleftarrow{\$} \text{Puncture}(1^\lambda, k, x), \quad r_0 \leftarrow \text{Eval}(1^\lambda, k, x), \quad r_1 \xleftarrow{\$} \{0,1\}^\lambda, \\
&\text{and send } (\hat{k}_x, r_b) \text{ to } \mathcal{A}.
\end{align*}
\]

• **Guess.** $\mathcal{A}$ outputs a bit $b'$, which is the output of the experiment.

**Public-Key Encryption.** Our *ad hoc* broadcast, trace, and revoke scheme can be based on any public-key encryption scheme.

**Definition 5** (PKE). A public-key encryption (PKE) scheme (with message space $\{0,1\}^\lambda$ and public key length $M_0(\lambda)$) consists of 3 efficient algorithms:

- **Gen**$(1^\lambda)$ outputs a pair $(pk, sk)$ of public and secret keys with $|pk| = M_0(\lambda)$.
- **Enc**$(1^\lambda, pk, \mu) \in \{0,1\}^\lambda$ takes as input the public key and a message. It outputs a ciphertext $ct$.
- **Dec**$(1^\lambda, sk, ct)$ takes as input the secret key and a ciphertext. It outputs a message.
The scheme must be correct, i.e., for all $\lambda \in \mathbb{N}$, $\mu \in \{0,1\}^d$,
\[
\Pr \left[ (pk, sk) \xleftarrow{\$} \text{Gen}(1^d) \icolon \text{ct} \xleftarrow{\$} \text{Enc}(1^d, pk, \mu) : \text{Dec}(1^d, sk, \text{ct}) = \mu \right] = 1.
\]

**Definition 6** (PKE security). A PKE scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ per Definition 5 is semantically secure for random messages (or secure for the purpose of this work) if
\[
\left\{ (\mu_0, \mu_1, pk, ct_0) \right\} \approx \left\{ (\mu_0, \mu_1, pk, ct_1) \right\},
\]
where $(pk, sk) \xleftarrow{\$} \text{Gen}(1^d)$ and $\mu_b \xleftarrow{\$} \{0,1\}^d$, $\text{ct}_b \xleftarrow{\$} \text{Enc}(1^d, pk, \mu_b)$ for $b \in \{0,1\}$.

**Laconic Oblivious Transfer.** We rely on laconic oblivious transfer [CDG+17].

**Definition 7** (laconic OT [CDG+17]). A laconic oblivious transfer (OT) scheme (with message space $\{0,1\}^d$) consists of 4 efficient algorithms:

- $\text{Gen}(1^d, M \in \mathbb{N})$ takes the database length as input and outputs a hash key $hk$.
- $\text{Hash}(1^d, hk, D \in \{0,1\}^M)$ takes as input a hash key and a database. It is deterministic, runs in time $O(M) \text{poly}(\lambda, \log M)$, and outputs a hash $h$ of length $\text{poly}(\lambda, \log M)$ and a processed database $\hat{D}$.
- $\text{Send}(1^d, hk, h, m \in [M], L_0 \in \{0,1\}^d, L_1 \in \{0,1\}^d)$ takes as input a hash key, a hash, an index, and two labels (messages). It outputs a ciphertext $ct$.
- $\text{Recv}^{\hat{D}}(1^d, hk, h, m \in [M], ct)$ is given random access to a processed database, and takes as input a hash key, a hash, an index, and a ciphertext. It runs in time $\text{poly}(\lambda, \log M)$ and outputs a label (message).

The scheme must be correct, i.e., for all $\lambda, M \in \mathbb{N}$, $D \in \{0,1\}^M$, $m \in [M]$, $L_0, L_1 \in \{0,1\}^d$,
\[
\Pr \left[ \begin{array}{c}
\text{hk} \xleftarrow{\$} \text{Gen}(1^d, M) \\
(h, \hat{D}) \leftarrow \text{Hash}(1^d, hk, D) \\
\text{ct} \xleftarrow{\$} \text{Send}(1^d, hk, h, m, L_0, L_1)
\end{array} \right] = 1.
\]

We only need database-selective security [AL18]. The following indistinguishability-based definition is equivalent to the usual simulation-based formulation.

**Definition 8** (laconic OT security [CDG+17,AL18,KNTY19]). A laconic OT scheme $(\text{Gen}, \text{Hash}, \text{Send}, \text{Recv})$ per Definition 7 is database-selectively sender-private (or secure for the purpose of this work) if $\text{Exp}^0_{\text{LOT}} \approx \text{Exp}^1_{\text{LOT}}$, where $\text{Exp}^h_{\text{LOT}}(1^d)$ with adversary $\mathcal{A}$ proceeds as follows:

- **Setup.** Launch $\mathcal{A}(1^d)$ and receive from it some $M \in \mathbb{N}$ and a database $D \in \{0,1\}^M$. Run
  \[
  \text{hk} \xleftarrow{\$} \text{Gen}(1^d, M), \quad (h, \hat{D}) \leftarrow \text{Hash}(1^d, hk, D),
  \]
  and send $hk$ to $\mathcal{A}$.
**Challenge.** \( A \) submits an index \( m \in [M] \) and two labels (messages) \( L_0, L_1 \in \{0,1\}^d \). Run

\[
\text{ct} \leftarrow \begin{cases} 
\text{Send}(1^k, hk, h, m, L_0 \quad , L_1 ), & \text{if } b = 0; \\
\text{Send}(1^k, hk, h, m, L_{D[m]}, L_{D[m]}), & \text{if } b = 1; 
\end{cases}
\]

and send \( \text{ct} \) to \( A \).

**Guess.** \( A \) outputs a bit \( b' \), which is the output of the experiment.

**Obfuscation.** We rely on indistinguishability obfuscator for polynomial-sized domain.

**Definition 9** ((circuit) obfuscator [BGI*01]). A (circuit) obfuscator is an efficient algorithm \( \text{Obf}(1^k, C) \) taking a circuit \( C : \{0,1\}^n \rightarrow \{0,1\}^{n'} \) as input and producing a circuit \( \tilde{C} : \{0,1\}^n \rightarrow \{0,1\}^{n'} \) as output. The scheme must be correct, i.e., for all \( \lambda \in \mathbb{N}, n,n' \in \mathbb{N}, C : \{0,1\}^n \rightarrow \{0,1\}^{n'}, x \in \{0,1\}^n \),

\[
\Pr[\text{Obf}(1^k, C)(x) = C(x)] = 1.
\]

**Definition 10** (iO [BGI*01] for poly(\( \lambda \))-sized domain). An obfuscator \( \text{Obf} \) (Definition 9) is an indistinguishability obfuscator for polynomial-sized domain (iO for poly(\( \lambda \))-sized domain) if \( \text{Exp}^0_{\text{iO}} \approx \text{Exp}^1_{\text{iO}} \), where \( \text{Exp}^b_{\text{iO}}(1^k) \) with adversary \( A \) proceeds as follows:

- **Challenge.** Launch \( A(1^k) \) and receive from it the domain size \( 1^{2^k} \) and two circuits \( C_0, C_1 : \{0,1\}^n \rightarrow \{0,1\}^{n'} \). Send \( \text{Obf}(1^k, C_b) \) to \( A \).

- **Guess.** \( A \) outputs a bit \( b' \). The output of the experiment is \( b' \) if \( C_0, C_1 \) have the same (description) size and \( C_0(x) = C_1(x) \) for all \( x \in \{0,1\}^n \). Otherwise, the output is set to 0.

**Assumption.** All of the primitives defined in this section are implied by the existence of weakly selectively secure, single key, and sublinearly succinct public-key functional encryption for general circuits (so-called obfuscation-minimum PKFE), of which we refer the reader to [KNTY19] for the precise definition.

**Lemma 1.** Suppose there exists an obfuscation-minimum PKFE with polynomial security, then there exist

- [Yao86,LP09,BHR12] a secure circuit garbling scheme (Definitions 1 and 2),
- [GGM84,BW13,KPTZ13,BGI14] a secure PPRF (Definitions 3 and 4),
- [folklore] a secure PKE scheme (Definitions 5 and 6),
- [CDG*17,LZ17,AL18,KNTY19] a secure laconic OT scheme (Definitions 7 and 8), and
- [LT17,LZ17] an iO for poly(\( \lambda \))-sized domain (Definitions 9 and 10),

with polynomial security.
Alternatively, those primitives can be based on the existence of \( iO \) and one-way function. However, \( iO \) security (for circuits whose domains are not necessarily poly(\( \lambda \))-sized) is not known to be falsifiable [GW11] and it is hard to conceive [GGSW13] a reduction of \( iO \) security to complexity assumptions [GK16]. Since all of the security notions defined in this section are falsifiable, it is unsatisfactory to base them on \( iO \) from a theoretical point of view.

In contrast, obfuscation-minimum PKFE security is falsifiable and there are constructions [JLS21b,JLS21a] from well-studied complexity assumptions. The point of Lemma 1 is to base our constructions solely on one falsifiable assumption, or even complexity assumptions.

3 Ad Hoc Broadcast, Trace, and Revoke

This section concerns the definitions for ad hoc broadcast, trace, and revoke. After formally defining the syntax and correctness of AH-BTR, we present an intuitive definition of its security. While the security definition is comprehensive, it is not the easiest to work with, so we turn to define two simpler security notions, whose conjunction is equivalent to the comprehensive definition. The proof of their equivalence follows the definitions. Later in this paper, we will only work with the simpler notions.

**Definition 11** (AH-BTR). An ad hoc broadcast, trace, and revoke (AH-BTR) scheme (with message space \( \{0,1\}^4 \) and public key length \( M_0(\lambda) \)) consists of 4 efficient algorithms:

- **Gen**\( (1^\lambda) \) outputs a pair \((pk, sk)\) of public and secret keys with \( |pk| = M_0(\lambda) \).
- **Enc**\( (1^\lambda, \{pk_j\}_{j \in [N]}, \mu \in \{0,1\}^4) \) takes as input a list of public keys and a message. It outputs a ciphertext \( ct \).
- **Dec**\( \{pk_j\}_{j \in [N]}, ct \)\( (1^\lambda, N, i \in [N], sk_i) \) is given random access to a list of public keys and a ciphertext, and takes as input the length of the list, an index, and a secret key. It outputs a message.
- **Trace**\( D \)\( (1^\lambda, \{pk_j^*\}_{j \in [N]}, 1/\varepsilon^*) \) is given oracle access to a (stateless randomized) distinguisher \( D \) and takes as input a list of public keys and an error bound. It outputs an index \( i^* \in \{\bot\} \cup [N] \).

The scheme must be robustly correct, i.e., for all \( \lambda \in \mathbb{N}, N \in \mathbb{N}, i \in [N], \{pk_j\}_{j \in [N] \setminus \{i\}} \)\(^{10} \) such that \( |pk_j| = M_0(\lambda) \) for all \( j \in [N] \setminus \{i\} \), and \( \mu \in \{0,1\}^4 \),

\[
\Pr \left[ (pk_i, sk_i) \xleftarrow{\$} \text{Gen}(1^\lambda) \bigg| \begin{array}{c} \text{ct} \xleftarrow{\$} \text{Enc}(1^\lambda, \{pk_j\}_{j \in [N]}, \mu) \\
\quad \text{Dec}(\{pk_j\}_{j \in [N]}, \text{ct}(1^\lambda, N, i, sk_i) = \mu \bigg) = 1. \right]
\]

**Definition 12** (traceability). An AH-BTR scheme (Gen, Enc, Dec, Trace) per Definition 11 is traceable if all efficient adversary wins \( \text{Exp}_{\text{trace}} \) only with negligible probability, where \( \text{Exp}_{\text{trace}}(1^\lambda) \) with adversary \( B \) proceeds as follows:

\(^{10}\)These public keys could be out of the support of Gen, i.e., malformed.
• **Setup.** Launch $B(1^λ)$ and receive $1^Q$ from it. Let $S ← [Q]$ and run

$$(pk_q, sk_q) ← \text{Gen}(1^λ) \quad \text{for } q ∈ [Q],$$

and send $\{pk_q\}_{q ∈ [Q]}$ to $B$.

• **Query.** Repeat the following for arbitrarily many rounds determined by $B$. In each round, $B$ submits $t ∈ [Q]$ for $sk_t$. Upon the query, let $S ← S \setminus \{t\}$ and send $sk_t$ to $B$.

• **Challenge.** $B$ outputs a (probabilistic) circuit $D$, a list $\{pk_j^*\}_{j ∈ [N]}$ of public keys, and an error bound $1/\varepsilon^*$. Run

$$i^* ← \text{Trace}^D(1^λ, \{pk_j^*\}_{j ∈ [N]}, 1/\varepsilon^*).$$

Let

- $\text{FalsePos}$ be the event that $i^* ∈ [N]$ and $pk_i^* = pk_s$ for some $s ∈ S$,
- $\text{GoodDist}$ the event that

$$\left| \Pr \left[ \begin{array}{c}
\mu_0 ← \{0, 1\}^λ, \\
\mu_1 ← \{0, 1\}^λ \\
\beta ← \{0, 1\} \\
\text{ct} ← \text{Enc}(1^λ, \{pk_j^*\}_{j ∈ [N]}, \mu_\beta)
\end{array} \right| : D(\mu_0, \mu_1, \text{ct}) = \beta - \frac{1}{2} ≥ \varepsilon^*,
\right.$$  

- and $\text{NotFound}$ the event that $i^* ∉ [N]$ (i.e., $i^* = \perp$).

$B$ wins if and only if $\text{FalsePos} ∨ (\text{GoodDist} ∧ \text{NotFound})$.

Similar to Remark 3 in [Zha20b], traceability implies KEM security (omitted here).

### 3.1 Simplified Security Notions

The traceability of AH-BTR guarantees that a traitor must be found (if the decoder is good enough) and innocent users must not be accused (whether or not the decoder is good enough). Decomposing the two requirements (plus some apparent weakening) makes each of them simpler (in particular, non-interactive). The first requirement is called **completeness**, and the second **soundness**.

**Definition 13 (completeness).** An AH-BTR scheme $(\text{Gen, Enc, Dec, Trace})$ per Definition 11 is **complete** if all efficient adversary wins $\text{Exp}_{\text{complete}}$ only with negligible probability, where $\text{Exp}_{\text{complete}}(1^λ)$ with adversary $C$ proceeds as follows:

- **Challenge.** Launch $C(1^λ)$, which outputs a (probabilistic) circuit $D$, a list $\{pk_j^*\}_{j ∈ [N]}$ of public keys, and an error bound $1/\varepsilon^*$. Run

$$i^* ← \text{Trace}^D(1^λ, \{pk_j^*\}_{j ∈ [N]}, 1/\varepsilon^*).$$

Let

- $\text{GoodDist}$ be the event that

$$\left| \Pr \left[ \begin{array}{c}
\mu_0 ← \{0, 1\}^λ, \\
\mu_1 ← \{0, 1\}^λ \\
\beta ← \{0, 1\} \\
\text{ct} ← \text{Enc}(1^λ, \{pk_j^*\}_{j ∈ [N]}, \mu_\beta)
\end{array} \right| : D(\mu_0, \mu_1, \text{ct}) = \beta - \frac{1}{2} ≥ \varepsilon^*,
\right.$$  

- and $\text{NotFound}$ the event that $i^* ∉ [N]$ (i.e., $i^* = \perp$).

$C$ is **secure** if $\text{Exp}_{\text{complete}}(1^λ)$ at every stage is negligible.
– and NotFound the event that \( i^* \notin [N] \) (i.e., \( i^* = \perp \)).

\( C \) wins if and only if GoodDist \( \land \) NotFound.

**Definition 14** (soundness). An AH-BTR scheme \((\text{Gen}, \text{Enc}, \text{Dec}, \text{Trace})\) per Definition 11 is *sound* if all efficient adversary wins \( \text{Exp}_{\text{sound}} \) only with negligible probability, where \( \text{Exp}_{\text{sound}}(1^\lambda) \) with adversary \( C \) proceeds as follows:

- **Challenge.** Run \((pk, sk) \leftarrow \text{Gen}(1^\lambda)\), then run \( C(1^\lambda, pk) \), which outputs a (probabilistic) circuit \( D \), some \( N \in \mathbb{N} \), a challenge index \( i^*_\perp \in [N] \), a list \( \{pk_j^*\}_{j \in [N]} \backslash \{i^*_\perp\} \) of public keys, and an error bound \( 1/\epsilon^* \). Let \( pk^*_{i^*_\perp} \leftarrow pk \) and run

  \[
  i^* \leftarrow \text{Trace}^D(1^\lambda, \{pk_j^*\}_{j \in [N]}, 1^{1/\epsilon^*}).
  \]

\( C \) wins if and only if \( i^* = i^*_\perp \) (the event FalsePos).

**Theorem 2** (¶). An AH-BTR scheme is traceable if and only if it is both complete and sound.

**Proof** (Theorem 2). The reductionist proof of necessity is straight-forward by setting \( Q = 0 \) (resp. \( Q = 1 \)) for completeness (resp. soundness).

To show sufficiency, suppose the AH-BTR scheme \((\text{Gen}, \text{Enc}, \text{Dec}, \text{Trace})\) is both complete and sound and let \( B \) be an efficient adversary against its traceability. We consider two efficient adversaries. \( C_1 \) is against the completeness of the scheme. It works by internally simulating the traceability game for \( B \) and outputting (in the completeness experiment) whatever \( B \) outputs. Denoting probabilities and events for adversary \( \lambda \) in its security experiment with subscript \( \lambda \),

\[
\text{GoodDist}_{C_1} \iff \text{GoodDist}_B \quad \text{and} \quad \text{NotFound}_{C_1} \iff \text{NotFound}_B.
\]

Therefore,

\[
\Pr_B[\text{GoodDist}_B \land \text{NotFound}_B] = \Pr_{C_1}[\text{GoodDist}_{C_1} \land \text{NotFound}_{C_1}].
\]

\( C_2 \) is against the soundness of the scheme. Let \( B = \text{poly}(\lambda) > 1 \) be an upper bound of \( Q \) that \( B \) might ever output (\( B \) exists since \( B \) outputs \( 1^Q \) in polynomial time). \( C_2 \) does the following:

- \( C_2(pk) \) launches \( B \), receives \( 1^Q \) from it, samples and sets

  \[
  s^* \leftarrow [B], \quad pk_{s^*} \leftarrow pk, \quad S \leftarrow [Q], \quad (pk_q, sk_q) \leftarrow \text{Gen()} \quad \text{for } q \in [Q] \setminus \{s^*\},
  \]

  and sends \( \{pk_q\}_{q \in [Q]} \) to \( B \).

- \( C_2 \) answers queries from \( B \) and updates \( S \) as stipulated by the query phase of the traceability experiment, except that it aborts if \( B \) queries for \( sk_{s^*} \).

- After the query phase, \( B \) outputs

  \[
  D, \quad \{pk_j^*\}_{j \in [N]}, \quad 1^{1/\epsilon^*},
  \]
and $C_2$ samples or sets

$$i^*_\perp \begin{cases} i^* \perp, & \text{if } i^* \perp \leftarrow \{ i \in [N] : \text{pk}_i^* = \text{pk} \neq \emptyset \}; \\ \perp, & \text{otherwise.} \end{cases}$$

It aborts if $i^*_\perp = \perp$. Otherwise, $C_2$ outputs

$$D, \ N, \ i^*_\perp, \ \{ \text{pk}_j^* \}_{j \in [N] \setminus \{ i^*_\perp \}}, \ 1^{1/\varepsilon^*}.$$ 

Routine calculation yields

$$\Pr_{C_2}[\text{FalsePos}_{C_2}] \geq \frac{1}{B^2} \Pr[B_2[\text{FalsePos}_{B_2}]].$$

By the union bound,

$$\Pr[B_2[\text{FalsePos}_{B_2} \lor (\text{GoodDist}_{B_2} \land \text{NotFound}_{B_2})]] \leq \Pr[B_2[\text{FalsePos}_{B_2}]] + \Pr[B_2[\text{GoodDist}_{B_2} \land \text{NotFound}_{B_2}]] \leq B^2 \Pr[C_2[\text{FalsePos}_{C_2}]] + \Pr[C_1[\text{GoodDist}_{C_1} \land \text{NotFound}_{C_1}]] = (\text{poly}(\lambda))^2 \text{negl}(\lambda) + \text{negl}(\lambda) = \text{negl}(\lambda). \quad \square$$

## 4 Ad Hoc Private Linear Broadcast Encryption

Our construction of AH-BTR follows that of traitor tracing schemes in [BSW06]. We define ad hoc private broadcast linear encryption (AH-PLBE) by adapting the notion of PLBE [BSW06] to the ad hoc setting.

**Definition 15 (AH-PLBE).** An ad hoc private linear broadcast encryption (AH-PLBE) scheme (with message space $\{0,1\}^\lambda$ and public key length $M_0(\lambda)$) consists of 3 efficient algorithms:

- **Gen($1^\lambda$)** outputs a pair $(\text{pk}, \text{sk})$ of public and secret keys with $|\text{pk}| = M_0(\lambda)$.

- **Enc($1^\lambda, \{\text{pk}_j\}_{j \in [N]}, i^*_\perp \in [0..N], \mu \in \{0,1\}^\lambda$)** takes as input a list of public keys, a cut-off index, and a message. It outputs a ciphertext $ct$.

- **Dec($\{\text{pk}_j\}_{j \in [N]}$,$ct$,$1^\lambda, N, i, \text{sk}_i$)** is given random access to a list of public keys and a ciphertext, and takes as input the length of the list, an index, and a secret key. It outputs a message.

The scheme must be **robustly correct**, i.e., for all $\lambda \in \mathbb{N}$, $N \in \mathbb{N}$, $i \in [N]$, $\{\text{pk}_j\}_{j \in [N] \setminus \{i\}}^{11}$ such that $|\text{pk}_j| = M_0(\lambda)$ for all $j \in [N] \setminus \{i\}$, and $\mu \in \{0,1\}^\lambda$,

$$\Pr \left[ \begin{array}{c} (\text{pk}_i, \text{sk}_i) \sample Gen(1^\lambda) \\ ct \sample Enc(1^\lambda, \{\text{pk}_j\}_{j \in [N]}, 0, \mu) \end{array} : \text{Dec}(\{\text{pk}_j\}_{j \in [N]}, ct(1^\lambda, N, i, \text{sk}_i) = \mu) \right] = 1.$$ 

---

11These public keys could be out of the support of Gen, i.e., malformed.
Security. We define security notions of AH-PLBE analogously to those in [BSW06], except “mode indistinguishability” (Game 1 in [BSW06]), which is not needed here. The two security definitions have a one-to-one correspondence to the simplified security notions of AH-BTR in Section 3.1. Namely, message-hiding translates to completeness, and index-hiding translates to soundness.

Definition 16 (message-hiding). An AH-PLBE scheme \((\text{Gen}, \text{Enc}, \text{Dec})\) per Definition 15 is message-hiding if \(\text{Exp}^0_{\text{MH}} \approx \text{Exp}^1_{\text{MH}}\), where \(\text{Exp}^b_{\text{MH}}(\lambda)\) with adversary \(\mathcal{A}\) proceeds as follows:

- **Challenge.** Launch \(\mathcal{A}(1^\lambda)\) and receive from it a list \(\{pk^*_j\}_{j \in [N]}\) of public keys. Run
  \[
  \mu_0 \leftarrow \{0, 1\}^\lambda, \quad \mu_1 \leftarrow \{0, 1\}^\lambda, \quad \text{ct} \leftarrow \text{Enc}(1^\lambda, \{pk^*_j\}_{j \in [N]}, N, \mu_b),
  \]
  and send \((\mu_0, \mu_1, \text{ct})\) to \(\mathcal{A}\).

- **Guess.** \(\mathcal{A}\) outputs a bit \(b'\), which is the output of the experiment.

Definition 17 (index-hiding). An AH-PLBE scheme \((\text{Gen}, \text{Enc}, \text{Dec})\) per Definition 15 is index-hiding if \(\text{Exp}^0_{\text{IH}} \approx \text{Exp}^1_{\text{IH}}\), where \(\text{Exp}^b_{\text{IH}}(\lambda)\) with adversary \(\mathcal{A}\) proceeds as follows:

- **Challenge.** Run \((pk, sk) \leftarrow \text{Gen}(1^\lambda)\), launch \(\mathcal{A}(1^\lambda, pk)\), and receive from it some \(N \in \mathbb{N}\), a cut-off index \(i^*_\perp \in [N]\), and a list \(\{pk^*_j\}_{j \in [N]\setminus \{i^*_\perp\}}\) of public keys. Let \(pk^*_{i^*_\perp} \leftarrow pk\), run
  \[
  \mu \leftarrow \{0, 1\}^\lambda, \quad \text{ct} \leftarrow \text{Enc}(1^\lambda, \{pk^*_j\}_{j \in [N]}, i^*_\perp - 1 + b, \mu),
  \]
  and send \((\mu, \text{ct})\) to \(\mathcal{A}\).

- **Guess.** \(\mathcal{A}\) outputs a bit \(b'\), which is the output of the experiment.

4.1 Construction

Ingredients of Construction 1. Let

- \(\text{GC} = (\text{GC.Garble}, \text{GC.Eval}, \text{GC.SimGarble})\) be a circuit garbling scheme such that \(\text{GC.Garble}\) uses \(\lambda\)-bit randomness,

- \(\text{PPRF} = (\text{PPRF.Puncture}, \text{PPRF.Eval})\) a PPRF,

- \(\text{PKE} = (\text{PKE.Gen}, \text{PKE.Enc}, \text{PKE.Dec})\) a PKE scheme such that \(\text{PKE.Enc}\) uses \(\lambda\)-bit randomness and whose public keys are (exactly) of polynomial length \(M_0\),

- \(\text{LOT} = (\text{LOT.Gen}, \text{LOT.Hash}, \text{LOT.Send}, \text{LOT.Recv})\) a laconic OT scheme,

- \(\text{Obf}\) an obfuscator.

Construction 1 (AH-PLBE). Our AH-PLBE works as follows:

- \(\text{Gen}\) is the same as \(\text{PKE.Gen}\).
Hardwired. \( N \), number of users;  
\( hk \), laconic OT hash key;  
\( h \), laconic OT hash of \( D = pk_1 || \cdots || pk_N \);  
\( i \), cut-off index;  
\( \mu \), cut-off message;  
\( \mu' \), message;  
\( k_{GC} \), PPRF key for circuit garbling;  
\( k_{PKE} \), PPRF key for public-key encryption;  
\( k_{LOT} \), PPRF key for sending the \( m_0 \)th label using laconic OT.

Input. \( i \in [N] \), index of recipient.

Output. Computed as follows.  
\[
\begin{align*}
      r_{GC}^i &\leftarrow \text{PPRF.Eval}(k_{GC}, i) \\
      r_{PKE}^i &\leftarrow \text{PPRF.Eval}(k_{PKE}, i) \\
      r_{LOT}^i &\leftarrow \text{PPRF.Eval}(k_{LOT}, i) \quad \text{for } m_0 \in [M_0] \\
      (\widehat{C}_{ct,i}, \{L_{i,m_0,b}\}_{m_0 \in [M_0], b \in \{0,1\}}) &\leftarrow \begin{cases} \\
      \text{GC.Garble}(\widehat{C}_{ct}, (\mu, r_{PKE}^i); r_{GC}^i), & \text{if } i \leq i_\perp; \\
      \text{GC.Garble}(\widehat{C}_{ct}, (\mu', r_{PKE}^i); r_{GC}^i), & \text{if } i > i_\perp; \\
\end{cases} \\
      \text{LOT.ct}_{i,m_0} &\leftarrow \text{LOT.Send}(hk, h, (i-1)M_0 + m_0, \\
      &\quad L_{i,m_0,0}, L_{i,m_0,1}; i_{LOT}^i) \quad \text{for } m_0 \in [M_0] \\
\end{align*}
\]

\[
\begin{align*}
\text{output} &\quad (\widehat{C}_{ct,i}, \{\text{LOT.ct}_{i,m_0}\}_{m_0 \in [M_0]}) \\
\end{align*}
\]

\[
\begin{align*}
\text{Hardwired.} &\quad \mu', \text{ message or cut-off message; } \\
&\quad r_{PKE}^i, \text{ public-key encryption randomness.} \\
\text{Input.} &\quad pk_i, \text{ public key of recipient.} \\
\text{Output.} &\quad \text{PKE.ct}_i \leftarrow \text{PKE.Enc}(pk_i, \mu'; r_{PKE}^i). \\
\end{align*}
\]

Figure 1. The circuits \( C_{GC} \) and \( C_{ct} \) in Construction 1.
• Enc(\{pk_j\}_{j \in [N], i_\perp, \mu}) first checks whether |pk_j| = M_0 for all j \in [N]. If not, it outputs ct = \bot and terminates. Otherwise, the algorithm hashes down the public keys by running

\begin{align*}
M &\leftarrow NM_0, & D &\leftarrow pk_1 \parallel \cdots \parallel pk_N, \\
\text{hk} &\leftarrow \text{LOT} \cdot \text{Gen}(M), & (h, \tilde{D}) &\leftarrow \text{LOT} \cdot \text{Hash}(\text{hk}, D).
\end{align*}

It samples the cut-off message \mu_\perp \leftarrow \{0, 1\}^\lambda and PPRF keys

\begin{align*}
k_G^\text{GC} &\leftarrow \{0, 1\}^\lambda, & k_P^\text{PKE} &\leftarrow \{0, 1\}^\lambda, & k_{\text{lot}}^\mu &\leftarrow \{0, 1\}^\lambda \text{ for } m_0 \in [M_0],
\end{align*}

and obfuscates \( C_G \) (Figure 1) by running

\[ \tilde{C}_G \leftarrow \text{Obf}(C_G[N, \text{hk}, i_\perp, \mu_\perp, \mu, k_G^\text{GC}, k_P^\text{PKE}, \{k_{\text{lot}}^\mu\}_{m_0 \in [M_0]}) \right. \]

The algorithm outputs ct = (hk, \tilde{C}_G) as the ciphertext.

• Dec(\{pk_j\}_{j \in [N], ct}(N, i, sk_i) first parses ct = (hk, \tilde{C}_G) and recomputes

\begin{align*}
M &\leftarrow NM_0, & D &\leftarrow pk_1 \parallel \cdots \parallel pk_N, & (h, \tilde{D}) &\leftarrow \text{LOT} \cdot \text{Hash}(\text{hk}, D).
\end{align*}

The algorithm next runs the obfuscated circuit,

\[ (\tilde{C}_{ct, i}, \{\text{LOT} \cdot \text{ct}_{i, m_0}\}_{m_0 \in [M_0]})) \leftarrow \tilde{C}_G(i), \]

to obtain the garbled \( C_{ct} \) (Figure 1) for the decryptor and the laconic OT ciphertexts sending its labels. It then receives the labels,

\[ L_{i, m_0, pk_{[m_0]} \leftarrow \text{LOT} \cdot \text{Recv}(hk, h, (i - 1)M_0 + m_0, \text{LOT} \cdot \text{ct}_{i, m_0}) \text{ for } m_0 \in [M_0], \]

and evaluates the garbled circuit,

\[ \text{PKE} \cdot \text{ct}_i \leftarrow \text{GC} \cdot \text{Eval}(\tilde{C}_{ct, i}, \text{pk}_i, \{L_{i, m_0, pk_{[m_0]}\}_{m_0 \in [M_0]}), \]

to obtain the PKE ciphertext under the decryptor’s public key. Lastly, the algorithm runs and outputs (as the decrypted message)

\[ \mu \leftarrow \text{PKE} \cdot \text{Dec}(\text{sk}_i, \text{PKE} \cdot \text{ct}_i). \]

**Robustness Correctness.** This can be verified by inspection.

**Efficiency.** By the efficiency of laconic OT, LOT.Gen takes time poly(\lambda, \log N), LOT.Hash takes time \( O(N) \cdot \text{poly}(\lambda, \log N) \), and \( |hk|, |h| = \text{poly}(\lambda, \log N) \). As we shall see later, it suffices to pad \( C_G \) to size poly(\lambda, \log N) for the security proofs to go through. Putting these together,

\[ T_{\text{Enc}} = O(N) \cdot \text{poly}(\lambda, \log N), \quad |ct| = \text{poly}(\lambda, \log N), \quad T_{\text{Dec}} = O(N) \cdot \text{poly}(\lambda, \log N). \]

In practice and for security reasons, we always assume \( N \leq 2^\lambda \) and \( \log N \) is absorbed by \( \lambda \). Therefore, with \( \text{poly}(\lambda) \) factors ignored, both encryption and decryption take linear time, and the ciphertext is constant-size.
Compatibility. Since the key generation algorithm of Construction 1 is just the key generation algorithm of the underlying PKE scheme (which only has to be semantically secure for random messages), it is compatible with the existing public-key encryption schemes, i.e., existing users possessing PKE key pairs can utilize our AH-PLBE without regenerating their keys.

4.2 Message-Hiding Property

**Theorem 3 (4).** Suppose in Construction 1, the obfuscator Obf is an iO for poly(\(\lambda\))-sized domain, then the resultant AH-PLBE is message-hiding.

**Proof (Theorem 3).** For Construction 1, the only difference between Exp_{MH}^0 and Exp_{MH}^1 is whether \(C_{\text{GC}}\) used to create \(ct = (hk, C_{\text{GC}})\) has \(\mu_0\) or \(\mu_1\) hardwired as \(\mu\). In \(C_{\text{GC}}\) (Figure 1), \(\mu\) is used only in the branch \(i > i_\bot\), which is never taken in \(\text{Exp}_{\text{MH}}^0\) or \(\text{Exp}_{\text{MH}}^1\) because \(i_\bot\) is hardwired to be \(N\) and the domain of \(i\) is \([N]\). Therefore, the two \(C_{\text{GC}}\)'s in \(\text{Exp}_{\text{MH}}^0\) and \(\text{Exp}_{\text{MH}}^1\) being obfuscated are functionally equivalent and have the same size. Moreover, their domain size is \(N\) (polynomially large). Therefore, \(\text{Exp}_{\text{MH}}^0 \approx \text{Exp}_{\text{MH}}^1\) reduces to the iO security for poly(\(\lambda\))-sized domain of Obf. \(\square\)

4.3 Index-Hiding Property

**Theorem 4 (4).** Suppose in Construction 1, all of the ingredients are secure, then the resultant AH-PLBE is index-hiding.

**Proof (Theorem 4).** The only difference between \(\text{Exp}_{\text{IH}}^0\) and \(\text{Exp}_{\text{IH}}^1\) is whether the \(C_{\text{GC}}\) being obfuscated hardwires \(\mu\) (in \(\text{Exp}_{\text{IH}}^0\)) or \(\mu_\bot\) (in \(\text{Exp}_{\text{IH}}^1\)) into \(C_{\text{ct},i_\bot}\), which only affects the output of \(C_{\text{GC}}\) at \(i = i_\bot^*\). We consider the following hybrids, each (except the first) described by the changes from the previous one:

- \(H_0^b\) (for \(b \in \{0,1\}\)) is \(\text{Exp}_{\text{IH}}^b\), where
  
  \[
  \begin{align*}
  &\text{hk} \leftarrow \text{LOT.Gen}(NM_0), \quad (h, \tilde{D}) \leftarrow \text{LOT.Hash}(hk, pk_i^* || \cdots || pk_N^*), \\
  &k_{\text{GC}} \leftarrow \{0,1\}^\lambda, \quad k_{\text{PKE}} \leftarrow \{0,1\}^\lambda, \quad k_{\text{LOT}} \leftarrow \{0,1\}^\lambda \text{ for } m_0 \in [M_0], \\
  &C_{\text{GC}} \leftarrow \text{Obf}(C_{\text{GC}}[N, \text{hk}, h, i_\bot^* + b], \mu_\bot, \mu, k_{\text{GC}}, k_{\text{PKE}}, k_{\text{LOT}}, \{k_{m_0} m_0 \in [M_0]\}), \\
  &ct = (hk, C_{\text{GC}}).
  \end{align*}
  \]

- \(H_1^b\) alters the obfuscation into
  
  \[
  \begin{align*}
  &\tilde{C}_{\text{GC}} \leftarrow \text{Obf}(C'_{\text{GC}}[N, \text{hk}, h, i_\bot^*, \mu, \\
  &i_\bot^*, k_{\text{GC}}^*, k_{\text{PKE}}^*, k_{\text{LOT}}^*, \{k_{m_0,i_\bot^*} m_0 \in [M_0]\}, \tilde{C}_{\text{ct},i_\bot^*}, \{\text{LOT.ct}_{i_\bot^*} m_0 \in [M_0]\}),
  \end{align*}
  \]

where

- \(C_{\text{GC}}'\) is defined in Figure 2,
- the PPRF keys are punctured at \(i_\bot^*\) by running
  
  \[
  \begin{align*}
  &k_{\text{GC}}^* \leftarrow \text{PPRF.Puncture}(k_{\text{GC}}, i_\bot^*), \\
  &k_{\text{PKE}}^* \leftarrow \text{PPRF.Puncture}(k_{\text{PKE}}, i_\bot^*), \\
  &k_{\text{LOT}}^* \leftarrow \text{PPRF.Puncture}(k_{\text{LOT}}, i_\bot^*) \text{ for } m_0 \in [M_0],
  \end{align*}
  \]

17 / 34
and the output \((\widetilde{\text{C}}_{\text{ct},i^*_1}, \{\text{LOT}.\text{ct}_{i^*_1,m_0}\}_{m_0 \in [M_0])}\) of \(C'_{\text{GC}}\) at \(i = i^*_1\) is computed as

\[
\begin{align*}
  r^\text{GC} & \leftarrow \text{PPRF.Eval}(k^\text{GC}_{i^*_1}), \\
r^\text{PKE} & \leftarrow \text{PPRF.Eval}(k^\text{PKE}_{i^*_1}), \\
r^{\text{LOT}}_{\text{ct},i^*_1,m_0} & \leftarrow \text{PPRF.Eval}(k_{\text{ct},i^*_1,m_0}) \quad \text{for } m_0 \in [M_0], \\
(\widetilde{\text{C}}_{\text{ct},i^*_1}, \{L_{i^*_1,m_0,b}\}_{m_0 \in [M_0], b \in \{0,1\}}) & \leftarrow \begin{cases} \\
  \text{GC.Garble}(\text{C}_{\text{ct}}, (\mu, \mu^*, r^\text{PKE}_{i^*_1}); r^\text{GC}_{i^*_1}), & \text{if } b = 0; \\
  \text{GC.Garble}(\text{C}_{\text{ct}}, (\mu, r^\text{PKE}_{i^*_1}); r^\text{GC}_{i^*_1}), & \text{if } b = 1; \\
\end{cases} \\
\text{LOT}.\text{ct}_{i^*_1,m_0} & \leftarrow \text{LOT}.\text{Send}(hk, (i^*_1 - 1)M_0 + m_0, \\
L_{i^*_1,m_0,0}, L_{i^*_1,m_0,1}; r^{\text{LOT}}_{\text{ct},i^*_1,m_0}) \quad \text{for } m_0 \in [M_0].
\end{align*}
\]

\(H^b_j\) changes \(r^\text{GC}_{i^*_1}, r^\text{PKE}_{i^*_1}\), and \(r^{\text{LOT}}_{\text{ct},i^*_1,m_0}\)'s into true randomness, i.e.,

\[
\begin{align*}
  r^\text{GC} & \leftarrow \{0, 1\}^\lambda, \\
r^\text{PKE} & \leftarrow \{0, 1\}^\lambda, \\
r^{\text{LOT}}_{\text{ct},i^*_1,m_0} & \leftarrow \{0, 1\}^\lambda \quad \text{for } m_0 \in [M_0].
\end{align*}
\]

**Figure 2.** The circuit \(C'_{\text{GC}}\) in the proof of Theorem 4.
* $H^b_3$ removes the unused labels from LOT.$ct_{i_1^*,m_0}'$s by setting

$\text{LOT.}ct_{i_1^*,m_0} \leftarrow \text{LOT.}\text{Send}(hk, h, (i_1^* - 1)M_0 + m_0, \allowbreak L_{i_1^*,m_0,pk_{i_1^*}^{s}[m_0]}, L_{i_1^*,m_0,pk_{i_1^*}^{s}[m_0]}; \tau_{i_1^*,m_0}^{\text{LOT}})$ for $m_0 \in [M_0]$.

* $H^b_4$ changes $\hat{C}_{ct,i_1^*}$ into simulation, i.e.,

$$
\text{PKE.}ct_{i_1^*} \leftarrow \begin{cases} 
\text{PKE.}\text{Enc}(pk_{i_1^*}^{s}, \mu ; r_{\text{PKE}}), & \text{if } b = 0; \\
\text{PKE.}\text{Enc}(pk_{i_1^*}^{s}, \mu_1 ; r_{\text{PKE}}), & \text{if } b = 1;
\end{cases}
$$

$$(\hat{C}_{ct,i_1^*}, \{L_{i_1^*,m_0,pk_{i_1^*}^{s}[m_0]}\}_{m_0 \in [M_0]}) \leftarrow \text{GC.}\text{SimGarble}(C_{ct}, pk_{i_1^*}^{s}, \text{PKE.}ct_{i_1^*})$$

where $pk_{i_1^*}^{s} = pk$ is sampled by the experiment (not adversarially controlled).

The following claims hold, all of which are immediate by inspection:

Claim 5. $H^b_5 \equiv H^b_1$ for $b \in \{0, 1\}$ if Obf is an iO for poly($\lambda$)-sized domain.

Claim 6. $H^b_5 \equiv H^b_2$ for $b \in \{0, 1\}$ if PPRF is pseudorandom at the punctured point.

Claim 7. $H^b_5 \equiv H^b_3$ for $b \in \{0, 1\}$ if LOT is database-selectively sender-private.

Claim 8. $H^b_5 \equiv H^b_4$ for $b \in \{0, 1\}$ if GC is $w$-hiding.

Claim 9. $H^b_5 \equiv H^b_1$ if PKE is semantically secure for random messages.

$\text{Exp}^b_{11} \approx \text{Exp}^b_{11}$ follows from a hybrid argument. $\square$

### 5 AH-BTR from AH-PLBE

**Ingredient of Construction 2.** Let ahPLBE = (ahPLBE.Gen, ahPLBE.Enc, ahPLBE.Dec) be an AH-PLBE scheme.

**Construction 2** (adapted from Section 2.2 in [BSW06]). Our AH-BTR works as follows:

- Gen is the same as ahPLBE.Gen.
- Enc$(\{pk_j\}_{j \in [N]}, \mu)$ runs and outputs $ct \leftarrow ^\$ ahPLBE.Enc$(\{pk_j\}_{j \in [N]}, 0, \mu)$.
- Dec is the same as ahPLBE.Dec.
- Trace$^D(\{pk_j\}_{j \in [N]}, 1^{1/\varepsilon'})$ defines for $i \in [0..N],

$$
\epsilon_i = \Pr[\mu_0 \xleftarrow{\$} \{0, 1\}^\lambda, \mu_1 \xleftarrow{\$} \{0, 1\}^\lambda, \beta \xleftarrow{\$} \{0, 1\}, \allowbreak ct \xleftarrow{\$} \text{ahPLBE.Enc}(1^\lambda, \{pk_j\}_{j \in [N]}, i, \mu_1, \mu_1, \beta) \in D(\mu_0, \mu_1, ct) = \beta] - \frac{1}{2}.
$$

experiment $\epsilon_i$ (sampling and testing) and event $E_i$ (correct guessing)

Setting $\delta \leftarrow \frac{\varepsilon'}{10N}$ and $\eta \leftarrow \left[\frac{\lambda \log(2N+2)}{2\delta^2}\right]$, for each $i \in [0..N]$, the algorithm runs $E_i$ for $\eta$ times independently, counts the absolute frequency $\xi_i \in [0..\eta]$ of $E_i$, and computes $\tilde{\xi}_i = \frac{\xi_i}{\eta} - \frac{1}{2}$. It outputs

$$
i^* = \begin{cases} 
\min T, & \text{if } T \leftarrow \{ i \in [N] : |\tilde{\xi}_i - \tilde{\xi}_{i-1}| \geq 3\delta \} \neq \emptyset; \\
\bot, & \text{if } T = \emptyset.
\end{cases}
$$
**Robustness Correctness, Efficiency, Compatibility.** These are inherited from the underlying AH-PLBE. When based on Construction 1, the resultant AH-BTR has

$$T_{\text{Enc}} = O(N) \poly(\lambda), \quad |\text{ct}| = \poly(\lambda), \quad T_{\text{Dec}} = O(N) \poly(\lambda),$$

and is compatible with the existing public-key encryption schemes.

**Theorem 10.** Suppose in Construction 2, the AH-PLBE scheme $\text{ahPLBE}$ is message-hiding, then the resultant AH-BTR is complete.

**Theorem 11.** Suppose in Construction 2, the AH-PLBE scheme $\text{ahPLBE}$ is index-hiding, then the resultant AH-BTR is sound.

**Proof (Theorem 10).** Consider any efficient adversary $C$ against the completeness of Construction 2. Let $\text{GoodEst}$ be the event that $|\varepsilon_i - \varepsilon_i| \leq \delta$ for all $i \in [0..N]$. By the Chernoff bound, the union bound, and the law of total probability,

$$\Pr[\neg \text{GoodEst}] = \mathbb{E}[\Pr[\neg \text{GoodEst} | \varepsilon^*, N]] \leq \mathbb{E}[2(N + 1) \exp(-2\delta^2 \eta)] \leq 2^{-\lambda}.$$ 

Let $\text{BadEnd}$ be the event that $|\varepsilon_N| > \varepsilon^*_2$, then $\text{GoodDist} \land \neg \text{BadEnd}$ implies

$$\max_{i \in [N]} |\varepsilon_{i-1} - \varepsilon_i| \geq \frac{1}{N} \sum_{i=1}^{N} |\varepsilon_{i-1} - \varepsilon_i| \geq \frac{1}{N} \left| \sum_{i=1}^{N} (\varepsilon_{i-1} - \varepsilon_i) \right| = \frac{1}{N} |\varepsilon_0 - \varepsilon_N|$$

$$\geq \frac{1}{N} (|\varepsilon_0| - |\varepsilon_N|) \geq \frac{1}{N} \left( \varepsilon^* - \frac{\varepsilon^*}{2} \right) = \frac{\varepsilon^*}{2N} = 5\delta.$$ 

Therefore, $\text{GoodDist} \land \neg \text{BadEnd} \land \text{GoodEst}$ implies

$$\max_{i \in [N]} |\varepsilon_{i-1} - \varepsilon_i| \geq \max_{i \in [N]} (|\varepsilon_{i-1} - \varepsilon_i| - 2\delta) \geq 5\delta - 2\delta = 3\delta,$$

which in turn implies $T \neq \emptyset$ hence $i^* \in [N]$, i.e., $\neg \text{NotFound}$. By contraposition,

$$\text{GoodDist} \land \text{NotFound} \land \text{GoodEst} \implies \text{BadEnd}.$$ 

By the union bound,

$$\Pr[C \text{ wins}] \leq \Pr[\neg \text{GoodEst}] + \Pr[(C \text{ wins}) \land \text{GoodEst}]$$

$$= \Pr[\neg \text{GoodEst}] + \Pr[\text{GoodDist} \land \text{NotFound} \land \text{GoodEst}]$$

$$\leq 2^{-\lambda} + \Pr[\text{BadEnd}],$$

so it remains to show $\Pr[\text{BadEnd}] = \text{negl}(\lambda)$.

Consider the following efficient adversary $A$ against the message-hiding property of $\text{ahPLBE}$:

- $A$ runs $C$ to obtain

  $$\mathcal{D}, \quad \{pk_j^*\}_{j \in [N]}, \quad 1^{1/\varepsilon^*}.$$ 

- $A$ runs $E_N$ once and notes down $\alpha \in \{0, 1\}$ indicating whether $E_N$ happened, i.e., $\alpha = 1$ if and only if $\mathcal{D}$ guessed correctly in the trial.
• \( A \) submits \( \{pk_j^*\}_{j \in [N]} \) to the message-hiding experiment, receives \((\mu_0, \mu_1, ct)\) back, and runs and outputs \( b' \leftarrow D(\mu_0, \mu_1, ct) \oplus a \).

Routine calculation shows that the advantage of \( A \) is \( E[4e_N^2] \), which must be negligible by the message-hiding property of ahPLBE. Let \( B = \text{poly}(\lambda) \) be an upper bound of \( 1/e^* \) (\( B \) exists since \( C \) outputs \( 1/\epsilon^* \) in polynomial time). By Markov’s inequality,

\[
\Pr[\text{BadEnd}] = \Pr[4e_N^2 > (\epsilon^*)^2] \leq \Pr[4e_N^2 > B^{-2}] \\
\leq B^2 E[4e_N^2] = (\text{poly}(\lambda))^2 \text{negl}(\lambda) = \text{negl}(\lambda). \quad \square
\]

**Proof (Theorem 11).** Consider any efficient adversary \( C \) against the soundness of Construction 2. Similarly to the proof of Theorem 10, define GoodEst and recall that \( \Pr[\neg \text{GoodEst}] \leq 2^{-\lambda} \). We have

\[
\Pr[C \text{ wins}] \leq \Pr[\neg \text{GoodEst}] + \Pr[(C \text{ wins}) \land \text{GoodEst}] \\
= \Pr[\neg \text{GoodEst}] + \Pr[\text{FalsePos} \land \text{GoodEst}] \\
\leq 2^{-\lambda} + \Pr[\text{FalsePos} \land \text{GoodEst}],
\]

and it suffices to prove \( \Pr[\text{FalsePos} \land \text{GoodEst}] = \text{negl}(\lambda) \).

Let \( \alpha \) be a random element in an execution of Trace with

\[
\alpha = \begin{cases} 
0, & \text{if } i^* \in [N] \text{ and } \tilde{\epsilon}_{i^*} - \tilde{\epsilon}_{i^*-1} \geq 3\delta; \\
1, & \text{if } i^* \in [N] \text{ and } \tilde{\epsilon}_{i^*} - \tilde{\epsilon}_{i^*-1} \leq -3\delta; \\
\bot, & \text{if } i^* = \bot.
\end{cases}
\]

Consider the following efficient adversary \( A \) against the index-hiding property of ahPLBE:

• \( A(pk) \) runs \( C(pk) \) to obtain

\[
D, \: N, \quad i^* \parallel, \quad \{pk_j^*\}_{j \in [N] \setminus \{i^*\}}, \quad 1^{1/e^*},
\]

and sets \( pk_{i^*}^* \leftarrow pk \).

• \( A \) runs

\[
i^* \leftarrow \text{Trace}^D(\{pk_j^*\}_{j \in [N]}, 1^{1/e^*}),
\]

and aborts if \( i^* \neq i^*_{\parallel} \).

• \( A \) notes down \( \alpha \in \{0, 1\} \) from the above execution of Trace, submits

\[
N, \quad i^* \parallel, \quad \{pk_j^*\}_{j \in [N] \setminus \{i^*\}}
\]

to the index-hiding experiment, gets \((\mu, ct)\) back, samples and sets

\[
\beta \leftarrow \{0, 1\}, \quad \mu_\beta \leftarrow \mu, \quad \mu_{\beta^*} \leftarrow \{0, 1\}^\lambda,
\]

and runs and outputs \( b' \leftarrow D(\mu_0, \mu_1, ct) \oplus \beta \oplus \alpha \).
Routine calculation shows that the advantage of $A$ is

$$E[I_{\text{FalsePos}} \cdot (-1)^a(\epsilon_{i',-1} - \epsilon_{i'})],$$

which must be negligible by the index-hiding property of $\text{ahPLBE}$.

Let $B = \text{poly}(\lambda)$ be an upper bound of $10N/\epsilon^*$ ($B$ exists since $C$ outputs $1^N$ and $1^{1/\epsilon^*}$ in polynomial time). The event $\text{FalsePos} \wedge \text{GoodEst}$ implies

$$|\epsilon_{i',-1} - \epsilon_{i'}| - (\hat{\epsilon}_{i',-1} - \hat{\epsilon}_{i'})| \leq 2\delta < 3\delta \leq |\hat{\epsilon}_{i',-1} - \hat{\epsilon}_{i'}|$$

$$\implies (-1)^a(\epsilon_{i',-1} - \epsilon_{i'}) = |\epsilon_{i',-1} - \epsilon_{i'}| \geq 3\delta - 2\delta = \frac{\epsilon^*}{10N} \geq B^{-1}.$$ 

Moreover, $(-1)^a(\epsilon_{i',-1} - \epsilon_{i'}) \geq -1$ always holds. These together show that

$$\Pr[\text{FalsePos} \wedge \text{GoodEst}]$$

$$= B E[I_{\text{FalsePos}} \cdot I_{\text{GoodEst}} \cdot B^{-1}]$$

$$\leq B E[I_{\text{FalsePos}} \cdot I_{\text{GoodEst}} \cdot (-1)^a(\epsilon_{i',-1} - \epsilon_{i'})]$$

$$\leq B \left(E[I_{\text{FalsePos}} \cdot I_{\text{GoodEst}} \cdot (-1)^a(\epsilon_{i',-1} - \epsilon_{i'})] + E[I_{\text{FalsePos}} \cdot \neg I_{\text{GoodEst}}] \right)$$

$$= B \left(E[I_{\text{FalsePos}} \cdot (-1)^a(\epsilon_{i',-1} - \epsilon_{i'})] + \Pr[\text{FalsePos} \wedge \neg I_{\text{GoodEst}}] \right)$$

$$\leq B \left(E[I_{\text{FalsePos}} \cdot (-1)^a(\epsilon_{i',-1} - \epsilon_{i'})] + 2^{-1} \right)$$

$$= \text{poly}(\lambda)(\text{negl}(\lambda) + 2^{-1}) = \text{negl}(\lambda). \quad \square$$

### 6 Trading Ciphertext Size for Decryption Time in AH-BTR

While Construction 2 achieves constant ciphertext size, it takes time $\Omega(N)$ to decrypt. In contrast, the naïve scheme that encrypts to each user separately has $\Omega(N)$-size ciphertext, yet decryption only takes constant time. By grouping the recipients and encrypting to each group separately, we can trade ciphertext size for decryption time. Previous work [Zha20a] already systemizes the idea of grouping in the context of traditional traitor tracing.

**Ingredients of Construction 3.** Let $\text{old} = (\text{old.Gen, old.Enc, old.Dec, old.Trace})$ be an AH-BTR scheme and $\gamma$ some constant ($0 < \gamma < 1$).

**Construction 3** (adapted from Theorem 1 in [Zha20a]). Our new AH-BTR works as follows:

- $\text{Gen}$ is the same as $\text{old.Gen}$.

---

12 Alternatively, one can reformulate Construction 2 as a compiler that trades decryption time for ciphertext size, by grouping the recipients and compressing the groups. We refrained from such a formulation because the “transformation” uses a quite strong additional assumption, namely functional encryption for general circuits.

13 We require that $N \mapsto [N^\gamma]$ can be computed in (deterministic) time $\text{poly}(\log N)$. 

22 / 34
• Enc(\{pk_j\}_{j \in [N]}, \mu) sets N_1 = \lceil N^\gamma \rceil \text{ and } N_2 = \lceil N/N_1 \rceil. It runs

$$\text{old.ct}_{j_1} \xleftarrow{} \text{old.Enc}(\{pk_j\}_{(j_1-1)N_2 < j \leq j_1N_2}, \mu) \quad \text{ for } j_1 \in [N_1].$$

The algorithm outputs ct = \{old.ct_{j_1}\}_{j_1 \in [N_1]}.

• Dec(\{pk_j\}_{j \in [N]}, ct(N, i, sk_i)) sets N_1 = \lceil N^\gamma \rceil, N_2 = \lceil N/N_1 \rceil. It parses ct as \{old.ct_{j_1}\}_{j_1 \in [N_1]}, finds i_1 \in [N_1] such that (i_1 - 1)N_2 < i \leq i_1N_2, and sets N'_2 = \min \{N_2, N - (i_1 - 1)N_2 \}. The algorithm runs and outputs

$$\text{old.Dec}(\{pk_j\}_{i_1-1N_2 < j \leq i_1N_2}, \text{old.ct}_{j_1}(N'_2, i - (i_1 - 1)N_2, sk_i).$$

• Trace(\{pk_j\}_{j \in [N]}, 1^{1/\epsilon'}) sets N_1 = \lceil N^\gamma \rceil \text{ and } N_2 = \lceil N/N_1 \rceil. It runs

$$i_{j_1}^* \xleftarrow{} \text{old.Trace}(\{pk_j\}_{(j_1-1)N_2 < j \leq j_1N_2}, 1^{N_2/\epsilon'}) \quad \text{ for } j_1 \in [N_1],$$

where \(D_{j_1}(\mu_0, \mu_1, \text{old.ct})^*\) runs and outputs \(D(\mu_0, \mu_1, \{\text{old.ct}_{j_1}\}_{j_1 \in [N_1]})\) with

$$\text{old.ct}_{j_1} \leftarrow \text{old.ct}^*,$$

where

$$\left\{ \begin{array}{ll}
\text{old.Enc}(\{pk_j\}_{(j_1-1)N_2 < j \leq j_1N_2}, \mu_0), & \text{ if } j_1' < j_1;
\text{old.Enc}(\{pk_j\}_{(j_1-1)N_2 < j \leq j_1N_2}, \mu_1), & \text{ if } j_1' > j_1.
\end{array} \right.$$

The algorithm outputs

$$\left\{ \begin{array}{ll}
(j_1 - 1)N_2 + i_{j_1}^*, & \text{ if } i_{j_1}^* = \bot \text{ for all } j_1' < j_1 \text{ and } i_{j_1}^* \neq \bot; \\
\bot, & \text{ if } i_{j_1}^* = \bot \text{ for all } j_1' \in [N_1].
\end{array} \right.$$

Robustness Correctness and Compatibility. These are inherited from the underlying AH-BTR. When based on Construction 2, the resultant AH-BTR is compatible with the existing public-key encryption schemes.

Efficiency. Let \(\gamma_1, \gamma_2, \gamma_3\) be constants such that the AH-BTR efficiency is

$$T_{\text{Enc}} = O(N^{\gamma_1}) \text{ poly}(\lambda), \quad |\text{ct}| = O(N^{\gamma_2}) \text{ poly}(\lambda), \quad T_{\text{Dec}} = O(N^{\gamma_3}) \text{ poly}(\lambda),$$

then the underlying efficiency is mapped to the resultant efficiency\(^{14}\) by

$$(\gamma_1, \gamma_2, \gamma_3) \mapsto (1 - \gamma + \gamma \gamma_1, 1 - \gamma + \gamma \gamma_2, \gamma \gamma_3).$$

When based on Construction 2, the resultant AH-BTR enjoys

$$T_{\text{Enc}} = O(N) \text{ poly}(\lambda), \quad |\text{ct}| = O(N^{1-\gamma}) \text{ poly}(\lambda), \quad T_{\text{Dec}} = O(N^\gamma) \text{ poly}(\lambda).$$

**Theorem 12 (\(\heartsuit\)).** Suppose in Construction 3, the underlying AH-BTR scheme old is complete, then so is the resultant AH-BTR.

**Theorem 13 (\(\heartsuit\)).** Suppose in Construction 3, the underlying AH-BTR scheme old is sound, then so is the resultant AH-BTR.

\(^{14}\)We assume that old.ct's are of deterministic length so Dec knows the location of each particular old.ct. Alternatively, Enc can store a look-up table of their locations in ct.
Proof (Theorem 12). Let $C$ be an efficient adversary against the completeness of the resultant scheme. Consider the following efficient adversary $C_{\text{old}}$ against the completeness of old:

- $C_{\text{old}}$ launches $C$ to obtain

\[ D, \{pk_j^*\}_{j \in [N]}, 1^{1/\epsilon}. \]

It computes $N_1, N_2$ as specified by the resultant scheme.

- $C_{\text{old}}$ samples $j_1^* \triangleq [N_1]$, prepares $D_{j_1^*}$ (using $D$, as specified by the resultant scheme), and outputs

\[ D_{j_1^*}, \{pk_j^*\}_{(j_1^* - 1)N_2 < j \leq j_1^*N_2}, 1^{N_1/\epsilon}. \]

Let $B = \text{poly}(\lambda)$ be an upper bound of $N_1$. Routine calculation shows

\[ \Pr[C_{\text{old}} \text{ wins}] \geq \frac{1}{B} \Pr[C \text{ wins}], \]

hence by the completeness of old,

\[ \Pr[C \text{ wins}] \leq B \Pr[C_{\text{old}} \text{ wins}] = \text{poly}(\lambda) \text{ negl}(\lambda) = \text{negl}(\lambda). \]

Proof (Theorem 13). Let $C$ be an efficient adversary against the soundness of the resultant scheme. Consider the following efficient adversary $C_{\text{old}}$ against the soundness of old:

- $C_{\text{old}}(pk)$ launches $C(pk)$ to obtain

\[ D, N, i_\perp^*, \{pk_j^*\}_{j \in [N] \setminus \{i_\perp^*\}}, 1^{1/\epsilon}. \]

It computes $N_1, N_2$ as specified by the resultant scheme.

- $C_{\text{old}}$ computes $j_1^* = \lceil i_\perp^*/N_2 \rceil$ and outputs

\[ D_{j_1^*}, \min \{N_2, N - (j_1^* - 1)N_2\}, i_\perp^* - (j_1^* - 1)N_2, \{pk_j^*\}_{(j_1^* - 1)N_2 < j \leq j_1^*N_2, j \neq i_\perp^*}, 1^{N_1/\epsilon}. \]

Routine calculation and the soundness of old yield

\[ \Pr[C \text{ wins}] \leq \Pr[C_{\text{old}} \text{ wins}] = \text{negl}(\lambda). \]

\[ \square \]

7 Lower Bound on Ciphertext Size and Decryption Time

In this section, we prove that for all secure AH-BTR,

\[ |ct| \cdot T_{\text{dec}} = \Omega(N), \]

and therefore, we have constructed all the optimal (ignoring $\text{poly}(\lambda)$ factors) AH-BTR schemes in this work, completely pinning down the Pareto front of its efficiency. In fact, we will show a related bound against a restricted kind of broadcast encryption,\(^{15}\) which can be implemented using AH-BTR in a straight-forward manner.

The scheme is restricted in the sense that the users are paired and encryption only broadcasts to those sets for which there is precisely one recipient from each pair. The required security notion is also weaker — it does not consider collusion among multiple non-recipients nor adaptive attacks.

\(^{15}\)This result is thus also a lower bound for general attribute-based encryption.
Definition 18 (restricted broadcast encryption and its security). A restricted broadcast encryption (BE) scheme (for the purpose of this work) consists of 3 efficient algorithms:

- \( \text{Gen}(1^\lambda, 1^N) \) takes a length parameter as input. It outputs a master public key mpk and a list \( \{sk_{j,s}\}_{j\in[N], s\in\{0,1\}} \) of secret keys.
- \( \text{Enc}(1^\lambda, \text{mpk}, R, \mu) \) takes as input the master public key \( \text{mpk} \), an \( N \)-bit string \( R \in \{0,1\}^N \), and a message \( \mu \in \{0,1\}^\lambda \). It outputs a ciphertext \( ct_R \).
- \( \text{Dec}^{\text{mpk}, i, r, sk_i, R, ct_R}(1^\lambda) \) is given random access to the master public key mpk, a secret key with its description \( (i, r, sk_{i,r}) \), a ciphertext with its attribute \((R, ct_R)\). It is supposed to recover \( \mu \) if and only if \( R[i] = r \).

The scheme must be correct, i.e., for all \( \lambda, N \in \mathbb{N}, R \in \{0,1\}^N, i \in [N], \mu \in \{0,1\}^\lambda \),
\[
\Pr \left[ \left( \text{mpk}, \{sk_{j,s}\}_{j\in[N], s\in\{0,1\}} \right) \leftarrow \text{Gen}(1^\lambda, 1^N) \\
ct_R \leftarrow \text{Enc}(1^\lambda, \text{mpk}, R, \mu) \\
: \text{Dec}^{\text{mpk}, i, r, sk_i, R, ct_R}(1^\lambda) = \mu \right] = 1.
\]

The scheme is 1-key secure for random challenge against uniform adversaries (or secure for the purpose of this work) if
\[
\{(1^\lambda, \text{mpk}, R, i^*, \mu_0, sk_{i^*,-R[i^*]}, ct_0)\} \approx \{(\cdots, ct_1)\},
\]
where
\[
R \leftarrow \{0,1\}^N, \quad i^* \leftarrow [N],
\]
\[
\left( \text{mpk}, \{sk_{j,s}\}_{j\in[N], s\in\{0,1\}} \right) \leftarrow \text{Gen}(1^\lambda, 1^N),
\]
\[
\mu_b \leftarrow \{0,1\}^\lambda, \quad ct_b \leftarrow \text{Enc}(1^\lambda, \text{mpk}, R, \mu_b) \quad \text{for } b \in \{0,1\},
\]
for all polynomially bounded \( N = N(\lambda) \), where the computational indistinguishability only has to hold against uniform adversaries.\(^\text{16}\)

Theorem 14 (¶). For all secure restricted BE,
\[
\max |\text{ct}| \cdot \max T_{\text{dec}} \geq \frac{N}{1000}
\]
for all polynomially bounded \( N = N(\lambda) \) and sufficiently large \( \lambda \), where ct runs through all possible ciphertexts and \( T_{\text{dec}} \) the time to probe \( R \) and produce output by \( \text{Dec} \), both for \( R \) of length \( N \).

We remark that while the statement and the proof here apply to perfectly correct schemes with polynomial security, it is straightforward to adapt them for schemes with sufficient (say, constant) gap between correctness and security.

To prove Theorem 14, we need the following lemma:

Lemma 15 (adapted from Theorem 2 in [Unr07]). For all \( N, P \in \mathbb{N} \) subject to \( 1 \leq P \leq N \), distribution \( D \) supported over \( Z \), function \( F : Z \times \{0,1\}^N \rightarrow \{0,1\}^S \), there exists a function \( G : Z \times \{0,1\}^N \rightarrow \{0,1, \bot\}^N \) such that
\[
|\{j \in [N] : G(z, j) \neq \bot\}| \leq P \quad \text{for all } z \in Z
\]

\(^{16}N \) need not be a computable function of \( \lambda \). This does not make the security definition “non-uniform”, as a standard guessing argument (with advantage sign correction) applies to an interactive formulation in which the uniform and efficient \( A \) chooses \( N \).
and for all (non-efficient) oracle (randomized) algorithm \( B^Y \) making at most \( T \) queries to \( Y \),

\[
\left| \Pr \left[ B^R(z, F(z, R)) \rightarrow 1 \right] - \Pr \left[ B^H(z, F(z, R)) \rightarrow 1 \right] \right| \leq \frac{\sqrt{ST}}{2P},
\]

where

\[
R \xleftarrow{\$} \{0, 1\}^N, \quad z \xleftarrow{\$} D, \quad H[j] \begin{cases} = G(z, R)[j], & \text{if } G(z, R)[j] \neq \bot; \\ \xleftarrow{\$} \{0, 1\}, & \text{if } G(z, R)[j] = \bot. \end{cases}
\]

**Proof (Theorem 14).** Define

\[
S = 1 + \max |ct|, \quad T = 1 + \max \{\text{number of bits in } R \text{ probed by } \text{Dec}\}.
\]

For \( \lambda, N \geq 1 \), it is necessary that \( |ct| \geq 1 \) because \( ct \) can encode any string \( \mu \) of length \( \lambda \), and that \( \max T_{\text{Dec}} \geq T \) because \( \text{Dec} \) performs all the probes and, in addition, produces at least 1 bit of output. Therefore,

\[
\max |ct| \cdot \max T_{\text{Dec}} \geq \frac{\max |ct| + 1}{2} \cdot \max T_{\text{Dec}} \geq \frac{ST}{2}.
\]

It remains to prove \( ST \geq \frac{2N}{1000} \) for sufficiently large \( \lambda \). It suffices to consider the case when \( N \geq 2 \) and \( ST \leq 2N \).

We prepare for Lemma 15. Let \( P \) be determined later, and

\[
z = \left( \mu, z_{\text{Enc}}, \text{mpk}, \{sk_{j,s}\}_{j \in [N], s \in (0, 1)} \right) \sim D = \left\{ \begin{array}{l}
\mu \xleftarrow{\$} \{0, 1\}^{\lambda} \\
\text{randomness for } \text{Enc} \\
(\text{mpk}, \{sk_{j,s}\}_{j \in [N], s \in (0, 1)}) \xleftarrow{\$} \text{Gen}(1^N)
\end{array} \right\},
\]

\[
F(z, R) = 0^{S-|ct|-1} ||ct|, \quad \text{where } ct \leftarrow \text{Enc}(\text{mpk}, R, \mu; z_{\text{Enc}}).
\]

Let \( G \) be the function guaranteed by Lemma 15 and make \( B^Y(z, f) \) do the following:

- Sample \( i^* \xleftarrow{\$} [N] \) and query \( r^* \leftarrow Y[i^*] \).
- Read \( \mu, \text{mpk}, sk_{i^*, r^*}, ct \) from \( z, f \).
- Run \( \mu' \xleftarrow{\$} \text{Dec}^{\text{mpk}, i^*, r^*, sk_{i^*, r^*}, Y, ct}() \).
- Output 1 if and only if \( \mu = \mu' \).

Note that \( B \) indeed makes at most \( T \) queries to \( Y \), the first to obtain \( r^* \) and the rest to run Dec.

For \( w \in \{1, 2, 3, 4, 5\} \), write \( p_w \) for \( \Pr[B^Y_w(z, f; i^*) \rightarrow 1] \), where

\[
i^* \xleftarrow{\$} [N], \quad Y_1 = R,
\]

\[
Y_2[j] = G(z, F(z, R))[j], \quad \begin{cases} = G(z, F(z, R))[j], & \text{if } G(z, F(z, R))[j] \neq \bot; \\ \xleftarrow{\$} \{0, 1\}, & \text{if } G(z, F(z, R))[j] = \bot; \end{cases}
\]

\[
Y_3[j] = G(z, F(z, R))[j], \quad \begin{cases} = G(z, F(z, R))[j], & \text{if } j \neq i^* \text{ and } G(z, F(z, R))[j] \neq \bot; \\ \xleftarrow{\$} \{0, 1\}, & \text{if } j \neq i^* \text{ and } G(z, F(z, R))[j] = \bot; \\ \xleftarrow{\$} \{0, 1\}, & \text{if } j = i^*. \end{cases}
\]

\[
Y_4[j] = R[j], \quad \begin{cases} \xleftarrow{\$} \{0, 1\}, & \text{if } j \neq i^*; \\ \xleftarrow{\$} \{0, 1\}, & \text{if } j = i^*. \end{cases}
\]

\[
Y_5[j] = R[j], \quad \begin{cases} \xleftarrow{\$} \{0, 1\}, & \text{if } j \neq i^*; \\ \xleftarrow{\$} \{0, 1\}, & \text{if } j = i^*. \end{cases}
\]
By the correctness of the restricted BE scheme, \( p_1 = 1 \).

From Lemma 15,
\[
|p_1 - p_2| \leq \sqrt{\frac{ST}{2P}}, \quad |p_4 - p_3| \leq \sqrt{\frac{ST}{2P}}.
\]

Here, the second inequality is obtained by applying the lemma to
\[
C^Y(z, f) = B^Y(z, f; i^*) , \quad \text{where } i^* \overset{\$}{\leftarrow} [N] , \quad Y'[j] \begin{cases} = Y[j] , & \text{if } j \neq i^* ; \\
\leftarrow \{0,1\} , & \text{if } j = i^* .
\end{cases}
\]

Clearly, \( |p_2 - p_3| \leq \frac{P}{N} \). Setting \( P = \left\lceil \frac{3STN^2}{2} \right\rceil \), we have
\[
|p_1 - p_4| \leq |p_1 - p_2| + |p_2 - p_3| + |p_3 - p_4| \leq \sqrt{\frac{ST}{2P}} + \frac{P}{N} + \sqrt{\frac{ST}{2P}} \leq 3\sqrt{\frac{ST}{2N}} + \frac{1}{N} < 4\sqrt{\frac{ST}{2N}} ,
\]
where the last inequality follows from \( N \geq 2 \). By how \( Y[i^*] \) is set,
\[
p_4 = \frac{p_1 + p_5}{2} \quad \implies \quad p_5 = p_1 - 2(p_1 - p_4) \geq p_1 - 2|p_1 - p_4| > 1 - 8\sqrt{\frac{ST}{2N}}.
\]

Consider the following adversary \( \mathcal{A}(\text{mpk}, R, i^*, \mu_0, \text{sk}_{i^*}, R[i^*], \text{ct}) \) against the security of the restricted BE scheme:

- Construct \( Y_5 \) from \( R \) and let \( r^* \leftarrow Y_5[i^*] = -R[i^*] \).
- Run \( \mu' \overset{\$}{\leftarrow} \text{Dec}^{\text{mpk}, i^*, r^*, \text{sk}_{i^*}, Y_5, \text{ct}}() \), i.e., pretend \( R[i^*] \) were \( -R[i^*] \) and try decrypting using the key given to \( \mathcal{A} \).
- Output 1 if and only if \( \mu' = \mu_0 \).

If \( \text{ct} = \text{ct}_1 \) is an encryption of \( \mu_1 \), then \( \mu_0 \) is uniformly random and independent of everything else, hence
\[
\Pr[\mathcal{A}(\cdots) \rightarrow 1 \text{ with } \text{ct} = \text{ct}_1] \leq 2^{-\lambda}.
\]

Note that \( \mathcal{A} \) is a uniform adversary. By the security of the restricted BE scheme,
\[
p_5 = \Pr[B^{Y_5}(z, f; i^*) \rightarrow 1] = \Pr[\mathcal{A}(\cdots) \rightarrow 1 \text{ with } \text{ct} = \text{ct}_0] \leq 2^{-\lambda} + \text{negl}(\lambda) < \frac{1}{5}
\]
for sufficiently large \( \lambda \), which gives
\[
1 - 8\sqrt{\frac{ST}{2N}} < p_5 < \frac{1}{5} \quad \implies \quad ST > \frac{2N}{1000} . \quad \Box
\]

**Corollary 16 (¶).** For all secure AH-BTR,
\[
\max |\text{ct}| \cdot \max T_{\text{dec}} \geq \frac{N}{1000}
\]
for all polynomially bounded \( N = N(\lambda) \) and sufficiently large \( \lambda \), where \( T_{\text{dec}} \) only counts the time to probe \( \text{pk}_j \)'s and produce output. Ignoring \( \text{poly}(\lambda) \) factors, Construction 3 achieves all possible optimal trade-offs in terms of the exponents over \( N \) in the dependency of ciphertext size and (actual) decryption time, fully demonstrating the Pareto front of AH-BTR efficiency.
Proof (Corollary 16). Let $ahBTR = (ahBTR.Gen, ahBTR.Enc, ahBTR.Dec, ahBTR.Trace)$ be a secure AH-BTR and construct the following restricted BE scheme:

- $Gen(1^N)$ runs $\mathbf{(pk_j, sk_j, s) \leftarrow ahBTR.Gen() \text{ for } j \in [N], s \in \{0, 1\}}$ and outputs $\mathbf{mpk = \{pk_j, s\}_{j \in [N], s \in \{0, 1\}}}$.
- $Enc(mpk, R, \mu)$ runs and outputs $\mathbf{ct \leftarrow ahBTR.Enc(\{pk_j, R\}_{j \in [N]}, \mu)}$.
- $Dec(mpk, i, r, sk_i, R, ct)$ runs $\mathbf{ahBTR.Dec^K, ct(N, i, sk_i, R)},$ where $K$ is an oracle implemented by Dec for $ahBTR.Dec$ to probe $pk_j$’s. Whenever $ahBTR.Dec$ probes $pk_j[m_0]$, we make Dec probe $R[j]$ and answer $pk_j[R[j]][m_0]$.

It is straightforward to verify that the constructed scheme is correct and secure. Since a restricted BE ciphertext is precisely an AH-BTR ciphertext, each probe to $pk_j$’s by $ahBTR.Dec$ translates to exactly one probe to $R[j]$ by Dec with no more additional probes by Dec on its own, and Dec outputs whatever $ahBTR.Dec$ outputs, the corollary follows from Theorem 14.

Acknowledgement. The author was supported by NSF grants CNS-1936825 (CAREER), CNS-2026774, a J.P. Morgan AI Research Award, and a Simons Collaboration on the Theory of Algorithmic Fairness. The views expressed in this work are those of the author and do not reflect the official policy or position of any of the supporters. The author thanks Huijia Lin for her encouragement and support for him to pursue an independent research project (culminating this paper), Hoeteck Wee for helpful suggestions on writing from another context, and Daniel Wichs for helpful discussions. He started researching this topic after rereading a post [sil21] on V2EX.

References


[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or


<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
<th>Conference</th>
<th>Volume</th>
<th>Pages</th>
<th>Publisher</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>Dalit Naor, Moni Naor, and Jeffery Lotspiech</td>
<td>Revocation and tracing schemes for stateless receivers</td>
<td>CRYPTO 2001</td>
<td>LNCS 2139</td>
<td>41–62</td>
<td>Springer, Heidelberg</td>
<td>August 2001</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Duong Hieu Phan, David Pointcheval, and Mario Strefler</td>
<td>Decentralized dynamic broadcast encryption</td>
<td>SCN 12</td>
<td>LNCS 7485</td>
<td>166–183</td>
<td>Springer, Heidelberg</td>
<td>September 2012</td>
<td></td>
</tr>
<tr>
<td>2021</td>
<td>sillydaddy</td>
<td>gpg 加密文件文件：一份加密文件，可以被不同的密码解密 (GPG file encryption: One encrypted file can be decrypted by many keys)</td>
<td>V2EX.com</td>
<td></td>
<td></td>
<td></td>
<td>March 2021</td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>Amit Sahai and Brent Waters</td>
<td>How to use indistinguishability obfuscation: deniable encryption, and more</td>
<td>46th ACM STOC</td>
<td></td>
<td>475–484</td>
<td>ACM Press, May / June 2014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer</td>
<td>Ad hoc broadcast encryption (poster presentation)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


