Ad Hoc (Decentralized) Broadcast, Trace, and Revoke

罗辑 (Ji Luo)

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

July 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. 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-source. See `https://github.com/GeeLaw/ahbtr`.

Contents

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,GKRW18,CVW+18,GQWW19,GKW19,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

$$\begin{aligned} \text{encryption time} \qquad & T_{\mathsf{Enc}} = \mathrm{O}(N), \\ \text{ciphertext size} \qquad & |\mathsf{ct}| = \mathrm{O}(N^{1-\gamma}), \\ \text{decryption time} \qquad & T_{\mathsf{Dec}} = \mathrm{O}(N^{\gamma}), \end{aligned}$$

  for any constant $0 \leq \gamma \leq 1$.

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

$$|\mathsf{ct}| \cdot T_{\mathsf{Dec}} = \Omega(N),$$

so our schemes offer all possible optimal trade-offs between $|\mathsf{ct}|$ and T_{Dec}. Better yet, the bound holds for (weak) attribute-based encryption (ABE) [SW05,GPSW06] schemes with a specific simple functionality, thus also shedding new insights into the efficiency of ABE.

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 weakenings include making the scheme bounded,[1] 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.[2] 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].[3] 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

[1]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.

[2]We aim for a scheme without any trusted party, so there should be no global set-up.

[3]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.

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[4] 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 straight-forward 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,[5] 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. It can be seen later that the 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:[6]

- Everyone generates their own public and secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- Encryption uses a list $\{\mathsf{pk}_j\}_{j \in [N]}$ of N public keys of the recipients as well as a cut-off index $0 \le i_\perp \le N$.
- Decryption is possible with sk_j if $j > i_\perp$.

[4]The general transformation [KY01,BSW06] to deal with stateful decoder applies to our definition of AH-BTR, *mutatis mutandis*.

[5]While some previous works [BF99,GKW18,Zha20a] separate traceability into multiple notions, those notions still share one single complicated security experiment.

[6]AH-PLBE can be cast as multi-authority attribute-based encryption [Cha07] for 1-local monotone functions without global set-up.

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 $\mathsf{sk}_1, \ldots, \mathsf{sk}_{i_\perp}$, 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 ε, 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 $\Omega(\varepsilon/N)$ 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,[7] 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

[7]We ignore *fixed* polynomial factors in the security parameter. The point is that the size does not grow with N (the number of recipients).

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 $|\mathsf{ct}| \cdot T_{\mathsf{Dec}} = \Omega(N)$, where $|\mathsf{ct}|$ is the ciphertext length and T_{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 $|\mathsf{ct}|$ and T_{Dec} constant in a traditional traitor tracing scheme [BZ14]. In existing broadcast encryption (or revocation) schemes [BGW05,Del07,GW09,BZ14,AY20,AWY20,BV20] for N users, encrypting to arbitrary subsets of size S or $(N-S)$ makes $|\mathsf{ct}| \cdot T_{\mathsf{Dec}} = \Omega(S)$. 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 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 $R \xleftarrow{\$} \{0,1\}^N$ 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 $(\mathsf{pk}_{j,b}, \mathsf{sk}_{j,b})$ for $j \in [N]$ and $b \in \{0,1\}$. Consider a ciphertext ct encrypting a random plaintext to $\{\mathsf{pk}_{j,R[j]}\}_{j \in [N]}$ for a random string R and regard ct as the advice. Let's try decrypting ct using $\mathsf{sk}_{j,R[j]}$ for a random $j \xleftarrow{\$} [N]$. Each time the AH-BTR decryption algorithm queries pk_j, we probe $R[j]$ and respond with $\mathsf{pk}_{j,R[j]}$. By way of contradiction, suppose $|\mathsf{ct}| \cdot T_{\mathsf{Dec}} \ll N$, which would translate to the setting of the corollary as $S = |\mathsf{ct}|$, $T \leq T_{\mathsf{Dec}}$, 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 $R[j]$ should also lead to successful recovery. But if $R[j]$ is flipped after ct is computed, by the security of AH-BTR, the attempted decryption should fail to recover the plaintext except for negligible probability, yielding a contradiction.

2 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter, by $\mathrm{poly}(\cdot)$ a polynomial function, and by $\mathrm{negl}(\lambda)$ a negligible function of λ. Efficient algorithms are probabilistic random-access machines $M^w(x)$ of running time $\mathrm{poly}(|x|, |w|)$. Efficient adversaries (in interactive experiments) are probabilistic Turing machines of (total) running time $\mathrm{poly}(\lambda)$, with or without $\mathrm{poly}(\lambda)$-long advices. (All of the proofs in this work are uniform.) The advantage of $\mathcal{A}$ in distinguishing Exp_0 and Exp_1 is $\Pr[\mathsf{Exp}_0^{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}_1^{\mathcal{A}}(1^\lambda) = 1]$. We write $\approx, \approx_{\mathrm{s}}, \equiv$ 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 λ-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 $C : \{0,1\}^{n+M_0} \to \{0,1\}^{n'}$ 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 $\mathbb{1}_X$. 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:

- $\mathsf{Garble}(1^\lambda, C, w)$ takes as input a circuit $C : \{0,1\}^{n+M_0} \to \{0,1\}^{n'}$ 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 $m_0 \in [M_0]$, $b \in \{0,1\}$.
- $\mathsf{Eval}(1^\lambda, \widehat{C}, x, \{L_{m_0}\}_{m_0 \in [M_0]})$ 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 $\lambda \in \mathbb{N}$, $n, M_0, n' \in \mathbb{N}$, $C : \{0,1\}^{n+M_0} \to \{0,1\}^{n'}$, $w \in \{0,1\}^n$, $x \in \{0,1\}^{M_0}$,

$$\Pr\left[\begin{array}{l}(\widehat{C}, \{L_{m_0,b}\}_{m_0 \in [M_0], b \in \{0,1\}}) \xleftarrow{\$} \mathsf{Garble}(1^\lambda, C, w) \\ \quad : \mathsf{Eval}(1^\lambda, \widehat{C}, x, \{L_{m_0,x[m_0]}\}_{m_0 \in [M_0]}) = C[w](x)\end{array}\right] = 1.$$

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

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

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 $\mathsf{SimGarble}$ such that $\mathsf{Exp}^0_{\mathsf{GC}} \approx \mathsf{Exp}^1_{\mathsf{GC}}$, where $\mathsf{Exp}^b_{\mathsf{GC}}(1^\lambda)$ with adversary $\mathcal{A}$ proceeds as follows:

- **Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive a circuit $C : \{0,1\}^{n+M_0} \to \{0,1\}^{n'}$, a hardwired input $w \in \{0,1\}^n$, and a non-hardwired input $x \in \{0,1\}^{M_0}$ from it. Run

$$\begin{array}{lll}\text{if } b = 0, & (\widehat{C}, \{L_{m_0,b}\}_{m_0 \in [M_0], b \in \{0,1\}}) \xleftarrow{\$} & \mathsf{Garble}(1^\lambda, C, w); \\ \text{if } b = 1, & (\widehat{C}, \{L_{m_0,x[m_0]}\}_{m_0 \in [M_0]}) \xleftarrow{\$} & \mathsf{SimGarble}(1^\lambda, C, x, C[w](x));\end{array}$$

  and send $(\widehat{C}, \{L_{m_0,x[m_0]}\}_{m_0 \in [M_0]})$ 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:

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

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

$$\Pr\left[\begin{array}{l} k \xleftarrow{\$} \{0,1\}^\lambda \\ \mathring{k}_x \xleftarrow{\$} \mathsf{Puncture}(1^\lambda, k, x) \end{array} : \mathsf{Eval}(1^\lambda, k, x') = \mathsf{Eval}(1^\lambda, \mathring{k}_x, x')\right] = 1.$$

Definition 4 (PPRF security [BW13,KPTZ13,BGI14,SW14])**.** A PPRF $(\mathsf{Puncture}, \mathsf{Eval})$ per Definition 3 is *pseudorandom at the punctured point* (or *secure* for the purpose of this work) if $\mathsf{Exp}^0_{\mathrm{PPRF}} \approx \mathsf{Exp}^1_{\mathrm{PPRF}}$, where $\mathsf{Exp}^b_{\mathrm{PPRF}}(1^\lambda)$ 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

$$k \xleftarrow{\$} \{0,1\}^\lambda, \quad \mathring{k}_x \xleftarrow{\$} \mathsf{Puncture}(1^\lambda, k, x), \quad r_0 \leftarrow \mathsf{Eval}(1^\lambda, k, x), \quad r_1 \xleftarrow{\$} \{0,1\}^\lambda,$$

  and send $(\mathring{k}_x, r_b)$ to $\mathcal{A}$.
- **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:

- $\mathsf{Gen}(1^\lambda)$ outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of public and secret keys with $|\mathsf{pk}| = M_0(\lambda)$.
- $\mathsf{Enc}(1^\lambda, \mathsf{pk}, \mu \in \{0,1\}^\lambda)$ takes as input the public key and a message. It outputs a ciphertext ct.
- $\mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct})$ takes as input the secret key and a ciphertext. It outputs a message.

The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, $\mu \in \{0,1\}^\lambda$,

$$\Pr\left[\begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda) \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{pk}, \mu) \end{array} : \mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct}) = \mu\right] = 1.$$

Definition 6 (PKE security)**.** A PKE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ per Definition 5 is *semantically secure for random messages* (or *secure* for the purpose of this work) if

$$\{(\mu_0, \mu_1, \mathsf{pk}, \mathsf{ct}_0)\} \approx \{(\mu_0, \mu_1, \mathsf{pk}, \mathsf{ct}_1)\},$$

where $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and $\mu_b \xleftarrow{\$} \{0,1\}^\lambda$, $\mathsf{ct}_b \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{pk}, \mu_b)$ 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 $\{0,1\}^\lambda$) consists of 4 efficient algorithms:

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

The scheme must be *correct*, i.e., for all $\lambda, M \in \mathbb{N}$, $D \in \{0,1\}^M$, $m \in [M]$, $L_0, L_1 \in \{0,1\}^\lambda$,

$$\Pr\left[\begin{array}{r} \mathsf{hk} \xleftarrow{\$} \mathsf{Gen}(1^\lambda, M) \\ (h, \widehat{D}) \leftarrow \mathsf{Hash}(1^\lambda, \mathsf{hk}, D) \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Send}(1^\lambda, \mathsf{hk}, h, m, L_0, L_1) \end{array} : \mathsf{Recv}^{\widehat{D}}(1^\lambda, \mathsf{hk}, h, m, \mathsf{ct}) = L_{D[m]}\right] = 1.$$

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 $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Recv})$ per Definition 7 is *database-selectively sender-private* (or *secure* for the purpose of this work) if $\mathsf{Exp}^0_{\mathrm{LOT}} \approx \mathsf{Exp}^1_{\mathrm{LOT}}$, where $\mathsf{Exp}^b_{\mathrm{LOT}}(1^\lambda)$ with adversary $\mathcal{A}$ proceeds as follows:

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive from it some $M \in \mathbb{N}$ and a database $D \in \{0,1\}^M$. Run

$$\mathsf{hk} \xleftarrow{\$} \mathsf{Gen}(1^\lambda, M), \qquad (h, \widehat{D}) \leftarrow \mathsf{Hash}(1^\lambda, \mathsf{hk}, D),$$

  and send hk to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ submits an index $m \in [M]$ and two labels (messages) $L_0, L_1 \in \{0,1\}^\lambda$. Run

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

  and send ct to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ 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 $\mathsf{Obf}(1^\lambda, C)$ taking a circuit $C : \{0,1\}^n \to \{0,1\}^{n'}$ as input and producing a circuit $\widetilde{C} : \{0,1\}^n \to \{0,1\}^{n'}$ as output. The scheme must be *correct,* i.e., for all $\lambda \in \mathbb{N}$, $n, n' \in \mathbb{N}$, $C : \{0,1\}^n \to \{0,1\}^{n'}$, $x \in \{0,1\}^n$,

$$\Pr\left[\mathsf{Obf}(1^\lambda, C)(x) = C(x)\right] = 1.$$

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

- **Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive from it the domain size 1^{2^n} and two circuits $C_0, C_1 : \{0,1\}^n \to \{0,1\}^{n'}$. Send $\mathsf{Obf}(1^\lambda, C_b)$ to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ 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 $i\mathcal{O}$ for* poly(λ)*-sized domain (Definitions 9 and 10),*

with polynomial security.

Alternatively, those primitives can be based on the existence of $i\mathcal{O}$ and one-way function. However, $i\mathcal{O}$ security (for circuits whose domains are not necessarily poly(λ)-sized) is not known to be *falsifiable* [GW11] and it is hard to conceive [GGSW13] a reduction of $i\mathcal{O}$ security to *complexity assumptions* [GK16]. Since all of the security notions defined in this section are falsifiable, it is unsatisfactory to base them on $i\mathcal{O}$ 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 $\{0,1\}^\lambda$ and public key length $M_0(\lambda)$) consists of 4 efficient algorithms:

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

The scheme must be *robustly correct*, i.e., for all $\lambda \in \mathbb{N}$, $N \in \mathbb{N}$, $i \in [N]$, $\{\mathsf{pk}_j\}_{j\in[N]\setminus\{i\}}$[8] such that $|\mathsf{pk}_j| = M_0(\lambda)$ for all $j \in [N] \setminus \{i\}$, and $\mu \in \{0,1\}^\lambda$,

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

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

- **Setup.** Launch $\mathcal{B}(1^\lambda)$ and receive 1^Q from it. Let $S \leftarrow [Q]$ and run

$$(\mathsf{pk}_q, \mathsf{sk}_q) \xleftarrow{\$} \mathsf{Gen}(1^\lambda) \qquad \text{for } q \in [Q],$$

  and send $\{\mathsf{pk}_q\}_{q\in[Q]}$ to $\mathcal{B}$.
- **Query.** Repeat the following for arbitrarily many rounds determined by $\mathcal{B}$. In each round, $\mathcal{B}$ submits $t \in [Q]$ for sk_t. Upon the query, let $S \leftarrow S \setminus \{t\}$ and send sk_t to $\mathcal{B}$.
- **Challenge.** $\mathcal{B}$ outputs a (probabilistic) circuit $\mathcal{D}$, a list $\{\mathsf{pk}^*_j\}_{j\in[N]}$ of public keys, and an error bound $1^{1/\varepsilon^*}$. Run

$$i^* \xleftarrow{\$} \mathsf{Trace}^{\mathcal{D}}(1^\lambda, \{\mathsf{pk}^*_j\}_{j\in[N]}, 1^{1/\varepsilon^*}).$$

  Let

[8]These public keys could be out of the support of Gen, i.e., malformed.

- FalsePos be the event that $i^* \in [N]$ and $\mathsf{pk}^*_{i^*} = \mathsf{pk}_s$ for some $s \in S$,
- GoodDist the event that

$$\left| \Pr\left[\begin{array}{ll} \mu_0 \xleftarrow{\$} \{0,1\}^\lambda, & \mu_1 \xleftarrow{\$} \{0,1\}^\lambda \\ \beta \xleftarrow{\$} \{0,1\} & \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \{\mathsf{pk}^*_j\}_{j\in[N]}, \mu_\beta) & \end{array} : \mathcal{D}(\mu_0, \mu_1, \mathsf{ct}) = \beta \right] - \frac{1}{2} \right| \ge \varepsilon^*,$$

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

$\mathcal{B}$ *wins* if and only if FalsePos $\vee$ (GoodDist $\wedge$ 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 (Gen, Enc, Dec, Trace) per Definition 11 is *complete* if all efficient adversary wins $\mathsf{Exp}_{\mathsf{complete}}$ only with negligible probability, where $\mathsf{Exp}_{\mathsf{complete}}(1^\lambda)$ with adversary $\mathcal{C}$ proceeds as follows:

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

$$i^* \xleftarrow{\$} \mathsf{Trace}^{\mathcal{D}}(1^\lambda, \{\mathsf{pk}^*_j\}_{j\in[N]}, 1^{1/\varepsilon^*}).$$

Let

- GoodDist be the event that

$$\left| \Pr\left[\begin{array}{ll} \mu_0 \xleftarrow{\$} \{0,1\}^\lambda, & \mu_1 \xleftarrow{\$} \{0,1\}^\lambda \\ \beta \xleftarrow{\$} \{0,1\} & \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \{\mathsf{pk}^*_j\}_{j\in[N]}, \mu_\beta) & \end{array} : \mathcal{D}(\mu_0, \mu_1, \mathsf{ct}) = \beta \right] - \frac{1}{2} \right| \ge \varepsilon^*,$$

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

$\mathcal{C}$ *wins* if and only if GoodDist $\wedge$ NotFound.

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

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

$$i^* \xleftarrow{\$} \mathsf{Trace}^{\mathcal{D}}(1^\lambda, \{\mathsf{pk}^*_j\}_{j\in[N]}, 1^{1/\varepsilon^*}).$$

$\mathcal{C}$ *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 $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ is both complete and sound and let $\mathcal{B}$ be an efficient adversary against its traceability. We consider two efficient adversaries. $\mathcal{C}_1$ is against the completeness of the scheme. It works by internally simulating the traceability game for $\mathcal{B}$ and outputting (in the completeness experiment) whatever $\mathcal{B}$ outputs. Denoting probabilities and events for adversary $\mathcal{X}$ in its security experiment with subscript $\mathcal{X}$,

$$\mathsf{GoodDist}_{\mathcal{C}_1} \iff \mathsf{GoodDist}_{\mathcal{B}} \quad \text{and} \quad \mathsf{NotFound}_{\mathcal{C}_1} \iff \mathsf{NotFound}_{\mathcal{B}}.$$

Therefore,

$$\Pr_{\mathcal{B}}[\mathsf{GoodDist}_{\mathcal{B}} \wedge \mathsf{NotFound}_{\mathcal{B}}] = \Pr_{\mathcal{C}_1}[\mathsf{GoodDist}_{\mathcal{C}_1} \wedge \mathsf{NotFound}_{\mathcal{C}_1}].$$

$\mathcal{C}_2$ is against the soundness of the scheme. Let $B = \mathrm{poly}(\lambda) > 1$ be an upper bound of Q that $\mathcal{B}$ might ever output (B exists since $\mathcal{B}$ outputs 1^Q in polynomial time). $\mathcal{C}_2$ does the following:

- $\mathcal{C}_2(\mathsf{pk})$ launches $\mathcal{B}$, receives 1^Q from it, samples and sets

$$s^* \xleftarrow{\$} [B], \qquad \mathsf{pk}_{s^*} \leftarrow \mathsf{pk},$$
$$S \leftarrow [Q], \qquad (\mathsf{pk}_q, \mathsf{sk}_q) \xleftarrow{\$} \mathsf{Gen}() \quad \text{for } q \in [Q] \setminus \{s^*\},$$

  and sends $\{\mathsf{pk}_q\}_{q \in [Q]}$ to $\mathcal{B}$.

- $\mathcal{C}_2$ answers queries from $\mathcal{B}$ and updates S as stipulated by the query phase of the traceability experiment, except that it aborts if $\mathcal{B}$ queries for sk_{s^*}.

- After the query phase, $\mathcal{B}$ outputs

$$\mathcal{D}, \quad \{\mathsf{pk}^*_j\}_{j \in [N]}, \quad 1^{1/\varepsilon^*},$$

  and $\mathcal{C}_2$ samples or sets

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

  It aborts if $i^*_\perp = \perp$. Otherwise, $\mathcal{C}_2$ outputs

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

Routine calculation yields

$$\Pr_{\mathcal{C}_2}[\mathsf{FalsePos}_{\mathcal{C}_2}] \geq \frac{1}{B^2} \Pr_{\mathcal{B}}[\mathsf{FalsePos}_{\mathcal{B}}].$$

By the union bound,

$$\begin{aligned} &\Pr_{\mathcal{B}}[\mathsf{FalsePos}_{\mathcal{B}} \vee (\mathsf{GoodDist}_{\mathcal{B}} \wedge \mathsf{NotFound}_{\mathcal{B}})] \\ \leq\ &\Pr_{\mathcal{B}}[\mathsf{FalsePos}_{\mathcal{B}}] + \Pr_{\mathcal{B}}[\mathsf{GoodDist}_{\mathcal{B}} \wedge \mathsf{NotFound}_{\mathcal{B}}] \\ \leq\ &B^2 \Pr_{\mathcal{C}_2}[\mathsf{FalsePos}_{\mathcal{C}_2}] + \Pr_{\mathcal{C}_1}[\mathsf{GoodDist}_{\mathcal{C}_1} \wedge \mathsf{NotFound}_{\mathcal{C}_1}] \\ =\ &(\mathrm{poly}(\lambda))^2 \, \mathrm{negl}(\lambda) + \mathrm{negl}(\lambda) = \mathrm{negl}(\lambda). \end{aligned}$$

□

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:

- $\mathsf{Gen}(1^\lambda)$ outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of public and secret keys with $|\mathsf{pk}| = M_0(\lambda)$.
- $\mathsf{Enc}(1^\lambda, \{\mathsf{pk}_j\}_{j\in[N]}, i_\perp \in [0..N], \mu \in \{0,1\}^\lambda)$ takes as input a list of public keys, a cut-off index, and a message. It outputs a ciphertext ct.
- $\mathsf{Dec}^{\{\mathsf{pk}_j\}_{j\in[N]},\mathsf{ct}}(1^\lambda, N, i \in [N], \mathsf{sk}_i)$ 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 $\lambda \in \mathbb{N}$, $N \in \mathbb{N}$, $i \in [N]$, $\{\mathsf{pk}_j\}_{j\in[N]\setminus\{i\}}$[9] such that $|\mathsf{pk}_j| = M_0(\lambda)$ for all $j \in [N] \setminus \{i\}$, and $\mu \in \{0,1\}^\lambda$,

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

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 $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ per Definition 15 is *message-hiding* if $\mathsf{Exp}^0_{\mathrm{MH}} \approx \mathsf{Exp}^1_{\mathrm{MH}}$, where $\mathsf{Exp}^b_{\mathrm{MH}}(1^\lambda)$ with adversary $\mathcal{A}$ proceeds as follows:

- **Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive from it a list $\{\mathsf{pk}^*_j\}_{j\in[N]}$ of public keys. Run

$$\mu_0 \xleftarrow{\$} \{0,1\}^\lambda, \quad \mu_1 \xleftarrow{\$} \{0,1\}^\lambda, \qquad \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \{\mathsf{pk}^*_j\}_{j\in[N]}, N, \mu_b),$$

  and send $(\mu_0, \mu_1, \mathsf{ct})$ 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 $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ per Definition 15 is *index-hiding* if $\mathsf{Exp}^0_{\mathrm{IH}} \approx \mathsf{Exp}^1_{\mathrm{IH}}$, where $\mathsf{Exp}^b_{\mathrm{IH}}(1^\lambda)$ with adversary $\mathcal{A}$ proceeds as follows:

- **Challenge.** Run $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, launch $\mathcal{A}(1^\lambda, \mathsf{pk})$, and receive from it some $N \in \mathbb{N}$, a cut-off index $i^*_\perp \in [N]$, and a list $\{\mathsf{pk}^*_j\}_{j\in[N]\setminus\{i^*_\perp\}}$ of public keys. Let $\mathsf{pk}^*_{i^*_\perp} \leftarrow \mathsf{pk}$, run

$$\mu \xleftarrow{\$} \{0,1\}^\lambda, \qquad \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \{\mathsf{pk}^*_j\}_{j\in[N]}, i^*_\perp - 1 + b, \mu),$$

  and send (μ, ct) to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit b', which is the output of the experiment.

[9]These public keys could be out of the support of Gen, i.e., malformed.

4.1 Construction

Ingredients of Construction 1. Let

- $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval}, \mathsf{GC.SimGarble})$ be a circuit garbling scheme such that $\mathsf{GC.Garble}$ uses λ-bit randomness,
- $\mathsf{PPRF} = (\mathsf{PPRF.Puncture}, \mathsf{PPRF.Eval})$ a PPRF,
- $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ a PKE scheme such that $\mathsf{PKE.Enc}$ uses λ-bit randomness and whose public keys are (exactly) of polynomial length M_0,
- $\mathsf{LOT} = (\mathsf{LOT.Gen}, \mathsf{LOT.Hash}, \mathsf{LOT.Send}, \mathsf{LOT.Recv})$ a laconic OT scheme,
- Obf an obfuscator.

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

- Gen is the same as $\mathsf{PKE.Gen}$.
- $\mathsf{Enc}(\{\mathsf{pk}_j\}_{j\in[N]}, i_\perp, \mu)$ first checks whether $|\mathsf{pk}_j| = M_0$ for all $j \in [N]$. If not, it outputs $\mathsf{ct} = \perp$ and terminates. Otherwise, the algorithm hashes down the public keys by running

$$M \leftarrow NM_0, \qquad D \leftarrow \mathsf{pk}_1 \| \cdots \| \mathsf{pk}_N,$$
$$\mathsf{hk} \xleftarrow{\$} \mathsf{LOT.Gen}(M), \qquad (h, \widehat{D}) \leftarrow \mathsf{LOT.Hash}(\mathsf{hk}, D).$$

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

$$k^{\mathsf{GC}} \xleftarrow{\$} \{0,1\}^\lambda, \qquad k^{\mathsf{PKE}} \xleftarrow{\$} \{0,1\}^\lambda, \qquad k^{\mathsf{LOT}}_{m_0} \xleftarrow{\$} \{0,1\}^\lambda \quad \text{for } m_0 \in [M_0],$$

  and obfuscates C_{GC} (Figure 1) by running

$$\widetilde{C}_{\mathsf{GC}} \xleftarrow{\$} \mathsf{Obf}(C_{\mathsf{GC}}[N, \mathsf{hk}, h, i_\perp, \mu_\perp, \mu, k^{\mathsf{GC}}, k^{\mathsf{PKE}}, \{k^{\mathsf{LOT}}_{m_0}\}_{m_0\in[M_0]}]).$$

  The algorithm outputs $\mathsf{ct} = (\mathsf{hk}, \widetilde{C}_{\mathsf{GC}})$ as the ciphertext.

- $\mathsf{Dec}^{\{\mathsf{pk}_j\}_{j\in[N]}, \mathsf{ct}}(N, i, \mathsf{sk}_i)$ first parses $\mathsf{ct} = (\mathsf{hk}, \widetilde{C}_{\mathsf{GC}})$ and recomputes

$$M \leftarrow NM_0, \qquad D \leftarrow \mathsf{pk}_1 \| \cdots \| \mathsf{pk}_N, \qquad (h, \widehat{D}) \leftarrow \mathsf{LOT.Hash}(\mathsf{hk}, D).$$

  The algorithm next runs the obfuscated circuit,

$$(\widehat{C}_{\mathsf{ct},i}, \{\mathsf{LOT.ct}_{i,m_0}\}_{m_0\in[M_0]}) \leftarrow \widetilde{C}_{\mathsf{GC}}(i),$$

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

$$L_{i,m_0,\mathsf{pk}_i[m_0]} \leftarrow \mathsf{LOT.Recv}^{\widehat{D}}(\mathsf{hk}, h, (i-1)M_0 + m_0, \mathsf{LOT.ct}_{i,m_0}) \quad \text{for } m_0 \in [M_0],$$

  and evaluates the garbled circuit,

$$\mathsf{PKE.ct}_i \leftarrow \mathsf{GC.Eval}(\widehat{C}_{\mathsf{ct},i}, \mathsf{pk}_i, \{L_{i,m_0,\mathsf{pk}_i[m_0]}\}_{m_0\in[M_0]}),$$

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

$$\mu \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_i, \mathsf{PKE.ct}_i).$$

$$C_{\mathsf{GC}}[N, \mathsf{hk}, h, i_\perp, \mu_\perp, \mu, k^{\mathsf{GC}}, k^{\mathsf{PKE}}, \{k^{\mathsf{LOT}}_{m_0}\}_{m_0 \in [M_0]}](i)$$

Hardwired.	N,	number of users;
	hk,	laconic OT hash key;
	h,	laconic OT hash of $D = \mathsf{pk}_1 \| \cdots \| \mathsf{pk}_N$;
	$i_\perp$,	cut-off index;
	$\mu_\perp$,	cut-off message;
	μ,	message;
	k^{GC},	PPRF key for circuit garbling;
	k^{PKE},	PPRF key for public-key encryption;
	$k^{\mathsf{LOT}}_{m_0}$,	PPRF key for sending the m_0^{th} label using laconic OT.
Input.	$i \in [N]$,	index of recipient.

Output. Computed as follows.

$$\begin{aligned}
r_i^{\mathsf{GC}} &\leftarrow \mathsf{PPRF.Eval}(k^{\mathsf{GC}}, i) \\
r_i^{\mathsf{PKE}} &\leftarrow \mathsf{PPRF.Eval}(k^{\mathsf{PKE}}, i) \\
r_{i,m_0}^{\mathsf{LOT}} &\leftarrow \mathsf{PPRF.Eval}(k^{\mathsf{LOT}}_{m_0}, i) \quad \text{for } m_0 \in [M_0] \\
(\widehat{C}_{\mathsf{ct},i}, \{L_{i,m_0,b}\}_{m_0 \in [M_0], b \in \{0,1\}}) & \\
&\leftarrow \begin{cases} \mathsf{GC.Garble}(\widehat{C}_{\mathsf{ct}}, (\mu_\perp, r_i^{\mathsf{PKE}}); r_i^{\mathsf{GC}}), & \text{if } i \le i_\perp; \\ \mathsf{GC.Garble}(\widehat{C}_{\mathsf{ct}}, (\mu\ \ , r_i^{\mathsf{PKE}}); r_i^{\mathsf{GC}}), & \text{if } i > i_\perp; \end{cases} \\
\mathsf{LOT.ct}_{i,m_0} &\leftarrow \mathsf{LOT.Send}(\mathsf{hk}, h, (i-1)M_0 + m_0, \\
&\qquad\qquad L_{i,m_0,0}, L_{i,m_0,1}; r_{i,m_0}^{\mathsf{LOT}}) \quad \text{for } m_0 \in [M_0]
\end{aligned}$$

output $(\widehat{C}_{\mathsf{ct},i}, \{\mathsf{LOT.ct}_{i,m_0}\}_{m_0 \in [M_0]})$

$$C_{\mathsf{ct}}[\mu_i', r_i^{\mathsf{PKE}}](\mathsf{pk}_i)$$

Hardwired.	μ_i',	message or cut-off message;
	r_i^{PKE},	public-key encryption randomness.
Input.	pk_i,	public key of recipient.
Output.	$\mathsf{PKE.ct}_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_i, \mu_i'; r_i^{\mathsf{PKE}})$.	

Figure 1. The circuits C_{GC} and C_{ct} in Construction 1.

Robustness Correctness. This can be verified by inspection.

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

$$T_{\mathsf{Enc}} = \mathrm{O}(N)\,\mathrm{poly}(\lambda, \log N), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, \log N), \quad T_{\mathsf{Dec}} = \mathrm{O}(N)\,\mathrm{poly}(\lambda, \log N).$$

In practice and for security reasons, we always assume $N \le 2^{\lambda}$ and $\log N$ is absorbed by λ. Therefore, with $\mathrm{poly}(\lambda)$ 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 (¶). *Suppose in Construction 1, the obfuscator* Obf *is an* $i\mathcal{O}$ *for* $\mathrm{poly}(\lambda)$*-sized domain, then the resultant AH-PLBE is message-hiding.*

Proof (Theorem 3). For Construction 1, the only difference between $\mathsf{Exp}^0_{\mathrm{MH}}$ and $\mathsf{Exp}^1_{\mathrm{MH}}$ is whether C_{GC} used to create $\mathsf{ct} = (\mathsf{hk}, \widetilde{C}_{\mathsf{GC}})$ has μ_0 or μ_1 hardwired as μ. In C_{GC} (Figure 1), μ is used only in the branch $i > i_\perp$, which is never taken in $\mathsf{Exp}^0_{\mathrm{MH}}$ or $\mathsf{Exp}^1_{\mathrm{MH}}$ because $i_\perp$ is hardwired to be N and the domain of i is $[N]$. Therefore, the two C_{GC}'s in $\mathsf{Exp}^0_{\mathrm{MH}}$ and $\mathsf{Exp}^1_{\mathrm{MH}}$ being obfuscated are functionally equivalent and have the same size. Moreover, their domain size is N (polynomially large). Therefore, $\mathsf{Exp}^0_{\mathrm{MH}} \approx \mathsf{Exp}^1_{\mathrm{MH}}$ reduces to the $i\mathcal{O}$ security for $\mathrm{poly}(\lambda)$-sized domain of Obf. □

4.3 Index-Hiding Property

Theorem 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 $\mathsf{Exp}^0_{\mathrm{IH}}$ and $\mathsf{Exp}^1_{\mathrm{IH}}$ is whether the C_{GC} being obfuscated hardwires μ (in $\mathsf{Exp}^0_{\mathrm{IH}}$) or $\mu_\perp$ (in $\mathsf{Exp}^1_{\mathrm{IH}}$) into $C_{\mathsf{ct}, i^*_\perp}$, which only affects the output of C_{GC} at $i = i^*_\perp$. We consider the following hybrids, each (except the first) described by the changes from the previous one:

- H^b_0 (for $b \in \{0,1\}$) is $\mathsf{Exp}^b_{\mathrm{IH}}$, where

$$\begin{gathered}
\mathsf{hk} \xleftarrow{\$} \mathsf{LOT.Gen}(NM_0), \qquad (h, \widehat{D}) \xleftarrow{\$} \mathsf{LOT.Hash}(\mathsf{hk}, \mathsf{pk}^*_1 \| \cdots \| \mathsf{pk}^*_N), \\
k^{\mathsf{GC}} \xleftarrow{\$} \{0,1\}^{\lambda}, \qquad k^{\mathsf{PKE}} \xleftarrow{\$} \{0,1\}^{\lambda}, \qquad k^{\mathsf{LOT}}_{m_0} \xleftarrow{\$} \{0,1\}^{\lambda} \quad \text{for } m_0 \in [M_0], \\
\widetilde{C}_{\mathsf{GC}} \xleftarrow{\$} \mathsf{Obf}(C_{\mathsf{GC}}[N, \mathsf{hk}, h, i^*_\perp - 1 + b, \mu_\perp, \mu, k^{\mathsf{GC}}, k^{\mathsf{PKE}}, \{k^{\mathsf{LOT}}_{m_0}\}_{m_0 \in [M_0]}]), \\
\mathsf{ct} = (\mathsf{hk}, \widetilde{C}_{\mathsf{GC}}).
\end{gathered}$$

$C'_{\mathsf{GC}}[N, \mathsf{hk}, h, \mu_\perp, \mu, i^*_\perp, \mathring{k}^{\mathsf{GC}}_{i^*_\perp}, \mathring{k}^{\mathsf{PKE}}_{i^*_\perp}, \{\mathring{k}^{\mathsf{LOT}}_{m_0,i^*_\perp}\}_{m_0\in[M_0]}, \widehat{C}_{\mathsf{ct},i^*_\perp}, \{\mathsf{LOT.ct}_{i^*_\perp,m_0}\}_{m_0\in[M_0]}](i)$

Hardwired.	$N, \mathsf{hk}, h, \mu_\perp, \mu,$	see Figure 1;
	$i^*_\perp,$	challenge cut-off index;
	$\mathring{k}^{\cdots}_{\cdots,i^*_\perp},$	PPRF keys punctured at $i^*_\perp$;
	$\widehat{C}_{\mathsf{ct},i^*_\perp}, \mathsf{LOT.ct}_{i^*_\perp,\dots},$	hardwired output of C'_{GC} at $i = i^*_\perp$;
Input.	$i \in [N],$	index of recipient.

Output. Computed as follows.

```
if $\boxed{i = i^*_\perp}$:
    output $(\widehat{C}_{\mathsf{ct},i^*_\perp}, \{\mathsf{LOT.ct}_{i^*_\perp,m_0}\}_{m_0\in[M_0]})$ as hardwired
else:
    $r^{\mathsf{GC}}_i \leftarrow \mathsf{PPRF.Eval}(\mathring{k}^{\mathsf{GC}}_{i^*_\perp}, i)$
    $r^{\mathsf{PKE}}_i \leftarrow \mathsf{PPRF.Eval}(\mathring{k}^{\mathsf{PKE}}_{i^*_\perp}, i)$
    $r^{\mathsf{LOT}}_{i,m_0} \leftarrow \mathsf{PPRF.Eval}(\mathring{k}^{\mathsf{LOT}}_{m_0,i^*_\perp}, i)$   for $m_0 \in [M_0]$
    $(\widehat{C}_{\mathsf{ct},i}, \{L_{i,m_0,b}\}_{m_0\in[M_0],b\in\{0,1\}}) \leftarrow \begin{cases} \mathsf{GC.Garble}(\widehat{C}_{\mathsf{ct}}, (\mu_\perp, r^{\mathsf{PKE}}_i); r^{\mathsf{GC}}_i), & \text{if } \boxed{i < i^*_\perp}; \\ \mathsf{GC.Garble}(\widehat{C}_{\mathsf{ct}}, (\mu\ \ , r^{\mathsf{PKE}}_i); r^{\mathsf{GC}}_i), & \text{if } \boxed{i > i^*_\perp}; \end{cases}$
    $\mathsf{LOT.ct}_{i,m_0} \leftarrow \mathsf{LOT.Send}(\mathsf{hk}, h, (i-1)M_0 + m_0, L_{i,m_0,0}, L_{i,m_0,1}; r^{\mathsf{LOT}}_{i,m_0})$   for $m_0 \in [M_0]$
    output $(\widehat{C}_{\mathsf{ct},i}, \{\mathsf{LOT.ct}_{i,m_0}\}_{m_0\in[M_0]})$
```

Figure 2. The circuit C'_{GC} in the proof of Theorem 4.

- H^b_1 alters the obfuscation into

$$\widetilde{C}_{\mathsf{GC}} \xleftarrow{\$} \mathsf{Obf}(C'_{\mathsf{GC}}[N, \mathsf{hk}, h, \mu_\perp, \mu, i^*_\perp, \mathring{k}^{\mathsf{GC}}_{i^*_\perp}, \mathring{k}^{\mathsf{PKE}}_{i^*_\perp}, \{\mathring{k}^{\mathsf{LOT}}_{m_0,i^*_\perp}\}_{m_0\in[M_0]}, \widehat{C}_{\mathsf{ct},i^*_\perp}, \{\mathsf{LOT.ct}_{i^*_\perp,m_0}\}_{m_0\in[M_0]}]),$$

  where

  - C'_{GC} is defined in Figure 2,
  - the PPRF keys are punctured at $i^*_\perp$ by running

$$\mathring{k}^{\mathsf{GC}}_{i^*_\perp} \xleftarrow{\$} \mathsf{PPRF.Puncture}(k^{\mathsf{GC}}, i^*_\perp),$$
$$\mathring{k}^{\mathsf{PKE}}_{i^*_\perp} \xleftarrow{\$} \mathsf{PPRF.Puncture}(k^{\mathsf{PKE}}, i^*_\perp),$$
$$\mathring{k}^{\mathsf{LOT}}_{m_0,i^*_\perp} \xleftarrow{\$} \mathsf{PPRF.Puncture}(k^{\mathsf{LOT}}_{m_0}, i^*_\perp) \qquad \text{for } m_0 \in [M_0],$$

  - and the output $(\widehat{C}_{\mathsf{ct},i^*_\perp}, \{\mathsf{LOT.ct}_{i^*_\perp,m_0}\}_{m_0\in[M_0]})$ of C'_{GC} at $i = i^*_\perp$ is computed as

$$r^{\mathsf{GC}} \leftarrow \mathsf{PPRF.Eval}(k^{\mathsf{GC}}, i^*_\perp), \qquad r^{\mathsf{PKE}} \leftarrow \mathsf{PPRF.Eval}(k^{\mathsf{PKE}}, i^*_\perp),$$
$$r^{\mathsf{LOT}}_{i^*_\perp,m_0} \leftarrow \mathsf{PPRF.Eval}(k^{\mathsf{LOT}}_{m_0}, i^*_\perp) \qquad \text{for } m_0 \in [M_0],$$

$$(\widehat{C}_{\mathsf{ct},i_\perp^*}, \{L_{i_\perp,m_0,b}\}_{m_0\in[M_0],b\in\{0,1\}}) \leftarrow \begin{cases} \mathsf{GC.Garble}(C_{\mathsf{ct}}, (\mu\ \ , r_{i_\perp^*}^{\mathsf{PKE}}); r_{i_\perp^*}^{\mathsf{GC}}), & \text{if } b = 0; \\ \mathsf{GC.Garble}(C_{\mathsf{ct}}, (\mu_\perp, r_{i_\perp^*}^{\mathsf{PKE}}); r_{i_\perp^*}^{\mathsf{GC}}), & \text{if } b = 1; \end{cases}$$

$$\mathsf{LOT.ct}_{i_\perp^*,m_0} \leftarrow \mathsf{LOT.Send}(\mathsf{hk}, h, (i_\perp^* - 1)M_0 + m_0, L_{i_\perp^*,m_0,0}, L_{i_\perp^*,m_0,1}; r_{i_\perp^*,m_0}^{\mathsf{LOT}}) \qquad \text{for } m_0 \in [M_0].$$

- H_2^b changes $r_{i_\perp^*}^{\mathsf{GC}}$, $r_{i_\perp^*}^{\mathsf{PKE}}$, and $r_{i_\perp^*,m_0}^{\mathsf{LOT}}$'s into true randomness, i.e.,

$$r^{\mathsf{GC}} \xleftarrow{\$} \{0,1\}^\lambda, \qquad r^{\mathsf{PKE}} \xleftarrow{\$} \{0,1\}^\lambda, \qquad r_{i_\perp^*,m_0}^{\mathsf{LOT}} \xleftarrow{\$} \{0,1\}^\lambda \quad \text{for } m_0 \in [M_0].$$

- H_3^b removes the unused labels from $\mathsf{LOT.ct}_{i_\perp^*,m_0}$'s by setting

$$\mathsf{LOT.ct}_{i_\perp^*,m_0} \leftarrow \mathsf{LOT.Send}(\mathsf{hk}, h, (i_\perp^* - 1)M_0 + m_0, L_{i_\perp^*,m_0,\mathsf{pk}_{i_\perp^*}[m_0]}, L_{i_\perp^*,m_0,\mathsf{pk}_{i_\perp^*}[m_0]}; r_{i_\perp^*,m_0}^{\mathsf{LOT}}) \qquad \text{for } m_0 \in [M_0].$$

- H_4^b changes $\widehat{C}_{\mathsf{ct},i_\perp^*}$ into simulation, i.e.,

$$\mathsf{PKE.ct}_{i_\perp^*} \leftarrow \begin{cases} \mathsf{PKE.Enc}(\mathsf{pk}_{i_\perp^*}^*, \mu\ \ ; r^{\mathsf{PKE}}), & \text{if } b = 0; \\ \mathsf{PKE.Enc}(\mathsf{pk}_{i_\perp^*}^*, \mu_\perp; r^{\mathsf{PKE}}), & \text{if } b = 1; \end{cases}$$

$$(\widehat{C}_{\mathsf{ct},i_\perp^*}, \{L_{i_\perp,m_0,\mathsf{pk}_{i_\perp^*}^*[m_0]}\}_{m_0\in[M_0]}) \xleftarrow{\$} \mathsf{GC.SimGarble}(C_{\mathsf{ct}}, \mathsf{pk}_{i_\perp^*}^*, \mathsf{PKE.ct}_{i_\perp^*}),$$

  where $\mathsf{pk}_{i_\perp^*}^* = \mathsf{pk}$ is sampled by the experiment (not adversarially controlled).

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

Claim 5. $\mathsf{H}_0^b \approx \mathsf{H}_1^b$ *for* $b \in \{0,1\}$ *if* Obf *is an* $i\mathcal{O}$ *for* $\mathsf{poly}(\lambda)$*-sized domain.*

Claim 6. $\mathsf{H}_1^b \approx \mathsf{H}_2^b$ *for* $b \in \{0,1\}$ *if* PPRF *is pseudorandom at the punctured point.*

Claim 7. $\mathsf{H}_2^b \approx \mathsf{H}_3^b$ *for* $b \in \{0,1\}$ *if* LOT *is database-selectively sender-private.*

Claim 8. $\mathsf{H}_3^b \approx \mathsf{H}_4^b$ *for* $b \in \{0,1\}$ *if* GC *is* w*-hiding.*

Claim 9. $\mathsf{H}_4^0 \approx \mathsf{H}_4^1$ *if* PKE *is semantically secure for random messages.*

$\mathsf{Exp}_{\mathrm{IH}}^0 \approx \mathsf{Exp}_{\mathrm{IH}}^1$ follows from a hybrid argument. □

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.
- $\mathsf{Enc}(\{\mathsf{pk}_j\}_{j\in[N]}, \mu)$ runs and outputs $\mathsf{ct} \xleftarrow{\$} \mathsf{ahPLBE.Enc}(\{\mathsf{pk}_j\}_{j\in[N]}, 0, \mu)$.

- Dec is the same as ahPLBE.Dec.
- $\mathsf{Trace}^{\mathcal{D}}(\{\mathsf{pk}_j^*\}_{j\in[N]}, 1^{1/\varepsilon^*})$ defines for $i \in [0..N]$,

$$\varepsilon_i = \Pr\underbrace{\left[\begin{matrix}\mu_0 \xleftarrow{\$} \{0,1\}^\lambda, \quad \mu_1 \xleftarrow{\$} \{0,1\}^\lambda, \quad \beta \xleftarrow{\$} \{0,1\} \\ \mathsf{ct} \xleftarrow{\$} \mathsf{ahPLBE.Enc}(1^\lambda, \{\mathsf{pk}_j^*\}_{j\in[N]}, i, \mu_\beta)\end{matrix} : \mathcal{D}(\mu_0,\mu_1,\mathsf{ct}) = \beta\right]}_{\text{experiment } \mathcal{E}_i \text{ (sampling and testing) and event } E_i \text{ (correct guessing)}} - \frac{1}{2}.$$

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

$$i^* = \begin{cases} \min T, & \text{if } T \leftarrow \{\, i \in [N] \,:\, |\widehat{\varepsilon}_i - \widehat{\varepsilon}_{i-1}| \geq 3\delta \,\} \neq \varnothing; \\ \perp, & \text{if } T = \varnothing. \end{cases}$$

Robustness Correctness, Efficiency, Compatibility. These are inherited from the underlying AH-PLBE. When based on Construction 1, the resultant AH-BTR has

$$T_{\mathsf{Enc}} = \mathrm{O}(N)\,\mathrm{poly}(\lambda), \qquad |\mathsf{ct}| = \mathrm{poly}(\lambda), \qquad T_{\mathsf{Dec}} = \mathrm{O}(N)\,\mathrm{poly}(\lambda),$$

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

Theorem 10 (¶)**.** *Suppose in Construction 2, the AH-PLBE scheme* ahPLBE *is message-hiding, then the resultant AH-BTR is complete.*

Theorem 11 (¶)**.** *Suppose in Construction 2, the AH-PLBE scheme* ahPLBE *is index-hiding, then the resultant AH-BTR is sound.*

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

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

Let BadEnd be the event that $|\varepsilon_N| > \frac{\varepsilon^*}{2}$, then $\mathsf{GoodDist} \wedge \neg\mathsf{BadEnd}$ implies

$$\begin{aligned}\max_{i\in[N]} |\varepsilon_{i-1} - \varepsilon_i| &\geq \frac{1}{N}\sum_{i=1}^{N} |\varepsilon_{i-1} - \varepsilon_i| \geq \frac{1}{N}\left|\sum_{i=1}^{N}(\varepsilon_{i-1} - \varepsilon_i)\right| = \frac{1}{N}|\varepsilon_0 - \varepsilon_N| \\ &\geq \frac{1}{N}(\underset{\substack{\uparrow \\ \mathsf{GoodDist}}}{|\varepsilon_0|} - \underset{\substack{\uparrow \\ \neg\mathsf{BadEnd}}}{|\varepsilon_N|}) \geq \frac{1}{N}\left(\varepsilon^* - \frac{\varepsilon^*}{2}\right) = \frac{\varepsilon^*}{2N} = 5\delta.\end{aligned}$$

Therefore, $\mathsf{GoodDist} \wedge \neg\mathsf{BadEnd} \wedge \mathsf{GoodEst}$ implies

$$\max_{i\in[N]} |\widehat{\varepsilon}_{i-1} - \widehat{\varepsilon}_i| \underset{\substack{\uparrow \\ \mathsf{GoodEst}}}{\geq} \max_{i\in[N]} (|\varepsilon_{i-1} - \varepsilon_i| - 2\delta) \underset{\substack{\uparrow \\ \mathsf{GoodDist}\wedge\neg\mathsf{BadEnd}}}{\geq} 5\delta - 2\delta = 3\delta,$$

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

$$\mathsf{GoodDist} \wedge \mathsf{NotFound} \wedge \mathsf{GoodEst} \quad \Longrightarrow \quad \mathsf{BadEnd}.$$

By the union bound,

$$\begin{aligned}\Pr[\mathcal{C}\text{ wins}] &\le \Pr[\neg\mathsf{GoodEst}] + \Pr[(\mathcal{C}\text{ wins}) \wedge \mathsf{GoodEst}]\\ &= \Pr[\neg\mathsf{GoodEst}] + \Pr[\mathsf{GoodDist} \wedge \mathsf{NotFound} \wedge \mathsf{GoodEst}]\\ &\le 2^{-\lambda} + \Pr[\mathsf{BadEnd}],\end{aligned}$$

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

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

- $\mathcal{A}$ runs $\mathcal{C}$ to obtain
$$\mathcal{D}, \quad \{\mathsf{pk}_j^*\}_{j\in[N]}, \quad 1^{1/\varepsilon^*}.$$
- $\mathcal{A}$ runs $\mathcal{E}_N$ 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.
- $\mathcal{A}$ submits $\{\mathsf{pk}_j^*\}_{j\in[N]}$ to the message-hiding experiment, receives $(\mu_0, \mu_1, \mathsf{ct})$ back, and runs and outputs $b' \xleftarrow{\$} \mathcal{D}(\mu_0, \mu_1, \mathsf{ct}) \oplus \alpha$.

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

$$\begin{aligned}\Pr[\mathsf{BadEnd}] = \Pr[4\varepsilon_N^2 > (\varepsilon^*)^2] &\le \Pr[4\varepsilon_N^2 > B^{-2}]\\ &\le B^2\,\mathbb{E}[4\varepsilon_N^2] = (\mathrm{poly}(\lambda))^2\,\mathrm{negl}(\lambda) = \mathrm{negl}(\lambda). \qquad \square\end{aligned}$$

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

$$\begin{aligned}\Pr[\mathcal{C}\text{ wins}] &\le \Pr[\neg\mathsf{GoodEst}] + \Pr[(\mathcal{C}\text{ wins}) \wedge \mathsf{GoodEst}]\\ &= \Pr[\neg\mathsf{GoodEst}] + \Pr[\mathsf{FalsePos} \wedge \mathsf{GoodEst}]\\ &\le 2^{-\lambda} + \Pr[\mathsf{FalsePos} \wedge \mathsf{GoodEst}],\end{aligned}$$

and it suffices to prove $\Pr[\mathsf{FalsePos} \wedge \mathsf{GoodEst}] = \mathrm{negl}(\lambda)$.

Let α be a random element in an execution of Trace with

$$\alpha = \begin{cases} 0, & \text{if } i^* \in [N] \text{ and } \widehat{\varepsilon}_{i^*-1} - \widehat{\varepsilon}_{i^*} \ge 3\delta;\\ 1, & \text{if } i^* \in [N] \text{ and } \widehat{\varepsilon}_{i^*-1} - \widehat{\varepsilon}_{i^*} \le -3\delta;\\ \perp, & \text{if } i^* = \perp.\end{cases}$$

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

- $\mathcal{A}(\mathsf{pk})$ runs $\mathcal{C}(\mathsf{pk})$ to obtain
$$\mathcal{D}, \quad N, \quad i_\perp^*, \quad \{\mathsf{pk}_j^*\}_{j\in[N]\setminus\{i_\perp^*\}}, \quad 1^{1/\varepsilon^*},$$
and sets $\mathsf{pk}_{i_\perp^*}^* \leftarrow \mathsf{pk}$.

- $\mathcal{A}$ runs
$$i^* \xleftarrow{\$} \mathsf{Trace}^{\mathcal{D}}(\{\mathsf{pk}^*_j\}_{j\in[N]}, 1^{1/\varepsilon^*}),$$
and aborts if $i^* \neq i^*_\perp$.
- $\mathcal{A}$ notes down $\alpha \in \{0,1\}$ from the above execution of Trace, submits
$$N, \quad i^*_\perp, \quad \{\mathsf{pk}^*_j\}_{j\in[N]\setminus\{i^*_\perp\}}$$
to the index-hiding experiment, gets (μ, ct) back, samples and sets
$$\beta \xleftarrow{\$} \{0,1\}, \qquad \mu_\beta \leftarrow \mu, \quad \mu_{\neg\beta} \xleftarrow{\$} \{0,1\}^\lambda,$$
and runs and outputs $b' \xleftarrow{\$} \mathcal{D}(\mu_0, \mu_1, \mathsf{ct}) \oplus \neg\beta \oplus \alpha$.

Routine calculation shows that the advantage of $\mathcal{A}$ is
$$\mathbb{E}[\mathbb{1}_{\mathsf{FalsePos}} \cdot (-1)^\alpha(\varepsilon_{i^*-1} - \varepsilon_{i^*})],$$
which must be negligible by the index-hiding property of ahPLBE.

Let $B = \mathrm{poly}(\lambda)$ be an upper bound of $10N/\varepsilon^*$ (B exists since $\mathcal{C}$ outputs 1^N and $1^{1/\varepsilon^*}$ in polynomial time). The event $\mathsf{FalsePos} \wedge \mathsf{GoodEst}$ implies
$$\begin{aligned} &|(\varepsilon_{i^*-1} - \varepsilon_{i^*}) - (\widehat{\varepsilon}_{i^*-1} - \widehat{\varepsilon}_{i^*})| \le 2\delta < 3\delta \le |\widehat{\varepsilon}_{i^*-1} - \widehat{\varepsilon}_{i^*}| \\ \implies \quad &(-1)^\alpha(\varepsilon_{i^*-1} - \varepsilon_{i^*}) = |\varepsilon_{i^*-1} - \varepsilon_{i^*}| \ge 3\delta - 2\delta = \frac{\varepsilon^*}{10N} \ge B^{-1}. \end{aligned}$$
Moreover, $(-1)^\alpha(\varepsilon_{i^*-1} - \varepsilon_{i^*}) \ge -1$ always holds. These together show that
$$\begin{aligned} &\Pr[\mathsf{FalsePos} \wedge \mathsf{GoodEst}] \\ &= B\,\mathbb{E}[\mathbb{1}_{\mathsf{FalsePos}} \cdot \mathbb{1}_{\mathsf{GoodEst}} \cdot B^{-1}] \\ &\le B\,\mathbb{E}[\mathbb{1}_{\mathsf{FalsePos}} \cdot \mathbb{1}_{\mathsf{GoodEst}} \cdot (-1)^\alpha(\varepsilon_{i^*-1} - \varepsilon_{i^*})] \\ &\le B\Big(\mathbb{E}[\mathbb{1}_{\mathsf{FalsePos}} \cdot \mathbb{1}_{\mathsf{GoodEst}} \cdot (-1)^\alpha(\varepsilon_{i^*-1} - \varepsilon_{i^*})] \\ &\qquad + \mathbb{E}[\mathbb{1}_{\mathsf{FalsePos}} \cdot \mathbb{1}_{\neg\mathsf{GoodEst}} \cdot (-1)^\alpha(\varepsilon_{i^*-1} - \varepsilon_{i^*})] + \mathbb{E}[\mathbb{1}_{\mathsf{FalsePos}} \cdot \mathbb{1}_{\neg\mathsf{GoodEst}}]\Big) \\ &= B\Big(\mathbb{E}[\mathbb{1}_{\mathsf{FalsePos}} \cdot (-1)^\alpha(\varepsilon_{i^*-1} - \varepsilon_{i^*})] + \Pr[\mathsf{FalsePos} \wedge \neg\mathsf{GoodEst}]\Big) \\ &\le B\Big(\mathbb{E}[\mathbb{1}_{\mathsf{FalsePos}} \cdot (-1)^\alpha(\varepsilon_{i^*-1} - \varepsilon_{i^*})] + 2^{-\lambda}\Big) \\ &= \mathrm{poly}(\lambda)\big(\mathrm{negl}(\lambda) + 2^{-\lambda}\big) = \mathrm{negl}(\lambda). \end{aligned}$$
□

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.[10] Previous work [Zha20a] already systemizes the idea of grouping in the context of traditional traitor tracing.

[10] 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.

Ingredients of Construction 3. Let old = (old.Gen, old.Enc, old.Dec, old.Trace) be an AH-BTR scheme and γ some[11] constant ($0 < \gamma < 1$).

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

- Gen is the same as old.Gen.
- $\mathsf{Enc}(\{\mathsf{pk}_j\}_{j\in[N]}, \mu)$ sets $N_1 = \lceil N^\gamma \rceil$ and $N_2 = \lceil N/N_1 \rceil$. It runs
$$\mathsf{old.ct}_{j_1} \xleftarrow{\$} \mathsf{old.Enc}(\{\mathsf{pk}_j\}_{(j_1-1)N_2<j\le j_1N_2}, \mu) \qquad \text{for } j_1 \in [N_1].$$
The algorithm outputs $\mathsf{ct} = \{\mathsf{old.ct}_{j_1}\}_{j_1\in[N_1]}$.
- $\mathsf{Dec}^{\{\mathsf{pk}_j\}_{j\in[N]},\mathsf{ct}}(N, i, \mathsf{sk}_i)$ sets $N_1 = \lceil N^\gamma \rceil$, $N_2 = \lceil N/N_1 \rceil$. It parses ct as $\{\mathsf{old.ct}_{j_1}\}_{j_1\in[N_1]}$, finds $i_1 \in [N_1]$ such that $(i_1-1)N_2 < i \le i_1N_2$, and sets $N_2' = \min\{N_2, N-(i_1-1)N_2\}$. The algorithm runs and outputs
$$\mathsf{old.Dec}^{\{\mathsf{pk}_j\}_{(i_1-1)N_2<j\le i_1N_2},\mathsf{old.ct}_{i_1}}(N_2', i-(i_1-1)N_2, \mathsf{sk}_i).$$
- $\mathsf{Trace}^{\mathcal{D}}(\{\mathsf{pk}_j^*\}_{j\in[N]}, 1^{1/\varepsilon^*})$ sets $N_1 = \lceil N^\gamma \rceil$ and $N_2 = \lceil N/N_1 \rceil$. It runs
$$i_{j_1}^* \xleftarrow{\$} \mathsf{old.Trace}^{\mathcal{D}_{j_1}}(\{\mathsf{pk}_j^*\}_{(j_1-1)N_2<j\le j_1N_2}, 1^{N_1/\varepsilon^*}) \qquad \text{for } j_1 \in [N_1],$$
where $\mathcal{D}_{j_1}(\mu_0, \mu_1, \mathsf{old.ct}^*)$ runs and outputs $\mathcal{D}(\mu_0, \mu_1, \{\mathsf{old.ct}_{j_1'}\}_{j_1'\in[N_1]})$ with
$$\mathsf{old.ct}_{j_1'} \begin{cases} \xleftarrow{\$} \mathsf{old.Enc}(\{\mathsf{pk}_j^*\}_{(j_1'-1)N_2<j\le j_1'N_2}, \mu_0), & \text{if } j_1' < j_1; \\ \leftarrow \mathsf{old.ct}^*, & \text{if } j_1' = j_1; \\ \xleftarrow{\$} \mathsf{old.Enc}(\{\mathsf{pk}_j^*\}_{(j_1'-1)N_2<j\le j_1'N_2}, \mu_1), & \text{if } j_1' > j_1. \end{cases}$$
The algorithm outputs
$$\begin{cases} (j_1-1)N_2 + i_{j_1}^*, & \text{if } i_{j_1'}^* = \perp \text{ for all } j_1' < j_1 \text{ and } i_{j_1}^* \ne \perp; \\ \perp, & \text{if } i_{j_1'}^* = \perp \text{ for all } j_1' \in [N_1]. \end{cases}$$

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
$$T_{\mathsf{Enc}} = \mathrm{O}(N^{\gamma_1})\,\mathrm{poly}(\lambda), \qquad |\mathsf{ct}| = \mathrm{O}(N^{\gamma_2})\,\mathrm{poly}(\lambda), \qquad T_{\mathsf{Dec}} = \mathrm{O}(N^{\gamma_3})\,\mathrm{poly}(\lambda),$$
then the underlying efficiency is mapped to the resultant efficiency[12] 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
$$T_{\mathsf{Enc}} = \mathrm{O}(N)\,\mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \mathrm{O}(N^{1-\gamma})\,\mathrm{poly}(\lambda), \quad T_{\mathsf{Dec}} = \mathrm{O}(N^{\gamma})\,\mathrm{poly}(\lambda).$$

[11]We require that $N \mapsto \lceil N^\gamma \rceil$ can be computed in (deterministic) time $\mathrm{poly}(\log N)$.
[12]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.

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

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

Proof (Theorem 12). Let $\mathcal{C}$ be an efficient adversary against the completeness of the resultant scheme. Consider the following efficient adversary $\mathcal{C}_{\mathsf{old}}$ against the completeness of old:

- $\mathcal{C}_{\mathsf{old}}$ launches $\mathcal{C}$ to obtain
$$\mathcal{D}, \quad \{\mathsf{pk}_j^*\}_{j\in[N]}, \quad 1^{1/\varepsilon^*}.$$
It computes N_1, N_2 as specified by the resultant scheme.
- $\mathcal{C}_{\mathsf{old}}$ samples $j_1^* \xleftarrow{\$} [N_1]$, prepares $\mathcal{D}_{j_1^*}$ (using $\mathcal{D}$, as specified by the resultant scheme), and outputs
$$\mathcal{D}_{j_1^*}, \quad \{\mathsf{pk}_j^*\}_{(j_1^*-1)N_2<j\le j_1^*N_2}, \quad 1^{N_1/\varepsilon^*}.$$

Let $B = \mathrm{poly}(\lambda)$ be an upper bound of N_1. Routine calculation shows
$$\Pr[\mathcal{C}_{\mathsf{old}} \text{ wins}] \ge \frac{1}{B}\Pr[\mathcal{C} \text{ wins}],$$
hence by the completeness of old,
$$\Pr[\mathcal{C} \text{ wins}] \le B\Pr[\mathcal{C}_{\mathsf{old}} \text{ wins}] = \mathrm{poly}(\lambda)\,\mathrm{negl}(\lambda) = \mathrm{negl}(\lambda). \qquad \square$$

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

- $\mathcal{C}_{\mathsf{old}}(\mathsf{pk})$ launches $\mathcal{C}(\mathsf{pk})$ to obtain
$$\mathcal{D}, \quad N, \quad i_\perp^*, \quad \{\mathsf{pk}_j^*\}_{j\in[N]\setminus\{i_\perp^*\}}, \quad 1^{1/\varepsilon^*}.$$
It computes N_1, N_2 as specified by the resultant scheme.
- $\mathcal{C}_{\mathsf{old}}$ computes $j_1^* = \lceil i_\perp^*/N_2\rceil$ and outputs
$$\mathcal{D}_{j_1^*}, \quad \min\{N_2, N-(j_1^*-1)N_2\}, \quad i_\perp^*-(j_1^*-1)N_2, \quad \{\mathsf{pk}_j^*\}_{(j_1^*-1)N_2<j\le j_1^*N_2,\, j\ne i_\perp^*}, \quad 1^{N_1/\varepsilon^*}.$$

Routine calculation and the soundness of old yield
$$\Pr[\mathcal{C} \text{ wins}] \le \Pr[\mathcal{C}_{\mathsf{old}} \text{ wins}] = \mathrm{negl}(\lambda). \qquad \square$$

7 Lower Bound on Ciphertext Size and Decryption Time

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

$$|\mathsf{ct}| \cdot T_{\mathsf{Dec}} = \Omega(N),$$

and therefore, we have constructed all the optimal (ignoring poly(λ) factors) AH-BTR schemes in this work. In fact, we will show a related bound against a restricted kind of broadcast encryption,[13] 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.

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:

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

The scheme must be *correct*, i.e., for all $\lambda, N \in \mathbb{N}$, $R \in \{0,1\}^N$, $j \in [N]$, $\mu \in \{0,1\}^\lambda$,

$$\Pr\left[\begin{array}{rl} (\mathsf{mpk}, \{\mathsf{sk}_{j,b}\}_{j\in[N],b\in\{0,1\}}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^N) \\ \mathsf{ct}_R \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, R, \mu) \\ : \quad \mathsf{Dec}^{\mathsf{mpk},j,R[j],\mathsf{sk}_{j,R[j]},R,\mathsf{ct}_R}(1^\lambda) = \mu \end{array}\right] = 1.$$

The scheme is *1-key secure for random challenge against uniform adversaries* (or *secure* for the purpose of this work) if

$$\begin{aligned} &\big\{(1^\lambda, \mathsf{mpk}, R, i^*, \mu_0, \mathsf{sk}_{i^*,\neg R[i^*]}, \mathsf{ct}_0)\big\} \approx \big\{(\cdots, \mathsf{ct}_1)\big\}, \text{ where} \\ &R \xleftarrow{\$} \{0,1\}^N, \quad i^* \xleftarrow{\$} [N], \\ &(\mathsf{mpk}, \{\mathsf{sk}_{j,b}\}_{j\in[N],b\in\{0,1\}}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^N), \\ &\mu_b \xleftarrow{\$} \{0,1\}^\lambda, \quad \mathsf{ct}_b \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, R, \mu_b) \quad \text{for } b \in \{0,1\}, \end{aligned}$$

for all polynomially bounded $N = N(\lambda)$, where the computational indistinguishability only has to hold against *uniform* adversaries.[14]

[13] This result is thus also a lower bound for general attribute-based encryption.

[14] N need not be a computable function of λ. This does not make the security definition "non-uniformy", as a standard guessing argument (with advantage sign correction) applies to an interactive formulation in which the uniform and efficient $\mathcal{A}$ chooses N.

Theorem 14 (¶)**.** *For all secure restricted BE,*

$$\max|\mathsf{ct}|\cdot\max T_{\mathsf{Dec}}\geq\frac{N}{1000}$$

for all polynomially bounded $N = N(\lambda)$ and sufficiently large λ, where ct *runs through all possible ciphertexts and T_{Dec} the time to probe R and produce output by* 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 straight-forward 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 $F : Z \times \{0,1\}^N \to \{0,1\}^S$, there exists a function $G : Z \times \{0,1\}^N \to \{0,1,\perp\}^N$ such that*

$$|\{j\in[N]:G(z,j)\neq\perp\}|\leq P\qquad\text{for all } z\in Z$$

and for all (non-efficient) oracle (randomized) algorithm $\mathcal{B}^Y$ making at most T queries to Y,

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

where

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

Proof (Theorem 14). Define

$$S=1+\max|\mathsf{ct}|,\qquad T=1+\max\{\text{number of bits in } R \text{ probed by } \mathsf{Dec}\}.$$

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

$$\max|\mathsf{ct}|\cdot\max T_{\mathsf{Dec}}\geq\frac{\max|\mathsf{ct}|+1}{2}\cdot\max T_{\mathsf{Dec}}\geq\frac{ST}{2}.$$

It remains to prove $ST \geq \frac{2N}{1000}$ for sufficiently large λ. 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

$$z=\begin{pmatrix}\mu, z_{\mathsf{Enc}}, \mathsf{mpk},\\ \{\mathsf{sk}_{j,b}\}_{j\in[N],b\in\{0,1\}}\end{pmatrix}\sim D=\left\{\begin{matrix}\mu\xleftarrow{\$}\{0,1\}^\lambda,\qquad i^*\xleftarrow{\$}[N]\\ z_{\mathsf{Enc}}:\text{randomness for }\mathsf{Enc}\\ (\mathsf{mpk},\{\mathsf{sk}_{j,b}\}_{j\in[N],b\in\{0,1\}})\xleftarrow{\$}\mathsf{Gen}(1^N)\end{matrix}\right\},$$

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

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

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

Note that $\mathcal{B}$ indeed makes at most T queries to Y, the first to obtain b^* and the rest to run Dec.

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

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

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

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

$$Y_4[j] \begin{cases} = R[j], & \text{if } j \neq i^*; \\ \xleftarrow{\$} \{0,1\}, & \text{if } j = i^*; \end{cases} \qquad Y_5[j] \begin{cases} = R[j], & \text{if } j \neq i^*; \\ = \neg R[i^*], & \text{if } j = i^*. \end{cases}$$

By the correctness of the restricted BE scheme, $p_1 = 1$.

From Lemma 15,

$$|p_1 - p_2| \leq \sqrt{\frac{ST}{2P}}, \qquad |p_4 - p_3| \leq \sqrt{\frac{ST}{2P}}.$$

Here, the second inequality is obtained by applying the lemma to

$$\mathcal{C}^Y(z,f) = \mathcal{B}^{Y'}(z,f;i^*), \qquad \text{where } i^* \xleftarrow{\$} [N], \quad Y'[j] \begin{cases} = Y[j], & \text{if } j \neq i^*; \\ \xleftarrow{\$} \{0,1\}, & \text{if } j = i^*. \end{cases}$$

Clearly, $|p_2 - p_3| \leq \frac{P}{N}$. Setting $P = \left\lceil \sqrt[3]{\frac{STN^2}{2}} \right\rceil$, we have

$$\begin{aligned} |p_1 - p_4| &\leq |p_1 - p_2| + |p_2 - p_3| + |p_3 - p_4| \\ &\leq \sqrt{\frac{ST}{2P}} + \frac{P}{N} + \sqrt{\frac{ST}{2P}} \leq 3\sqrt[3]{\frac{ST}{2N}} + \frac{1}{N} < 4\sqrt[3]{\frac{ST}{2N}}, \end{aligned}$$

where the last inequality follows from $N \geq 2$. By how $Y[i^*]$ is set,

$$p_4 = \frac{p_1 + p_5}{2} \qquad \Longrightarrow \qquad p_5 = p_1 - 2(p_1 - p_4) \geq p_1 - 2|p_1 - p_4| > 1 - 8\sqrt[3]{\frac{ST}{2N}}.$$

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

- Construct Y_5 from R and let $b^* \leftarrow Y_5[i^*] = \neg R[i^*]$.

- Run $\mu' \xleftarrow{\$} \mathsf{Dec}^{\mathsf{mpk},i^*,b^*,\mathsf{sk}_{i^*,b^*},Y_5,\mathsf{ct}}()$, i.e., pretend $R[i^*]$ were $\neg R[i^*]$ and try decrypting using the key given to $\mathcal{A}$.
- Output 1 if and only if $\mu' = \mu$.

If $\mathsf{ct} = \mathsf{ct}_1$ is an encryption of μ_1, then μ_0 is uniformly random and independent of everything else, hence

$$\Pr[\mathcal{A}(\cdots) \to 1 \text{ with } \mathsf{ct} = \mathsf{ct}_1] \leq 2^{-\lambda}.$$

Note that $\mathcal{A}$ is a uniform adversary. By the security of the restricted BE scheme,

$$p_5 = \Pr[\mathcal{B}^{Y_5}(z, f; i^*) \to 1] = \Pr[\mathcal{A}(\cdots) \to 1 \text{ with } \mathsf{ct} = \mathsf{ct}_0] \leq 2^{-\lambda} + \mathrm{negl}(\lambda) < \frac{1}{5}$$

for sufficiently large λ, which gives

$$1 - 8\sqrt[3]{\frac{ST}{2N}} < p_5 < \frac{1}{5} \qquad \Longrightarrow \qquad ST > \frac{2N}{1000}. \qquad \square$$

Corollary 16 (¶). *For all secure AH-BTR,*

$$\max|\mathsf{ct}| \cdot \max T_{\mathsf{Dec}} \geq \frac{N}{1000}$$

for all polynomially bounded $N = N(\lambda)$ and sufficiently large λ, where T_{Dec} only counts the time to probe pk_j's and produce output. Ignoring $\mathrm{poly}(\lambda)$ 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.

Proof (Corollary 16). Let $\mathsf{ahBTR} = (\mathsf{ahBTR.Gen}, \mathsf{ahBTR.Enc}, \mathsf{ahBTR.Dec}, \mathsf{ahBTR.Trace})$ be a secure AH-BTR and construct the following restricted BE scheme:

- $\mathsf{Gen}(1^N)$ runs

$$(\mathsf{pk}_{j,b}, \mathsf{sk}_{j,b}) \xleftarrow{\$} \mathsf{ahBTR.Gen}() \qquad \text{for } j \in [N], b \in \{0,1\}$$

  and outputs $\mathsf{mpk} = \{\mathsf{pk}_{j,b}\}_{j\in[N],b\in\{0,1\}}$ with $\{\mathsf{sk}_{j,b}\}_{j\in[N],b\in\{0,1\}}$.

- $\mathsf{Enc}(\mathsf{mpk}, R, \mu)$ runs and outputs

$$\mathsf{ct} \xleftarrow{\$} \mathsf{ahBTR.Enc}(\{\mathsf{pk}_{j,R[j]}\}_{j\in[N]}, \mu).$$

- $\mathsf{Dec}^{\mathsf{mpk},j,b,\mathsf{sk}_{j,b},R,\mathsf{ct}}()$ runs $\mathsf{ahBTR.Dec}^{K,\mathsf{ct}}(N, j, \mathsf{sk}_{j,b})$, where K is an oracle implemented by Dec for $\mathsf{ahBTR.Dec}$ to probe pk_j's. Whenever $\mathsf{ahBTR.Dec}$ probes $\mathsf{pk}_j[m_0]$, we make Dec probe $R[j]$ and answer $\mathsf{pk}_{j,R[j]}[m_0]$.

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

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). He thanks Hoeteck Wee for helpful suggestions on writing from another context. He started researching this topic after rereading a post [sil21] on V2EX.

References

[AL18] Prabhanjan Ananth and Alex Lombardi. Succinct garbling schemes from functional encryption through a local simulation paradigm. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 455–472. Springer, Heidelberg, November 2018.

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 149–178. Springer, Heidelberg, November 2020.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 13–43. Springer, Heidelberg, May 2020.

[BF99] Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353. Springer, Heidelberg, August 1999.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, Heidelberg, August 2005.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012.

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 501–510. ACM Press, October 2008.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, Heidelberg, May / June 2006.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

[BV20] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext-policy ABE. Cryptology ePrint Archive, Report 2020/191, 2020. https://eprint.iacr.org/2020/191.

[BW06] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 211–220. ACM Press, October / November 2006.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014.

[CDG$^+$17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Heidelberg, August 2017.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270. Springer, Heidelberg, August 1994.

[Cha07] Melissa Chase. Multi-authority attribute based encryption. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 515–534. Springer, Heidelberg, February 2007.

[CVW$^+$18] Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from LWE made simple and attribute-based. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 341–369. Springer, Heidelberg, November 2018.

[Del07] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 200–215. Springer, Heidelberg, December 2007.

[DHMR08] Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. *Ad-hoc* threshold broadcast encryption with shorter ciphertexts. *Electronic Notes in Theoretical*

Computer Science, 192(2):3–15, 2008. Proceedings of the Third Workshop on Cryptography for Ad-hoc Networks (WCAN 2007).

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *PAIRING 2007*, volume 4575 of *LNCS*, pages 39–59. Springer, Heidelberg, July 2007.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer, Heidelberg, August 1994.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

[GK16] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 505–522. Springer, Heidelberg, January 2016.

[GKRW18] Rishab Goyal, Venkata Koppula, Andrew Russell, and Brent Waters. Risky traitor tracing and new differential privacy negative results. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 467–497. Springer, Heidelberg, August 2018.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 660–670. ACM Press, June 2018.

[GKW19] Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 149–179. Springer, Heidelberg, December 2019.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

[GQWW19] Rishab Goyal, Willy Quach, Brent Waters, and Daniel Wichs. Broadcast and trace with N^{ε} ciphertext size from standard assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 826–855. Springer, Heidelberg, August 2019.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 171–188. Springer, Heidelberg, April 2009.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[IW14] Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, Heidelberg, July 2014.

[JLS21a] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. Cryptology ePrint Archive, Report 2021/1334, 2021. https://eprint.iacr.org/2021/1334.

[JLS21b] Aayush Jain, Huijia Lin, and Amit Sahai. *Indistinguishability Obfuscation from Well-Founded Assumptions*, pages 60–73. Association for Computing Machinery, New York, NY, USA, 2021.

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Heidelberg, August 2019.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013.

[KY01] Aggelos Kiayias and Moti Yung. On crafty pirates and foxy tracers. In *ACM Workshop on Security and Privacy in Digital Rights Management*, DRM '01, pages 22–39, Berlin, Heidelberg, 2001. Springer-Verlag.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 630–660. Springer, Heidelberg, August 2017.

[LZ17] Qipeng Liu and Mark Zhandry. Decomposable obfuscation: A framework for building applications of obfuscation from polynomial hardness. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 138–169. Springer, Heidelberg, November 2017.

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62. Springer, Heidelberg, August 2001.

[NP98] Moni Naor and Benny Pinkas. Threshold traitor tracing. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 502–517. Springer, Heidelberg, August 1998.

[NP01] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *FC 2000*, volume 1962 of *LNCS*, pages 1–20. Springer, Heidelberg, February 2001.

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 388–419. Springer, Heidelberg, May 2016.

[PPS12] Duong Hieu Phan, David Pointcheval, and Mario Strefler. Decentralized dynamic broadcast encryption. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 166–183. Springer, Heidelberg, September 2012.

[sil21] sillydaddy. gpg 加密文件：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). https://v2ex.com/t/759538, March 2021. Retrieved on 20 May 2022, archived at https://web.archive.org/web/20220520040245/https://v2ex.com/t/759538.

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 205–223. Springer, Heidelberg, August 2007.

[WQZD10] Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer. Ad hoc broadcast encryption (poster presentation). In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 741–743. ACM Press, October 2010.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[Zha20a] Mark Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 652–682. Springer, Heidelberg, August 2020.

[Zha20b] Mark Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. Cryptology ePrint Archive, Report 2020/954, 2020. https://eprint.iacr.org/2020/954.

[Zha20c] Mark Zhandry. Schrödinger's pirate: How to trace a quantum decoder. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 61–91. Springer, Heidelberg, November 2020.

[Zha21] Mark Zhandry. White box traitor tracing. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 303–333, Virtual Event, August 2021. Springer, Heidelberg.

A Table of Non-Descriptive Symbols

Table 1 lists non-descriptive symbols used in this work.

Table 1. Non-descriptive symbols.

Symbol	**Meaning**
λ	security parameter
b, β, α	bit for distinguishing, for guessing, for sign correction
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$	adversary, adversary, adversary, distinguisher
μ	message
N, i, j	number of recipients, index, dummy index
$i_\perp, i^*$	cut-off index, traitor index
Q, t	number of key pairs, traitor index
S, T	set of non-traitors, set of traitors
$\varepsilon, \widehat{\varepsilon}$	guessing advantage, empirical
$\mathcal{E}, E$	distinguisher experiment, event of correct guessing
η, ξ, δ	number of trials, absolute frequency of events, estimation error
$C, \widehat{C}, \widetilde{C}$	circuit, garbled, obfuscated
M_0, m_0	length of non-hardwired input (pk), index
L	label of garbled circuit
$D, \widehat{D}$	database (pk's), processed
M, m	length of database (NM_0), index
$k, \mathring{k}_x$	PPRF key, punctured at x
R, A	random oracle (bitset of recipients), oracle algorithm
Y, K	oracle placeholder, oracle implemented during reduction
F, G, H	advice function, presampling function, presampled oracle