Risky Traitor Tracing and New Differential Privacy Negative Results

Rishab Goyal[1], Venkata Koppula[1], Andrew Russell[1], and Brent Waters[1]

University of Texas at Austin, Austin, USA
{rgoyal,kvenkata,ahr,bwaters}@cs.utexas.edu

Abstract. In this work we seek to construct collusion-resistant traitor tracing systems with small ciphertexts from standard assumptions that also move toward practical efficiency. In our approach we will hold steadfast to the principle of collusion resistance, but relax the requirement on catching a traitor from a successful decoding algorithm. We define a f-risky traitor tracing system as one where the probability of identifying a traitor is $f(\lambda, n)$ times the probability a successful box is produced. We then go on to show how to build such systems from prime order bilinear groups with assumptions close to those used in prior works. Our core system achieves, for any $k > 0$, $f(\lambda, n) \approx \frac{k}{n+k-1}$ where ciphertexts consists of $(k+4)$ group elements and decryption requires $(k+3)$ pairing operations.

At first glance the utility of such a system might seem questionable since the f we achieve for short ciphertexts is relatively small. Indeed an attacker in such a system can more likely than not get away with producing a decoding box. However, we believe this approach to be viable for four reasons:

1. A risky traitor tracing system will provide deterrence against risk averse attackers. In some settings the consequences of being caught might bear a high cost and an attacker will have to weigh his utility of producing a decryption D box against the expected cost of being caught.
2. Consider a broadcast system where we want to support low overhead broadcast encrypted communications, but will periodically allow for a more expensive key refresh operation. We refer to an adversary produced algorithm that maintains the ability to decrypt across key refreshes as a persistent decoder. We show how if we employ a risky traitor tracing systems in this setting, even for a small f, we can amplify the chances of catching such a "persistent decoder" to be negligibly close to 1.
3. In certain resource constrained settings risky traitor tracing provides a best tracing effort where there are no other collusion-resistant alternatives. For instance, suppose we had to support 100K users over a radio link that had just 10KB of additional resources for extra ciphertext overhead. None of the existing $\sqrt{N}$ bilinear map systems can fit in these constraints. On the other hand a risky traitor tracing system provides a spectrum of tracing probability versus overhead tradeoffs and can be configured to at least give some deterrence in this setting.

4. Finally, we can capture impossibility results for differential privacy from $\frac{1}{n}$-risky traitor tracing. Since our ciphertexts are short ($O(\lambda)$), we get the negative result which matches what one would get plugging in the obfuscation based tracing system Boneh-Zhandry [9] solution into the prior impossibility result of Dwork et al. [14].

1 Introduction

A traitor tracing [11] system is an encryption system in which a setup algorithm produces a public key pk, master secret key msk and n private keys $\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n$ that are distributed to n user devices. One can encrypt a message m using the public key to produce a ciphertext ct which can be decrypted using any of the private keys; however, is inaccessible by an attacker that is bereft of any keys. The tracing aspect comes into play if we consider an attacker that corrupts some subset $S \subseteq \{1, \ldots, n\}$ of the devices and produces a decryption algorithm D that decrypts ciphertext with some non-negligible probability $\epsilon(\lambda)$ where λ is the security parameter. An additional Trace algorithm will take as input the master secret key msk and with just oracle access to D will identify at least one user from the corrupted set S (and no one outside it). Importantly, any secure system must be able to handle attackers that will construct D in an arbitrary manner including using techniques such as obfuscation.

While the concept of traitor tracing was originally motivated by the example of catching users that created pirate decoder boxes in broadcast TV systems, there are several applications that go beyond that setting. For example ciphertexts could be encryptions of files stored on cloud storage. Or one might use a broadcast to transmit sensitive information to first responders on an ad-hoc deployed wireless network. In addition, the concepts and techniques of traitor tracing have had broader impacts in cryptography and privacy. Most notably Dwork et al. [14] showed that the existence of traitor tracing schemes leads to certain impossibility results in the area of differential privacy [13]. Briefly, they consider the problem of constructing a "sanitizer" $\mathcal{A}$ that takes in a database $x_1, \ldots, x_n$ of entries and wishes to efficiently produce a sanitized summary of database that can evaluate a set of predicate queries on the database. The sanitized database should both support giving an average of answers without too much error and the database should be differentially private in that no one entry should greatly impact the output of the sanitization process. The authors show that an efficient solution to such a problem is impossible to achieve (for certain parameters) assuming the existence of a (collusion resistant) traitor tracing system. The strength of their negative results is directly correlated with the size of ciphertexts in the traitor tracing system.

A primary obstacle in building traitor tracing systems is achieving (full) collusion resistance. There have been several proposals [5, 25, 20, 29, 4, 6, 19] for building systems that are k-collusion resistant where the size of the ciphertexts grows as some polynomial function of k. These systems are secure as long as the number of corrupted keys $|S| \leq k$; however, if the size of the corrupted set exceeds

k the attacker will be able to produce a decryption box that is untraceable. Moreover, the collusion bound of k is fixed at system setup so an attacker will know how many keys he needs to exceed to beat the system. In addition, the impossibility results of Dwork et al. [14] only apply for fully collusion resistant encryption systems. For these reasons we will focus on collusion resistant systems in the rest of the paper.

The existing approaches for achieving collusion resistant broadcast encryption can be fit in the framework of Private Linear Broadcast Encryption (PLBE) introduced by Boneh, Sahai and Waters [7]. In a PLBE system the setup algorithm takes as input a security parameter λ and the number of users n. Like a traitor tracing system it output a public key pk, master secret key msk and n private keys $\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n$ where a user with index j is given key sk_j. Any of the private keys is capable of decrypting a ciphertext ct created using pk. However, there is an additional $\mathsf{TrEncrypt}$ algorithm that takes in the master secret key, a message and an index i. This produces a ciphertext that only users with index $j \geq i$ can decrypt. Moreover, any adversary produced decryption box D that was created with a set of S where $i \notin S$ would not be able to distinguish between encryption to index i or $i+1$. These properties lead to a tracing system where the tracer measures for each index the probability that D decrypts a ciphertext encrypted (using $\mathsf{TrEncrypt}$) for that index and reports all indices i where there is a significant discrepancy between i and $i+1$. These properties imply that such a PLBE based traitor tracing system will catch at least one user in S with all but negligible probability and not falsely accuse anyone in S.

The primary difficulty in achieving collusion resistant traitor tracing is to do so with short ciphertext size. There are relatively few approaches for achieving this goal. First, one can achieve PLBE in a very simple way from public key encryption. Simply create n independent public and private key pairs from the PKE system and lump all the individual public keys together as the PLBE public key. To encrypt one just encrypts to each sub public key in turn. The downside of this method is that the ciphertext size grows as $O(n \cdot \lambda)$ as each of the n users need their own slot in the PLBE ciphertext. If one plugs this into the Dwork et al. [14] impossibility result it rules out systems with a query set $\mathcal{Q}$ of size $2^{O(n \cdot \lambda)}$ or larger. Boneh, Sahai and Waters [7] showed how ciphertexts in a PLBE system can be compressed to $O(\sqrt{n} \cdot \lambda)$ using bilinear maps of composite order. Future variants [17, 15] moved this to the decision linear assumption in prime order groups. While this was an improvement and worked under standard assumptions, there was still a large gap between this and the ideal case where ciphertext size has only polylogarithmic dependence on n.

To achieve really short ciphertexts one needs to leverage heavier tools such as collusion resistant functional encryption or indistingishability obfuscation [2, 16]. For instance, a simple observation shows that one can make a PLBE scheme directly from a collusion resistant FE scheme such as the [16]. Boneh and Zhandry [9] gave a construction of PLBE from indistinguishability obfuscation. These two approaches get ciphertexts that grow proportionally to $\log n$ and thus leading to differential privacy impossibility results with smaller query sets of size $n \cdot 2^{O(\lambda)}$.

However, general functional encryption and indistinguishability obfuscation candidates currently rely on multilinear map candidates, many of which have been broken and the security of which is not yet well understood. In addition, the actual decryption time resulting from using obfuscation is highly impractical.

Our Results. In this work we seek to construct collusion resistant traitor tracing systems with small ciphertexts from standard assumptions geared towards practical efficiency. In our approach we will hold steadfast to the principle of collusion resistance, but relax the requirement on catching a traitor from a successful decoding algorithm. We define a f-risky traitor tracing system as one where the probability of identifying a traitor is $f(\lambda, n)$ times the probability a successful box is produced. We then go on to show how to build such systems from prime order bilinear groups. Our core system achieves $f(\lambda, n) \approx \frac{k}{n+k-1}$ where ciphertexts consist of $(k+4)$ group elements and decryption requires $(k+3)$ pairing operations, where $k > 0$ is a system parameter fixed at setup time. For the basic setting, i.e. $k = 1$, this gives us a success probability of $\frac{1}{n}$, ciphertext consisting of 5 group elements, and decryption requiring just 4 pairing operations in primer order groups.[1] In addition, we show a generic way to increase f by approximately a factor of c at the cost of increasing the size of the ciphertext and decryption time also by a factor of c.

Finally, we show that the argument of Dwork et al. applies to $\frac{1}{n}$-risky traitor tracing. Interestingly, when we structure our argument carefully we achieve the same negative results as when it is applied to a standard traitor tracing system. Since our ciphertexts are short ($O(\lambda)$), we get the negative result which matches what one would get plugging in the obfuscation based tracing system Boneh-Zhandry [9] solution into the prior impossibility result of Dwork et al. [14].

1.1 Technical Overview

In this section, we give a brief overview of our technical approach. We start by discussing the definitional work. That is, we discuss existing traitor tracing definitions, mention their limitations and propose a stronger (and possibly more useful) definition, and finally introduce a weaker notion of traitor tracing which we call *risky* traitor tracing. Next, we describe our construction for risky traitor tracing from bilinear maps. Lastly, we discuss the differential privacy negative results implied by existence of risky traitor tracing schemes.

Definitional Work. A traitor tracing system consists of four poly-time algorithms — Setup, Enc, Dec, and Trace. The setup algorithm takes as input security parameter λ, and number of users n and generates a public key pk, a master secret key msk, and n private keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$. The encrypt algorithm encrypts messages using pk and the decrypt algorithm decrypts a ciphertext using any

[1] In addition to our construction from prime-order bilinear groups, we also provide a construction from composite order bilinear groups where ciphertexts consist of three group elements and decryption requires two pairing operations only.

one of the private keys sk_i. The tracing algorithm takes msk as input and is given a black-box oracle access to a pirate decoder D. It either outputs a special failure symbol $\perp$, or an index $i \in \{1, \ldots, n\}$ signalling that the key sk_i was used to create the pirate decoder.

Traditionally, a traitor tracing scheme is required to satisfy two security properties. First, it must be IND-CPA secure, i.e. any PPT adversary, when given no private keys, should not be able to distinguish between encryptions of two different messages. Second, it is required that if an adversary, given private keys $\{\mathsf{sk}_i\}_{i \in S}$ for any set S of its choice, builds a good pirate decoding box D (that is, a decoding box that can can decrypt encryptions of random messages with non-negligible probability), then the trace algorithm should be able to catch one of the private keys used to build the pirate decoding box. Additionally, the trace algorithm should not falsely accuse any user with non-negligible probability. This property is referred to as secure traitor tracing.

Now a limitation of the traitor tracing property as traditionally defined is that a pirate box is labeled as a *good decoder* only if it extracts the entire message from a non-negligible fraction of ciphertexts.[2] In numerous practical scenarios such a definition could be useless and problematic. For instance, consider a pirate box that can always decrypt encryptions of messages which lie in a certain smaller set but does not work on others. If the size of this special set is negligible, then it won't be a good decoder as per existing definitions, but might still be adversarially useful in practice. There are also other reasons why the previous definitions of traitor tracing are problematic (see Section 3.2 for more details). To this end, we use an indistinguishability-based secure-tracing definition, similar to that used in [26], in which a pirate decoder is labeled to a good decoder if it can distinguish between encryptions of messages chosen by the adversary itself. We discuss this in more detail in Section 3.2.

In this work, we introduce a weaker notion of traitor tracing called *f-risky* traitor tracing, where f is a function that takes the security parameter λ and number of users n as inputs. The syntax as well as IND-CPA security requirement is identical to that of standard traitor tracing schemes. The difference is in the way security of tracing traitors is defined. In an f-risky system, we only require that the trace algorithm must catch a traitor with probability at least $f(\lambda, n)$ whenever the adversary outputs a good decoder. This property is referred to as f-risky secure traitor tracing. Note that a 1-risky traitor tracing scheme is simply a standard traitor tracing scheme, and as f decreases, this progressively becomes weaker.

Constructing Risky Traitor Tracing from Bilinear Maps. As mentioned before, our main construction is based on prime order bilinear groups, and leads to a $\frac{k}{n+k-1}$-risky traitor tracing where k is chosen at setup time. However, for ease of technical exposition we start with a simpler construction that uses composite order bilinear groups and leads to $\frac{1}{n}$-risky traitor tracing scheme. This scheme conveys the basic idea and will serve as a basis for our prime order construction.

[2] The tracing algorithm only needs to work when the pirate box is a good decoder.

Let $\mathbb{G}, \mathbb{G}_T$ be groups of order $N = p_1p_2p_3p_4$ such that there exists a bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ (that is, a mapping which maps (g^a, g^b) to $e(g, g)^{a \cdot b}$ for all $a, b \in \mathbb{Z}_N$). Since these groups are of composite order, $\mathbb{G}$ has subgroups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \mathbb{G}_4$ of prime order p_1, p_2, p_3 and p_4 respectively. Moreover, pairing any element in $\mathbb{G}_i$ with an element in $\mathbb{G}_j$ (for $i \neq j$) results in the identity element (we will say that elements in $\mathbb{G}_i$ and $\mathbb{G}_j$ are orthogonal to each other).

At a high level, our construction works as follows. There are three key-generation algorithms: 'less-than' key-generation, 'equal' key-generation and 'greater-than' key-generation. Similarly, we have three encryption algorithms : 'standard' encryption, 'less-than' encryption and 'less-than-equal' encryption. Out of these encryption algorithms, the 'less-than' and 'less-than-equal' encryptions require the master secret key, and are only used for tracing traitors. The decryption functionality can be summarized by Table 1.

	'less-than' keygen	'equal' keygen	'greater-than' keygen
standard enc	✓	✓	✓
'less-than' enc	✗	✓	✓
'less-than-equal' enc	✗	✗	✓

Table 1: Decryption functionality for different encryption/key-generation algorithms. The symbol ✓ denotes that decryption works correctly, while ✗ denotes that decryption fails.

The master secret key consists of a 'cutoff' index i chosen uniformly at random from $\{1, \ldots, n\}$. For any index $j < i$, it uses the 'less-than' key-generation algorithm to generate keys. For $j > i$, it uses the 'greater-than' key-generation algorithm, and for $j = i$, it uses the 'equal' key-generation algorithm. The ciphertext for a message m is a 'standard' encryption of m. From Table 1, it is clear that decryption works. The trace algorithm tries to identify if the cutoff index i is used by the pirate box D. It first checks if D can decrypt 'less-than' encryptions. If so, then it checks if D can decrypt 'less-than-equal' encryptions. If D works in the 'less-than' case, but not in the 'less-than-equal' case, then the trace algorithm identifies index i as one of the traitors.

Let us now look at how the encryption/key generation algorithms work at a high level. The public key in our scheme consists of $g_1 \in \mathbb{G}_1$ and $e(g_1, g_1)^\alpha$, while the master secret key has the cut-off index i, element α, as well as generators for all subgroups of $\mathbb{G}$. The 'less-than' keys are set to be $g_1^\alpha \cdot w_3 \cdot w_4$, where w_3, w_4 are random group elements from $\mathbb{G}_3, \mathbb{G}_4$ respectively. The 'equal' key is $g_1^\alpha \cdot w_2 \cdot w_4$, where $w_2 \leftarrow \mathbb{G}_2, w_4 \leftarrow \mathbb{G}_4$. Finally, the 'greater-than' key has no $\mathbb{G}_2$ or $\mathbb{G}_3$ terms, and is set to be $g_1^\alpha \cdot w_4$.

The 'standard' encryption of message m is simply $(m \cdot e(g_1, g_1)^{\alpha \cdot s}, g_1^s)$. In the 'less-than' and 'less-than-equal' ciphertexts, the first component is computed similarly but the second component is modified. For 'less-than' encryptions, the ciphertext is $(m \cdot e(g_1, g_1)^{\alpha \cdot s}, g_1^s \cdot h_3)$, where h_3 is a uniformly random group element in $\mathbb{G}_3$. For 'less-than-equal' encryptions the ciphertext is

$(m \cdot e(g_1, g_1)^{\alpha \cdot s}, g_1^s \cdot h_2 \cdot h_3)$, where h_2 and h_3 are uniformly random group elements in $\mathbb{G}_2$ and $\mathbb{G}_3$ respectively.

To decrypt a ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ using a key K, one must compute $\mathsf{ct}_1 / e(\mathsf{ct}_2, K)$. It is easy to verify that the keys and encryptions follow the decryption behavior described in Table 1. For instance, an 'equal' key $K = g_1^{\alpha} \cdot w_2 \cdot w_4$ can decrypt a 'less-than' encryption $(m \cdot e(g_1, g_1)^{\alpha \cdot s}, g_1^s \cdot h_3)$ because $e(\mathsf{ct}_2, K) = e(g_1, g_1)^{\alpha \cdot s}$. However, an 'equal' key cannot decrypt a 'less-than-equal' ciphertext $\mathsf{ct} = (m \cdot e(g_1, g_1)^{\alpha \cdot s}, g_1^s \cdot h_2)$ because $e(\mathsf{ct}_2, K) = e(g_1, g_1)^{\alpha \cdot s} \cdot e(h_2, w_2)$.

Given this construction, we need to prove two claims. First, we need to show that no honest party is implicated by our trace algorithm; that is, if an adversary does not receive key for index i, then the trace algorithm must not output index i. We show that if an adversary does not have key for index i, then the pirate decoding box must not be able to distinguish between 'less-than' and 'less-than-equal' encryptions (otherwise we can break the subgroup-decision assumption on composite order bilinear groups). Next, we show that if an adversary outputs a pirate decoding box that works with probability ρ, then we can identify a traitor with probability ρ/n. To prove this, we show that if ρ_i denotes the probability that the adversary outputs a ρ-functional box and i is the cutoff-index, then the sum of all these ρ_i quantities is close to ρ. The above scheme is formally described in the full version along with a detailed security proof. Next we move on to our risky traitor tracing construction from prime order bilinear groups.

Moving to Prime Order Bilinear Maps and $\frac{k}{n+k-1}$-Risky. The starting point for building $\frac{k}{n+k-1}$-risky traitor tracing scheme from prime order bilinear groups is the aforementioned scheme. Now to increase the success probability of the tracing algorithm by a factor k, we increase the types of secret keys and ciphertexts from 3 to $k+2$ such that the decryptability of ciphertexts w.r.t. secret keys can again be described as an upper-triangular matrix of dimension $k+2$ as follows.

	'$< w$' keygen	'$= w$' keygen	'$= w+1$' keygen	$\cdots$	'$= w+k-1$' keygen	'$\geq w+k$' keygen
standard enc	✓	✓	✓	$\cdots$	✓	✓
'$< w$' enc	✗	✓	✓	$\cdots$	✓	✓
'$< w+1$' enc	✗	✗	✓	$\cdots$	✓	✓
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$	$\vdots$	$\vdots$
'$< w+k-1$' enc	✗	✗	✗	$\cdots$	✓	✓
'$< w+k$' enc	✗	✗	✗	$\cdots$	✗	✓

Table 2: New Decryption Functionality.

The basic idea will similar to the one used previously, except now we choose a cutoff window $W = \{w, w+1, \ldots, w+k-1\}$ of size k uniformly at random. (Earlier the window had size 1, that is we choose a single index.) The first $w-1$ users are given '$< w$' keys. For $w \leq j < w+k$, the j^{th} user gets '$= j$' key, and

rest of the users get the '$\geq w + k$' keys. The remaining idea is similar to what we used which is that the tracer estimates the successful decryption probability for a decoder D on all the special index encryptions (i.e., '$< j$' encryptions), and outputs the indices of all those users where there is a gap in decoding probability while moving from type '$< j$' to '$< j + 1$'.

Now instead of directly building a scheme that induces such a decryption functionality, we provide a general framework for building risky traitor tracing schemes. In this work, we introduce a new intermediate primitive called Mixed Bit Matching Encryption (mBME) and show that it is sufficient to build risky traitor tracing schemes. In a mBME system, the secret keys and ciphertexts are associated with bit vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^\ell$ (respectively) for some ℓ. And decryption works whenever $f(\mathbf{x}, \mathbf{y}) = 1$ where f computes an 'AND-of-ORs' over vectors $\mathbf{x}, \mathbf{y}$ (i.e., for every $i \leq \ell$, either $\mathbf{x}_i = 1$ or $\mathbf{y}_i = 1$). Using the public parameters, one could encrypt to the 'all-ones' vector, and using the master secret key one could sample a ciphertext (or secret key) for any vector. For security, we require that the ciphertexts and the secret keys should not reveal non-trivial information about their associated vectors. In other words, the only information an adversary learns about these vectors is by running the decryption algorithm. In the sequel, we provide a generic construction of risky traitor tracing from a mBME scheme, and also give a construction of mBME scheme using prime order bilinear groups. They are described in detail later in Sections 5 and 6.

Finally, we also provide a performance evaluation of our risky traitor tracing scheme in Section 7.

Relation to BSW traitor tracing scheme. Boneh, Sahai and Waters [7] constructed a (fully) collusion-resistant traitor tracing scheme with $O(\sqrt{n} \cdot \lambda)$ size ciphertexts. The BSW construction introduced the *private linear broadcast encryption* (PLBE) abstraction, showed how to build traitor tracing using PLBE, and finally gave a PLBE construction using composite-order bilinear groups.

Our framework deviates from the PLBE abstraction in that we support encryptions to only $k+1$ adjacent indices (that is, if w is starting index of the cutoff window, then we support encryptions to either $w, \ldots, w + k)$ and index 0. As a result, the trace algorithm can only trace an index in the window $w, \ldots, w + k$. The main difficulty in our proof argument is that encrypting to index j is not defined for indices oustide the cutoff window, i.e. $j \notin \{0, w, w+1, \ldots, w+k\}$. As a result, we need to come up with a new way to link success probabilities across different setups and weave these into an argument.

Negative Results for Differential Privacy. Given a database $D = (x_1, x_2, \ldots, x_n) \in \mathcal{X}^n$, in which each row represents a single record of some sensitive information contributed by an individual and each record is an element in the data universe $\mathcal{X}$, the problem of privacy-preserving data analysis is to allow statistical analyses of D while protecting the privacy of individual contributors. The problem is formally defined in the literature by representing the database with a *sanitized* data structure s that can be used to answer all queries q in some query

class $\mathcal{Q}$ with reasonable accuracy, with the restriction that the sanitization of any two databases D, D' which differ at only a single position are indistinguishable. In this work, we will focus on counting (or statistical) queries. Informally, a counting query q on a database D tells what fraction of records in D satisfy the property associated with q.

Dwork et al. [14] first showed that secure traitor tracing schemes can be used to show hardness results for efficient differentially private sanitization. In their hardness result, the data universe is the private key space of traitor tracing scheme and the query space is the ciphertext space. A database consists of n private keys and each query is associated with either an encryption of 0 or 1. Formally, for a ciphertext ct, the corresponding query q_{ct} on input a private key sk outputs the decryption of ct using sk. They show that if the underlying traitor tracing scheme is secure, then there can not exist sanitizers that are simultaneously accurate, differentially private, and efficient. At a very high level, the idea is as follows. Suppose there exists an efficient sanitizer A that, on input $D = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ outputs a sanitization s. The main idea is to use sanitizer A to build a pirate decoding box such that the tracing algorithm falsely accuses a user with non-negligible probability, thereby breaking secure traitor traitor property. Concretely, let $\mathcal{B}$ be an attacker on the secure tracing property that works as follows — $\mathcal{B}$ queries for private keys of all but i^{th} party, and then uses sanitizer A to generate sanitization s of the database containing all the queried private keys, and finally it outputs the pirate decoding box as the sanitization evaluation algorithm which has s hardwired inside and on input a ciphertext ouputs its evaluation given sanitization s.[3]

To prove that the tracing algorithm outputs i (with non-negligible probability) given such a decoding box, Dwork et al. crucially rely on the fact that A is differentially private. First, they show that if an adversary uses all private keys to construct the decoding box, then the tracing algorithm always outputs an index and never aborts.[4] Then, they argue that there must exist an index i such that tracing algorithm outputs i with probability $p \geq 1/n$. Finally, to complete the claim they show that even if i^{th} key is removed from the database, the tracing algorithm will output i with non-negligible probability since the sanitizer is differentially private with parameters $\epsilon = O(1)$ and $\delta = o(1/n)$.

In this work, we show that their analysis can be adapted to risky traitor tracing as well. Concretely, we show that f-risky secure traitor tracing schemes can be used to show hardness results for efficient differentially private sanitization, where f directly relates to the differential privacy parameters. At a high level, the proof strategy is similar, i.e. we also show that an efficient sanitizer could be used to build a good pirate decoding box. The main difference is that now we can only claim that if an adversary uses all private keys to construct the decoding box, then (given oracle access to the box) the tracing algorithm outputs an index

[3] Technically, the decoding box must round the output of evaluation algorithm in order to remove evaluation error.

[4] In the full proof, one could only argue that tracing algorithm outputs an index with probability at least $1 - \beta$ where β is the accuracy parameter of sanitizer A.

with probability at least f, i.e. the trace algorithm could potentially abort with non-negligible probability. Next, we can argue that there must exist an index i such that tracing algorithm outputs i with probability $p \geq f/n$. Finally, using differential privacy of A we can complete the argument. An important caveat in the proof is that since the lower bounds in the probability terms have an additional multiplicative factor of f, thus f-risky traitor tracing could only be used to argue hardness of differential privacy with slightly lower values of parameter δ, i.e. $\delta = o(f/n)$.

However, we observe that if the risky traitor tracing scheme additionally satisfies what we call "singular trace" property, then we could avoid the $1/n$ loss. Informally, a risky scheme is said to satisfy the singular trace property if the trace algorithm always outputs either a fixed index or the empty set. One could visualize the fixed index to be tied to the master secret and public keys. Concretely, we show that f-risky traitor tracing with singular trace property implies hardness of differential privacy for $\delta = o(f)$, thereby matching that achieved by previous obfuscation based result of [9]. We describe our hardness result in detail in Section 8.2.

Amplifying the Probability of Tracing — Catching Persistent Decoders. While an f-risky traitor tracing system by itself gives a small probability of catching a traitor, there can be ways to deploy it that increase this dramatically. We discuss one such way informally here.

Consider a broadcast system where we want to support low overhead broadcast encrypted communications, but will periodically allow for a more expensive key refresh operation. Suppose that we generate the secret keys $\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n$ for a risky traitor tracing system and in addition generate standard secret keys $\mathrm{SK}_1, \ldots, \mathrm{SK}_n$. In this system an encryptor can use the traitor tracing public key pk to compute a ciphertext. A user i will use secret key sk_i to decrypt. The system will allow this to continue for a certain window of time. (Note during the window different ciphertexts may be created by different users.) Then at some point in time the window will close and a new risky tracing key pk' and secret keys $\mathsf{sk}'_1, \mathsf{sk}'_2, \ldots, \mathsf{sk}'_n$ will be generated. The tracing secret keys will be distributed by encrypting each sk'_i under the respective permanent secret key SK_i. And the encryptors will be instructed to only encrypt using the new public key pk'. This can continue for an arbitrary number of windows followed by key refreshes. Note that each key refresh requires $O(n\lambda)$ size communication.

Consider an attacker that wishes to disperse a stateless decoder D that is capable of continuing to work through multiple refresh cycles. Such a "persistent decoder" can be traced with very high probability negligibly close to 1. The tracing algorithm must simply give it multiple key refreshes followed by calls to the Trace algorithm and by the risky property it will eventually pick one that can trace one of the contributors.

We emphasize that care must be taken when choosing the refresh size window. If the window is too small the cost of key refreshes will dominate communication — in one extreme if a refresh happens at the frequency that ciphertexts are created then the communication is as bad as the trivial PLBE system. In addition,

dispersing new public keys very frequently can be an issue. On the other hand if a refresh window is very long, then an attacker might decide there is value in producing a decoding box that works only for the given window and we are back to having only an $f(\lambda, n)$ chance of catching him.

1.2 Additional Related Work

Our traitor tracing system allows for public key encryption, but requires a master secret key to trace users as do most works. However, there exists exceptions [27, 28, 31, 21, 10, 8, 9] where the tracing can be done using a public key. In a different line of exploration, Kiayias and Yung [20] argue that a traitor tracing system with higher overhead can be made "constant rate" with long enough messages. Another interesting point in the space of collusion resistant systems is that of Boneh and Naor [6]. They show how to achieve short ciphertext size, but require private keys that grow quadratically in the number of users as $O(n^2\lambda)$. In addition, this is only achievable assuming a perfect decoder. If the decoder D works with probability δ then the secret key grows to $O(n^2\lambda/\delta^2)$. Furthermore, the system must be configured a-priori with a specific δ value and once it is set one will not necessarily be able to identify a traitor from a box D that works with smaller probability. Such systems have been called threshold traitor tracing systems [24, 12]. Both [24, 12] provide combinatorial and probabilistic constructions in which the tracing algorithm is guaranteed to work with high probability, and to trace t traitors they get private keys of size $O(t \cdot \log n)$. In contrast we can capture any traitor strategy that produces boxes that work with any non-negligible function $\epsilon(\lambda)$. Chor et al. [12] also considered a setting for traitor tracing in which the tracing algorithm only needs to correctly trace with probability $1 - p$, where p could the scheme parameter. However, this notion has not been formally defined or explored since then.

Dwork et al. [14] first showed that existence of collusion resistant traitor tracing schemes implies hardness results for efficient differentially private sanitization. In their hardness result, the database consists of n secret keys and each query is associated with an encryption of 0/1. Thus, the size of query space depends on the size of ciphertexts. Instantiating the result of Dwork et al. with the traitor tracing scheme of Boneh et al. [7], we get that under assumptions on bilinear groups, there exist a distribution on databases of size n and a query space of size $O(2^{\sqrt{n}\cdot\lambda})$ such that it is not possible to efficiently sanitize the database in a differentially private manner.

Now the result of Dwork et al. gives hardness of *one-shot* sanitization. A one-shot sanitizer is supposed to produce a summary of an entire database from which approximate answers to any query in the query set could be computed. A weaker setting could be where we consider interactive sanitization, in which the queries are fixed and given to the sanitizer as an additional input and the sanitizer only needs to output approximate answers to all those queries instead of a complete summary. Ullman [30] showed that, under the assumption that one-way functions exist, there is no algorithm that takes as input a database of n records along with an arbitrary set of about $O(n^2)$ queries, and approximately answers each query

in polynomial time while preserving differential privacy. Ullman's result differs from the result of Dwork et al. in that it applies to algorithms answering any arbitrary set of $O(n^2)$ queries, whereas Dwork et al. show that it is impossible to sanitize a database with respect to a fixed set of $O(2^{\sqrt{n}\cdot\lambda})$ queries.

Recently a few works [9, 23] have improved the size of query space for which (one-shot) sanitization is impossible from $O(2^{\sqrt{n}\cdot\lambda})$ to $n \cdot O(2^{\lambda})$ to $\mathsf{poly}(n)$.[5] [9] showed the impossibility by first constructing a fully collusion resistant scheme with short ciphertexts, and later simply applying the Dwork et al. result. On the other hand, [23] first construct a weakly secure traitor tracing scheme by building on top of PLBE abstraction, and later adapt the Dwork et al. impossibility result for this weaker variant. These works however assume existence of a stronger cryptographic primitive called *indistinguishability-obfuscator* ($i\mathcal{O}$) [2, 16]. Currently we do not know of any construction of $i\mathcal{O}$ from a standard cryptographic assumption. In this work, we are interested in improving the state-of-the-art hardness results in differential privacy based on standard assumptions.

More recent related work. In an independent and concurrent work, Kowalczyk, Malkin, Ullman and Wichs [22] gave similar differential privacy negative results from one way functions. The negative results they achieve are similar to ours, but also apply for slightly smaller database sizes. However, the paths taken in our and their work diverge significantly. Our approach has been to focus on weaker notion of traitor tracing that suffices for DP impossibility while still being useful as a standalone primitive. On the other hand, KMUW instead closely follow the approach taken in [23], and they build special purpose functional encryption scheme for comparisons that supports 2 ciphertexts and bounded number of secret keys (succinctly) and achieves a very weak notion of IND-based security. Thus, the focus of their work is on negative results for differential privacy, whereas our risky tracing framework has both positive applications as well as lead to differential privacy impossibility results.

Subsequent to our work, Goyal, Koppula and Waters [18] gave a collusion resistant tracing system from the Learning with Errors assumption where the ciphertext size grows polynomially in $\lambda, \lg(N)$. Their result could be directly plugged into the Dwork et al. [14] differential privacy result as is, however they do not develop paths for weakening TT.

2 Preliminaries

Notations. For any set $\mathcal{X}$, let $x \leftarrow \mathcal{X}$ denote a uniformly random element drawn from the set $\mathcal{X}$. Given a PPT algorithm D, let A^D denote an algorithm $\mathcal{A}$ that uses D as an oracle (that is, A sends queries to D, and for each query x, it receives $D(x)$). Throughout this paper, we use PPT to denote probabilistic polynomial-time. We will use lowercase bold letters for vectors (e.g. $\mathbf{v}$), and we will sometimes represent bit vectors $\mathbf{v} \in \{0,1\}^{\ell}$ as bit-strings of appropriate length.

[5] In this work, we only focus on the size of query space.

2.1 Assumptions

In this work, we will be using bilinear groups. Let $\mathsf{Grp\text{-}Gen}$ be a PPT algorithm that takes as input security parameter λ (in unary), and outputs a λ-bit prime p, an efficient description of groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order p, generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and an efficient non-degenerate bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ (that is, $e(g_1, g_2) \neq 1_{G_T}$, and for all $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{a \cdot b}$).

We will be using the following assumptions in this work.

Assumption 1 *For every PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ s.t. for all $\lambda \in \mathbb{N}$,*

$$\Pr\left[b \leftarrow \mathcal{A}\begin{pmatrix}\mathsf{params},\\ g_1^x, g_1^y, g_1^{y \cdot z}, g_2^y, g_2^z, T_b\end{pmatrix} : \begin{matrix}\mathsf{params} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e(\cdot,\cdot)) \leftarrow \mathsf{Grp\text{-}Gen}(1^\lambda);\\ x, y, z, r \leftarrow \mathbb{Z}_p, T_0 = g_1^{x \cdot y \cdot z}, T_1 = g_1^{x \cdot y \cdot z + r}, b \leftarrow \{0,1\}\end{matrix}\right] \leq 1/2 + negl(\lambda).$$

Assumption 2 *For every PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ s.t. for all $\lambda \in \mathbb{N}$,*

$$\Pr\left[b \leftarrow \mathcal{A}\begin{pmatrix}\mathsf{params},\\ g_1^y, g_1^z, g_2^x, g_2^y, T_b\end{pmatrix} : \begin{matrix}\mathsf{params} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e(\cdot,\cdot)) \leftarrow \mathsf{Grp\text{-}Gen}(1^\lambda);\\ x, y, z, r \leftarrow \mathbb{Z}_p, T_0 = g_2^{x \cdot y \cdot z}, T_1 = g_2^{x \cdot y \cdot z + r}, b \leftarrow \{0,1\}\end{matrix}\right] \leq 1/2 + negl(\lambda).$$

3 Risky Traitor Tracing

In this section, we will first introduce the traditional definition of traitor tracing based on that given by Boneh, Sahai and Waters [7]. We provide a "public key" version of the definition in which the encryption algorithm is public, but the tracing procedure will require a master secret key. Our definition will by default capture full collusion resistance.

A limitation of this definition is that the tracing algorithm is only guaranteed to work on decoders that entirely decrypt encryptions of randomly selected messages with non-negligible probability. We we will discuss why this definition can be problematic and then provide an *indistinguishability* based definition for secure tracing.

Finally, we will present our new notion of *risky* traitor tracing which captures the concept of a trace algorithm that will identify a traitor from a working pirate box with probability close to $f(\lambda, n)$. Our main definition for risky traitor tracing will be a public key one using the indistinguishability; however we will also consider some weaker variants that will be sufficient for obtaining our negative results in differential privacy.

3.1 Public Key Traitor Tracing

A traitor tracing scheme with message space $\mathcal{M}$ consists of four PPT algorithms $\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}$ and Trace with the following syntax:

- $(\mathsf{msk}, \mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$: The setup algorithm takes as input the security parameter λ, number of users n, and outputs a master secret key msk, a public key pk and n secret keys $\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m \in \mathcal{M})$: The encryption algorithm takes as input a public key pk, message $m \in \mathcal{M}$ and outputs a ciphertext ct.
- $y \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key sk, ciphertext ct and outputs $y \in \mathcal{M} \cup \{\perp\}$.
- $S \leftarrow \mathsf{Trace}^D(\mathsf{msk}, 1^y)$: The tracing algorithm takes a parameter $y \in \mathbb{N}$ (in unary) as input, has black box access to an algorithm D, and outputs a set $S \subseteq \{1, 2, \ldots, n\}$.

Correctness For correctness, we require that if ct is an encryption of message m, then decryption of ct using one of the valid secret keys must output m. More formally, we require that for all $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, $(\mathsf{msk}, \mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, $m \in \mathcal{M}$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ and $i \in \{1, 2, \ldots, n\}$, $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}) = m$.

Security A secure traitor tracing scheme must satisfy two security properties. First, the scheme must be IND-CPA secure (that is, any PPT adversary, when given no secret keys, cannot distinguish between encryptions of m_0, m_1). Next, we require that if an adversary, using some secret keys, can build a pirate decoding box, then the trace algorithm should be able to catch at least one of the secret keys used to build the pirate decoding box. In this standard definition, the trace algorithm identifies a traitor if the pirate decoding box works with non-negligible probability in extracting the entire message from an encryption of a random message.

Definition 1 (IND-CPA security). *A traitor tracing scheme* $\mathcal{T} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ *is* IND-CPA *secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, polynomial* $n(\cdot)$*, there exists a negligible function* $negl(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,* $|\Pr[1 \leftarrow \mathsf{Expt\text{-}IND\text{-}CPA}^{\mathcal{T}}_{\mathcal{A}}(1^\lambda, 1^n)] - 1/2| \leq negl(\lambda)$*, where* $\mathsf{Expt\text{-}IND\text{-}CPA}_{\mathcal{T},\mathcal{A}}$ *is defined below.*

- $(\mathsf{msk}, \mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)) \leftarrow \mathsf{Setup}(1^\lambda, 1^{n(\lambda)})$
- $(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(\mathsf{pk})$
- $b \leftarrow \{0, 1\}$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
- $b' \leftarrow \mathcal{A}_2(\sigma, \mathsf{ct})$. *Experiment outputs* 1 *iff* $b = b'$.

Definition 2 (Secure traitor tracing). *Let* $\mathcal{T} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ *a traitor tracing scheme. For any polynomial* $n(\cdot)$*, non-negligible function* $\epsilon(\cdot)$ *and PPT adversary* $\mathcal{A}$*, consider the following experiment* $\mathsf{Expt}^{\mathcal{T}}_{\mathcal{A},n,\epsilon}(\lambda)$*:*

- $\left(\mathsf{msk}, \mathsf{pk}, \left(\mathsf{sk}_1, \ldots, \mathsf{sk}_{n(\lambda)}\right)\right) \leftarrow \mathsf{Setup}(1^\lambda, 1^{n(\lambda)})$.
- $D \leftarrow A^{O(\cdot)}(\mathsf{pk})$

– $S_D \leftarrow \mathsf{Trace}^D(\mathsf{msk}, 1^{1/\epsilon(\lambda)})$.

Here, $O(\cdot)$ is an oracle that has $\{\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_{n(\lambda)}\}$ hardwired, takes as input an index $i \in \{1, 2, \ldots, n(\lambda)\}$ and outputs sk_i. Let S be the set of indices queried by $\mathcal{A}$. Based on this experiment, we will now define the following (probabilistic) events and the corresponding probabilities (which is a function of λ, parameterized by $\mathcal{A}, n, \epsilon$):

– $\mathsf{Good\text{-}Decoder}$ *:* $\Pr[D(\mathsf{ct}) = m \ : \ m \leftarrow \mathcal{M}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)] \geq \epsilon(\lambda)$
$\Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},n,\epsilon}(\lambda) = \Pr[\mathsf{Good\text{-}Decoder}]$.
– $\mathsf{Cor\text{-}Tr}$ *:* $S_D \subseteq S \ \wedge \ S_D \neq \emptyset$
$\Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) = \Pr[\mathsf{Cor\text{-}Tr}]$.
– $\mathsf{Fal\text{-}Tr}$ *:* $S_D \setminus S \neq \emptyset$
$\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) = \Pr[\mathsf{Fal\text{-}Tr}]$.

A traitor tracing scheme $\mathcal{T}$ is said to be secure if for every PPT adversary $\mathcal{A}$, polynomials $n(\cdot)$, $p(\cdot)$ and non-negligible function $\epsilon(\cdot)$, there exists negligible functions $negl_1(\cdot)$, $negl_2(\cdot)$ such that for all $\lambda \in \mathbb{N}$ such that $\epsilon(\lambda) > 1/p(\lambda)$, $\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) \leq negl_1(\lambda)$ *and* $\Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) \geq \Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},n,\epsilon}(\lambda) - negl_2(\lambda)$.

3.2 Indistinguishability Security Definition for Traitor Tracing Schemes

A limitation of the previous definition is that the tracing algorithm is only guaranteed to work on decoders that *entirely* decrypt a *randomly* selected message with non-negligible probability. This definition can be problematic for the following reasons.

– First, there could be pirate boxes which do not extract the entire message from a ciphertext, but can extract some information about the message underlying a ciphertext. For example, a box could paraphrase English sentences or further compress an image. Such boxes could be very useful to own in practice yet the tracing definition would give no guarantees on the ability to trace them.
– Second, a pirate decoder may not be very successful in decrypting random ciphertexts, but can decrypt encryptions of messages from a smaller set. In practice the set of useful or typical messages might indeed fall in a smaller set.
– Finally, if the message space is small (that is, of polynomial size), then one can always construct a pirate decoder which succeeds with non-negligible probability and can not get caught (the pirate decoder box simply outputs a random message for each decryption query. If $\mathcal{M}$ is the message space, then decryption will be successful with probability $1/|\mathcal{M}|$). Since such a strategy does not use any private keys, it cannot be traced. Therefore the above definition is only sensible for superpolynomial sized message spaces.

To address these issues, we provide a stronger definition, similar to that used in [26], in which a pirate decoder is successful if it can distinguish between encryptions of messages chosen by the decoder itself. For this notion, we also need to modify the syntax of the Trace algorithm. Our security notion is similar to the one above except that an attacker will output a box D along with two messages (m_0, m_1). If the box D is able to distinguish between encryptions of these two messages with non-negligible probability then the tracing algorithm can identify a corroborating user.

$\mathsf{Trace}^D(\mathsf{msk}, 1^y, m_0, m_1)$: The trace algorithm has oracle access to a program D, it takes as input a master secret key msk, y (in unary) and two messages m_0, m_1. It outputs a set $S \subseteq \{1, 2, \ldots, n\}$.

Definition 3 (Ind-secure traitor tracing). *Let $\mathcal{T} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ be a traitor tracing scheme. For any polynomial $n(\cdot)$, non-negligible function $\epsilon(\cdot)$ and PPT adversary $\mathcal{A}$, consider the experiment $\mathsf{Expt\text{-}TT}^{\mathcal{T}}_{\mathcal{A},n,\epsilon}(\lambda)$ defined in Figure 1. Based on this experiment, we will now define the following (probabilistic) events and the corresponding probabilities (which is a function of λ, parameterized by $\mathcal{A}, n, \epsilon$):*

- $\mathsf{Good\text{-}Decoder}$ *:* $\Pr[D(\mathsf{ct}) = b \ : \ b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)] \geq 1/2 + \epsilon(\lambda)$
  $\Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},n,\epsilon}(\lambda) = \Pr[\mathsf{Good\text{-}Decoder}]$.
- $\mathsf{Cor\text{-}Tr}$ *:* $S_D \subseteq S \ \wedge \ S_D \neq \emptyset$
  $\Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) = \Pr[\mathsf{Cor\text{-}Tr}]$.
- $\mathsf{Fal\text{-}Tr}$ *:* $S_D \setminus S \neq \emptyset$
  $\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) = \Pr[\mathsf{Fal\text{-}Tr}]$.

A traitor tracing scheme $\mathcal{T}$ is said to be ind-secure if for every PPT adversary $\mathcal{A}$, polynomials $n(\cdot)$, $p(\cdot)$ and non-negligible function $\epsilon(\cdot)$, there exists negligible functions $negl_1(\cdot)$, $negl_2(\cdot)$ such that for all $\lambda \in \mathbb{N}$ satisfying $\epsilon(\lambda) > 1/p(\lambda)$, $\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) \leq negl_1(\lambda)$ *and* $\Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) \geq \Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},n,\epsilon}(\lambda) - negl_2(\lambda)$.

Experiment $\mathsf{Expt\text{-}TT}^{\mathcal{T}}_{\mathcal{A},n,\epsilon}(\lambda)$

- $\left(\mathsf{msk}, \mathsf{pk}, \left(\mathsf{sk}_1, \ldots, \mathsf{sk}_{n(\lambda)}\right)\right) \leftarrow \mathsf{Setup}(1^\lambda, 1^{n(\lambda)})$.
- $(D, m_0, m_1) \leftarrow \mathcal{A}^{O(\cdot)}(\mathsf{pk})$
- $S_D \leftarrow \mathsf{Trace}^D(\mathsf{msk}, 1^{1/\epsilon(\lambda)}, m_0, m_1)$.

Here, $O(\cdot)$ is an oracle that has $\{\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_{n(\lambda)}\}$ hardwired, takes as input an index $i \in \{1, 2, \ldots, n(\lambda)\}$ and outputs sk_i. Let S be the set of indices queried by $\mathcal{A}$.

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

3.3 Risky Traitor Tracing

In this section, we will introduce the notion of risky traitor tracing. The syntax is same as that of ind-secure traitor tracing. However, for security, if the adversary outputs a good decoder, then the trace algorithm will catch a traitor with probability f where f is a function of λ and the number of users.

Definition 4 (f-risky secure traitor tracing). *Let $f : \mathbb{N} \times \mathbb{N} \to [0,1]$ be a function and $\mathcal{T} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ a traitor tracing scheme. For any polynomial $n(\cdot)$, non-negligible function $\epsilon(\cdot)$ and PPT adversary $\mathcal{A}$, consider the experiment $\mathsf{Expt\text{-}TT}^{\mathcal{T}}_{\mathcal{A},n,\epsilon}(\lambda)$ (defined in Figure 1). Based on this experiment, we will now define the following (probabilistic) events and the corresponding probabilities (which are functions of λ, parameterized by $\mathcal{A}, n, \epsilon$):*

- $\mathsf{Good\text{-}Decoder}$ *:* $\Pr[D(\mathsf{ct}) = b \ : \ b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)] \geq 1/2 + \epsilon(\lambda)$
  $\Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},n,\epsilon}(\lambda) = \Pr[\mathsf{Good\text{-}Decoder}]$.
- $\mathsf{Cor\text{-}Tr}$ *:* $S_D \subseteq S \ \wedge \ S_D \neq \emptyset$
  $\Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) = \Pr[\mathsf{Cor\text{-}Tr}]$.
- $\mathsf{Fal\text{-}Tr}$ *:* $S_D \setminus S \neq \emptyset$
  $\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) = \Pr[\mathsf{Fal\text{-}Tr}]$.

A traitor tracing scheme $\mathcal{T}$ is said to be f-risky secure if for every PPT adversary $\mathcal{A}$, polynomials $n(\cdot)$, $p(\cdot)$ and non-negligible function $\epsilon(\cdot)$, there exists negligible functions $negl_1(\cdot)$, $negl_2(\cdot)$ such that for all $\lambda \in \mathbb{N}$ satisfying $\epsilon(\lambda) > 1/p(\lambda)$, $\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) \leq negl_1(\lambda)$ and $\Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},n,\epsilon}(\lambda) \geq \Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},n,\epsilon}(\lambda) \cdot f(\lambda, n(\lambda)) - negl_2(\lambda)$.

We also define another interesting property for traitor tracing schemes which we call "singular" trace. Informally, a scheme satisfies it if the trace algorithm always outputs either a fixed index or the reject symbol. The fixed index could depend on the master secret and public keys. Below we define it formally.

Definition 5 (Singular Trace). *A traitor tracing scheme $\mathcal{T} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ is said to satisfy* singular trace *property if for every polynomial $n(\cdot)$, $\lambda \in \mathbb{N}$, keys $(\mathsf{msk}, \mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, there exists an index $i^* \in \{1, \ldots, n\}$ such that for every poly-time algorithm D, parameter $y \in \mathbb{N}$, any two messages m_0, m_1,*

$$\Pr[\mathsf{Trace}^D(\mathsf{msk}, 1^y, m_0, m_1) \in \{\{i^*\}, \emptyset\}] = 1,$$

where the probability is taken over random coins of Trace.

One can analogously define the notion of private key risky traitor tracing, which suffices for our differential privacy lower bound. We present this notion in the full version.

4 A New Abstraction for Constructing Risky Traitor Tracing

Let $\{\mathcal{M}_\lambda\}_\lambda$ denote the message space. A mixed bit matching encryption scheme for $\mathcal{M}$ consists of five algorithms with the following syntax.

- $\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{pk}, \mathsf{msk})$: The setup algorithm takes as input security parameter λ, a parameter ℓ and outputs a public key pk and master secret key msk.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x} \in \{0,1\}^\ell) \to \mathsf{sk}$: The key generation algorithm takes as input the master secret key msk and a vector $\mathbf{x} \in \{0,1\}^\ell$. It outputs a secret key sk corresponding to $\mathbf{x}$.
- $\mathsf{Enc\text{-}PK}(\mathsf{pk}, m \in \mathcal{M}) \to \mathsf{ct}$: The public-key encryption algorithm takes as input a public key pk and a message m, and outputs a ciphertext ct.
- $\mathsf{Enc\text{-}SK}(\mathsf{msk}, m \in \mathcal{M}, \mathbf{y} \in \{0,1\}^\ell) \to \mathsf{ct}$: The secret-key encryption algorithm takes as input master secret key msk, message m, and an attribute vector $\mathbf{y} \in \{0,1\}^\ell$. It outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to z$: The decryption algorithm takes as input a ciphertext ct, a secret key sk and outputs $z \in \mathcal{M} \cup \{\perp\}$.

Permissions Define $f : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}$ by the following:

$$f(\mathbf{x}, \mathbf{y}) = \bigwedge_{i=1}^{\ell} x_i \vee y_i$$

We will use this function to determine when secret keys with attribute vectors $\mathbf{x}$ are "permitted" to decrypt ciphertexts with attribute vectors $\mathbf{y}$.

Correctness We require the following properties for correctness:

- For every $\lambda \in \mathbb{N}, \ell \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, $\mathbf{x} \in \{0,1\}^\ell$, $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$, message $m \in \mathcal{M}_\lambda$ and $\mathsf{ct} \leftarrow \mathsf{Enc\text{-}PK}(\mathsf{pk}, m)$, $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m$.
- For every $\lambda \in \mathbb{N}, \ell \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, $\mathbf{x} \in \{0,1\}^\ell$, $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$, message $m \in \mathcal{M}_\lambda$, $\mathbf{y} \in \{0,1\}^\ell$ and $\mathsf{ct} \leftarrow \mathsf{Enc\text{-}SK}(\mathsf{msk}, m, \mathbf{y})$, if $f(\mathbf{x}, \mathbf{y}) = 1$ then $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m$.

4.1 Security

Oracles To begin, we define two oracles we use to enable the adversary to query for ciphertexts and secret keys. Let m be a message, and $\mathbf{x}, \mathbf{y}, \in \{0,1\}^\ell$.

- $\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}(\mathbf{x}) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$.
- $\mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}(m, \mathbf{y}) \leftarrow \mathsf{Enc\text{-}SK}(\mathsf{msk}, m, \mathbf{y})$.

Experiments We will now define three security properties that a mixed bit matching encryption scheme must satisfy. These definitions are similar to the indistinguishability-based data/function privacy definitions for attribute based encryption. For each of these experiments we restrict the adversary's queries to the ciphertext and secret key oracles to prevent trivial distinguishing strategies. Also, we will be considering selective definitions, since our constructions achieve selective security, and selective security suffices for our risky traitor tracing application. One could also consider full (adaptive) versions of these security definitions.

Definition 6. *A mixed bit matching encryption scheme* $\mathsf{mBME} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc\text{-}PK}, \mathsf{Enc\text{-}SK}, \mathsf{Dec})$ *is said to satisfy* pk-sk ciphertext indistinguishability *if for any polynomial* $\ell(\cdot)$ *and stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for all security parameters* $\lambda \in \mathbb{N}$*,* $\Pr[1 \leftarrow \mathsf{Expt\text{-}pk\text{-}sk\text{-}ct}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)] \leq 1/2 + negl(\lambda)$*, where* $\mathsf{Expt\text{-}pk\text{-}sk\text{-}ct}$ *is defined in Figure 2.*

Experiment $\mathsf{Expt\text{-}pk\text{-}sk\text{-}ct}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)$

- $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell(\lambda)})$
- $m \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{pk})$.
- $\mathsf{ct}_0 \leftarrow \mathsf{Enc\text{-}SK}(\mathsf{msk}, m, 1^{\ell(\lambda)})$, $\mathsf{ct}_1 \leftarrow \mathsf{Enc\text{-}PK}(\mathsf{pk}, m)$, $b \leftarrow \{0, 1\}$
- $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{ct}_b)$.
- Output 1 if $b = b'$, and 0 otherwise.

Fig. 2: Public-key vs Secret-key Ciphertext Indistinguishability Experiment

Definition 7. *A mixed bit matching encryption scheme* $\mathsf{mBME} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc\text{-}PK}, \mathsf{Enc\text{-}SK}, \mathsf{Dec})$ *is said to satisfy* selective ciphertext hiding *if for any polynomial* $\ell(\cdot)$ *and stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for all security parameters* $\lambda \in \mathbb{N}$*,* $\Pr[1 \leftarrow \mathsf{Expt\text{-}ct\text{-}ind}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)] \leq 1/2 + negl(\lambda)$*, where* $\mathsf{Expt\text{-}ct\text{-}ind}$ *is defined in Figure 3.*

Definition 8. *A mixed bit matching encryption scheme* $\mathsf{mBME} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc\text{-}PK}, \mathsf{Enc\text{-}SK}, \mathsf{Dec})$ *is said to satisfy* selective key hiding *if for any polynomial* $\ell(\cdot)$ *and stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for all security parameters* $\lambda \in \mathbb{N}$*,* $\Pr[1 \leftarrow \mathsf{Expt\text{-}key\text{-}ind}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)] \leq 1/2 + negl(\lambda)$*, where* $\mathsf{Expt\text{-}key\text{-}ind}$ *is defined in Figure 4.*

4.2 Simplified Ciphertext Hiding

As a tool for proving mixed bit matching encryption constructions secure, we define two simplified ciphertext hiding experiments, and then show that they imply the original (selective) ciphertext hiding security game.

Experiment $\mathsf{Expt\text{-}ct\text{-}ind}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)$

- $(\mathbf{y}_0, \mathbf{y}_1) \leftarrow \mathcal{A}(1^\lambda)$.
- $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell(\lambda)})$
- $(m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{pk})$
- $b \leftarrow \{0,1\}$, $\mathsf{ct}_b \leftarrow \mathsf{Enc\text{-}SK}(\mathsf{msk}, m_b, \mathbf{y}_b)$
- $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{ct}_b)$
- Output 1 if $b = b'$, and 0 otherwise.

Adversarial Restrictions: For all queries $\mathbf{x}$ made by $\mathcal{A}$ to $\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}$ the following conditions must hold:

- If $m_0 = m_1$, then $f(\mathbf{x}, \mathbf{y}_0) = f(\mathbf{x}, \mathbf{y}_1)$.
- If $m_0 \neq m_1$, then $f(\mathbf{x}, \mathbf{y}_0) = f(\mathbf{x}, \mathbf{y}_1) = 0$.

Fig. 3: Ciphertext Hiding Experiment

Experiment $\mathsf{Expt\text{-}key\text{-}ind}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)$

- $(\mathbf{x}_0, \mathbf{x}_1) \leftarrow \mathcal{A}(1^\lambda)$
- $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell(\lambda)})$
- $b \leftarrow \{0,1\}$, $\mathsf{sk}_b \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x}_b)$
- $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{pk}, \mathsf{sk}_b)$
- Output 1 if $b = b'$, and 0 otherwise.

Adversarial Restrictions: For all queries $(m, \mathbf{y})$ made by $\mathcal{A}$ to $\mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}$ the following equality must hold: $f(\mathbf{x_0}, \mathbf{y}) = f(\mathbf{x_1}, \mathbf{y})$.

Fig. 4: Key Hiding Experiment

Definition 9. *A mixed bit matching encryption scheme* $\mathsf{mBME} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc\text{-}PK}, \mathsf{Enc\text{-}SK}, \mathsf{Dec})$ *is said to satisfy* selective 1-attribute ciphertext hiding *if for any polynomial* $\ell(\cdot)$ *and stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for all security parameters* $\lambda \in \mathbb{N}$, $\Pr[1 \leftarrow \mathsf{Expt\text{-}1\text{-}attr\text{-}ct\text{-}ind}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)] \leq 1/2 + negl(\lambda)$*, where* $\mathsf{Expt\text{-}1\text{-}attr\text{-}ct\text{-}ind}$ *is defined in Figure 5.*

Definition 10. *A mixed bit matching encryption scheme* $\mathsf{mBME} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc\text{-}PK}, \mathsf{Enc\text{-}SK}, \mathsf{Dec})$ *is said to satisfy* selective ciphertext indistinguishability under chosen attributes *if for any polynomial* $\ell(\cdot)$ *and stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for all security parameters* $\lambda \in \mathbb{N}$, $\Pr[1 \leftarrow \mathsf{Expt\text{-}IND\text{-}CA}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)] \leq 1/2 + negl(\lambda)$*, where* $\mathsf{Expt\text{-}IND\text{-}CA}$ *is defined in Figure 6.*

Theorem 1. *If a mixed bit matching encryption scheme* $\mathsf{mBME} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc\text{-}PK}, \mathsf{Enc\text{-}SK}, \mathsf{Dec})$ *satisfies selective 1-attribute ciphertext hiding (Definition 9) and selective ciphertext indistinguishability under chosen attributes (Definition 10), then it also satisfies selective ciphertext hiding (Definition 7).*

Experiment $\mathsf{Expt\text{-}1\text{-}attr\text{-}ct\text{-}ind}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)$

- $(\mathbf{y}_0, \mathbf{y}_1) \leftarrow \mathcal{A}(1^\lambda)$ where $y_{0,i} \neq y_{1,i}$ for at most one $i \in \{1, \ldots, \ell(\lambda)\}$.
- $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell(\lambda)})$
- $m \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{pk})$
- $b \leftarrow \{0,1\}$, $\mathsf{ct}_b \leftarrow \mathsf{Enc\text{-}SK}(\mathsf{msk}, m_b, \mathbf{y}_b)$.
- $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{pk}, \mathsf{ct}_b)$
- Output 1 if $b = b'$, and 0 otherwise.

Adversarial Restrictions: For all queries $\mathbf{x}$ made by $\mathcal{A}$ to $\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}$ the following equality must hold: $f(\mathbf{x}, \mathbf{y}_0) = f(\mathbf{x}, \mathbf{y}_1)$.

Fig. 5: 1-Attribute Ciphertext Hiding Experiment

Experiment $\mathsf{Expt\text{-}IND\text{-}CA}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)$

- $\mathbf{y} \leftarrow \mathcal{A}(1^\lambda)$.
- $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell(\lambda)})$
- $(m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{pk})$
- $b \leftarrow \{0,1\}$, $\mathsf{ct}_b \leftarrow \mathsf{Enc\text{-}SK}(\mathsf{msk}, m_b, \mathbf{y})$.
- $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{pk}, \mathsf{ct}_b)$
- Output 1 if $b = b'$, and 0 otherwise.

Adversarial Restrictions: If $m_0 \neq m_1$, then for all queries $\mathbf{x}$ made by $\mathcal{A}$ to $\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}$ the following equality must hold: $f(\mathbf{x}, \mathbf{y}) = 0$.

Fig. 6: Ciphertext Indistinguishability under Chosen Attributes Experiment

The proof of above theorem is provided in the full version.

4.3 Simplified Key Hiding

We also define a similar simplified experiment for the key hiding security property.

Definition 11. *A mixed bit matching encryption scheme* $\mathsf{mBME} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc\text{-}PK}, \mathsf{Enc\text{-}SK}, \mathsf{Dec})$ *is said to satisfy* selective 1-attribute key hiding *if for any polynomial ℓ and stateful PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all security parameters $\lambda \in \mathbb{N}$,* $\Pr[1 \leftarrow \mathsf{Expt\text{-}1\text{-}attr\text{-}key\text{-}ind}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)] \leq 1/2 + negl(\lambda)$, *where* $\mathsf{Expt\text{-}1\text{-}attr\text{-}key\text{-}ind}$ *is defined in Figure 7.*

Theorem 2. *If a mixed bit matching encryption scheme* $\mathsf{mBME} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc\text{-}PK}, \mathsf{Enc\text{-}SK}, \mathsf{Dec})$ *satisfies 1-attribute key hiding (Definition 11) then it satisfies key hiding (Definition 8).*

The proof of above theorem is provided in the full version.

Experiment $\mathsf{Expt\text{-}1\text{-}attr\text{-}key\text{-}ind}^{\mathsf{mBME}}_{\ell(\lambda),\mathcal{A}}(1^\lambda)$

- $(\mathbf{x}_0, \mathbf{x}_1) \leftarrow \mathcal{A}(1^\lambda)$ where $x_{0,i} \neq x_{1,i}$ for at most one i.
- $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell(\lambda)})$.
- $b \leftarrow \{0,1\}$, $\mathsf{sk}_b \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x}_b)$.
- $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sk}}_{\mathsf{msk}}, \mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}}(\mathsf{pk}, \mathsf{sk}_b)$.
- Output 1 if $b = b'$.

Adversarial Restrictions: For all queries $(m, \mathbf{y})$ made by $\mathcal{A}$ to $\mathcal{O}^{\mathsf{ct}}_{\mathsf{msk}}$ the following equality must hold: $f(\mathbf{x}_0, \mathbf{y}) = f(\mathbf{x}_1, \mathbf{y})$.

Fig. 7: 1-Attribute Key Hiding Experiment

5 Building Risky Traitor Tracing using Mixed Bit Matching Encryption

In this section, we provide a generic construction for risky traitor tracing schemes from any mixed bit matching encryption scheme. Our transformation leads to a risky traitor tracing scheme with secret-key tracing. The *risky-ness* of the scheme will be $f = \frac{k}{n+k-1} - O\left(\frac{k(k-1)}{n^2}\right)$,[6] where k can be thought of as a scheme parameter fixed during setup, and the size of ciphertext will grow with k.

5.1 Construction

- $\mathsf{Setup}(1^\lambda, 1^n)$: The setup algorithm chooses a key pair for mixed bit matching encryption system as $(\mathsf{mbme.pk}, \mathsf{mbme.msk}) \leftarrow \mathsf{mBME.Setup}(1^\lambda, 1^{k+1})$. Next, it samples an index w as $w \leftarrow \{-k+2, -k+3, \ldots, n-1, n\}$, and sets vectors $\mathbf{x}_i$ for $i \in [n]$ as

$$\mathbf{x}_i = \begin{cases} 0^{k+1} & \text{if } i < w, \\ 0^{k-i+w} 1^{i-w+1} & \text{if } w \leq i < w+k, \\ 1^{k+1} & \text{otherwise.} \end{cases}$$

  It sets the master secret key as $\mathsf{msk} = (\mathsf{mbme.msk}, w)$, public key as $\mathsf{pk} = \mathsf{mbme.pk}$, and computes the n user secret keys as $\mathsf{sk}_i \leftarrow \mathsf{mBME.KeyGen}(\mathsf{mbme.msk}, \mathbf{x}_i)$ for $i \in [n]$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: The encryption algorithm outputs the ciphertext ct as $\mathsf{ct} \leftarrow \mathsf{mBME.Enc\text{-}PK}(\mathsf{pk}, m)$.
- $\mathsf{Dec}(\mathsf{sk}, m)$: The decryption algorithm outputs the message m as $m = \mathsf{mBME.Dec}(\mathsf{sk}, \mathsf{ct})$.

[6] We want to point out that for $k = 1$ we get the tight *risky-ness*, i.e. prove that our scheme is $\frac{1}{n}$-risky secure.

– $\mathsf{Trace}^D(\mathsf{msk}, 1^y, m_0, m_1)$: Let $\mathsf{msk} = (\mathsf{mbme.msk}, w)$. To define the trace algorithm, we first define a special index encryption algorithm $\mathsf{Enc\text{-}ind}$ which takes as input a master secret key msk, message m, and an index $i \in [k+1]$.

$\mathsf{Enc\text{-}ind}(\mathsf{msk}, m, i)$: The index encryption algorithm outputs $\mathsf{ct} \leftarrow \mathsf{mBME.Enc\text{-}SK}(\mathsf{msk}, m, 1^{k+1-i}0^i)$.

Next, consider the $\mathsf{Subtrace}$ algorithm defined in Figure 8. The sub-tracing algorithm simply tests whether the decoder box uses the key for user $i+w-1$ where i is one of the inputs provided to $\mathsf{Subtrace}$. Now the tracing algorithm simply runs the $\mathsf{Subtrace}$ algorithm for all indices $i \in [k]$, and for each index i where the $\mathsf{Subtrace}$ algorithm outputs 1, the tracing algorithm adds index $i + w - 1$ to the set of traitors. Concretely, the algorithm runs as follows:

- Let $S = \emptyset$. For $i = 1$ to k:
  - Compute $b \leftarrow \mathsf{Subtrace}(\mathsf{mbme.msk}, 1^y, m_0, m_1, i)$.
  - If $b = 1$, set $S := S \cup \{i + w - 1\}$.
- Output S.

Algorithm $\mathsf{Subtrace}(\mathsf{msk}, 1^y, m_0, m_1, i)$

Inputs: Key msk, parameter y, messages m_0, m_1, index i
Output: 0/1
Let $\epsilon = \lfloor 1/y \rfloor$. It sets $T = \lambda \cdot n/\epsilon$, and $\mathsf{count}_1 = \mathsf{count}_2 = 0$. For $j = 1$ to T, it computes the following:

1. It chooses $b_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}_{j,1} \leftarrow \mathsf{Enc\text{-}ind}(\mathsf{msk}, m_{b_j}, i)$ and sends $\mathsf{ct}_{j,1}$ to D. If D outputs b_j, set $\mathsf{count}_1 = \mathsf{count}_1 + 1$, else set $\mathsf{count}_1 = \mathsf{count}_1 - 1$.
2. It chooses $c_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}_{j,2} \leftarrow \mathsf{Enc\text{-}ind}(\mathsf{msk}, m_{c_j}, i+1)$ and sends $\mathsf{ct}_{j,2}$ to D. If D outputs c_j, set $\mathsf{count}_2 = \mathsf{count}_2 + 1$, else set $\mathsf{count}_2 = \mathsf{count}_2 - 1$.

If $\mathsf{count}_1 - \mathsf{count}_2 > T \cdot (\epsilon/4n)$, output 1, else output 0.

Fig. 8: $\mathsf{Subtrace}$

Correctness. Since the encryption algorithm simply runs the public-key encryption algorithm for mixed bit matching encryption the correctness of above scheme follows directly from the correctness of the mixed bit matching encryption scheme.

Singular Trace Property. Note that if k is fixed to be 1, then our scheme satisfies the singular trace property as defined in Definition 5. This is because the trace algorithm will either output the fixed index w (chosen during setup), or output an empty set.

Due to space constraints, the proof of security is provided in the full version.

6 Construction: Mixed Bit Matching Encryption Scheme

Let $\mathsf{Grp\text{-}Gen}$ be an algorithm that takes as input security parameter 1^λ and outputs $\mathsf{params} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot), g_1, g_2)$ where p is a λ bit prime, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order p, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable non-degenerate bilinear map and g_1, g_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively.

- $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{mBME.Setup}(1^\lambda, 1^\ell)$: The setup algorithm first chooses $\mathsf{params} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot), g_1, g_2) \leftarrow \mathsf{Grp\text{-}Gen}(1^\lambda)$. It chooses $\alpha \leftarrow \mathbb{Z}_p$, $a_i \leftarrow \mathbb{Z}_p, b_i \leftarrow \mathbb{Z}_p, c_i \leftarrow \mathbb{Z}_p$ for each $i \in [\ell]$. The public key consists of params, $e(g_1, g_2)^\alpha$, $\prod_{i\in[\ell]} g_1^{a_i \cdot b_i + c_i}$ and $\{g_1^{a_i}\}_{i\in[\ell]}$, while the master secret key consists of $\big(\mathsf{params}, \alpha, \{a_i, b_i, c_i\}_{i\in\ell}\big)$.
- $\mathsf{sk} \leftarrow \mathsf{mBME.KeyGen}(\mathbf{x}, \mathsf{msk})$: Let $\mathsf{msk} = \big(\mathsf{params}, \alpha, \{a_i, b_i, c_i\}_{i\in\ell}\big)$. The key generation algorithm first chooses $t \leftarrow \mathbb{Z}_p$ and $u_i \leftarrow \mathbb{Z}_p$ for each $i \in [\ell]$. It computes $K_0 = g_2^\alpha \cdot \Big(\prod_{i\in[\ell]} g_2^{-t\cdot c_i}\Big) \cdot \big(\prod_{i:x_i=0} g_2^{-u_i \cdot a_i}\big)$. Next, it sets $K_1 = g_2^t$, and for each $i \in [\ell]$, $K_{2,i} = g^{-t\cdot b_i}$ if $x_i = 1$, else $K_{2,i} = g_2^{-t\cdot b_i + u_i}$. The key is $\Big(K_0, K_1, \{K_{2,i}\}_{i\in[\ell]}\Big)$.
- $\mathsf{ct} \leftarrow \mathsf{mBME.Enc\text{-}SK}(m, \mathbf{y}, \mathsf{msk})$: Let $\mathsf{msk} = \big(\mathsf{params}, \alpha, \{a_i, b_i, c_i\}_{i\in\ell}\big)$. The secret key encryption algorithm first chooses $s \leftarrow \mathbb{Z}_p$, and for each $i \in [\ell]$ such that $y_i = 0$, it chooses $r_i \leftarrow \mathbb{Z}_p$. It sets $C = m \cdot e(g_1, g_2)^{\alpha\cdot s}$, $C_0 = g_1^s$, $C_1 = \Big(\prod_{i:y_i=1} g_1^{s\cdot(a_i\cdot b_i + c_i)}\Big) \cdot \Big(\prod_{i:y_i=0} g_1^{s\cdot c_i + a_i\cdot b_i \cdot r_i}\Big)$. For each $i \in [\ell]$, it sets $C_{2,i} = g_1^{a_i\cdot s}$ if $y_i = 1$, else $C_{2,i} = g_1^{a_i\cdot r_i}$ if $y_i = 0$. The ciphertext is $\Big(C, C_0, C_1, \{C_{2,i}\}_{i\in[\ell]}\Big)$.
- $\mathsf{ct} \leftarrow \mathsf{mBME.Enc\text{-}PK}(m, \mathsf{pk})$: Let $\mathsf{pk} = (\mathsf{params}, e(g_1, g_2)^\alpha, \prod_{i\in\ell} g_1^{a_i\cdot b_i + c_i}, \{g_1^{a_i}\}_{i\in[\ell]})$. The public key encryption algorithm is identical to the secret key encryption algorithm. It first chooses $s \leftarrow \mathbb{Z}_p$. It sets $C = m \cdot e(g_1, g_2)^{\alpha\cdot s}$, $C_0 = g_1^s$, $C_1 = \Big(\prod_{i\in\ell} g_1^{a_i\cdot b_i + c_i}\Big)^s$. For each $i \in [\ell]$, it sets $C_{2,i} = (g_1^{a_i})^s$. The ciphertext is $\Big(C, C_0, C_1, \{C_{2,i}\}_{i\in[\ell]}\Big)$.
- $z \leftarrow \mathsf{mBME.Dec}(\mathsf{ct}, \mathsf{sk})$: Let $\mathsf{ct} = \Big(C, C_0, C_1, \{C_{2,i}\}_{i\in[\ell]}\Big)$ and $\mathsf{sk} = \Big(K_0, K_1, \{K_{2,i}\}_{i\in[\ell]}\Big)$. The decryption algorithm outputs

$$\frac{C}{e(C_0, K_0) \cdot e(C_1, K_1) \cdot \prod_{i\in[\ell]} e(C_{2,i}, K_{2,i})}.$$

6.1 Correctness

Fix any security parameter λ, message m, vectors $\mathbf{x}, \mathbf{y}$ such that $f(\mathbf{x}, \mathbf{y}) = 1$ and public key $\mathsf{pk} = (\mathsf{params}, e(g_1, g_2)^\alpha, \prod_{i\in[\ell]} g_1^{a_i\cdot b_i + c_i}, \{g_1^{a_i}\}_{i\in[\ell]})$. Let $(s, \{r_i\}_{i:y_i=0})$ be the randomness used during encryption, $(t, \{u_i\}_{i:x_i=0})$ the randomness used during key generation, ciphertext $\mathsf{ct} = (C, C_0, C_1, \{C_{2,i}\}_{i\in[\ell]})$ and key $\mathsf{sk} =$

$(K_0, K_1, \{K_{2,i}\}_{i\in[\ell]})$. To show that decryption works correctly, it suffices to show that $e(C_0, K_0) \cdot e(C_1, K_1) \cdot \left(\prod_{i\in[\ell]} e(C_{2,i}, K_{2,i})\right) = e(g_1, g_2)^{\alpha\cdot s}$.

$$
\begin{aligned}
& e(C_0, K_0) \cdot e(C_1, K_1) \cdot \left(\prod_{i\in[\ell]} e(C_{2,i}, K_{2,i})\right) \\
&= \left(e(g_1, g_2)^{\alpha\cdot s - (\sum_i s\cdot t\cdot c_i) - (\sum_{i:x_i=0} s\cdot u_i\cdot a_i)}\right) \\
&\quad \cdot \left(e(g_1, g_2)^{(\sum_i s\cdot t\cdot c_i) + (\sum_{i:y_i=1} s\cdot t\cdot a_i\cdot b_i) + (\sum_{i:y_i=0} t\cdot a_i\cdot b_i\cdot r_i)}\right) \\
&\quad \cdot \left(e(g_1, g_2)^{-(\sum_{i:y_i=1} t\cdot s\cdot a_i\cdot b_i) - (\sum_{i:y_i=0} t\cdot a_i\cdot b_i\cdot r_i) + (\sum_{i:x_i=0} a_i\cdot s\cdot u_i)}\right)
\end{aligned}
$$

In the second step, we use the fact that since $f(\mathbf{x}, \mathbf{y}) = 1$, whenever $x_i = 0, y_i = 1$ (if this was not the case, then we would have, for all i such that $x_i = y_i = 0$, $e(g_1, g_2)^{u_i\cdot a_i\cdot r_i}$ terms in the product). Simplifying the expression, we get the desired product $e(g_1, g_2)^{\alpha\cdot s}$.

Due to space constraints, the proof of security is provided in the full version.

7 Performance Evaluation

We provide the performance evaluation of our risky traitor tracing scheme obtained by combining the mixed bit matching encryption scheme and the transformation to risky TT provided in Sections 6 and 5, respectively. Our performance evaluation is based on concrete measurements made using the RELIC library [1] written in the C language.

We use the BN254 curve for pairings. It provides 126-bit security level [3]. All running times below were measured on a server with 2.93 GHz Intel Xeon CPU and 40GB RAM. Averaged over 10000 iterations, the time taken to perform an exponentiation in the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ is approximately 0.28 ms, 1.60 ms and 0.90 ms, respectively. The time for perform a pairing operation is around 2.22 ms. The size of elements in group $\mathbb{G}_1$ is 96 bytes.

Based on the above measurements, for risky traitor tracing with parameter k we get the ciphertext size as $(96\cdot k + 288)$ bytes, encryption time $(0.28\cdot k + 1.74)$ ms, and decryption time $(2.226\cdot k + 6.66)$ ms.[7] We point out in the above evaluations we consider the KEM version of our risky traitor tracing in which the message is encrypted using a symmetric key encryption with the hash of the first component of ciphertext $e(g_1, g_2)^{\alpha\cdot s}$ is used as the secret key. That is, the hashed value could be used as an AES key to perform message encryptions. For the basic setting of risky traitor tracing, i.e. $k = 1$, we get the ciphertext size, encryption time, and decryption time to be around 384 bytes, 2.16 ms, 8.89 ms (respectively).

[7] In these estimations, we ignore the time to evaluate the hash function on the element in the target group $\mathbb{G}_T$ since it has an insiginicant effect on the running time.

8 Hardness of Differentially Private Sanitization

In this section, we show that the Dwork et al. [14] result works even if the traitor tracing scheme is f-risky secure. This, together with our risky TT constructions, results in a hardness result with query set size $2^{O(\lambda)}$ and based on assumptions over bilinear groups. First, we introduce some differential privacy related preliminaries following the notations from [23]. Next, we describe our hardness result.

8.1 Definitions

Differentially Private Algorithms. A database $D \in \mathcal{X}^n$ is a collection of n rows $x_1, \ldots, x_n$, where each row is an element of the date universe $\mathcal{X}$. We say that two databases $D, D' \in \mathcal{X}^*$ are adjacent, denoted by $D \sim D'$, if D' can be obtained from D by the addition, removal, or substitution of a single row (i.e., they differ only on a single row). Also, for any database $D \in \mathcal{X}^n$ and index $i \in \{1, 2, \ldots, n\}$, we use D_{-i} to denote a database where the i^{th} element/row in D is set removed. At a very high level, an algorithm is said to be differentially private if its behavior on all adjacent databases is similar. The formal definition is provided below.

Definition 12 (Differential Privacy [13]). *Let $A : \mathcal{X}^n \to \mathcal{S}_n$ be a randomized algorithm that takes a database as input and outputs a summary. A is (ϵ, δ)-differentially private if for every pair of adjacent databases $D, D' \in \mathcal{X}^n$ and every subset $T \subseteq \mathcal{S}_n$,*

$$\Pr[A(D) \in T] \leq e^{\epsilon} \Pr[A(D') \in T] + \delta.$$

Here parameters ϵ and δ could be functions in n, the size of the database.

Accuracy of Sanitizers. Note that any algorithm A that always outputs a fixed symbol, say $\perp$, already satisfies Definition 12. Clearly such a summary will never be useful as the summary does not contain any information about the underlying database. Thus, we also need to specify what it means for the sanitizer to be useful. As described before, in this work we study the notion of differentially private sanitizers that give accurate answers to *statistical* queries.[8] A statistical query on data universe $\mathcal{X}$ is defined by a binary predicate $q : \mathcal{X} \to \{0, 1\}$. Let $\mathcal{Q} = \{q : \mathcal{X} \to [0, 1]\}$ be a set of statistical queries on the data universe $\mathcal{X}$. Given any $n \in \mathbb{N}$, database $D \in \mathcal{X}^n$ and query $q \in \mathcal{Q}$, let $q(D) = \dfrac{\sum_{x \in D} q(x)}{n}$.

Before we define accuracy, we would like to point out that the algorithm A might represent the summary s of a database D is any arbitrary form. Thus, to extract the answer to each query q from summary s, we require that there exists an evaluator $\mathsf{Eval} : \mathcal{S} \times \mathcal{Q} \to [0, 1]$ that takes the summary and a query, and outputs an approximate answer to that query. As in prior works, we will abuse notation and simply write $q(s)$ to denote $\mathsf{Eval}(s, q)$, i.e. the algorithm's answer to query q. At a high level, an algorithm is said to be accurate if it answers every query to within some bounded error. The formal definition follows.

[8] Statistical queries are also referred as counting queries, predicate queries, or linear queries in the literature.

Definition 13 (Accuracy). *For a set $\mathcal{Q}$ of statistical queries on $\mathcal{X}$, a database $D \in \mathcal{X}^n$ and a summary $s \in \mathcal{S}$, we say that s is α-accurate for $\mathcal{Q}$ on D if*

$$\forall q \in \mathcal{Q}, |q(D) - q(s)| \leq \alpha.$$

A randomized algorithm $A : \mathcal{X}^n \to \mathcal{S}$ is said to be an (α, β)-accurate sanitizer if for every database $D \in \mathcal{X}^n$,

$$\Pr_{A\text{'s coins}}[A(D) \text{ is } \alpha\text{-accurate for } \mathcal{Q} \text{ on } D] \geq 1 - \beta.$$

The parameters α and β could be functions in n, the size of the database.

Efficiency of Sanitizers. In this work, we are interested in asymptotic efficiency, thus we introduce a computation parameter $\lambda \in \mathbb{N}$. The data universe and query space, both will be parameterized by λ; that is, for every $\lambda \in \mathbb{N}$, we have a data universe $\mathcal{X}_\lambda$ and a query space $\mathcal{Q}_\lambda$. The size of databases will be bounded by $n = n(\lambda)$, where $n(\cdot)$ is a polynomial. Now the algorithm A takes as input a database $\mathcal{X}_\lambda^n$ and output a summary in $\mathcal{S}_\lambda$, where $\{\mathcal{S}_\lambda\}_{\lambda \in \mathbb{N}}$ is a sequence of output ranges. And, there is an associated evaluator Eval that takes a query $q \in \mathcal{Q}_\lambda$ and a summary $S \in \mathcal{S}_\lambda$ and outputs a real-valued answer. The definitions of differential privacy and accuracy readily extend to such sequences.

Definition 14 (Efficiency). *A sanitizer A is efficient if, on input a database $D \in \mathcal{X}_\lambda^n$, A runs in time $\mathsf{poly}(\lambda, \log(|X_\lambda|), \log(|Q_\lambda|))$, as well as on input a summary $s \in \mathcal{S}_\lambda$ and query $q \in \mathcal{Q}_\lambda$, the associated evaluator Eval runs in time $\mathsf{poly}(\lambda, \log(|X_\lambda|), \log(|Q_\lambda|))$.*

8.2 Hardness of Efficient Differentially Private Sanitization from Risky Traitor Tracing

In this section, we prove hardness of efficient differentially private sanitization from risky traitor tracing schemes. The proof is an adaptation of the proofs in [14, 30, 23] to this restricted notion. At a high level, the idea is to set the data universe to the secret key space and each query will be associated with a ciphertext such that answer to a query on any secret key will correspond to the output of decryption of associated ciphertext using the secret key. Now to show hardness of sanitization we will prove by contradiction. The main idea is that if there exists an efficient (accurate) sanitizer, then that could be successfully used as a pirate box in the traitor tracing scheme. Next, assuming that the sanitizer satisfies differential privacy, we can argue that the sanitizer could still be a useful pirate box even if one of keys in the database is deleted, however the tracing algorithm will still output the missing key as a traitor with non-negligible probability, thereby contradicting the property that the tracing algorithm incorrectly traces with only negligible probability.

Below we state the formal theorem. The proof of this theorem can be found in the full version. Later we also show to get a stronger hardness result if the underlying risky traitor tracing schemes also satisfies "singular trace" property (Definition 5).

Hardness from Risky Traitor Tracing

Theorem 3. *If there exists a f-risky secure private-key no-query traitor tracing scheme $\mathcal{T} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$, then there exists a data universe and query family $\{\mathcal{X}_\lambda, \mathcal{Q}_\lambda\}_\lambda$ such that there does not any sanitizer $A : \mathcal{X}_\lambda^n \to \mathcal{S}_\lambda$ that is simultaneously — (1) (ϵ, δ)-differentially private, (2) (α, β)-accurate for query space $\mathcal{Q}_\lambda$ on $\mathcal{X}_\lambda^n$, and (3) computationally efficient — for any $\epsilon = O(\log \lambda), \alpha < 1/2, \beta = o(1)$ and $\delta \leq f \cdot (1-\beta)/4n$.*

Theorem 4. *If there exists a f-risky secure private-key no-query traitor tracing scheme $\mathcal{T} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ satisfying singular trace property (Definition 5), then there exists a data universe and query family $\{\mathcal{X}_\lambda, \mathcal{Q}_\lambda\}_\lambda$ such that there does not any sanitizer $A : \mathcal{X}_\lambda^n \to \mathcal{S}_\lambda$ that is simultaneously — (1) (ϵ, δ)-differentially private, (2) (α, β)-accurate for query space $\mathcal{Q}_\lambda$ on $\mathcal{X}_\lambda^n$, and (3) computationally efficient — for any $\epsilon = O(\log \lambda), \alpha < 1/2, \beta = o(1)$ and $\delta \leq f \cdot (1-\beta)/4$.*

Hardness from Assumptions over Bilinear Groups Combining Theorem 4 with our risky TT scheme over prime order bilinear groups, we get the following corollary.

Corollary 1. *If Assumption 1 and Assumption 2 hold, then there exists a data universe and query family $\{\mathcal{X}_\lambda, \mathcal{Q}_\lambda\}_\lambda$ such that there does not any sanitizer $A : \mathcal{X}_\lambda^n \to \mathcal{S}_\lambda$ that is simultaneously — (1) (ϵ, δ)-differentially private, (2) (α, β)-accurate for query space $\mathcal{Q}_\lambda$ on $\mathcal{X}_\lambda^n$, and (3) computationally efficient — for any $\epsilon = O(\log \lambda), \alpha < 1/2, \beta = o(1)$ and $\delta \leq (1-\beta)/4n$.*

Similary, combining Theorem 4 with our risky TT scheme over composite order bilinear groups, we get the following corollary.

Corollary 2. *Assuming subgroup decision and subgroup hiding in target group assumptions, there exists a data universe and query family $\{\mathcal{X}_\lambda, \mathcal{Q}_\lambda\}_\lambda$ such that there does not any sanitizer $A : \mathcal{X}_\lambda^n \to \mathcal{S}_\lambda$ that is simultaneously — (1) (ϵ, δ)-differentially private, (2) (α, β)-accurate for query space $\mathcal{Q}_\lambda$ on $\mathcal{X}_\lambda^n$, and (3) computationally efficient — for any $\epsilon = O(\log \lambda), \alpha < 1/2, \beta = o(1)$ and $\delta \leq (1-\beta)/4n$.*

Acknowledgements

The fourth author is supported by NSF CNS-1414082, DARPA SafeWare, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

References

1. Relic toolkit, `https://github.com/relic-toolkit/relic`

2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: CRYPTO. pp. 1–18 (2001)
3. Beuchat, J.L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In: International Conference on Pairing-Based Cryptography. pp. 21–39. Springer (2010)
4. Billet, O., Phan, D.H.: Efficient traitor tracing from collusion secure codes. In: Information Theoretic Security, Third International Conference, ICITS 2008, Calgary, Canada, August 10-13, 2008, Proceedings. pp. 171–182 (2008)
5. Boneh, D., Franklin, M.K.: An efficient public key traitor tracing scheme. In: Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. pp. 338–353 (1999)
6. Boneh, D., Naor, M.: Traitor tracing with constant size ciphertext. In: Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008. pp. 501–510 (2008)
7. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings. pp. 573–592 (2006)
8. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 211–220. ACM (2006)
9. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I. pp. 480–499 (2014)
10. Chabanne, H., Phan, D.H., Pointcheval, D.: Public traceability in traitor tracing schemes. In: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. pp. 542–558 (2005)
11. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: CRYPTO. pp. 257–270 (1994)
12. Chor, B., Fiat, A., Naor, M., Pinkas, B.: Tracing traitors. IEEE Transactions on Information Theory 46(3), 893–910 (2000)
13. Dwork, C., McSherry, F., Nissim, K., Smith, A.D.: Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings. pp. 265–284 (2006)
14. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.: On the complexity of differentially private data release: Efficient algorithms and hardness results. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing. pp. 381–390. STOC '09, ACM, New York, NY, USA (2009)
15. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings. pp. 44–61 (2010)

16. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
17. Garg, S., Kumarasubramanian, A., Sahai, A., Waters, B.: Building efficient fully collusion-resilient traitor tracing and revocation schemes. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010. pp. 121–130 (2010)
18. Goyal, R., Koppula, V., Waters, B.: Collusion resistant traitor tracing from learning with errors (2018)
19. Kiayias, A., Pehlivanoglu, S.: Encryption for Digital Content, Advances in Information Security, vol. 52. Springer (2010)
20. Kiayias, A., Yung, M.: Traitor tracing with constant transmission rate. In: Advances in CryptologyEUROCRYPT 2002. pp. 450–465. Springer (2002)
21. Kiayias, A., Yung, M.: Breaking and repairing asymmetric public-key traitor tracing. In: Digital Rights Management: ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers (2003)
22. Kowalczyk, L., Malkin, T., Ullman, J., Wichs, D.: Hardness of non-interactive differential privacy from one-way functions. Cryptology ePrint Archive, Report 2017/1107 (2017), `https://eprint.iacr.org/2017/1107`
23. Kowalczyk, L., Malkin, T., Ullman, J., Zhandry, M.: Strong hardness of privacy from weak traitor tracing. In: Proceedings of TCC 2016-B (2016)
24. Naor, M., Pinkas, B.: Threshold traitor tracing. In: Advances in Cryptology-CRYPTO'98. pp. 502–517. Springer (1998)
25. Naor, M., Pinkas, B.: Efficient trace and revoke schemes. In: Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings. pp. 1–20 (2000)
26. Nishimaki, R., Wichs, D., Zhandry, M.: Anonymous traitor tracing: How to embed arbitrary information in a key. In: Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. pp. 388–419 (2016)
27. Pfitzmann, B.: Trials of traced traitors. In: Information Hiding. pp. 49–64. Springer (1996)
28. Pfitzmann, B., Waidner, M.: Asymmetric fingerprinting for larger collusions. In: Proceedings of the 4th ACM conference on Computer and communications security. pp. 151–160. ACM (1997)
29. Sirvent, T.: Traitor tracing scheme with constant ciphertext rate against powerful pirates. Cryptology ePrint Archive, Report 2006/383 (2006), `http://eprint.iacr.org/2006/383`
30. Ullman, J.: Answering $n^{\{2+o(1)\}}$ counting queries with differential privacy is hard. In: Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013. pp. 361–370 (2013)
31. Watanabe, Y., Hanaoka, G., Imai, H.: Efficient asymmetric public-key traitor tracing without trusted agents. Topics in CryptologyCT-RSA 2001 pp. 392–407 (2001)