New Techniques for Traitor Tracing: Size $N^{1/3}$ and More from Pairings

Mark Zhandry
Princeton University & NTT Research, USA

Abstract. The best existing pairing-based traitor tracing schemes have $O(\sqrt{N})$-sized parameters, which has stood since 2006. This intuitively seems to be consistent with the fact that pairings allow for degree-2 computations, yielding a quadratic compression.
In this work, we show that this intuition is false by building a traitor tracing scheme from pairings with $O(\sqrt[3]{N})$-sized parameters. We additionally give schemes with a variety of parameter size trade-offs, including a scheme with constant-size ciphertexts and public keys (but linear-sized secret keys). We obtain our schemes by developing a number of new traitor tracing techniques, giving the first significant parameter improvements in pairings-based traitor tracing in over a decade.

1 Introduction

Traitor tracing [CFN94] allows a content distributor to trace the source of a pirate decoder. Every user is given a unique secret key that allows for decrypting ciphertexts. A "traitor" might distribute their key to un-authorized users, or even hide their key inside a pirate decoder capable of decrypting. A tracing algorithm can be run on the decoder that will identify the traitor. In a collusion-resistant scheme, even if several traitors collude, the tracing algorithm will be able to identify at least one of them[1], without ever falsely identifying an honest user. Much of the traitor tracing literature considers *fully collusion-resistant* schemes, where the coalition of traitors can be arbitrarily large. In this work, we will only consider fully collusion-resistant schemes.

The main goal of traitor tracing is to build schemes with short parameters, in particular short ciphertexts that depend minimally on the number N of users. Boneh, Sahai, and Waters [BSW06] demonstrated the first collusion-resistant scheme with $O(\sqrt{N})$-sized parameters using pairings[2]. Shortly after their work, Boneh and Waters [BW06] augmented the construction with a broadcast functionality, achieving a so-called broadcast and trace scheme also with $O(\sqrt{N})$-sized parameters. These works remain the state-of-the-art in pairings-based collusion-resistant traitor tracing. Using other tools such as obfuscation or LWE, better parameters are possible [GGH$^+$13, BZ14, GKW18].

[1] A traitor could be completely passive, so it is impossible to identify *all* traitors.
[2] Following convention, the Big-Oh notation throughout this paper will hide constants that depend on the security parameter, and focus on the dependence on N.

1.1 Some Existing Approaches to Traitor Tracing

Fingerprinting Codes. One of the earliest approaches to collusion-resistant tracing was shown by Boneh and Naor [BN08][3], who construct traitor tracing using an object called *fingerprinting codes* [BS95]. Their scheme is combinatorial, relying simply on generic public key encryption, and ciphertexts have optimal $O(1)$ size.

The Boneh-Naor scheme, however, is generally not considered to resolve the traitor tracing problem. Curiously, different authors seem to have different interpretations of why. Some works (e.g. [BZ14, GKSW10, TZ17]) note that Boneh-Naor requires very large secret keys — namely quadratic in the number of users — which is inherent to fingerprinting codes [Tar03]. The main limitation according to these works appears to be simultaneously achieving small ciphertext *and* small secret/public keys. Other works more or less ignore the secret key size limitation (e.g. [GKW18, KW19, GQWW19][4]), suggesting the main limitation of Boneh-Naor is that it is a *threshold* scheme: it can only trace decoders whose decryption probability exceeds some a priori threshold. These works appear to consider it an open problem, for example, to construct *non-threshold* traitor tracing with constant-sized ciphertexts (and any secret or public key size) from anything implied by pairings.

Private Linear Broadcast Encryption (PLBE). A Private Linear Broadcast Encryption (PLBE) scheme is a limited type of functional encryption that allows for encrypting to ranges of user identities, and is known to imply traitor tracing [BSW06]. Algebraic constructions of PLBE achieve simultaneously smaller parameters, and are not subject to the threshold restriction. PLBE is by far the most popular approach to traitor tracing today, being taken by the current best pairings-based constructions [BSW06, BW06], as well as the obfuscation and LWE-based constructions [GGH+13, BZ14, GKW18]. In fact, in the last five years (2014-2019) of traitor tracing papers, we could identify ten papers appearing at EUROCRYPT, CRYPTO, ASIACRYPT, TCC, STOC, and FOCS giving positive results for traitor tracing. With perhaps one exception (discussed below) *every single one* can be seen as following the PLBE or closely related approaches [BZ14, LPSS14, NWZ16, KMUZ16, GKRW18, KMUW18, CVW+18, GKW18, GQWW19, GKW19].

Risky Traitor Tracing. Recently, Goyal et al. [GKRW18] define a relaxed notion of "risky traitor tracing" where the pirate decoder is only guaranteed to be traced with some non-zero probability, say α for some $\alpha \ll 1$. Their approach follows the PLBE framework, but actually *strengthens* PLBE. Essentially, their scheme constructs PLBE for αN users, but then since $\alpha < 1$, it must assign multiple users to the same identity. In order to get tracing to work, however, it must be that users cannot tell what identity they were assigned to. This requires strengthening PLBE, as in standard PLBE every user knows their identity.

[3] The work originated from 2002, but was not published until 2008.

[4] Example: Goyal, Koppula, and Waters [GKW18] make the central claim of achieving a "secure traitor tracing with [constant]-sized ciphertexts from standard assumptions," without discussing the secret key size of their construction at all.

1.2 This work: New Techniques for Traitor Tracing

In this work, we explore the use of different structures to build traitor tracing, giving rich set of traitor tracing techniques beyond the usual approaches. We then use these techniques to build several new schemes from pairings and weaker primitives that offer new trade-offs that were not possible before. Below we describe our results, with a summary given in Table 1.

In the following, we will say a traitor tracing system has size (P, K, C) if its public key, secret keys, and ciphertexts have sizes at most $O(P), O(K)$, and $O(C)$, respectively, where constants hidden in the Big Oh notation are allowed to depend on the security parameter[5]. We abbreviate size (A, A, A) as simply A.

- The first scheme of size $(N^2, N^2, 1)$ *without the threshold limitation* from the minimal assumption of general public key encryption[6]. Thus, we remove the threshold limitation of fingerprinting code-based tracing schemes. The main limitation of these schemes is then the large public and secret key sizes. We note that we easily can compress the public keys to get a scheme of size $(1, N^2, 1)$, relying on the stronger assumption of identity-based encryption.
- The first pairings-based scheme of size $(1, N, 1)$, or generally $(1, N^{1-a}, N^a)$ for any constant $a \in [0, 1]$. For all constants $a < 1$, this gives a new parameter trade-off that was not possible before from pairings.
- An (N^{1-a}, N^{1-a}, N^a)-sized scheme from pairings, attaining the stronger notion of broadcast and trace [BW06], which augments traitor tracing with a broadcast functionality. For $a = 0$, this gives the first broadcast and trace scheme with constant-size ciphertexts from pairings. This improves on the recent work of [GQWW19] which attained arbitrarily-small polynomial ciphertext size, while also requiring lattices in addition to pairings.
- A new model for traitor tracing, which we call the *shared randomness model* (SRM), where encryption, decryption, and the decoder have access to a large source of randomness that is not included in the communication costs. While we define the model as a stepping stone toward a full tracing algorithm in the plain model, our shared randomness model may be useful in its own right. For example, the shared randomness could be derived from some publicly available data, such as stock market fluctuations or blockchains.
- A broadcast and trace scheme of size $(N, 1, 1)$, or more generally $(N^{1-a}, 1, N^a)$ for any constant $a \in [N]$, in the shared randomness model from pairings. The size of the shared randomness is N^{1-a}; thus, for $a \geq 1/2$, the shared randomness can simply be included in the ciphertext, in which case we get a scheme in the plain model. We note that for $a = 1/2$, we get the first broadcast and trace scheme of size $(N^{1/2}, 1, N^{1/2})$ from pairings, improving on the $(N^{1/2}, N^{1/2}, N^{1/2})$-sized scheme of [BW06].
- Putting it all together: a traitor tracing (non-broadcast) scheme of size $\sqrt[3]{N}$.

[5] We will also suppress $\log N$ terms. This is without loss of generality since it is always the case that $\log N < \lambda$, and the Big-Oh already hides $\mathsf{poly}(\lambda)$ terms.

[6] Our definition of traitor tracing has public encryption, which in particular implies public key encryption.

Our results are obtained by a number of new techniques that may have applications beyond the immediate scope of this work:

- A generic procedure to increase the number of users by expanding the ciphertext size, but in many cases keeping the other parameters fixed (Theorem 1).
- A generic procedure to convert any threshold scheme into a non-threshold scheme without affecting the dependence on N (Theorem 2).
- A generic procedure to convert a risky scheme into a non-risky scheme, without asymptotically affecting ciphertext size (Theorem 3).
- A conversion from a certain broadcast functionality into a traitor tracing scheme, with shared randomness (Theorem 4).
- New instantiations of broadcast encryption from pairings (Theorem 5).

Scheme	$\|pk\|$	$\|sk\|$	$\|ct\|$	Broadcast & Trace?	Tool	Limitations
Trivial	N	1	N	✓	PKE	
	1	1	N		IBE	
[BN08]	N^2	N^2	1	✗	PKE	Threshold
	1	N^2	1		IBE	
[BSW06]	$\sqrt{N}$	1	$\sqrt{N}$	✗	Pairing	
[BW06]	$\sqrt{N}$	$\sqrt{N}$	$\sqrt{N}$	✓		
Cor 1	N^{2-a}	N^{2-2a}	N^a	✗	PKE	
	1	N^{2-2a}	N^a		IBE	
Cor 2	1	N^{1-a}	N^a		Pairing	
Cor 3	N^{1-a}	N^{1-a}	N^a	✓		
Cor 4	$N^{1/2-a/2}$	1	$N^{1/2+a/2}$			
Cor 5	$\sqrt[3]{N}$	$\sqrt[3]{N}$	$\sqrt[3]{N}$	✗		

Table 1: Comparing parameters sizes of our schemes to some existing protocols. This table only includes schemes based on pairings or weaker assumptions implied by pairings. N is the number of users. All sizes hide multiplicative constants dependent on the security parameter (but not N). $a \in [0, 1]$ is any constant.

2 Technical Overview

In order to abstract and modularize the discussion, the central object we will consider is a generalization of a traitor tracing system, which we call a "multi-scheme," which can roughly be seen as a scaled-down version of "identity-based traitor tracing" as defined in [ADM⁺07]. Intuitively, a multi-scheme is M essentially independent tracing systems running in parallel, each with distinct secret keys and ciphertexts. All N users within a single instance can decrypt ciphertexts to that instance, but not to other instances. Tracing also works within an instance: any pirate decoder that decrypts for an instance can be traced to traitors within that instance. A plain traitor tracing scheme implies a multi-scheme by simply setting up M separate instances of the scheme. The point of a multi-scheme, however, is that the M schemes are allowed share a common public key, which may be smaller than M copies of a single public key. See Definition 1.

We will also consider broadcast and trace schemes [BW06], which augment plain traitor tracing with a broadcast functionality. That is, the encrypter can specify a subset $S \subseteq [N]$, and only users in S should be able to decrypt the ciphertext. S is also incorporated into the tracing definition. See Section 4.1.

We will say that a scheme Π has size (P, K, C) for functions $P = P(N, M)$, $K = K(N, M)$, and $C = C(N, M)$, if there is a polynomial $\mathsf{poly}(\lambda)$ such that, for all polynomials $N = N(\lambda)$ and $M = M(\lambda)$, we have $|\mathsf{pk}| \leq P(N, M) \times \mathsf{poly}(\lambda)$, $|\mathsf{sk}_{j,i}| \leq K(N, M) \times \mathsf{poly}(\lambda)$, and $|c| \leq C(N, M) \times \mathsf{poly}(\lambda)$. For example, if $|\mathsf{pk}| = |\mathsf{sk}_{j,i}| = |c| = 2N^{1/2}M\lambda^2 + \lambda^5$, we could set $\mathsf{poly}(\lambda) = 2\lambda^5$, which shows that the protocol has size $(N^{1/2}M, N^{1/2}M, N^{1/2}M)$.

2.1 User Expansion Compiler

Our first result shows how to expand the number of users by grouping different instances together. That is, we compile a scheme with N/T users and MT instances into a scheme with N users and M instances. Essentially, we just partition the MT instances into M sets of size T. Within each set, there are now N users (N/T for each instance, T instances). We then encrypt the message separately to each of the T instances within the set, ensuring that all N users in the set can decrypt. This conversion blows up the ciphertext size by a factor of T, but hopefully results in smaller public/secret keys. Concretely, we prove:

Theorem 1 (User Expansion). *Let $P = P(N, M), K = K(N, M), C = C(N, M), T = T(N, M)$ be polynomials such that $T(N, M) \leq N$. Suppose there exists a secure multi-scheme Π_0 with size (P, K, C). Then there exists a secure multi-scheme Π with size $(\ P(N/T, MT)\ ,\ K(N/T, MT)\ ,\ T \times C(N/T, MT)\ )$. If Π_0 is a broadcast and trace scheme, then so is Π.*

Our compiler can be seen as a generalization of the most basic traitor tracing scheme, which simply gives each user a different secret key for a public key encryption scheme and encrypts to each user separately. Abstracting the ideas behind this scheme will lead to useful results later in this paper.

The tracing algorithm in our compiler essentially views the construction as an instance of *private linear broadcast encryption* (PLBE), and then uses a tracing algorithm analogous to [BSW06]. Given a decoder D for the compiled scheme, we test the decoder on invalid ciphertexts where the first t components have been modified to encrypt gibberish, and see if the decoder still decrypts. For a good decoder, a simple hybrid argument shows that there will be some t where the decoder decrypts $t - 1$ with probability noticeably higher than it decrypts t. This will allow us to construct from the original decoder D a new decoder D_t for Π_0, targeting the t'th instance. We then run Π_0's tracing algorithm on D_t, which will accuse a set $A \subseteq [N/T]$. For each $i \in A$, we then accuse the user who was assigned index i within instance t. See Section 5 for details.

2.2 Threshold Elimination Compiler

Our next compiler converts a *threshold* scheme — which can only trace decoders that have *constant* decryption probability — into a full tracing scheme which can trace decoders arbitrary-small inverse-polynomial decryption probability.

Theorem 2 (Threshold Elimination). *Let P, K, C be polynomials in N, M. If there exists a* threshold *secure multi-scheme Π_{Thresh} with size (P, K, C), then there exists a (non-threshold) secure multi-scheme Π with size (P, K, C). If Π_{Thresh} is a broadcast and trace scheme, then so is Π.*

As an application, the Boneh-Naor traitor tracing scheme [BN08], when instantiated with "robust" Tardos fingerprinting codes [Tar03, BKM10], yields a threshold scheme of size $(N^2, N^2, 1)$, or a multi-scheme of size $(MN^2, N^2, 1)$. Applying Theorem 2 gives a non-threshold scheme with the same size. We can also eliminate the public key size by using identity-based encryption (IBE) instead of public key encryption. Finally, applying Theorem 1 with $T = N^a$ gives:

Corollary 1. *Assuming public key encryption, there exists a (non-threshold) secure multi-scheme of size $(MN^{2-a}, N^{2-2a}, N^a)$. Assuming IBE, there exists a secure (non-threshold) multi-scheme of size $(1, N^{2-2a}, N^a)$.*

Setting $a = 2/3$ and using IBE from the computational Diffie-Hellman (CDH) assumption in plain groups [DG17] gives a (non-threshold) scheme of size $(1, N^{2/3}, N^{2/3})$ from CDH, the first such scheme with sublinear size.

Proving Theorem 2. Our goal is to design Π such that any decoder D for the scheme — even one with small but noticeable decryption probability — can be converted into a decoder D' that decrypts with high probability, for the original scheme Π_{Thresh}. Importantly, we cannot asymptotically expand the parameters.

To encrypt a message m, our basic idea is to choose random $m_1, \ldots, m_n$ such that $m_1 \oplus m_2 \oplus \cdots \oplus m_n = m$. We encrypt each of the m_i separately using Π_{Thresh}, the final ciphertext for Π being the n encryptions of the m_i. To decrypt, simply decrypt each component to recover m_i, and then reconstruct m.

Since the m_i are an n-out-of-n secret sharing of m, a decoder needs to, in some sense, be able to recover *all* of the m_i in order to compute m. Supposing the "decryptability" of the n individual ciphertexts were independent events, then the decryptability of the individual ciphertexts must very high in order to have noticeable chance at decrypting all n ciphertexts simultaneously.

To turn this intuition into a proof, we show how to extract the m_i whenever the individual ciphertext is decryptable, in order to build a decoder D' for Π_0 with high-enough decryption probability so that it can be traced using Π_0. On input a ciphertext c, D' chooses a random $i \in [n]$ and sets $c_i = c$. It then fills in a ciphertext tuple $(c_1, \ldots, c_n)$ where the $c_j, j \neq i$ are encryptions of random messages m_j. When D gives a guess m' for m, D' can compute a guess m'_i for m_i using m' and the $m_j, j \neq i$. D' decrypts with the same probability as D, and by repeating the process many times on the same ciphertext c, the hope is to amplify the decryption probability.

Unfortunately, there are a few issues. For a fixed ciphertext c, the various trials share a common ciphertext, and therefore their success probabilities are not independent. Also, there is no obvious way to tell which of the trials produced the correct message. Finally, recent traitor tracing definitions [NWZ16, GKRW18, GKW18] actually require tracing to hold in the stronger indistinguishability setting, which means roughly that D does not have to actually produce the message, but only needs to distinguish it from, say, a random message.

We resolve these issues in a couple steps. We use Goldreich-Levin [GL89] to convert an indistinguishability decoder into a predicting decoder. We analyze the decoder's decryption probability on the correlated instances, and show that the success probability over multiple trials amplifies as necessary, when $n = \mathsf{poly}(\lambda)$. Finally, we leverage the indistinguishability security of Π_{Thresh} — meaning D' only needs to distinguish the correct message from random — which allows D' to tell when a trial produces the correct output. Details are given in Section 6.

Putting everything together, if D distinguishes with non-negligible probability, D' will distinguish with probability $1 - o(1)$. Our compiler leaves public and secret keys intact, and blows up the ciphertext by a factor independent of the number of users N, as desired. See Section 6 for additional details.

2.3 Risk Mitigation Compiler

Next, we give a compiler that eliminates risk from risky traitor tracing schemes:

Theorem 3 (Risk Mitigation). *Let $P = P(N, M), K = K(N, M), C = C(N, M)$ be polynomials. Let $\alpha = \alpha(N)$ be a polynomial. If there exists an α-risky multi-scheme Π_{Risky} with size (P, K, C), then there exists a secure (non-risky) multi-scheme Π with size $(\ P(N, M\alpha^{-1})\ ,\ \alpha^{-1} \times K(N, M\alpha^{-1})\ ,\ C(N, M\alpha^{-1})\ )$. If Π_{Risky} is a broadcast and trace scheme, then so is Π.*

Thus, by multiplying M by $O(\alpha^{-1})$ and increasing the secret key size by a factor of $O(\alpha^{-1})$, one can eliminate α-riskiness. In Section 7.2, we extend the risky scheme from [GKRW18] into a $1/N$-risky multi-scheme of size $(1, 1, 1)$. Theorem 3 plus Theorem 1 with $T = N^a$ gives:

Corollary 2. *For any $a \in [0, 1]$, if Assumptions 1 and 2 from [GKRW18] hold, there exists a secure multi-scheme of size $(1, N^{1-a}, N^a)$.*

Note that the computational assumptions are the same as in [GKRW18]. Also, for any $a < 1$, such parameters were not known before from pairings.

We also demonstrate how to add a broadcast functionality to the risky scheme of [GKRW18], at the cost of increasing the public key size and relying on the generic group model for security. Running through our compilers gives:

Corollary 3. *For any $a \in [0, 1]$, there exists a broadcast and trace multi-scheme of size (N^{1-a}, N^{1-a}, N^a) from pairings, with security in the generic group model.*

For $a = 0$, this gives the first broadcast and trace scheme with constant-sized ciphertexts from standard tools, and improves on [GQWW19], which attained N^{ϵ} ciphertext size for any $\epsilon > 0$, while also requiring lattices in addition to pairings[7].

Proving Theorem 3. Let Π_{Risky} be an α-risky multi-scheme. Consider a new protocol Π which runs Π_{Risky} with $T = \omega(\log \lambda)/\alpha$ instances. The secret key for a user consists of the all the secret keys for that user across the T instances. To encrypt, encrypt to a *single* random instance from Π_{Risky}. The overall ciphertext is simply the label of the instance (a number in $[T]$), and a ciphertext from Π_{Risky}. Since each user has a secret key from each instance, each user can decrypt.

Thus, we expand the secret key by a factor of $O(1/\alpha)$, and add $\log T = \log \lambda + \log(1/\alpha) = O(\log \lambda)$ bits to the ciphertext. We can easily extend the above to yield a riskless *multi*-scheme for M instances, by increasing the number of instances of Π_{Risky} to $M \times T$ and grouping them into sets of size T.

Analysis. Suppose a pirate decoder D for Π decrypts with *certainty*. Then it must decrypt, no matter which instance of Π_{Risky} is chosen during encryption. Thus, a perfect decoder for Π actually yields a decoder for each of the T instances of Π_{Risky}. α-riskiness means that each of the T decoders has an α chance of being traced to a traitor, and intuitively the probabilities should be independent. Over all T instances, we expect the tracing probability to be $1 - (1-\alpha)^T = 1 - \mathsf{negl}(\lambda)$.

Toward tracing imperfect decoders, suppose D instead only decrypts for a single instance of Π_0; D has non-negligible decryption probability $1/T$, but will only be traced with probability α. Thus, we cannot trace arbitrary decoders[8]. We will instead aim for a threshold scheme; we can then apply Theorem 2 to get a full tracing scheme.

Even in the threshold setting, however, difficulties arise. The decoder may only decrypt, say, half of the instances, which we will call "good" instances. The good instances are chosen adaptively, *after* the adversary interacts with the many instances of the scheme. This means that the tracing probabilities for the various good instances will not be independent. Nevertheless, we show by a careful argument that, for the right definition of security for a multi-scheme, the tracing probabilities cannot be too correlated, which is sufficient to get our proof to go through. More details are given in Section 7.

2.4 Traitor Tracing from Threshold Broadcast Encryption

We next turn to constructing traitor tracing from a certain type of attribute-based encryption which we call threshold broadcast encryption (this notion of "threshold" not to be confused with the notion of "threshold" for traitor tracing).

[7] The size of the broadcast and secret keys are never explicitly calculated in [GQWW19]. From personal communication with the authors of [GQWW19], we understand that the public key has size $\Omega(N)$ and the secret keys have size $\Omega(N^2)$. Thus, our scheme also improves on the secret key size from their work.

[8] This is similar to the reason behind why Boneh-Naor [BN08] is a threshold scheme.

A (plain) broadcast encryption scheme allows for broadcasting a ciphertext to arbitrary subsets of users with a single constant-sized ciphertext. Broadcast encryption with constant sized secret keys and ciphertexts (but linear-sized public keys) is possible using pairings, as first shown by Boneh, Gentry, and Waters [BGW05].

Describing an arbitrary subset of recipients takes linear space; therefore, broadcast schemes obtain sub-linear ciphertexts by assuming S is *public* and not counted in the ciphertext. On the other hand, traitor tracing typically requires a "private" broadcast, where the recipient set is at least partially hidden. For example, private linear broadcast encryption (PLBE) [BSW06] allows for encrypting to sets $[i]$, and only user i can distinguish between $[i-1]$ and $[i]$.

Our goal is to show how to use broadcast functionalities — with *public* recipient sets — to enable a *private* broadcast structure that allows for tracing.

Our Idea. To trace N users, we will instantiate a broadcast scheme with NT users, for some parameter T. We will think of the NT identities as being pairs $(i, x) \in [N] \times [T]$. For each user $i \in [N]$, we will choose a random $x_i \in [T]$, and give that user the secret key for broadcast identity (i, x_i). Only user i knows x_i. To encrypt, we will simply broadcast to a random subset $S \subseteq [N] \times [T]$.

For tracing, consider choosing S uniformly at random conditioned on $(i, x_i) \notin S$; doing so "turns off" user i, preventing them from decrypting. If i is honest, the adversary does not know x_i and hopefully cannot distinguish between this distribution and a truly uniform S. If turning off a user causes a change in decryption probability, we then accuse that user.

The description so far has several issues. First, in regular execution of the above scheme, any (i, x_i) will only be in the recipient set with probability $1/2$, meaning honest users can only decrypt half the time. Second, an attacker may guess x_i with non-negligible probability $1/T$, and create a decoder that fails if $(i, x_i) \notin S$, thus fooling the tracing algorithm into accusing an honest user with non-negligible probability. Finally, encoding an arbitrary subset S takes NT bits, meaning we have (at least) linear-sized ciphertexts.

Threshold Broadcast. To rectify the first two issues, we will rely on a stronger version of broadcast encryption, which we call *threshold* broadcast encryption[9]. Here, every secret key is associated with a set U; this key can decrypt a ciphertext to set S if and only if $|U \cap S| \geq t$ for some threshold t.

We now give users the secret key for disjoint sets U of identities. The size of $S \cap U$ for a random set S will concentrate around $|U|/2$; by setting t slightly smaller than $|U|/2$, users will be able to decrypt with overwhelming probability. For tracing, the attacker can only guess a small fraction of an honest user's identities. We turn off the identities the attacker *does not* guess, which will drop $|S \cap U|$ below t, thereby turning off the user while keeping the decoder on.

In slightly more detail, we set $T = 2\lambda$. We interpret the $N \times 2\lambda$ identities as triples $(i, j, b) \in [N] \times [\lambda] \times [2]$. For each user, we will choose a random vector

[9] The prior literature on this topic such as [AHL+12] uses the terminology of "threshold attribute based encryption"

$x^i \in \{0,1\}^\lambda$, and give the user the secret key for set $U_i = \{(i,j,x^i_j)\}_{j\in[\lambda]}$. When we trace, for each user i, we will iterate over all $j \in [\lambda]$, trying to turn off identity (i,j,x^i_j) by removing that element from S. If removing that element causes too-large a decrease in the decoder's decryption probability, we keep it in S; otherwise we remove it. We demonstrate that, if the user is outside the adversary's control (meaning in particular the adversary does not know x^i), that with high probability we can remove enough of the elements to completely turn off that user. A diagram illustrating our idea is given in Figure 1.

Interestingly, our tracing algorithm makes *adaptive* queries to the decoder: which elements are in the set S depends on the results of previous queries to the decoder. This is unlike the vast majority of tracing techniques (including both fingerprinting codes and PLBE), where all queries can be made in parallel.

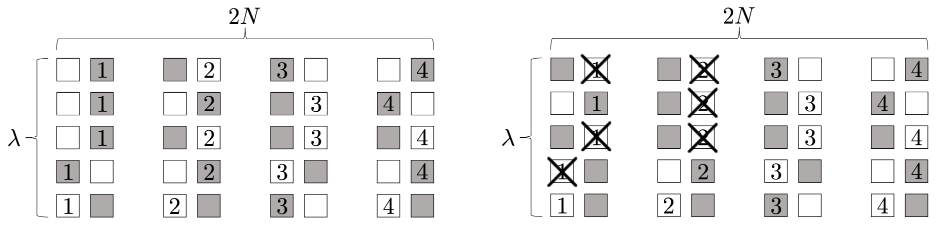

Fig. 1: An illustration in the case $\lambda = 5$, $N = 4$, $t = 2$. Here, the ith pair of columns corresponds to the identities $(i,j,b), j \in [\lambda], b \in \{0,1\}$. U_i is the set of boxes with the number i in them. Gray boxes are those contained in S. Left: Normal usage. In this case, if $t = 2$, all users would be able to decrypt. Right: An example tracing attempt. An "X" represents an element that has been explicitly removed from S. Here, removing $(1,2,1)$ (1st pair of columns, 2nd row) failed, and so $(1,2,1)$ was left in S. Tracing succeeds in fully turning off users 1 and 2.

The Shared Randomness Model. For now, we side-step the need to communicate S by considering a new model for traitor tracing, which we call the *shared randomness model*. Here, every ciphertext is encrypted using a large public source of randomness (in addition to private random coins). This public randomness is also available for decryption, but we will not count it as part of the ciphertext. In this model, we simply have S be derived from the shared randomness.

We update our size notation, to include a fourth term R which bounds the size of the shared randomness; C now only bounds the ciphertext component excluding the shared randomness. For example, a scheme of size $(P,K,C,R) = (N,N,1,N)$ would have linear-sized public and secret keys, constant-sized ciphertexts, and linear-sized shared randomness. We prove the following in Section 8:

Theorem 4 (Informal). *If there exists a secure threshold broadcast scheme of size* (P,K,C), *then there exists a secure broadcast and trace scheme of size* (P,K,C,N) *in the shared randomness model.*

Instantiation. We now turn to constructing a threshold broadcast scheme. Existing pairing-based constructions such as [AHL+12] have size $(N,N,1)$, allowing us to match Corollary 3 with entirely different techniques, but in the weaker shared randomness model. We observe, however, that we do not need a full threshold broadcast scheme. Prior works required security to hold, even if multiple users had overlapping sets U_i. In our case, all users have disjoint U_i. This turns out to let us strip away much of the secret key material, arriving at smaller secret keys.

In slightly more detail, the secret key for a set U consists of terms roughly of the form $g^{\beta \prod_{i\in U}(\gamma - i)^{-1}}$ where β, γ are hidden. The problem with overlapping U is that one can combine different secret keys to generate new keys for other subsets. For example, one can combine $\mathsf{sk}_{12} = g^{\beta(\gamma-1)^{-1}(\gamma-2)^{-1}}$ and $\mathsf{sk}_{13} = g^{\beta(\gamma-1)^{-1}(\gamma-3)^{-1}}$ into $\mathsf{sk}_{23} = \mathsf{sk}_{12}^{-1} \times \mathsf{sk}_{13}^{2} = g^{\beta(\gamma-2)^{-1}(\gamma-3)^{-1}}$ without knowing β, γ, invalidating security. Therefore, existing schemes add additional randomization to the secret key to prevent combinations; each user then needs a personalized version of the public key in order to strip away this extra randomization during decryption. This expands the secret keys to size $O(N)$.

Our main observation is that no such randomization is necessary if the U's are disjoint; we describe our scheme in Section 8. We justify the security of our scheme (for disjoint U) in the generic group model for pairings:

Theorem 5 (Informal). *There exists a threshold broadcast scheme with size* $(N, 1, 1)$ *from pairings with security for disjoint* U *in the generic group model.*

User Expansion in the Shared Randomness Model. Interestingly, in the shared randomness model, user expansion (Theorem 1) increases the ciphertext size, but *not* shared randomness size. Concretely, Theorem 1 becomes:

Theorem 1. *Let* $P = P(N, M), K = K(N, M), C = C(N, M), R = R(N, M)$ *and* $T = T(N, M)$ *be polynomials such that* $T(N, M) \leq N$. *If there exists a secure multi-scheme* Π_0 *with size* (P, K, C, R) *in the shared randomness model, then there exists a secure multi-scheme* Π *with size* $(\ P(N/T, MT)\ ,\ K(N/T, MT)\ ,\ T \times C(N/T, MT)\ ,\ R(N/T, MT)\ )$ *in the shared randomness model. If* Π_0 *is a broadcast and trace scheme, then so is* Π.

Next, note that if $R \leq C$, we can include the shared randomness in the ciphertext, giving a scheme with the same ciphertext size without shared randomness. Combining Theorems 4 and 5, and then applying our updated Theorem 1 gives:

Corollary 4. *For any constant* $a \in [0, 1]$, *there exists a broadcast and trace scheme of size* $(N^{1-a}, 1, N^a, N^{1-a})$ *from pairings in the shared randomness model, whose security is justified in the generic group model. For* $a \in [1/2, 1]$, *the scheme has size* $(N^{1-a}, 1, N^a)$ *in the* plain *model.*

Setting $a = 1/2$ gives the first pairing-based broadcast and trace scheme with size $(N^{1/2}, 1, N^{1/2})$, improving on $(N^{1/2}, N^{1/2}, N^{1/2})$ from [BW06].

2.5 Putting it All Together: Our $\sqrt[3]{N}$ Construction

Finally, we combine all of the ideas above to yield a traitor tracing scheme where all parameters have size $\sqrt[3]{N}$. At a high level, we take our shared randomness scheme of size $(N, 1, 1, N)$ for N users, augment the construction with ideas from [GKRW18] to expand it to N^2 users while hopefully keeping the size $(N, 1, 1, N)$, at the expense of only achieving $1/N$-riskiness. If this worked, scaling down $N^2 \mapsto N$ would give $1/\sqrt{N}$-risky scheme of size $(\sqrt{N}, 1, 1, \sqrt{N})$ for N users. Then we apply Theorem 3 to eliminate the risk, then Theorem 1 with $T = \sqrt[3]{N}$ to balance the number of users, and finally including the shared randomness in the ciphertext, achieving size $\sqrt[3]{N}$ in the plain model.

We follow the above idea, but unfortunately there are some subtle issues with the above approach which make the combination non-trivial. Concretely, when adding riskiness to our shared randomness scheme, we multiply the number of users by N. However, we cannot expand the set of recipients for the threshold broadcast scheme, since doing so would require expanding the public key. Since the recipient set is limited, the sets U_i for the various users will actually need to overlap. As discussed above, overlapping U_i requires expanding the secret key size, preventing us from achieving our goal.

While we are unable to achieve a $1/\sqrt{N}$-risky scheme of size $(\sqrt{N}, 1, 1, \sqrt{N})$, we build a scheme with large but redundant secret keys, so that the secret keys resulting from Theorem 3 can then be compressed by eliminating the redundancy. The result is the following, proved in Section 9:

Theorem 6. *There exists a secure multi-scheme with size $(\sqrt{N}, \sqrt{N}, 1, \sqrt{N})$ in the shared randomness model from pairings with security proved in the generic group model.*

Then, we apply the shared randomness version of Theorem 1 to obtain:

Corollary 5. *There exists a secure multi-scheme with size $\sqrt[3]{N}$ from pairings with security proved in the generic group model.*

3 Discussion, Other Related Work, and Open Problems

3.1 Takeaways

Beyond PLBE and Fingerprinting Codes. PLBE has been the stalwart abstraction in traitor tracing literature for some time, and PLBE and fingerprinting codes make up the vast majority of the fully collusion-resistant tracing literature. Our work demonstrates other useful approaches, and in doing so we hope motivate the further study of alternative approaches to traitor tracing.

Mind your public and secret key sizes. As a result of our work, the threshold limitation of fingerprinting code-based traitor tracing is eliminated. The only remaining limitation is the size of the other parameters. What is important for traitor tracing, therefore, is the *trade-off* between the various parameter sizes, rather than any one parameter on its own.

With this view in mind, perhaps a sub-quadratic scheme from pairings could have been anticipated. After all, the $\sqrt{N}$ scheme of Boneh, Sahai, and Waters [BSW06] has some "slack", in the sense that its secret keys are constant sized. On the other hand, Boneh and Naor [BN08] show that ciphertexts can potentially be compressed by expanding the secret key size. However, prior to our work there was no clear way to actually leverage this slack to get a $\sqrt[3]{N}$ scheme.

$|pk| \times |sk| \times |ct| = N$ *for pairings?:* Our pairing-based traitor tracing schemes, as well as [BSW06], all have size (N^a, N^b, N^c) where $a + b + c = 1$. We conjecture that *any* setting of $a, b, c \geq 0$ such that $a + b + c = 1$ should be possible from pairings. While me make progress towards this conjecture, there are still a number of gaps: for example, is a $(\sqrt{N}, \sqrt{N}, 1)$ scheme possible?

For broadcast and trace, we conjecture that any setting where $a + c \geq 1$ is satisfiable, matching what is known for plain broadcast from pairings. We achieve this in the shared randomness model, and for $c \geq 1/2$ in the plain model.

3.2 Limitations

Generic Groups. Some of our constructions, including our $\sqrt[3]{N}$-size scheme, have security proofs in the generic group model, as opposed to concrete assumptions on pairings. We believe the results are nevertheless meaningful. Our schemes are based on new attribute-based encryption-style primitives, and generic groups have been used in many such cases [BSW07, AY20]. We hope that further work will demonstrate a $\sqrt[3]{N}$ scheme based on concrete assumptions.

Concrete efficiency. While our schemes improve the dependence on N, they may be worse in terms of the dependence on the security parameter. We therefore view our schemes more as a proof-of-concept that improved asymptotics are possible, and leave as an important open question achieving better concrete efficiency. The same can be said of the prior LWE and obfuscation-based constructions, which incur enormous overhead (much worse than ours) due to non-black box techniques and other inefficiencies.

Private tracing. Our schemes all achieve only private traceability, meaning the tracing key must be kept secret. Most schemes from the literature, including the recent LWE schemes, also have private tracing. On the other hand, some schemes have public tracing, allowing the tracing key to be public [BW06, GGH$^+$13, BZ14].

3.3 Other Related Work

$(1, 1, 1)$ *traitor tracing.* Recent developments have given the first traitor tracing schemes where *all* parameters are independent of the number of users. These schemes, however, require tools other than pairings, namely LWE [GKW18, CVW$^+$18] or obfuscation-related objects [GGH$^+$13, BZ14, GVW19].

Embedded identities. Some tracing schemes [NWZ16, KW19, GKW19] allow for information beyond an index to be embedded into an identity and extracted during tracing. It is not obvious how to extend our scheme to handle embedded identities, and we leave this as an open question.

Bounded collusions. In this work, we only consider the unbounded collusion setting, where all users may conspire to build a pirate decoder that defeats tracing. It is also possible to consider bounded collisions, which often result in more efficient schemes [CFN94, BF99, KY02, ADM$^+$07, LPSS14, ABP$^+$17].

4 Traitor Tracing Definitions

In this section, we define traitor tracing, as well as some variants. The central object we will study is actually a slight generalization of traitor tracing, which we call a a "multi-scheme." Here, there are many separate instances of the traitor tracing scheme being run, but the public keys of the different instances are aggregated into a single common public key. Yet, despite this aggregation, the separate instances must behave as essentially independent traitor tracing schemes. Multi-schemes similar to *identity-based* traitor tracing [ADM⁺07], except that identity-based traitor tracing has an exponential number of instances.

In this work, we consider a key encapsulation variant of traitor tracing. A traitor tracing multi-scheme is a tuple $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ of PPT algorithms with the following syntax:

– $\mathsf{Gen}(1^N, 1^M, 1^\lambda)$ takes as input a security parameter, a number of users N, and a number of instances M. It outputs a public key pk, a (secret) tracing key tk, and $N \times M$ user secret keys $\{\mathsf{sk}_{j,i}\}_{i \in [N], j \in [M]}$.
– $\mathsf{Enc}(\mathsf{pk}, j)$ takes as input the public key bk and an instance number j, and outputs a ciphertext c together with a key k.
– $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}_{j,i}, c)$ takes as input the public key pk, the secret key $\mathsf{sk}_{j,i}$ for user i in instance j, and a ciphertext c; it outputs a message k.
– $\mathsf{Trace}^{\mathsf{D}}(\mathsf{tk}, j, \epsilon)$ takes as input the tracing key tk, and instance j, and an advantage ϵ. It then makes queries to a decoder D. Finally, it outputs a set $A \subseteq [N]$. We require that the running time of Trace, when counting queries as unit cost, is $\mathsf{poly}(\lambda, N, M, 1/\epsilon)$.

We require that Dec recovers k: for any polynomials $N = N(\lambda), M = M(\lambda)$, there exists a negligible function negl such that for all $i \in [N], j \in [M], \lambda > 0$:

$$\Pr\left[\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}_{j,i}, c) = k : \substack{(\mathsf{pk}, \mathsf{tk}, (\mathsf{sk}_{j',i'})_{i' \in [N], j' \in [M]}) \leftarrow \mathsf{Gen}(1^N, 1^M, 1^\lambda) \\ (c,k) \leftarrow \mathsf{Enc}(\mathsf{pk}, j)}\right] \geq 1 - \mathsf{negl}(\lambda)$$

For security, we generalize [GKRW18] to the case of multi-schemes. Let $\mathcal{A}$ be an adversary, and ϵ an inverse polynomial. Consider the following experiment:

– $\mathcal{A}$ receives the security parameter λ, written in unary.
– $\mathcal{A}$ sends numbers N, M (in unary) and commits to an instance $j^* \in [M]$. Run $(\mathsf{pk}, \mathsf{tk}, \{\mathsf{sk}_{j,i}\}_{i \in [N], j \in [M]}) \leftarrow \mathsf{Gen}(1^N, 1^M, 1^\lambda)$ and send pk to $\mathcal{A}$.
– $\mathcal{A}$ then makes two kinds of queries, in an arbitrary order.
  - Secret key queries, on pairs $(j, i) \in [M] \times [N]$. In response, it receives $\mathsf{sk}_{j,i}$. For $j \in [M]$, let $C_j \subseteq [N]$ be the set of queries (j, i) of this type.
  - Tracing queries, on pairs (j, D); D is a poly-sized circuit and $j \in [M] \setminus \{j^*\}$. All tracing queries must be on distinct j. Return $A_j \leftarrow \mathsf{Trace}^D(\mathsf{tk}, j, \epsilon)$.
– $\mathcal{A}$ produces a decoder D, and the challenger outputs $A_{j^*} \leftarrow \mathsf{Trace}^D(\mathsf{tk}, j^*, \epsilon)$.

We define the following events. BadTr is the event $A_{j^*} \not\subseteq C_{j^*}$. Let $\mathsf{GoodDec}$ be the event that $\Pr[D(c, k^b) = b] \geq 1/2 + \epsilon(\lambda)$, where $(c, k^0) \leftarrow \mathsf{Enc}(\mathsf{pk}, j^*)$, k^1 is chosen uniformly at random from the key space, and $b \leftarrow \{0,1\}$. In this case, we call D a "good" decoder. Finally, let GoodTr be the event that $|A_{j^*}| > 0$.

Definition 1. *A traitor tracing multi-scheme* Π *is* secure *if, for all PPT adversaries* $\mathcal{A}$ *and all inverse-polynomials* ϵ*, there exists a negligible function* negl *such that* $\Pr[\mathsf{BadTr}] \leq \mathsf{negl}(\lambda)$ *and* $\Pr[\mathsf{GoodTr}] \geq \Pr[\mathsf{GoodDec}] - \mathsf{negl}(\lambda)$.[10]

4.1 Variations, Special Cases, and Extensions

Standard Traitor Tracing. A standard tracing scheme is obtained by setting $M = 1$ in the multi-scheme definition. By a straightforward hybrid argument, a standard traitor tracing scheme also gives a multi-scheme by running independent instances for each $j \in [M]$. The result is that, if there exists a standard tracing scheme of size (P, K, C), then there exists a secure multi-scheme of size $(M \times P, K, C)$.

Threshold Schemes. A threshold scheme [NP98] is one where a malicious user is accused only for very good decoders that succeed a constant fraction of the time.

Definition 2. *A multi-scheme* Π *is* threshold secure *if there exists a constant* $\epsilon \in (0, 1/2)$ *such that, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that* $\Pr[\mathsf{BadTr}] \leq \mathsf{negl}(\lambda)$ *and* $\Pr[\mathsf{GoodTr}] \geq \Pr[\mathsf{GoodDec}] - \mathsf{negl}(\lambda)$.

In the case of threshold secure schemes, the constant ϵ is hard-coded into the algorithm Trace, and we omit ϵ as an input to Trace.

Risky Schemes. In a risky scheme [GKRW18], a traitor is only accused with some small but noticeable probability. Let $\alpha = \alpha(N, M, \lambda)$ be a polynomial.

Definition 3. *A traitor tracing multi-scheme* Π *is* α*-risky if, for all PPT adversaries* $\mathcal{A}$ *and all inverse-polynomials* ϵ*, there exists a negligible function* negl *such that* $\Pr[\mathsf{BadTr}] \leq \mathsf{negl}(\lambda)$ *and* $\Pr[\mathsf{GoodTr}] \geq \alpha \Pr[\mathsf{GoodDec}] - \mathsf{negl}(\lambda)$.

Broadcast and Trace. A broadcast and trace multi-scheme [BW06] is a multi-scheme augmented with a broadcast functionality. $\mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace}$ and the decoder all take as input a subset $S \subseteq [N]$. $\mathcal{A}$ additionally produces a set S at the beginning (when it produces N, M, j^*). $\mathsf{BadTr}, \mathsf{GoodDec}, \mathsf{GoodTr}$ are all defined relative to S, where BadTr happens when $A_{j^*} \nsubseteq S \cap C_{j^*}$. The ciphertext size does *not* include the description of S.

4.2 New Notion: The Shared Randomness Model

We now give a new model for traitor tracing, which we call the *shared randomness model*. In the shared randomness model, encryption has the form $(c = (r, c'), k) \leftarrow \mathsf{Enc}(\mathsf{bk}, j\ ;\ r, s)$. That is, some of the random coins for Enc are *public*, and included in the output of Enc. In this model, we will consider the "ciphertext length" to exclude the public random coins, and just be the length of c'.

[10] The definition given in [GKRW18] additionally introduces an inverse polynomial p, but relaxes ϵ to be *non-negligible*, with a more complicated condition for security. The simpler definition we use is readily shown to be equivalent to their definition.

The shared randomness model captures a setting where the sender and receiver have access to a common source of randomness, for example randomness beacons, stock market fluctuations, etc. The sender can use this randomness as r during encryption, but then does not actually need to send r to the receiver. Thus, communication costs depend only on c', rather than the entire length of (r, c').

For our parameter size notation, we will explicitly consider the size of c' and r separately. That is, for a traitor tracing multi-scheme in the shared randomness model, we say the scheme has parameter size (P, S, C, R) for functions P, K, C, R, where $C \times \mathsf{poly}(\lambda)$ is a bound on the size of c' and $R \times \mathsf{poly}(\lambda)$ is a bound on the size of r. We note that any multi-scheme with parameter size (P, K, C, R) in the shared randomness model is also a scheme with parameter size $(P, K, C + R)$ in the plain model, by having the encrypter choose r and send it as part of the ciphertext. We also note that any plain-model scheme with parameter size (P, K, C) is also a shared-randomness scheme with parameter size $(P, K, C, 0)$.

5 User Expansion Compiler

We now prove Theorem 1, which offers a trade-off between ciphertext size and number of users. For full generality, we give describe our compiler in the shared randomness model. By setting the shared randomness to be empty, our compiler immediately extends to the plain model.

Let $\Pi_0 = (\mathsf{Gen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0, \mathsf{Trace}_0)$ be a traitor tracing multi-scheme in the shared randomness model. We will assume without loss of generality that the encapsulated key has length at most the size of the ciphertext.

Construction 1 (User Expansion Compiler) *Let $T = T(N, M)$ be a polynomial. Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ be the tuple of PPT algorithms:*

- $\mathsf{Gen}(1^N, 1^M, 1^\lambda)$*: Run* $(\mathsf{pk}', \mathsf{tk}', (\mathsf{sk}'_{j',i'})_{i' \in [N'], j' \in [M']}) \leftarrow \mathsf{Gen}_0(1^{N'}, 1^{M'}, 1^\lambda)$ *where $N' = N/T$ and $M' = M \times T$. Set $\mathsf{pk} = \mathsf{pk}', \mathsf{tk} = \mathsf{tk}'$. Interpret $[M']$ as $[M] \times [T]$ and $[N]$ as $[N'] \times [T]$. Then set $\mathsf{sk}_{j,(i,t)} = \mathsf{sk}'_{(j,t),i}$*
- $\mathsf{Enc}(\mathsf{pk}, j, r)$*: Here, r is the shared randomness, which is taken from the same space of shared randomness as in Π_0. For each $t \in [T]$, run $(c_t, k_t) \leftarrow \mathsf{Enc}_0(\mathsf{pk}, (j, t), r)$, again using our interpretation of $[M']$ as $[M] \times [T]$. Choose a random key k from the key space. Output $c = (\ (c_t)_{t \in [T]}\ ,\ (k_t \oplus k)_{t \in [T]}\ )$ as the ciphertext and k as the key.*
- $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}_{j,i}, c, r)$*: Write i as (i', t) and $c = (\ (c_t)_{t \in [T]}\ ,\ (u_t)_{t \in [T]}\ )$. Compute $k'_t \leftarrow \mathsf{Dec}_0(\mathsf{pk}, \mathsf{sk}_{j,i}, c_t, r)$. Output $k' = u_t \oplus k'_t$.*
- $\mathsf{Trace}^D(\mathsf{tk}, j, \epsilon)$*: For each $t \in [T]$ run $A_t \leftarrow {\mathsf{Trace}_0}^{D_t}(\mathsf{tk}, (j, t), \epsilon/4T)$, and output $A = \cup_{t \in [T]} \{(i, t) : i \in A_t\}$. Here, D_t be the following decoder for instance (j, t) of Π_0: on input $(c, r), u$, do the following:*
  - *For $t' \neq t$, compute $(c_{t'}, k_{t'}) \leftarrow \mathsf{Enc}_0(\mathsf{pk}, (j, t'), r)$. Set $c_t = c$.*
  - *Choose a random bit $b \leftarrow \{0, 1\}$, and random keys k^0, k^1.*
  - *For $t' < t$, choose random $u_{t'}$. For $t' > t$, $u_{t'} = k_{t'} \oplus k^0$. Set $u_t = u \oplus k^0$.*
  - *Set $c' = (\ (c_{t'})_{t' \in [T]}\ ,\ (u_{t'})_{t' \in [T]}\ )$. Output $b \oplus D((c', r), k^b)$. (Note that XORing with b turns a distinguisher into a predictor)*

By the correctness of Π_0, we will have that $k'_t = k_t$, and therefore $k' = u_t \oplus k'_t = u_t \oplus k_t = k$, so Π is correct. Since the encapsulated key in Π_0 is at most the size of the ciphertext, we see that the desired sizes hold.

5.1 Security of Our Compiler

Theorem 7. *If Π_0 is a secure multi-scheme, then so is Π.*

Proof. Due to lack of space, we only sketch the proof, see the full version [Zha20] for a complete proof. Fix an adversary $\mathcal{A}$ for Π and inverse-polynomial ϵ. Let $\mathsf{GoodTr}, \mathsf{BadTr}, \mathsf{GoodDec}$ be the events as in Definition 1.

$\Pr[\mathsf{BadTr}] \leq \mathsf{negl}$ follows by a straightforward argument, using the fact that Trace_0 only accuses honest users with negligible probability. We now sketch why $\Pr[\mathsf{GoodTr}] \geq \Pr[\mathsf{GoodDec}] - \mathsf{negl}$. Our goal is to show that at least one of the decoders D_t will be traced. We set up a sequence of hybrid distributions by gradually replacing the u_t by independent random strings. Before any changes, a good decoder is correct with probability at least $1/2 + \epsilon$; after all the changes, the view of the decoder is statistically independent of b, and therefore it is correct with probability exactly $1/2$. Therefore, there is some t where changing u_t to random causes the decoder's success probability to drop by at least ϵ/T. This corresponds to the decoder D_t being a "good" decoder; by the security of Π_0, tracing this D_t will result in A_t being non-empty, as desired. □

6 Threshold Elimination Compiler

We now prove Theorem 2, generically removing thresholds from tracing schemes. For simplicity, we give our compiler for plain-model traitor tracing. Let $\Pi_{\mathsf{Thresh}} = (\mathsf{Gen}_{\mathsf{Thresh}}, \mathsf{Enc}_{\mathsf{Thresh}}, \mathsf{Dec}_{\mathsf{Thresh}}, \mathsf{Trace}_{\mathsf{Thresh}})$ be a multi-scheme.

Construction 2 (Threshold Elimination Compiler) *Assume the encapsulated key space of Π_{Thresh} is $\mathcal{K} = \{0,1\}^\ell$. Let $t = t(\lambda)$ be any polynomial. Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ be the tuple of the following PPT algorithms:*

- $\mathsf{Gen}(1^N, 1^M, 1^\lambda) = \mathsf{Gen}_{\mathsf{Thresh}}(1^N, 1^M, 1^\lambda)$
- $\mathsf{Enc}(\mathsf{pk}, j)$*: Let $n = \omega(\log \lambda)$. For each $u \in [n], v \in [t]$, run $(c_{u,v}, k_{u,v}) \leftarrow \mathsf{Enc}_{\mathsf{Thresh}}(\mathsf{pk}, j)$. Choose a random $s \leftarrow \mathcal{K}$. For each $v \in [t]$, let $k_v = k_{1,v} \oplus \cdots \oplus k_{n,v}$ and let $b_v = s \cdot k_v \bmod 2$ be the bit-wise inner product of s and k_v. Let $k = k_1 k_2 \cdot k_t$. Let $c = (s,\ (c_{u,v})_{u \in [n], v \in [t]}\ )$. Output (c, k).*
- $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}_{j,i}, c)$*: Write $c = (s,\ (c_{u,v})_{u \in [n], v \in [t]}\ )$. For each $u \in [n], v \in [t]$, run $k'_{u,v} \leftarrow \mathsf{Dec}_{\mathsf{Thresh}}(\mathsf{pk}, \mathsf{sk}_i, c_{u,v})$. For each $v \in [t]$, compute $k'_v = k'_{1,v} \oplus \cdots \oplus k'_{n,v}$ and $b'_v = r \cdots k'_v \bmod 2$. Output $k' = b'_1 b'_2 \cdots b'_t$.*
- *The algorithm $\mathsf{Trace}^D(\mathsf{tk}, \epsilon)$ will be described below.*

By the correctness of Π_{Thresh}, we have with overwhelming probability that $k'_{u,v} = k_{u,v}$ for all $u \in [n], v \in [t]$. This implies $k'_v = k_v$ and hence $b'_v = b_v$ for all $v \in [t]$, meaning $k' = k$. Thus Π is correct. We also see that Π has the desired parameter size: only the ciphertext is increased by a factor of $n \times t \leq \mathsf{poly}(\lambda)$. We now give our algorithm $\mathsf{Trace}^D(\mathsf{tk}, j, \epsilon)$, which proceeds in several stages:

Target Single Bit: First, we define a decoder $D_1(s, (c_u)_{u\in[n]})$, where $s \in \{0,1\}^\ell$, c_u are ciphertexts from Π_{Thresh}. The goal of D_1 is to predict the bit $s \cdot k$ where k is the XOR of all the keys encapsulated in the c_u. It does so by embedding its challenge into a random position of an input for D:

- Choose a random $v \in [t]$.
- Let $c_{u,v} = c_u$ and choose $(c_{u,v'}, k_{u,v'}) \leftarrow \mathsf{Enc}_{\mathsf{Thresh}}(\mathsf{pk}, j)$ for $u \in [n]$ and $v' \in [t] \setminus \{v\}$. Let $c = (s,\ (c_{u,v})_{u\in[n],v\in[t]}\ )$.
- For each $v' \in [t] \setminus \{v\}$, compute $k_{v'} = k_{1,v'} \oplus \cdots \oplus k_{n,v'}$. For $v' \leq v$, choose random $b_{v'} \leftarrow \{0,1\}$, and for $v' > v$, set $b_{v'} = r \cdot k'_{u,v'} \bmod 2$. Set $k = b_1 \cdots b_t$.
- Output $b_v \oplus D(c,k)$ (XORing with b_v turns a distinguisher into a predictor)

Apply Goldreich-Levin. Next, we will need the following theorem:

Theorem 8 ([GL89]). *There exists a constant Γ and oracle algorithm $\mathsf{GL}^D(\ell, \epsilon')$ running in time $\mathsf{poly}(\ell, \log(1/\epsilon'))$ and making $\mathsf{poly}(\ell, \log(1/\epsilon'))$ queries to D, such that the following holds. If there exists an $x \in \{0,1\}^\ell$ such that $\Pr[D(r) = x \cdot r \bmod 2 : r \leftarrow \{0,1\}^\ell] \geq 1/2 + \epsilon'$, then $\Pr[\mathsf{GL}^D(\ell, \epsilon') = x] \geq \Gamma \times (\epsilon')^2$.*

Trace will define $D_2(\ (c_u)_{u\in[n]}\ ) := \mathsf{GL}^{D_1(\cdot,(c_u)_{u\in[n]})}(\ell, \epsilon' = \epsilon/t)$; D_2 is given $(c_u)_{u\in[n]}$ that encrypt $k_1, \ldots, k_n$, and its goal is to compute $k_1 \oplus \cdot \oplus k_n$.

Generate List of Potential Decryptions: Let $D_3(c, k)$ be the following, where c is a ciphertext for Π_{Thresh} and $k \in \{0,1\}^\ell$. For $z = 1, \ldots, \xi = (2nt^3/\Gamma\epsilon^3) \times \omega(\log \lambda)$:

- Choose a random $u \in [n]$, and set $c_u = c$.
- Then for each $u' \in [n] \setminus \{u\}$, run $(c_{u'}, k_{u'}) \leftarrow \mathsf{Enc}_{\mathsf{Thresh}}(\mathsf{bk}, j)$.
- Run $k' \leftarrow D'_{v,b}(\ (c_u)_{u\in[n]})\ )$, and set $k^{(z)} = k' \oplus k_1 \oplus \cdots \oplus k_{u-1} \oplus k_{u+1} \cdots \oplus k_n$.

Next, if $k = k^{(z)}$ for any $z \in [\xi]$, output 1. Otherwise, output 0.

Trace. Finally, run and output $A \leftarrow {\mathsf{Trace}_{\mathsf{Thresh}}}^{D_3}(\mathsf{tk}, j)$

6.1 Security of Our Compiler

Theorem 9. *Set $n = \omega(\log \lambda), \epsilon' = \epsilon/t, \xi = (2nt^3/\Gamma\epsilon^3) \times \omega(\log \lambda)$. Suppose $\ell = \omega(\log \lambda)$. If Π_{Thresh} is a secure threshold multi-cheme, then Π is a secure (non-threshold) multi-scheme.*

Proof. Due to lack of space, we only sketch the proof, see the full version [Zha20] for a complete proof. Fix an adversary $\mathcal{A}$ for Π and inverse-polynomial ϵ. Let $\mathsf{GoodTr}, \mathsf{BadTr}, \mathsf{GoodDec}$ be the events as in Definition 1. That $\Pr[\mathsf{BadTr}]$ is negligible follows readily from an analogous argument to the proof of Theorem 7.

To show that $\Pr[\mathsf{GoodTr}] \geq \Pr[\mathsf{GoodDec}] - \mathsf{negl}$, we assume GoodTr happens (D guesses b with probability $\geq 1/2 + \epsilon$) and analyze the decoders D_1, D_2, D_3. If D is such a decoder, we can perform an analogous hybrid step as in Theorem 7; for a *randomly* selected position, we obtain that D can distinguish the bit of the key in that position from a random bit. Then, by a routine calculation, we can convert D into a *predictor* for said bit; the result is exactly the predictor D_1.

Claim. If $\mathsf{GoodDec}$ happens, then $\Pr[D_1(s,(c_u)_{u\in[n]}) = s\cdot k \bmod 2] \geq 1/2+2\epsilon/t$, where $s \leftarrow \{0,1\}^\ell$, $(c_u,k_u) \leftarrow \mathsf{Enc}_{\mathsf{Thresh}}(\mathsf{bk},j)$ for $u\in[n]$, and $k = k_1\oplus\cdots\oplus k_n$.

This claim is proved in the full version [Zha20]. Next, the following claim shows that D_2 actually guesses k, which follows from Goldreich-Levin (Theorem 8):

Claim. If $\mathsf{GoodDec}$ happens, then $\Pr[D_2((d_u,r_u)_{u\in[n]}) = k] \geq \Gamma\times(\epsilon')^3$, where $(c_u,k_u)\leftarrow \mathsf{Enc}_{\mathsf{Thresh}}(\mathsf{bk},j,r_u)$, r_u is uniformly random, and $k=k_1\oplus\cdots\oplus k_n$.

Next, we need to show that D_3 can decrypt with high probability. Let $\gamma>0$. Very roughly, we define S_γ to be the set of "good" ciphertexts, defined as: if we choose a random $u\in[n]$, a random c_u from S_γ, and choose the remaining ciphertexts from $\mathsf{Enc}_{\mathsf{Thresh}}(\mathsf{pk},j)$, then D_2 outputs the correct key with probability at least γ.

Claim. Let η be the fraction of $c\in S_\gamma$. Then $\Gamma\times(\epsilon')^3 \leq \eta^n + n(1-\eta)\gamma$.

The claim is proved in the full version [Zha20]; the intuition is that either (1) all n of the ciphertexts were in S_γ, or (2) at least one of the ciphertexts is not in S_γ. Case (1) happens with probability γ^n. For case (2), there are n possible positions for the "bad" ciphertext; for each position, the probability of being bad is $(1-\eta)$, and conditioned on being bad, the decryption probability is at most γ.

We choose $\gamma = \Gamma\times(\epsilon')^3/2n$, giving $\eta^n \geq \Gamma\times(\epsilon')^3/2$. Taking the nth root of both sides and using $n=\omega(\log\lambda)$ gives $\eta\geq 1-o(1)$, meaning most ciphertexts are good. We then set the number of trials D_3 runs to be high enough so that, on a good ciphertext, with overwhelming probability at least one of the trials will be correct. Thus, if D_3 is given the correct key, it will find the key amongst its trials with probably $1-o(1)$. If D_3 is given a random key as input, it will almost certainly not find the given key among its trials. Thus, D_3 is a good decoder for Π_{Thresh}. By the security of Π_{Thresh}, a user will be accused, as desired. □

7 Risk Mitigation Compiler

We now prove Theorem 3 by giving our risk mitigation compiler, converting any risky scheme into one that is not. For notational simplicity, we give our compiler for plain-model traitor tracing; our compiler is readily adapted to work in the shared randomness model as well. Let Π_{Risky} be an $\alpha(N)$-risky multi-scheme. For full generality, we will only assume that Π_{Risky} is a *threshold* scheme.

Construction 3 (Risk Mitigation Compiler) *Let Π_{Thresh} be a tuple of PPT algorithms $(\mathsf{Gen}_{\mathsf{Thresh}},\mathsf{Enc}_{\mathsf{Thresh}},\mathsf{Dec}_{\mathsf{Thresh}},\mathsf{Trace}_{\mathsf{Thresh}})$ where:*

- $\mathsf{Gen}_{\mathsf{Thresh}}(1^N,1^M,1^\lambda)$*: set $T=(1/\alpha)\times\omega(\log\lambda)$, $M'=M\times T$. Interpret $[M']$ as $[M]\times[T]$. Run $(\mathsf{pk},\mathsf{tk},\{\mathsf{sk}_{(j,t),i}\}_{i\in[N],j\in[M],t\in[T]}) \leftarrow \mathsf{Gen}_{\mathsf{Risky}}(1^N,1^{M'},1^\lambda)$. Output $(\mathsf{pk},\mathsf{tk},(\mathsf{sk}_{j,i})_{i\in[N],j\in[M]})$, where $\mathsf{sk}_{j,i}=(\mathsf{sk}_{(j,t),i})_{t\in[T]}$.*
- $\mathsf{Enc}_{\mathsf{Thresh}}(\mathsf{pk},j)$*: Run $(c,k)\leftarrow\mathsf{Enc}_{\mathsf{Risky}}(\mathsf{pk},(j,t),r)$ for a random choice of $t\in[T]$. Output the ciphertext (t,c) and encapsulated key k.*
- $\mathsf{Dec}_{\mathsf{Thresh}}(\mathsf{pk},\mathsf{sk}_{j,i},(j,c),r)$*: Run and output $k'\leftarrow\mathsf{Dec}_{\mathsf{Risky}}(\mathsf{pk},\mathsf{sk}_{(j,t),i},c)$.*

– $\mathsf{Trace_{Thresh}}^D(\mathsf{tk}, j)$: *Let D_t be the decoder $D_t(c, k) = D((t, c), k)$. For $t \in [T]$, run $A_t \leftarrow \mathsf{Trace_{Risky}}^{D_t}(\mathsf{tk}, (j, t))$. Output $A = \cup_t A_t$.*

Correctness follows readily from the correctness of Π_{Risky}. We also see that the desired parameter sizes hold.

7.1 Security of Our Compiler

Theorem 10. *Assume $T = (1/\alpha) \times \omega(\log \lambda)$. If Π_{Risky} is an α-risky threshold multi-scheme, then Π_{Thresh} is a secure* threshold *tracing scheme.*

Note that Theorem 10 only gives a threshold scheme; applying Theorem 2 then gives a non-threshold scheme of the same parameters, thus proving Theorem 3.

Proof. We say that $t \in [T]$ is "good" if D_t has a high chance of decrypting ciphertexts for instance (j, t) of Π_{Risky}. D can only decrypt ciphertexts for t where D_t is good; thus $\mathsf{GoodDec_{Thresh}}$ implies that the fraction of good t is large. Since each t represents a different instance of the risky scheme, each of the decoders D_t should have a $1/\alpha$ chance of being traced to some user. As long as the number of good t is larger than $\omega(\log \lambda)/\alpha$, then we would expect that, with overwhelming probability, at least one of the D_t traces. One challenge is that the attacker can choose adaptively which of the t will good and hence traceable, so the tracing probabilities are not independent events. Nonetheless, we show a careful security proof — and also show that $\Pr[\mathsf{BadTr_{Thresh}}]$ is negligible — in the full version [Zha20] which demonstrates that the intuition indeed holds. $\square$

7.2 Instantiation

Our goal now is to prove the following, which suffices to prove Corollary 2:

Theorem 11. *If Assumptions 1 and 2 of [GKRW18] on pairings hold, there exists a $1/N$-risky multi-scheme of size* $(1, 1, 1)$.

Due to lack of space, we only sketch the proof; see the full version [Zha20] for details. As a starting point, [GKRW18] build a $1/N$-risky traitor tracing tracing scheme of size $(1, 1, 1)$, based on pairing assumptions that they call Assumption 1 and 2. Their scheme is not a multi-scheme, but trivially gives a multi-scheme of size $(M, 1, 1)$ by running M instances in parallel. We show how to tweak the construction to obtain a $1/N$-risky *multi*-scheme of size $(1, 1, 1)$.

In more detail, [GKRW18] build a primitive called mixed Bit Matching Encryption (MBME). Here, ciphertexts and secret keys are associated to attribute vectors in $\{0, 1\}^n$. A secret key with attribute $\mathbf{x}$ can decrypt a ciphertext with attribute $\mathbf{y}$ if and only if $\mathbf{x} \cdot \mathbf{y} = 0$ (the inner product taken over the *integers*). For security, a message encrypted to attribute $\mathbf{x}$ stays hidden to all secret keys $\mathbf{y}$ that satisfy $\mathbf{x} \cdot \mathbf{y} > 0$. Moreover, given a secret key $\mathbf{x}$ and ciphertext $\mathbf{y}$, the attacker learns whether or not $\mathbf{x} \cdot \mathbf{y} = 0$, but learns nothing else about $\mathbf{x}, \mathbf{y}$.

[GKRW18] instantiate their scheme with $n = 2$. A random index $i^* \in [N]$ is chosen. Users $i < i^*, i = i^*$, and $i > i^*$ are given a secret key with attributes $(0,0),(1,0)$, and $(1,1)$, respectively. A normal ciphertext is encrypted with attribute $\mathbf{y}=(0,0)$ so that all users can decrypt. To trace a decoder D, D is tested on ciphertexts to attributes $(0,1)$, and $(1,1)$ to see whether it can distinguish. If so, accuse user i^*; otherwise accuse no one. MBME security implies that only user i^* can distinguish between encryptions to $(0,1)$ and $(1,1)$, so we only accuse i^* if they are indeed a traitor. A careful hybrid argument then shows that i^* is indeed accused with probability negligibly-close to $1/N$.

Adding Identities. Our idea is to add "identities" to get a multi-scheme, where each instance of the multi-scheme is an "independent" copy of the above. We set $n=2\kappa+2$, where κ is the bit-length of integers in $[M]$. Each $j\in[M]$ will give rise to a separate instance, which we distinguish using the first 2κ positions. The remaining 2 positions will be used as above to construct the risky scheme.

In slightly more detail, we group the first 2κ bit positions into κ pairs. For each $j\in[M]$, write j as a bit-vector $\mathbf{v}\in\{0,1\}^\kappa$. Let $\mathbf{v}^0\in\{0,1\}^{2\kappa}$ be vector where the tth pair of positions is $(1-v_t,v_t)$. Let $\mathbf{v}^1\in\{0,1\}^{2\kappa}$ be the opposite, setting the tth pair to $(v_t,1-v_t)$. Notice that $\mathbf{v}^0\cdot\mathbf{v}^1=0$, while for $\mathbf{v}\neq\mathbf{w}$, $\mathbf{v}^0\cdot\mathbf{w}^1>0$. For each $j\in[M]$, we set up the risky scheme as above, choosing a random i_j^*. To encrypt to the jth instance, we set the first 2κ positions of $\mathbf{y}$ to be $\mathbf{v}^0$, and for a secret key, we set the first 2κ positions to be $\mathbf{v}^1$. This ensures that only secret keys for the jth scheme can decrypt ciphertexts for the jth scheme, thus fully separating the j instances. We then set the last two positions analogous to the sketch above. By using essentially the same analysis as in [GKRW18], we can show that each separate scheme is $1/N$-risky, regardless of what secret keys and ciphertexts the adversary possesses for the various other schemes. This suffices to establish the $1/N$-riskiness of the entire multi-scheme. Theorem 11 follows from the fact that n is bounded by $O(\log M)$, which can be absorbed into $\mathsf{poly}(\lambda)$ terms. Then, applying Theorem 3 and then Theorem 1 gives Corollary 2.

7.3 A Broadcast and Trace Scheme

Our next result is the following, which suffices to prove Corollary 3:

Theorem 12. *There exists a $1/N$-risky broadcast and trace multi-scheme of size $(N,1,1)$ from pairings, with security proved in the generic group model.*

We sketch the construction; see the full version [Zha20] for additional details. The high-level idea is to add a mixed Bit Matching Encryption (MBME) functionality on top of a broadcast scheme of size $(N,1,1)$ (in particular, we use [Del07]), and then use the MBME functionality to create a $1/N$-risky tracing scheme.

The Delerablée Broadcast Scheme. We briefly recall Delerablée's scheme [Del07]. Let $\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}'$ be groups of prime order p with pairing operation $e:\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}'$. Let g_1,g_2 be generators of $\mathbb{G}_1,\mathbb{G}_2$, respectively. The set of possible user identities is $\mathbb{Z}_p\setminus\{0\}$. The public and secret keys are

$$\mathsf{pk}=(e(g_1,g_2)^\beta,g_1^{\beta\gamma},(g_2^{\gamma^i})_{j\in[0,N]})\qquad \mathsf{sk}_i=g_1^{\beta/(1-\gamma/i)}$$

for random secrets $\beta, \gamma \in \mathbb{Z}_p$. The public key allows for computing $J(P) := g_2^{P(\gamma)}$, for any polynomial P of degree at $\leq N$. The ciphertext to a set S is:

$$c_1 = g_1^{\alpha\beta\gamma} \quad , \quad c_2 = J\left(\prod_{i \in S}(1-\gamma/i)\right)^{\alpha} = g_2^{\alpha \prod_{i \in S}(1-\gamma/i)}$$

where $\alpha \in \mathbb{Z}_p$ is random. The encapsulated key is $k = e(g_1, g_2)^{\alpha\beta}$. Notice that any user in $\mathbb{Z}_p \setminus \{0\}$ (which has exponential size) can be a recipient, as long as the number of recipients is at most N.

To decrypt, let $Q(\gamma) = \prod_{j \in S \setminus \{i\}}(1-\gamma/j)$ and $P(\gamma) = (1-Q(\gamma))/\gamma$. Notice that $1 - Q(0) = 0$, meaning $1 - Q(\gamma)$ is a polynomial of degree $\leq N-1$ with a 0 constant term; thus $P(\gamma)$ is also a polynomial. Therefore, compute

$$e(c_1, J(P)) \cdot e(\mathsf{sk}_i, c_2) = e(g_1, g_2)^{\alpha\beta\gamma P(\gamma) + \alpha\beta Q(\gamma)} = e(g_1, g_2)^{\alpha\beta} = k$$

The intuition for security is that, for any $i \notin S$, pairing sk_i with c_2 will leave a pole in γ, which cannot be canceled; thus users outside of S cannot decrypt.

Our Construction. We now briefly explain how to augment Delerablée's scheme with a mixed Bit Matching functionality, in order to create a risky scheme. For a (row) vector $\mathbf{v} \in \mathbb{Z}_p^n$, we use the notation $g^{\mathbf{v}} = (g^{v_1}, \dots, g^{v_n})$.

Let $[\![\cdot]\!]$ be an arbitrary efficient injection from $[M] \times [N]$ into $\mathbb{Z}_p \setminus \{0\}$, which we use to embed instance/identity pairs into $\mathbb{Z}_p \setminus \{0\}$. We choose a random $R \in \mathbb{Z}_q^{4\times 4}$ in addition to α, β. Our public key is:

$$\mathsf{pk} = \left(e(g_1, g_2)^{\beta}, g_1^{(\beta\gamma\ ,\ 0\ ,\ 0\ ,\ 0)\cdot R^{-1}}, \left(g_2^{(\gamma^i\ ,\ 0\ ,\ 0\ ,\ 0)\cdot R^T} \right)_{j \in [0,N]} \right)$$

The secret key for user i in instance j, with attribute (x_0, x_1), is computed as

$$\mathsf{sk}_{j,i} = g_1^{(\beta(1-\gamma/[\![j,i]\!])^{-1}\ ,\ x_1 u_1\ ,\ x_2 u_2\ ,\ u_3)\cdot R^{-1}}$$

Here, u_1, u_2, u_3 are freshly chosen at random in $\mathbb{Z}_p$ for each secret key. A ciphertext to set S in instance j, with attribute (y_0, y_1) is set to

$$c_1 = g_1^{(\alpha\beta\gamma\ ,\ 0\ ,\ 0\ ,\ 0)\cdot R^{-1}} \quad , \quad c_2 = g_2^{(\alpha \prod_{i \in S}(1-\gamma/[\![j,i]\!])\ ,\ y_1 v_1\ ,\ y_2 v_2\ ,\ 0)\cdot R^T}$$

where α, v_1, v_2 are freshly chosen at random in $\mathbb{Z}_p$ for each ciphertext. The encapsulated key is $e(g_1, g_2)^{\alpha\beta}$. Let $J(P) := g_2^{(P(\gamma)\ ,\ 0\ ,\ 0\ ,\ 0)\cdot R^T}$, which can be computed from pk for any polynomial P of degree at most N. Notice if $(y_1, y_2) = (0, 0)$, then the ciphertext can thus be computed from pk. To decrypt, output $e(c_1, J(P)) \cdot e(\mathsf{sk}_{j,i}, c_2)$ as in Delerablée, where we use the notation $e(\mathbf{g}, \mathbf{h}) = \prod_i e(g_i, h_i)$, so that $e(g_1^{\mathbf{v}}, g_2^{\mathbf{w}}) = e(g_1, g_2)^{\mathbf{v}\cdot\mathbf{w}^T}$. Correctness follows by essentially the same calculation as in Delerablée, but working with vectors of group elements.

Notice that the construction has the desired size parameters, and readily gives a tracing scheme analogous to Section 7.2. We prove security in the full version [Zha20]. The rough intuition is that in the generic group model, we can prove that the MBME functionality combines correctly with the broadcast functionality, which allows us to trace as in [GKRW18].

8 Traitor Tracing from Threshold Broadcast

Here, we prove Corollary 4 by formalizing Theorems 4 and 5, showing how to construct traitor tracing from threshold broadcast, and then giving a new instantiation of threshold broadcast from pairings. A threshold broadcast scheme is a tuple $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Extract}, \mathsf{Dec})$ of PPT algorithms where:

- $\mathsf{Gen}(1^u, 1^v, 1^t, 1^\lambda)$ takes as input a security parameter, bounds $u, v \leq 2^\lambda$, and a threshold $t \leq u, v$. It outputs a public key pk and a master secret key msk.
- $\mathsf{Enc}(\mathsf{pk}, S)$ takes as input the public key pk and a set of users $S \subseteq [2^\lambda]$ of size at most v. It outputs a ciphertext c and key k.
- $\mathsf{Extract}(\mathsf{msk}, U)$ takes as input the master secret key and a subset $U \subseteq [2^\lambda]$ of size at most u. It outputs a secret key sk_U.
- $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}_U, S, c)$ takes as input the public key pk, the secret key sk_U for set U, and a ciphertext c; it outputs a key k.

For correctness, we require that Dec correctly recovers k, provided $|U \cap S| \geq t$: for any polynomials $v = v(\lambda), u = u(\lambda)$, there exists a negligible function negl such that for all $t \leq u, v$ and all $S, U \subseteq [2^\lambda]$ where $|U| \leq u, |S| \leq v$ and $|U \cap S| \geq t$:

$$\Pr\left[\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}_U, c) = k : {}^{(\mathsf{pk},\mathsf{msk}) \leftarrow \mathsf{Gen}(1^u,1^v,1^t,1^\lambda)}_{\mathsf{sk}_U \leftarrow \mathsf{Extract}(\mathsf{msk},U),(c,k)\leftarrow \mathsf{Enc}(\mathsf{pk},S)}\right] \geq 1 - \mathsf{negl}(\lambda)$$

We also use the same size notation as for traitor tracing schemes, except that the size parameters depend on u, v instead of M, N. For security, let $\mathcal{A}$ be an adversary, and consider the following experiment:

- $\mathcal{A}$ receives the security parameter λ, written in unary.
- $\mathcal{A}$ chooses numbers u, v, t, written in unary. It also chooses a set $S \subseteq [2^\lambda], |S| \leq v$, and a number of *disjoint* sets $U_i \subseteq [2^\lambda], |U_i| \leq u$ such that $|U_i \cap S| < t$. Send $u, v, t, (U_i)_{i \in [N]}, S$ to the challenger.
- The challenger runs $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(1^u, 1^v, 1^t, 1^\lambda)$, and for each i runs $\mathsf{sk}_{U_i} \leftarrow \mathsf{Extract}(\mathsf{msk}, U_i)$. It chooses a random bit b, samples random k^1, runs $(c, k^0) \leftarrow \mathsf{Enc}(\mathsf{pk}, S)$, and sends the adversary $(\ (\mathsf{sk}_{U_i})_{i \in [N]}\ ,\ c\ ,\ k^b\ )$.
- Finally, the adversary produces a guess b' for b.

Definition 4. *A threshold broadcast scheme is secure if, for all PPT adversaries $\mathcal{A}$, there exists a negligible* negl *such that* $\Pr[b' = b] \leq 1/2 + \mathsf{negl}(\lambda)$.

8.1 From Threshold Broadcast To Traitor Tracing

Here, we formalize and prove Theorem 4:

Theorem 4 (Formal Version). *Suppose there exists a secure threshold broadcast scheme which, for $u = \lambda$ has size $(P = P(v), S = S(v), C = C(v))$. Then there exists a secure broadcast and trace multi-scheme in the shared randomness model with size $(P(N), S(N), C(N), N)$.*

To prove the theorem, let $\Pi_0 = (\mathsf{Gen}_0, \mathsf{Enc}_0, \mathsf{Extract}_0, \mathsf{Dec}_0)$ be a threshold broadcast scheme satisfying the given size requirement.

Construction 4 *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ *be the tuple of the following PPT algorithms:*

- $\mathsf{Gen}(1^N, 1^M, 1^\lambda)$*: run* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Gen}_0(1^u, 1^v, 1^t, 1^\lambda)$*, where* $u = \omega(\log \lambda)$*,* $v = Nu$*, and* $t = (2/5)u$*. Let* $[\![\cdot]\!]$ *be an arbitrary efficient injection from* $[M] \times [N] \times [u] \times \{0,1\}$ *into the identity space* $[2^\lambda]$*. For each* $i \in [N], j \in [M]$*, choose a random* $x_{j,i} \in \{0,1\}^u$*. Set* $U_{j,i} = \{[\![j, i, \ell, x_{i,j,\ell}]\!]\}_{\ell \in [u]} \subseteq [2^\lambda]$ *and run* $\mathsf{sk}_{j,i} \leftarrow \mathsf{Extract}_0(\mathsf{msk}, U_{j,i})$*. Output* pk *as the public key,* $\mathsf{tk} = (x_{j,i})_{i \in [N], j \in [M]}$ *as the tracing key, and* $(\mathsf{sk}_{j,i})_{i \in [N], j \in [M]}$ *as the secret keys.*
- $\mathsf{Enc}(\mathsf{pk}, j, S, r)$*: here,* $r \in \{0,1\}^{N \times u}$ *is the public randomness, which will be interpreted as the list* $r = (r_{i,\ell})_{i \in [N], \ell \in [u]}$*,* $r_{i\ell} \in \{0,1\}$*. Let* $T_{j,S,r} = \{[\![j, i, \ell, r_{i,\ell}]\!]\}_{i \in S, \ell \in [u]}$*. Run and output* $(c, k) \leftarrow \mathsf{Enc}_0(\mathsf{pk}, T_{j,S,r})$*.*
- $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}_{j,i}, S, r, c)$*: Output* $k' \leftarrow \mathsf{Dec}_0(\mathsf{pk}, \mathsf{sk}_{j,i}, T_{j,S,r}, c)$ *for* $T_{j,S,r}$ *as above.*
- $\mathsf{Trace}^D(\mathsf{tk}, j, \epsilon)$ *will be described below.*

Notice that $|U_{j,i}| = u$ and $|T_{j,r}| = Nu$ to that $|T_{j,S,r}| \leq Nu = v$. Also, notice that if we set $u \leq \lambda$, Π will have the desired size parameter, since the factor of $u \leq \lambda$ can be absorbed into the terms hidden by the notation. Next, notice that, by the correctness of Π_0, we must have that $k' = k$, so Π is correct.

Trace. We now explain how to trace. Due to lack of space, we sketch the tracing algorithm, assuming the ability to *perfectly* estimate the decoder's success probability on distributions of ciphertexts. In reality, such probabilities will need to be estimated; it is straightforward but tedious to handle such estimates.

1. We will initialize a probability distribution Z over $\{0,1\}^{Nu}$, which is initially uniform. Let $p_Z = \Pr[D(c, r, k^b) = b : r \leftarrow Z, (c, k^0) \leftarrow \mathsf{Enc}_0(\mathsf{pk}, T_{j,S,r}), k^1 \leftarrow \mathcal{K}, b \leftarrow \{0,1\}]$ be the probability that D correctly distinguishes a random ciphertext using the set $T_{j,S,r}$ for $r \leftarrow Z$. Let p^* be the initial value of p_Z, the probability the decoder currectly guesses b uniform r.
2. Initialize an empty set A. Then, for each $i \in S$ do the following:
   (a) Initialize a counter $\mathsf{ctr}_i = 0$.
   (b) For $\ell = 1, \ldots, u$ do the following:
      i. Let Z_b be the current Z, but conditioned on $r_{i,\ell} = b$. Compute probabilities $p_0 := p_{Z_0}, p_1 := p_{Z_1}$.
      ii. If $p_{1-x_{j,i,\ell}} \geq p^*$, update Z to $Z_{1-x_{j,i,\ell}}$ and set $\mathsf{ctr}_i = \mathsf{ctr}_i + 1$. Otherwise do not update Z.
   (c) If $\mathsf{ctr}_i/u \leq 2/5$, add user i to A.
3. Output A.

The following theorem then establishes Theorem 4:

Theorem 13. *Assuming* Π_0 *is a secure threshold broadcast scheme and* $u = \omega(\log \lambda)$*, Construction 4 is a secure broadcast and trace multi-scheme.*

Proof. Due to lack of space, we only sketch the proof. Trace always maintains the invariant that $p_Z \geq p^*$. In Step 2(b)i, we thus have $(p_0 + p_1)/2 \geq p^*$, meaning at least one of p_0 or p_1 are at least p^*. If user i of instance j is honest, $x_{j,i,\ell}$ is independent of the attacker's view and therefore $p_{1-x_{j,i,\ell}} \geq p^*$ with probability $\geq 1/2$. For honest users, ctr_i will therefore concentrate around $u/2$, and be larger than $(2/5)u$ with overwhelming probability. Thus honest users are not accused.

On the other hand, consider a user i of instance j that is *not* accused. Consider the set $T_{j,S,r} \cap \{[\![j,i,\ell,b]\!]\}_{\ell\in[u],b\in\{0,1\}}$. Once we have finished processing user i, the distribution Z fixes at least $(2/5)u$ of the entries within this set to be *outside* of $U_{j,i}$, and the remaining $\leq (3/5)u$ of the entries are randomly chosen to be either in the set or outside. Therefore, the size of the overlap with $U_{j,i}$ will concentrate around $(3/10)u \leq (2/5)u$. Thus, any user that is not accused will, with overwhelming probability, be unable to decrypt by the end. Since by the end we know that some user can still decrypt (due to our invariant $\geq p^*$), this means *some* user must be accused. □

8.2 Construction of Threshold Broadcast Encryption

We prove the following, which combined with Theorem 13 gives Corollary 4:

Theorem 5 (Formal Version). *There exists a threshold broadcast scheme from pairings which, for $u = \lambda$ has size $(v, 1, 1)$, with security proved in the generic group model.*

Proof. Here, we only sketch the construction; the proof in the generic group model is given in the full version [Zha20]. The scheme is based on ideas from [Del07] (see Section 7.3) and from [AHL+12]. For an upper bound v on $|S|$, the public key is identical to Delerablée's scheme:

$$\mathsf{pk} = (e(g_1,g_2)^\beta, g_1^{\beta\gamma}, (g_2^{\gamma^j})_{j\in[0,v]})$$

Recalling $J(P) = g_2^{P(\gamma)}$, the ciphertext is also identical to Delerablée:

$$c_1 = g_1^{\alpha\beta\gamma} \ \ , \ \ c_2 = J\left(\prod_{i\in S}(1-\gamma/i)\right)^\alpha = g_2^{\alpha\prod_{i\in S}(1-\gamma/i)}$$

with encapsulated key $k = e(g_1,g_2)^{\alpha\beta}$. The secret key for a set U is

$$\mathsf{sk}_U = \left(g_1^{\beta\gamma^j/\prod_{i\in U}(1-\gamma/i)}\right)_{j=0,\dots,|U|-t}$$

where t is the threshold. Notice the size of the secret key is $O(|U|)$; in particular for $|U| = \lambda$ it is independent of v. Notice that, from sk_U, one can compute $g_1^{\beta Q(\gamma)/\prod_{i\in U}(1-\gamma/i)}$ for any polynomial Q of degree at most $|U|-t$.

To decrypt, notice that, since $U\cap S$ has size at least t, $\prod_{i\in U\setminus S}(1-\gamma/i)$ is a polynomial of degree at most $|U|-t$. Therefore, using sk_U, compute

$$g_1^{\beta\prod_{i\in U\setminus S}(1-\gamma/i)\Big/\prod_{i\in U}(1-\gamma/i)} = g_1^{\beta/\prod_{i\in U\cap S}(1-\gamma/i)} \ ,$$

canceling out all poles in sk_U that are not in S. From here, decryption proceeds analogously to Delerablée. Security — in the generic group model — follows a similar argument as in Delerablée, see the full version [Zha20] for details. □

9 Our $\sqrt[3]{N}$ Scheme

We now briefly explain how to prove Theorem 6, establishing a pairing-based multi-scheme of size $(\sqrt{N}, \sqrt{N}, 1, \sqrt{N})$ in the shared randomness model, with security proved in the generic group model. Combined with Theorem 1 setting $T = N^{1/3}$ gives our $\sqrt[3]{N}$-sized scheme.

Set $u = \omega(\log \lambda)$ and $t = (2/5)u$. We interpret the user identity space $[N]$ as $[\sqrt{N}] \times [\sqrt{N}]$; we will alternatively treat each identity i as either a number in $[N]$ or a pair $(i_0, i_1) \in [\sqrt{N}]^2$. Our public key is identical to our risky version of Delerablée (Section 7.3), just for $\sqrt{N}$ users:

$$\mathsf{pk} = \left(e(g_1, g_2)^\beta, g_1^{(\beta\gamma\ ,\ 0\ ,\ 0\ ,\ 0)\cdot R^{-1}}, \left(g_2^{(\gamma^i\ ,\ 0\ ,\ 0\ ,\ 0)\cdot R^T} \right)_{j \in [0, \sqrt{N}]} \right)$$

Let $[\![\cdot]\!]$ be an arbitrarily efficient injection from $[M] \times [\sqrt{N}] \times [u] \times \{0,1\}$ into $\mathbb{Z}_p \setminus \{0\}$. Let $\mathcal{X}$ be a polynomial-sized set and $f : [M] \times [N] \to \mathcal{X}$ be a function to be specified later.

For each instance j, choose a random $i_j^* \in [\sqrt{N}]$, and assign identity $i = (i_0, i_1)$ the attribute $(x_0, x_1) = (0,0), (1,0)$, and $(1,1)$, for $i_0 < i_j^*, i_0 = i_j^*$, and $i_0 > i_j^*$ respectively. Additionally, for each $\theta \in \mathcal{X}$, choose a random scalar $\tau_\theta \in \mathbb{Z}_p$.

To generate secret key for user i of instance j, let $\theta = f(j, i)$. Let $i = (i_0, i_1)$. Choose a random $x_{j,i} \in \{0,1\}^u$, and let $U_{j,i} = \{[\![j, i_1, \ell, x_{i,j,\ell}]\!]\}_{\ell \in [u]} \subseteq \mathbb{Z}_p \setminus \{0\}$. The secret key is very similar to our risky broadcast scheme, but modified to use τ_θ instead of β:

$$\mathsf{sk}_{j,i} = \left(g_1^{\left(\tau_\theta \gamma^\ell \Big/ \prod_{s \in U_{j,i}} (1 - \gamma/s)\ ,\ x_1 v_{1,\ell}\ ,\ x_2 v_{2,\ell}\ ,\ v_{3,\ell} \right) \cdot R^{-1}} \right)_{\ell = 0, \dots, u-t}$$

where the $v_{x,\ell}$ are chosen freshly for each secret key. Additionally, user i of instance j is given a "helper key":

$$\mathsf{hk}_\theta = \left(h_\theta := g_2^{\left(\frac{\beta - \tau_\theta}{\beta\gamma}\ ,\ 0\ ,\ 0\ ,\ 0 \right) \cdot R^T}, \left(g_2^{(\tau_\theta \gamma^i\ ,\ 0\ ,\ 0\ ,\ 0)\cdot R^T} \right)_{j \in [0, \sqrt{N}]} \right)$$

The ciphertext for attribute (y_1, y_2) is

$$c_1 = g_1^{(\alpha\beta\gamma\ ,\ 0\ ,\ 0\ ,\ 0)\cdot R^{-1}} \quad , \quad c_2 = g_2^{\left(\alpha \prod_{s \in T_{j,r}} (1 - \gamma/s)\ ,\ y_1 v_1\ ,\ y_2 v_2\ ,\ 0 \right) \cdot R^T}$$

where $r \in \{0,1\}^{u\sqrt{N}}$ is the $u\sqrt{N} = O(\sqrt{N})$ bits of shared randomness, and $T_{j,r} = \{[\![j, i_1, \ell, r_{i_1,\ell}]\!]\}_{i_1 \in [\sqrt{N}], \ell \in [u]}$. The encapsulated key is $e(g_1, g_2)^{\alpha\beta}$. A valid ciphertext has attribute $(0,0)$, and can be computed from the public key.

Decryption starts off analogously to our threshold broadcast scheme, allowing a user who is authorized to decrypt to compute $H = e(g_1, g_2)^{\alpha\tau_\theta}$, using the components of hk_θ in place of the public key. It remains to convert this into $e(g_1, g_2)^{\alpha\beta}$. This is accomplished by multiplying H by $e(c_1, h_\theta) = e(g_1, g_2)^{\alpha(\beta - \tau_\theta)}$.

(Risky) tracing works roughly as follows: first we perform a risky tracing, analogous to [GKRW18], and then we trace using our threshold broadcast technique from Section 8. In more detail, we first test a decoder on ciphertexts with attributes $(0,1),(1,1)$. If the decoder *cannot* distinguish $(0,1)$ from $(1,1)$, we abort and accuse no one. If the decoder *can* distinguish, then we accuse the "half identity" $i_0 = i_j^*$. We are not done, since we need to fill in the second half identity i_1. Here, we trace as in Section 8, gradually attempting to "turn off" all the users (i_j^*, i_1) for $i_1 = 1, \dots, \sqrt{N}$ by trying to remove elements in $U_{j,(i_j^*,i_1)}$ from $T_{j,r}$. We accuse any user (i_j^*, i_1) where turning off that user fails.

Note that two users with the same i_1 and j will have overlapping sets U. Turning off both users would thus place incompatible constraints on the set $T_{j,r}$, and hence both cannot be simultaneously turned off. Therefore, we can only freely turn off users for distinct i_1. This is why we trace i_0 first, and then i_1. Also note that, when tracing i_1, turning off an honest user (i_0, i_1) will succeed even if the adversary controls a different user with the same i_1 (but different i_0), since turning off users only required that the set $U_{j,i}$ was unknown to the adversary. This "independence" is crucial for this layered tracing approach to work[11].

Choosing f. There are two requirements we need from f. First, for security, we need that no two secret keys with overlapping U get mapped to the same θ, for reasons similar to why our threshold broadcast scheme is insecure for overlapping U. Therefore, we need that $f(j,(i_0,i_1)) \neq f(j,(i_0',i_1))$ for any j, i_1 and $i_0 \neq i_0'$. Once this requirement is met, the following is proved in the full version [Zha20]:

Lemma 1. *The scheme above is $1/\sqrt{N}$-risky in the generic group model.*

Second, excluding the helper keys hk_θ (which have size $O(\sqrt{N})$), the secret keys are constant-sized. Applying Theorem 3/Construction 3, the secret key will now contain the secret keys and helper keys from $O(\sqrt{N})$ different instances. In order to ensure that the overall secret key remains $O(\sqrt{N})$, we require all of the constituent instances to have the same helper key. Thus, we need $f(j_0,i) = f(j_1,i)$, for all j_0, j_1 that get mapped to the same secret key when applying Construction 3. Recall that Construction 3 interpreted $[M]$ as $[M'] \times [T]$, and instance $j = (j',t)$ gets mapped to j'. Thus, setting $\mathcal{X}$ to be the set $[M'] \times [N]$ and $f((j',t),i) = (j',i)$ will satisfy both conditions, giving Theorem 6.

10 Running Times

Here, we briefly discuss the running times of our constructions; see the full version [Zha20] for a more in-depth discussion.

[11] A natural question is whether a similar layer of risky tracing can be added on top of [BSW06], potentially giving a simpler path toward $\sqrt[3]{N}$. Unfortunately, [BSW06] in a sense "uses up" the pairing, preventing any risky layer from being independent of the underlying PLBE-based tracing. Concretely, the obvious approach would yield a scheme where it was *not* possible to turn off an honest user if the adversary controlled a different user with the same i_1. Our construction gets around this issue by having the second layer tracing happen "outside" the pairing, in the shared randomness.

We will say that a traitor tracing scheme Π is *asymptotically efficient* if each of $\mathsf{Gen}, \mathsf{Enc}$, and Dec have running times bounded by $(|\text{input}| + |\text{output}|) \times \mathsf{poly}(\lambda)$.

We note that all of our algebraic instantiations are asymptotically efficient, and that our threshold elimination and risk mitigation compilers (Theorems 2 and 3) preserve asymptotic efficiency. However, if the running time of Enc is longer than the ciphertext size (which is in particular possible when public keys are larger than ciphertexts), our user expansion compiler (Theorem 7) does *not* preserve asymptotic efficiency: the running time and ciphertext size get multiplied by a factor of T, but the input to Enc (namely, the public key) stays the same.

This issue affects our $(N^{1-a}, 1, N^a, N^{1-a})$ threshold broadcast-based construction, as well as our $\sqrt[3]{N}$ scheme. For our other schemes, the public key is smaller than the ciphertext, and hence this is not an issue.

In the full version, we explain how to remove the inefficiency from these two constructions; thus all of our constructions can be made asymptotically efficient. We carefully choose how the user identities are embedded in $\mathbb{Z}_p$. The result is that generating the multiple ciphertext components for Theorem 7 reduces to evaluating a polynomial at multiple points, except that the coefficients of the polynomial and the resulting evaluations are *in the exponent* of the pairing. Thus, we carry out fast multi-point polynomial evaluation methods "in the exponent"; this incurs a $\mathsf{polylog}(N)$ overhead, which can be absorbed into the $\mathsf{poly}(\lambda)$ term.

References

ABP$^+$17. Shweta Agrawal, Sanjay Bhattacherjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. Efficient public trace and revoke from standard assumptions: Extended abstract. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2277–2293. ACM Press, October / November 2017.

ADM$^+$07. Michel Abdalla, Alexander W. Dent, John Malone-Lee, Gregory Neven, Duong Hieu Phan, and Nigel P. Smart. Identity-based traitor tracing. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 361–376. Springer, Heidelberg, April 2007.

AHL$^+$12. Nuttapong Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie de Panafieu, and Carla Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. 2012.

AY20. Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and lwe. In *EUROCRYPT 2020*, 2020.

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

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

BKM10. Dan Boneh, Aggelos Kiayias, and Hart William Montgomery. Robust fingerprinting codes: A near optimal construction. In *Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management*, DRM'10, pages 3–12. Association for Computing Machinery, 2010.

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

BS95. Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 452–465. Springer, Heidelberg, August 1995.

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

BSW07. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007.

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

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

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

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

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

DG17. Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017.

GGH$^+$13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

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

GKSW10. Sanjam Garg, Abishek Kumarasubramanian, Amit Sahai, and Brent Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 121–130. ACM Press, October 2010.

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David

Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 660–670. ACM Press, June 2018.

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

GL89. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.

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

GVW19. Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. Collusion resistant broadcast and trace from positional witness encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 3–33. Springer, Heidelberg, April 2019.

KMUW18. Lucas Kowalczyk, Tal Malkin, Jonathan Ullman, and Daniel Wichs. Hardness of non-interactive differential privacy from one-way functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 437–466. Springer, Heidelberg, August 2018.

KMUZ16. Lucas Kowalczyk, Tal Malkin, Jonathan Ullman, and Mark Zhandry. Strong hardness of privacy from weak traitor tracing. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 659–689. Springer, Heidelberg, October / November 2016.

KW19. Sam Kim and David J. Wu. Collusion resistant trace-and-revoke for arbitrary identities from standard assumptions. Cryptology ePrint Archive, Report 2019/984, 2019. https://eprint.iacr.org/2019/984.

KY02. Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In David Naccache and Pascal Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 141–158. Springer, Heidelberg, February 2002.

LPSS14. San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-LWE and applications in traitor tracing. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 315–334. Springer, Heidelberg, August 2014.

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

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

Tar03. Gábor Tardos. Optimal probabilistic fingerprint codes. In *35th ACM STOC*, pages 116–125. ACM Press, June 2003.

TZ17. Bo Tang and Jiapeng Zhang. Barriers to black-box constructions of traitor tracing systems. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 3–30. Springer, Heidelberg, November 2017.

Zha20. Mark Zhandry. New techniques for traitor tracing, 2020. Full Version.