Collusion Resistant Trace-and-Revoke for Arbitrary Identities from Standard Assumptions*

Sam Kim[1] and David J. Wu[2]

[1] Stanford University, Stanford, CA, USA
[2] University of Virginia, Charlottesville, VA, USA

Abstract. A traitor tracing scheme is a multi-user public-key encryption scheme where each user in the system holds a decryption key that is associated with the user's identity. Using the public key, a content distributor can encrypt a message to all of the users in the system. At the same time, if a malicious group of users combine their respective decryption keys to build a "pirate decoder," there is an efficient tracing algorithm that the content distributor can use to identify at least one of the keys used to construct the decoder. A *trace-and-revoke* scheme is an extension of a standard traitor tracing scheme where there is an additional key-revocation mechanism that the content distributor can use to disable the decryption capabilities of compromised keys. Namely, during encryption, the content distributor can encrypt a message with respect to a list of revoked users such that only non-revoked users can decrypt the resulting ciphertext.

Trace-and-revoke schemes are challenging to construct. Existing constructions from standard assumptions can only tolerate bounded collusions (i.e., there is an *a priori* bound on the number of keys an adversary obtains), have system parameters that scale *exponentially* in the bit-length of the identities, or satisfy weaker notions of traceability that are vulnerable to certain types of "pirate evolution" attacks. In this work, we provide the first construction of a trace-and-revoke scheme that is fully collusion resistant and capable of supporting arbitrary identities (i.e., the identities can be drawn from an exponential-size space). Our scheme supports public encryption and secret tracing, and can be based on the sub-exponential hardness of the LWE problem (with a super-polynomial modulus-to-noise ratio). The ciphertext size in our construction scales logarithmically in the size of the identity space and linearly in the size of the revocation list. Our scheme leverages techniques from both combinatorial and algebraic constructions for traitor tracing.

1 Introduction

Traitor tracing schemes [CFN94] provide content distributors a way to identify malicious receivers and pirates. Specifically, a traitor tracing scheme is a public-key encryption scheme that is defined over a set of global public parameters

*The full version of this paper is available at `https://eprint.iacr.org/2019/984.pdf`

pp and many secret decryption keys $\{\mathsf{sk}_{\mathsf{id}}\}$. Each of the decryption keys $\mathsf{sk}_{\mathsf{id}}$ is associated with an identifier id (e.g., a user's name or profile picture). Anyone is able to encrypt a message using the public parameters pp and any user who holds a valid decryption key $\mathsf{sk}_{\mathsf{id}}$ can decrypt the resulting ciphertext. The main security property is *traceability*, which says that if a coalition of users combine their respective decryption keys to create a new decryption algorithm (i.e., a "pirate decoder"), there is an efficient tracing algorithm that, given (black-box) access to the decoder, will successfully identify at least one of the secret keys that was used to construct the pirate decoder. As such, traitor tracing schemes provide an effective way for content distributors to combat piracy.

In practice, simply identifying the keys that went into a pirate decoder is not enough; we also require a way for the content distributor to disable the decryption capabilities of a compromised key. Traitor tracing schemes that support efficient key-revocation mechanisms are called *trace-and-revoke* schemes [NP00]. In a trace-and-revoke scheme, the encryption algorithm additionally takes in a list of revoked users $\mathcal{L}$. A ciphertext that is generated with respect to a revocation list $\mathcal{L}$ can only be decrypted by keys for identities $\mathsf{id} \notin \mathcal{L}$. Furthermore, the revocation mechanism should remain compatible with tracing: namely, if an adversary builds a pirate decoder that can still decrypt ciphertexts encrypted with respect to a revocation list $\mathcal{L}$, the tracing algorithm should successfully identify at least one of the non-revoked decryption keys (i.e., some $\mathsf{id} \notin \mathcal{L}$) that went into the construction of the pirate decoder. We give the formal definition in Section 4.

Properties of trace-and-revoke schemes. There are a number of possible properties that a trace-and-revoke scheme could provide. We enumerate several important ones below:

- **Collusion resistance:** A trace-and-revoke scheme is t-collusion resistant if tracing works as long as the pirate obtains fewer than t decryption keys, and the scheme parameters are allowed to depend on t. When t can be an arbitrary polynomial, the scheme is *fully collusion resistant*.
- ***A priori* unbounded revocation:** Some trace-and-revoke schemes support bounded revocation where at setup time, there is an *a priori* bound r on the maximum number of revoked users the scheme supports. A scheme supports *a priori unbounded revocation* if the number of revoked users can be an arbitrary polynomial. We note here that while we can require an even stronger property that supports revoking a super-polynomial number of users, the scheme we develop in this work does not support this stronger property (except in certain restricted settings; see Section 1.1).
- **Black box tracing:** A trace-and-revoke scheme supports black box tracing if the tracing algorithm only requires oracle access to the pirate decoder. This means we do not need to impose any restrictions on the structure of the adversary's decoder. Tracing must work on *any* decoder that is able to decrypt (or even better, *distinguish*) ciphertexts.
- **Identity-based:** A trace-and-revoke scheme is "identity-based" or supports *arbitrary* identities if the set of possible identities $\mathcal{ID}$ the scheme supports

can be exponential in size [NWZ16]. In most trace-and-revoke schemes, the set of possible identities is assumed to have polynomial size (i.e., identities are represented by an element of the set $[N] = \{1, \ldots, N\}$). This means that there is an a priori bound on the maximum number of users supported by the system, and moreover, in practical scenarios, the tracing authority needs to separately maintain a database mapping from a numeric index $\mathsf{id} \in [N]$ to a user's actual identifier (which may not fit into a string of length $\log N$). In addition, as noted in [NWZ16], an added benefit of trace-and-revoke schemes that support arbitrary identities is *anonymity*: namely, a user can obtain a decryption key for their identity without needing to reveal their identity to the key issuer.

Our results. In this work, we focus on constructing trace-and-revoke schemes that provide each of the above guarantees. Namely, we seek schemes that are flexible (e.g., can support arbitrary identities of polynomial length and an arbitrary polynomial number of revocations) while providing strong security (i.e., full collusion resistance and security against arbitrary adversarial strategies). We achieve these properties assuming sub-exponential hardness of the learning with errors (LWE) assumption [Reg05]. Specifically, we show the following:

Theorem 1.1 (informal). *Let λ be a security parameter and $\mathcal{ID} = \{0,1\}^n$ be the set of possible identities. Assuming sub-exponential hardness of LWE, there exists a fully collusion resistant trace-and-revoke scheme where the secret key for an identity $\mathsf{id} \in \{0,1\}^n$ has size $n \cdot \mathsf{poly}(\lambda, \log n)$ and a ciphertext encrypting a message m with respect to a revocation list $\mathcal{L} \subseteq \{0,1\}^n$ has size $|m| + |\mathcal{L}| \cdot \mathsf{poly}(\lambda, \log n)$. Encryption in our scheme is a public operation while tracing requires knowledge of a secret key.*

Previous trace-and-revoke constructions were either not collusion resistant [NWZ16, ABP$^+$17], could only support a polynomial-size identity space [BW06, GKSW10, GQWW19], achieved weaker models of tracing [NNL01, DF02], or relied on strong assumptions such as indistinguishability obfuscation [NWZ16] or (positional) witness encryption [GVW19]. We refer to Section 1.2 for a more detailed comparison of our construction with existing ones.

Open questions. Before giving an overview of our construction, we highlight several interesting directions to further improve upon our trace-and-revoke scheme:

- *Public tracing:* Our tracing algorithm requires a secret key. It is an interesting open problem to obtain fully collusion resistant trace-and-revoke for arbitrary identities with public tracing from standard assumptions. In fact, even obtaining a collusion resistant traitor tracing scheme with succinct keys and public tracing from standard assumptions is currently open.
- *Succinct broadcast:* The length of the ciphertexts in our construction scales *linearly* in the size of the revocation list, and as such, our scheme only supports revocation for a polynomial number of users. It is an open question is to develop an scheme that supports arbitrary identities and where the

ciphertext size scales *sublinearly* in the number of revoked users (and more generally, where the ciphertext size scales with the *description length* of the revocation list rather than its size). Schemes with these properties are often called "broadcast, trace, and revoke" schemes [BW06] as they combine both the succinctness of a "broadcast encryption" [FN93] with the tracing capability of a traitor tracing scheme. Existing broadcast, trace, and revoke constructions [BW06, GKSW10, GQWW19] from standard assumptions can only handle a polynomial number of users. We provide a more thorough comparison in Section 1.2.

– *Polynomial hardness:* Security of our tracing construction relies on the *sub-exponential* hardness of LWE. Our reliance on sub-exponential hardness assumptions is due to our use of complexity leveraging [BB04] to instantiate *adaptively-secure* variants of the underlying cryptographic primitives we require in our construction. An important open problem is to base security on polynomial hardness. The work of Goyal et al. [GKW19] show how to obtain traitor tracing for an exponential-size identity space from a polynomial hardness assumption, but their scheme does not support revocation.

1.1 Construction Overview

In this section, we provide a high-level overview of our construction. Our approach combines an identity-based traitor tracing scheme based on the techniques developed in [NWZ16, GKW18] with the combinatorial revocation scheme from [NNL01]. We describe each of these components below.

Traitor tracing from private linear broadcast. Boneh et al. [BSW06] showed how to construct a collusion resistant traitor tracing scheme from a private linear broadcast encryption (PLBE) scheme. A PLBE scheme is an encryption scheme where decryption keys are associated with an index $i \in [N]$, and ciphertexts are associated with a secret index $j \in [N]$ and a message m. The correctness property guarantees that a decryption key sk_i for index i can decrypt all ciphertexts encrypted to indices j where $i \leq j$. There are two ways to generate a ciphertext. The *public* encryption algorithm allows anyone to encrypt to the index N, which can be decrypted by secret keys sk_i for all $i \in [N]$. The *secret* encryption algorithm allows the tracing authority who holds a tracing key to encrypt to indices $j \leq N$. The "index-hiding" requirement guarantees that an adversary who does not have a key for index j cannot distinguish an encryption to index j from an encryption to index $j + 1$. Finally, the "message-hiding" requirement says that ciphertexts encrypted to index 0 are semantically secure (given any subset of decryption keys for indices $1 \leq j \leq N$). These properties form the basis of the tracing algorithm described in [BSW06]. Boneh et al. showed how to construct PLBE from pairing-based assumptions where the ciphertexts have size $O(\sqrt{N})$. Hence their scheme only supports a polynomial-size identity space.

Recently, Goyal et al. [GKW18] gave a new construction of a PLBE scheme from the LWE assumption by combining a new cryptographic notion called mixed functional encryption (mixed FE) with an attribute-based encryption (ABE) scheme [SW05, GPSW06]. Their construction has the appealing property that the size of all of the system parameters (e.g., the public parameters, decryption keys, and ciphertexts) scale with $\mathsf{poly}(\lambda, \log N)$. Thus, the construction of Goyal et al. [GKW18] can in principle support arbitrary set of identities. However, the tracing algorithm in the PLBE framework runs in time that scales *linearly* with the size of the identity space. As a result, the [GKW18] construction does not support tracing over an exponential space of identities.

Identity-based traitor-tracing from functional encryption. In [NWZ16], Nishimaki et al. introduced a more general tracing algorithm for PLBE that supports an exponential identity space (by abstracting the tracing problem as an "oracle jump-finding" problem). Their construction relies on a PLBE scheme that satisfies a more general notion of index-hiding security. Namely a ciphertext encrypted to index j_1 should be indistinguishable from a ciphertext encrypted to index j_2 as long as the adversary does not have any keys in the interval $(j_1, j_2]$.[3] A limitation of this construction is that the ciphertexts scale linearly in the bit-length of the identities. Nishimaki et al. then show how to construct a traitor tracing scheme with *short* ciphertexts (i.e., one where the ciphertext size scales with $\mathsf{poly}(\log \log N)$) from a private broadcast encryption scheme that support slightly more general broadcast sets. Finally, they note that private broadcast is just a special case of general-purpose functional encryption which can be instantiated using indistinguishability obfuscation [GGH+13], or, in the bounded-collusion setting, from LWE [GKP+13] or even just public-key encryption [SS10, GVW12].

A more general view of [GKW18]. In this work, we take a more general view of the PLBE construction in [GKW18] and show that the construction in fact gives a *secret-key predicate encryption scheme with a broadcast functionality.* In turn, PLBE can be viewed as a specific instantiation of the predicate encryption scheme for the particular class of threshold predicates. This view will enable our generalization to identity-based traitor tracing with short ciphertexts (by following the approach of [NWZ16]) as well as enable an efficient mechanism for key revocation. Note that the "broadcast functionality" considered here refers to a method to *publicly* encrypt a message that can be decrypted by *all* secret keys in the system (i.e., broadcasting a message to all users in the system). We are not requiring the ability to succinctly broadcast messages to subsets of users (as in the setting of broadcast encryption [FN93]).

Specifically, in a secret-key (ciphertext-policy) predicate encryption scheme, ciphertexts are associated with a predicate f and a message m, while decryption keys are associated with an attribute x. Decrypting a ciphertext $\mathsf{ct}_{f,m}$ associated with a predicate f and a message m with a function key for an attribute x yields m if $f(x) = 1$ and $\perp$ otherwise. Moreover, the policy f associated with a ciphertext is hidden irrespective of whether decryption succeeds or not—this

[3] This property follows from the usual index-hiding security game by a standard hybrid argument when the indices are drawn from a polynomial-size space, but not when the indices are drawn from an exponentially-large one.

property is the analog of the "strong" attribute-hiding property considered in the study of key-policy predicate encryption [BW07, KSW08, SBC$^+$07]. Finally, while the predicate encryption scheme is secret-key, there exists a *public* encryption algorithm that allows anyone to encrypt a message with respect to the "always-accept" policy (i.e., $f(x) = 1$ for all inputs x). In Section 3.1, we show how to combine mixed FE (for general circuits) and attribute-based encryption (for general circuits) to obtain a secret-key ciphertext-policy predicate encryption scheme with broadcast. This construction is a direct analog of the [GKW18] construction of PLBE from the same set of underlying primitives. Next, we note that this type of predicate encryption directly implies a fully collusion resistant traitor tracing scheme with short ciphertexts via [NWZ16]. The one difference, however, is that since the predicate encryption scheme is in the secret-key setting, only the tracing authority who holds the master secret key is able to run the tracing algorithm. Thus in contrast to [NWZ16], our scheme only supports *secret tracing*. We note that working in the secret-key setting introduces some new challenges in the security analysis of the [NWZ16] construction. These can be handled using similar techniques as those developed in [GKW18], and we discuss this in greater detail in Section 4.1.

Trace-and-revoke via revocable predicate encryption. Thus far, we have shown how to combine ideas from [GKW18] and [NWZ16] to obtain a collusion resistant traitor tracing scheme for arbitrary identities. The next step is to develop a mechanism for key revocation. Previously, Nishimaki et al. showed how to use a revocable functional encryption scheme to construct a trace-and-revoke scheme. In this work, we show that a revocable variant of our secret-key predicate encryption scheme with broadcast also suffices for this general transformation. Namely, in a revocable predicate encryption scheme, each decryption key is additionally tagged with an identity id, and at encryption time (both secret and public), the encrypter provides both the decryption policy f and the revocation list $\mathcal{L}$. The resulting ciphertext can then be decrypted by all keys $\mathsf{sk}_{\mathsf{id},x}$ associated with an identity id and an attribute x such that $f(x) = 1$ and $\mathsf{id} \notin \mathcal{L}$.

A natural approach to support revocation is to include the revocation list $\mathcal{L}$ as part of the ciphertext policy in the predicate encryption scheme. We would then embed the identity id as part of the decryption key, and the final decryption policy would first check that $\mathsf{id} \notin \mathcal{L}$ and then check that $f(x) = 1$. While this basic approach seems straightforward, it unfortunately does not apply in our setting. As noted above, the predicate encryption scheme we construct is a *secret-key* scheme, and the only public operation it supports is the broadcast functionality.[4] Obtaining a public-key analog of collusion resistant, strong attribute-hiding predicate encryption seems quite challenging (and in fact, implies public-key

[4]The recent work of Goyal et al. [GQWW19] introduces a notion of *broadcast mixed FE* that supports a *succinct* public broadcast to a restricted set of identities (of polynomial size). The notion we develop in this work supports an exponential-sized identity space, but in a *non-succinct* manner (i.e., the ciphertext size scales linearly with the size of the revocation list).

functional encryption). But as we note in Remark 3.3, even in the bounded-collusion setting (where we can construct public-key predicate encryption from standard assumptions), this basic approach seems to run into a barrier, and any such instantiation from standard assumptions would likely have to assume a bound on the maximum number of revoked users. In this work, we seek solutions from standard assumptions that are collusion resistant and support unbounded revocation.

Revocable predicate encryption via subset cover set systems. As we described above, constructing a collusion resistant trace-and-revoke scheme for arbitrary identities reduces to constructing a secret-key revocable predicate encryption scheme with a broadcast functionality. To build the necessary revocable predicate encryption scheme, we leverage ideas from combinatorial constructions of traitor tracing. We note that while we rely on combinatorial ideas in our construction, we do not provide a generic transformation of any predicate encryption scheme into a revocable analog. Rather, our construction relies on a careful integration of the algebraic approach from [GKW18] with the combinatorial approach from [NNL01].

The core combinatorial ingredient that we use for our construction is a subset-cover set system, a notion that has featured in several traitor tracing constructions [NNL01, DF02, HS02]. Let $[N]$ be the identity space. A subset-cover set system for $[N]$ is a set of indices $[K]$ with the following two properties. Each identity $\mathsf{id} \in [N]$ is associated with a small number of indices $\mathcal{I}_{\mathsf{id}} \subseteq [K]$. Moreover, given a revocation list $\mathcal{L} \subseteq [N]$, there is an efficient algorithm to compute a "covering" set of indices $\mathcal{J}_{\mathcal{L}} \subseteq [K]$ with the property that $\mathsf{id} \in \mathcal{L}$ if and only if $\mathcal{I}_{\mathsf{id}} \cap \mathcal{J}_{\mathcal{L}} = \varnothing$. If we instantiate using the subset-cover set system from [NNL01], then $K = O(N)$, $|\mathcal{I}_{\mathsf{id}}| = O(\log N)$, and $|\mathcal{J}_{\mathcal{L}}| = O(|\mathcal{L}| \log(N/|\mathcal{L}|))$.

Given a subset-cover set system, a first attempt to construct a revocable predicate encryption scheme is as follows. We associate a set of public parameters pp_i and master secret key msk_i with each index $i \in [K]$. A key for an identity $\mathsf{id} \in [N]$ and an attribute x would consist of predicate encryption keys $\mathsf{sk}_{\mathsf{id},x} \leftarrow \mathsf{KeyGen}(\mathsf{msk}_i, x)$ for all the predicate encryption schemes $i \in \mathcal{I}_{\mathsf{id}}$ associated with id. Finally, an encryption of a message m with respect to the revocation list $\mathcal{L} \subseteq [N]$ would consist of a collection of ciphertexts $\{\mathsf{ct}_i\}_{i \in \mathcal{J}_{\mathcal{L}}}$ where each ct_i is an encryption of m with respect to pp_i for $i \in \mathcal{J}_{\mathcal{L}}$. By the property described above, if $\mathsf{id} \notin \mathcal{L}$, then $\mathcal{I}_{\mathsf{id}} \cap \mathcal{J}_{\mathcal{L}} \neq \varnothing$. This means that all non-revoked users $\mathsf{id} \notin \mathcal{L}$ will possess a key $\mathsf{sk}_{i,x}$ for some $i \in \mathcal{J}_{\mathcal{L}}$, and therefore, will be able to decrypt (provided that $f(x) = 1$). For a revoked user, it will be the case that $i \notin \mathcal{J}_{\mathcal{L}}$ for all $i \in \mathcal{I}_{\mathsf{id}}$, and they will be unable to decrypt. The problem though is that the size of the public parameters now scale *linearly* with K (which is as large as N). As such, this scheme only supports a polynomial number of identities. Thus, we need a different approach. We describe two candidate ideas below:

– If the underlying predicate encryption scheme has the property where the master secret key msk can be sampled *after* the public parameters pp, then in principle, the construction above would suffice. Namely, we would use a single set of public parameters for all of the predicate encryption schemes, and derive the master secret key msk_i for each $i \in [K]$ from a pseudorandom function (PRF). Unfortunately, such a predicate encryption scheme cannot be secure since the adversary can always generate for itself a master secret key and use it to decrypt.

- If the scheme supports a *public* encryption algorithm, then we can support revocation by including the index $i \in [K]$ as part of the policy associated with the ciphertext as well as the attribute in the decryption key. Then, the decryption policy would additionally check that the index associated with the key matched the index associated with the ciphertext. Essentially, we ensure that a decryption key for i can only be used to decrypt ciphertexts encrypted to index i. However, this revocation approach also does not seem to apply in our setting because our predicate encryption scheme is in the secret-key setting, and it is not clear how to generalize to a public-key encryption algorithm that can support more general policies (while retaining the same security properties).[5]

While neither of these approaches directly apply in our setting, we can combine *both* ideas in our construction to obtain a revocable predicate encryption scheme. As noted above, our basic secret-key predicate encryption scheme with broadcast combines a mixed FE scheme with an ABE scheme. Without getting into too many details, the construction has the following properties. Each ciphertext in the scheme consists of a mixed FE ciphertext and an ABE ciphertext, and analogously, each decryption key consists of a mixed FE decryption key and an ABE decryption key. The mixed FE scheme is a secret-key scheme that supports a broadcast mechanism while the ABE scheme is a standard public-key scheme. The key observation is that if *both* the underlying mixed FE scheme and the ABE scheme support revocation, then the resulting predicate encryption scheme also supports revocation. For our construction it is critical that both schemes support revocation as we rely on the mixed FE scheme to hide the ciphertext policy and the ABE scheme to hide the message. If only one of the underlying schemes supports revocation, then one or both of these security properties become incompatible with revocation. We now describe how we implement revocation for the underlying mixed FE and ABE schemes:

- The mixed FE scheme is a secret-key scheme that supports public broadcast. Unlike standard predicate encryption, the security properties of mixed FE can be satisfied by schemes where the master secret key is sampled *after* the public parameters, and this property is satisfied by existing constructions [GKW18, CVW$^+$18]. This means that we can associate a different mixed FE scheme with each index $i \in [K]$ where the master secret key associated with each

[5]While the notion of attribute-based mixed FE from [CVW$^+$18] seems like it would also provide this functionality, this revocation approach only preserves the message hiding property and not the mixed FE attribute hiding property of the underlying attribute-based mixed FE scheme. For our trace-and-revoke scheme, we require both message hiding and attribute hiding (which we refer to as "function hiding"). Obtaining the latter property seemingly requires a way to revoke mixed FE decryption keys.

instance is derived from a PRF. All of the mixed FE schemes share a common set of public parameters. We can now use the first revocation idea described above to implement revocation for the mixed FE scheme.

– Next, the ABE scheme is a public-key encryption scheme, and thus, we can use the second type of revocation described above. Namely, we require a single set of ABE parameters and simply include the index $i \in [K]$ in both the decryption key and the ciphertext to identity which index is being targeted.

By combining these two approaches for revocation, we show in Section 3.1 how to construct a secret-key revocable predicate encryption with broadcast scheme from the sub-exponential hardness of LWE. Notably, our final revocation mechanism relies critically on both the combinatoric properties of the subset-cover set system as well as the specific algebraic nature of the predicate encryption construction. Together, this yields the first collusion resistant trace-and-revoke scheme for arbitrary identities from the same underlying assumptions (Theorem 1.1).

A simple extension: more general revocation policies. While the basic scheme we described above supports revoking any polynomial number of identities, it naturally extends to support any revocation policy supported by the underlying subset-cover set system. Specifically, if we use the prefix-based subset-cover set system by Naor et al. [NNL01], our scheme supports revoking any number of identities that can be specified by a polynomial number of *prefix-based patterns*. For instance, we can revoke all users whose identity starts with a fixed prefix—which may consist of an *exponential* number of identities. In a concrete application, if the first few bits of a user's identity specifies a region, then we can use prefix-based policies to efficiently revoke all of the users from one or more regions. We provide more discussion in Remark 3.10.

1.2 Related Work

In this section, we survey some of the related work on traitor tracing and trace-and-revoke schemes and compare our results to existing notions.

Traitor tracing and trace-and-revoke. Numerous works have studied constructions of both traitor tracing and trace-and-revoke schemes from a wide range of assumptions and settings. Very broadly, most existing constructions can be categorized into two main categories: *combinatorial* approaches [CFN94, NP98, SSW01, CFNP00, NNL01, HS02, DF02, SSW01, BN08] and *algebraic* approaches [KD98, NP00, BSW06, BW06, GKSW10, LPSS14, KT15, NWZ16, ABP$^+$17, GKW18, CVW$^+$18, GVW19, GQWW19]. We refer to these works and the references therein for a survey of the field.

Many existing traitor-tracing and trace-and-revoke schemes (from standard assumptions) are only secure against bounded collusions [CFN94, KD98, NP00, SSW01, LPSS14, KT15, NWZ16, ABP$^+$17]. Other schemes are fully collusion resistant, but can only handle a polynomial-size identity space [BSW06, BW06, GKSW10, GKW18, CVW$^+$18, GQWW19]. In this work, we focus on schemes that are fully collusion resistant and support arbitrary identity spaces. While there are schemes that are both collusion resistant and support a super-polynomial identity space [NWZ16, GVW19], these construction require strong assumptions such as indistinguishability obfuscation [BGI$^+$12] or positional witness encryption and cannot currently be based on standard intractability assumptions.

Several of the aforementioned schemes from standard assumptions [BW06, GKSW10, GQWW19] additionally provide a *succinct* broadcast mechanism where anyone can encrypt a message to any subset of the users with a ciphertext whose size scales with $N^{1/2}$ [BW06, GKSW10] or with N^ε [GQWW19] for any constant $\varepsilon > 0$, where N is the total number of users in the system. Such schemes are commonly referred to as "broadcast, trace, and revoke" schemes. Notably, the ciphertext size in these constructions is *independent* of the number of revoked users and only depends on the total number of users. In our trace-and-revoke construction (Theorem 1.1), the ciphertext size scales *linearly* with the number of revoked users (which can be $\Omega(N)$ in the worst case). Thus, in the setting where we have a polynomial-size identity space and when the number of revoked users is a sufficiently-large fraction of the total number of users, existing broadcast, trace, and revoke constructions will have shorter ciphertexts. In the setting where there is an exponential identity space, the ciphertexts in these existing constructions are also exponential, and they do not provide a compelling solution.

Several works [NP98, CFNP00, BN08] consider a threshold notion of traitor tracing where the tracing algorithm is only guaranteed to work for decoders that succeed with probability at least $\delta = 1/\mathsf{poly}(\lambda)$ (and the scheme parameters are allowed to depend on the parameter δ). In this work, we focus on schemes that work for any decoder that succeeds with non-negligible probability.

Some combinatorial constructions [NNL01, HS02, DF02] are fully collusion resistant, but they only satisfy a weaker notion of traceability where the tracing algorithm either succeeds in extracting a pirated key *or* identifies an encryption strategy that disables the pirate decoder (this latter strategy increases the ciphertext size). This weaker traceability notion has led to pirate evolution [KP07] and Pirate 2.0 attacks [BP09] on schemes satisfying this weaker security notion. In this work, we focus on the strong notion of traceability where the tracing algorithm always succeeds in extracting at least one pirate key from any functional decoder. This notion is not affected by the pirate evolution attacks.

Cryptographic watermarking. A closely-related notion to traitor tracing is cryptographic watermarking [BGI$^+$12, CHN$^+$16]. Very briefly, a cryptographic watermarking scheme allows an authority to embed arbitrary data into the secret key of a cryptographic function such that the marked program preserves the original functionality, and moreover, it is difficult to remove the watermark from the program without destroying its functionality. A collusion resistant watermarking scheme for a public-key encryption scheme would imply a collusion resistant traitor tracing scheme. Existing constructions [KW17, QWZ18, KW19b] of watermarking from standard assumptions are not collusion resistant and they are also limited to watermarking PRFs, which are not sufficient for traitor tracing. The recent construction of watermarking for public-key primitives [GKM$^+$19] does imply a traitor tracing scheme for general identities (with public tracing), but only provides bounded collusion resistance (in fact, in this setting, their construction precisely coincides with the bounded collusion resistant traitor tracing construction from [NWZ16]). Moreover, it is not clear that existing constructions of watermarking can be extended to support key revocation.

Concurrent work. In a recent and concurrent work, Goyal et al. [GKW19] also study the problem of identity-based traitor tracing for arbitrary identities (i.e., which they call "traitor tracing with embedded identities"). Their focus is on traitor tracing (without revocation) and achieving security based on *polynomial* hardness assumptions. In contrast, our focus is on supporting both tracing *and* revocation while still supporting arbitrary identities. Security of our construction, however, does rely on making a stronger sub-exponential hardness assumption.

2 Preliminaries

We begin by introducing some notation. We use λ (often implicitly) to denote the security parameter. We write $\mathsf{poly}(\lambda)$ to denote a quantity that is bounded by a fixed polynomial in λ and $\mathsf{negl}(\lambda)$ to denote a function that is $o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We say that an event occurs with overwhelming probability if its complement occurs with negligible probability. We say an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. For two families of distributions $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda\in\mathbb{N}}$, we write $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$ if the two distributions are computationally indistinguishable (i.e., no efficient algorithm can distribution $\mathcal{D}_1$ from $\mathcal{D}_2$ except with negligible probability).

For an integer $n \geq 1$, we write $[n]$ to denote the set of integers $\{1, \ldots, n\}$. For integers $1 \leq m \leq n$, we write $[m, n]$ to denote the set of integers $\{m, m+1, \ldots, n\}$, and $[m, n]_{\mathbb{R}}$ to denote the closed interval between m and n (inclusive) over the real numbers. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote that x is drawn from $\mathcal{D}$. For a finite set S, we write $x \xleftarrow{\text{R}} S$ to denote that x is drawn uniformly at random from S.

Cryptographic primitives. We now recall the standard definition of pseudorandom functions and collision-resistant hash functions.

Definition 2.1 (Pseudorandom Function [GGM84]). *A pseudorandom function (PRF) with key-space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, and range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ is an efficiently-computable function $F \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that for all efficient adversaries $\mathcal{A}$,*

$$\Pr[k \xleftarrow{\text{R}} \mathcal{K} : \mathcal{A}^{F(k,\cdot)}(1^\lambda) = 1] - \Pr[f \xleftarrow{\text{R}} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}] : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1] = \mathsf{negl}(\lambda).$$

Definition 2.2 (Keyed Collision-Resistant Hash Function). *A keyed collision-resistant hash function with key-space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, and range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ is an efficiently-computable function $H \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that for all efficient adversaries $\mathcal{A}$ and sampling $k \xleftarrow{\text{R}} \mathcal{K}$,*

$$\Pr[(x_0, x_1) \leftarrow \mathcal{A}(1^\lambda, k) : x_0 \neq x_1 \text{ and } H(k, x_0) = H(k, x_1)] = \mathsf{negl}(\lambda).$$

Subset-cover set systems. As discussed in Section 1.1, the subset-cover framework introduced by Naor et al. [NNL01] is the basis for many *combinatorial* trace-and-revoke schemes. We provide the formal definition below:

Definition 2.3 (Subset-Cover Set System [NNL01]). *Let N be a positive integer. A subset-cover set system for $[N]$ is a set of indices $[K]$ where $K = \mathsf{poly}(N)$ together with a pair of algorithms $(\mathsf{Encode}, \mathsf{ComputeCover})$ satisfying the following properties:*

- $\mathsf{Encode}(x) \to \mathcal{I}_x$*: On input an element $x \in [N]$, the encoding algorithm outputs a set of indices $\mathcal{I}_x \subseteq [K]$.*
- $\mathsf{ComputeCover}(\mathcal{L}) \to \mathcal{J}_\mathcal{L}$*: On input a revocation list $\mathcal{L} \subseteq [N]$, the cover-computation algorithm outputs a collection of indices $\mathcal{J}_\mathcal{L} \subseteq [K]$.*

We require the following efficiency and security requirements for a subset-cover set system.

- ***Efficiency**: Take any element $x \in [N]$ and any revocation list $\mathcal{L} \subseteq [N]$. Then, $\mathsf{Encode}(x)$ runs in time $\mathsf{poly}(\log N)$ and $\mathsf{ComputeCover}(\mathcal{L})$ runs in time $\mathsf{poly}(|\mathcal{L}|, \log N)$.*
- ***Correctness**: Take any element $x \in [N]$ and revocation list $\mathcal{L} \subseteq [N]$, and let $\mathcal{I}_x \leftarrow \mathsf{Encode}(x)$, $\mathcal{J}_\mathcal{L} \leftarrow \mathsf{ComputeCover}(\mathcal{L})$. Then, $x \in \mathcal{L}$ if and only if $\mathcal{I}_x \cap \mathcal{J}_\mathcal{L} = \varnothing$.*

In this work, we will use the "complete subtree" system from [NNL01, §3.1]. The details of this construction are not essential to our construction, so we omit them and just summarize the main properties below:

Fact 2.4 (Subset-Cover Set System [NNL01, §3.1]). Let N be a positive integer. Then there exists a subset-cover set system $[K]$ for $[N]$ where $K = 2N - 1$, and where the algorithms $(\mathsf{Encode}, \mathsf{ComputeCover})$ satisfy the following properties:

- For all elements $x \in [N]$, if $\mathcal{I}_x \leftarrow \mathsf{Encode}(x)$, then $|\mathcal{I}_x| = \log N + 1$.
- For all revocation lists $\mathcal{L} \subseteq [N]$, if $\mathcal{J}_\mathcal{L} \leftarrow \mathsf{ComputeCover}(\mathcal{L})$, then $|\mathcal{J}_\mathcal{L}| = O(|\mathcal{L}| \log(N/|\mathcal{L}|))$.

The generalized jump-finding problem. Next, we recall the generalized jump-finding problem introduced by Nishimaki et al. [NWZ16, §3.1] for constructing identity-based traitor tracing schemes with succinct ciphertexts. We note that [NWZ16] also introduced a simpler variant of the jump-finding problem that essentially abstracts out the algorithmic core of the traitor tracing construction from private linear broadcast. Here, we consider the generalized version because it enables shorter ciphertexts (where the ciphertext size scales logarithmically with the bit-length of the identities)

Definition 2.5 (Generalized Jump-Finding Problem [NWZ16, Definition 3.9]). *For positive integers $N, r, q \in \mathbb{N}$ and $\delta, \varepsilon > 0$, the $(N, r, q, \delta, \varepsilon)$ generalized jump-finding problem is defined as follows. An adversary begins by choosing a set C of up to q tuples $(s, b_1, \ldots, b_r) \in [N] \times \{0,1\}^r$ where all of the s*

are distinct. Each tuple $(s, b_1, \dots, b_r)$ describes a curve between grid points from the top to bottom of the grid $[1, r] \times [0, 2N]$, which oscillates about the column at position $2s-1$, with $b = (b_1, \dots, b_r)$ specifying which side of the column the curve is on in each row. The curves divide the grid into $|C|+1$ contiguous regions. For each pair $(i, x) \in [1, r] \times [0, 2N]$, the adversary chooses a probability $p_{i,x} \in [0, 1]_{\mathbb{R}}$ with the following properties:

– *For any two pairs $(i, 2x)$, $(j, 2x) \in [1, r] \times [0, 2N]$, it holds that $|p_{i,2x} - p_{j,2x}| < \delta$.*
– *Let $C_i = \{(s, b_1, \dots, b_r) \in C : 2s - b_i\}$ be the set of values $2s - b_i$ for tuples in C. For any two pairs $(i, x), (i, y) \in [1, r] \times [0, 2N]$ such that $(x, y] \cap C_i = \varnothing$, then $|p_{i,x} - p_{i,y}| < \delta$.*
– *For all $i, j \in [r]$, it holds that $p_{i,0} = p_{j,0}$ and $p_{i,2N} = p_{j,2N}$. Define $p_0 = p_{i,0}$ and $p_{2N} = p_{i,2N}$.*
– *Finally, $|p_{2N} - p_0| > \varepsilon$.*

Next, define the oracle $Q \colon [1, r] \times [0, 2N] \to \{0, 1\}$ to be a randomized *oracle that on input (i, x) outputs 1 with probability $p_{i,x}$. Repeated calls to Q on the same input (i, x) will yield a fresh and independently-sampled bit. The $(N, r, q, \delta, \varepsilon)$ generalized jump-finding problem is to output some element in C given oracle access to Q.*

Theorem 2.6 (Generalized Jump-Finding Algorithm [NWZ16, Theorem 3.10]). *There is an efficient algorithm $\mathsf{QTrace}^Q(\lambda, N, r, q, \delta, \varepsilon)$ that runs in time $t = \mathsf{poly}(\lambda, \log N, r, q, 1/\delta)$ and makes at most t queries to Q that solves the $(N, r, q, \delta, \varepsilon)$ generalized jump-finding problem with probability $1 - \mathsf{negl}(\lambda)$ whenever $\varepsilon \geq \delta(9 + 4(\lceil \log N \rceil - 1)q)$. Moreover, any element $(s, b_1, \dots, b_r) \in [N] \times \{0, 1\}^r$ output by QTrace^Q satisfies the following property (with overwhelming probability):*

– *For all $i \in [r]$, $|P(i, 2s - b_i) - P(i, 2s - 1 - b_i)| \geq \delta$, where $P(i, x) := \Pr[Q(i, x) = 1]$.*

Remark 2.7 (Cheating Oracles [NWZ16, Remark 3.8]). The algorithm QTrace^Q from Theorem 2.6 succeeds in solving the $(N, r, q, \delta, \varepsilon)$ generalized jump-finding problem even if the oracle Q does not satisfy all of the requirements in Definition 2.5. As long as the first two properties hold for all pairs (i, x) and (j, y) queried by QTrace^Q, the algorithm succeeds in outputting an element in C.

2.1 Functional Encryption

In this section, we recall the notions of attribute-based encryption (ABE) and mixed functional encryption (mixed FE) that we use in this work.

Mixed FE. A mixed FE scheme [GKW18] is a secret-key FE scheme (i.e., a secret key is needed to encrypt) where ciphertexts are associated with binary-valued functions $f \colon \mathcal{X} \to \{0, 1\}$ and decryption keys are associated with inputs $x \in \mathcal{X}$. When a secret key sk_x associated with an input x is used to decrypt a ciphertext encrypting a message f, the decryption algorithm outputs $f(x)$. The special property in a mixed FE scheme is that there additionally exists a *public-key* encryption algorithm that can be used to encrypt to the "always-accept" function (i.e., the function f where $f(x) = 1$ for all $x \in \mathcal{X}$). Moreover, ciphertexts encrypted using the public key are computationally indistinguishable from ciphertexts produced by using the secret key to encrypt the "always-accept" function. Finally, for our constructions, we require an additional property where the master public key and the master secret key for the mixed FE scheme can be generated *independently*. This means that we can have a family of mixed FE schemes sharing a common set of public parameters. As we discuss in the full version of this paper [KW19a], all existing mixed FE schemes satisfy this requirement.

Definition 2.8 (Mixed Functional Encryption [GKW18]). *A mixed functional encryption scheme Π_{MFE} with domain $\mathcal{X}$ and function family $\mathcal{F} = \{f \colon \mathcal{X} \to \{0,1\}\}$ is a tuple of algorithms $\Pi_{\mathsf{MFE}} = (\mathsf{PrmsGen}, \mathsf{MSKGen}, \mathsf{KeyGen}, \mathsf{PKEnc}, \mathsf{SKEnc}, \mathsf{Dec})$ with the following properties:*

– $\mathsf{PrmsGen}(1^\lambda) \to \mathsf{pp}$*: On input the security parameter λ, the parameter generation algorithm outputs the public parameters* pp.
– $\mathsf{MSKGen}(\mathsf{pp}) \to \mathsf{msk}$*: On input the public parameters* pp*, the master secret key generation algorithm outputs a master secret key* msk.
– $\mathsf{KeyGen}(\mathsf{msk}, x) \to \mathsf{sk}_x$*: On input the master secret key* msk *and an input $x \in \mathcal{X}$, the key-generation algorithm outputs a secret key* sk_x.
– $\mathsf{PKEnc}(\mathsf{pp}) \to \mathsf{ct}$*: On input the public parameters* pp*, the public encryption algorithm outputs a ciphertext* ct.
– $\mathsf{SKEnc}(\mathsf{msk}, f) \to \mathsf{ct}_f$*: On input the master secret key* msk *and a function $f \in \mathcal{F}$, the secret encryption algorithm outputs a ciphertext* ct_f.
– $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to b$*: On input a secret key* sk *and a ciphertext* ct*, the decryption algorithm outputs a bit $b \in \{0,1\}$.*

A mixed FE scheme should satisfy the following properties:

– ***Correctness:*** *For all functions $f \in \mathcal{F}$ and all inputs $x \in \mathcal{X}$, and setting* $\mathsf{pp} \leftarrow \mathsf{PrmsGen}(1^\lambda)$, $\mathsf{msk} \leftarrow \mathsf{MSKGen}(\mathsf{pp})$, $\mathsf{sk}_x \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x)$, $\mathsf{ct} \leftarrow \mathsf{PKEnc}(\mathsf{pp})$, $\mathsf{ct}_f \leftarrow \mathsf{SKEnc}(\mathsf{msk}, f)$*, it follows that*

$$\Pr[\mathsf{Dec}(\mathsf{sk}_x, \mathsf{ct}) = 1] = 1 - \mathsf{negl}(\lambda) \quad \textit{and} \quad \Pr[\mathsf{Dec}(\mathsf{sk}_x, \mathsf{ct}_f) = f(x)] = 1 - \mathsf{negl}(\lambda).$$

– ***Semantic security:*** *For a bit $b \in \{0,1\}$, we define the security experiment* $\mathsf{ExptMFE}_{\mathsf{SS}}[\lambda, \mathcal{A}, b]$ *between a challenger and an adversary $\mathcal{A}$. The challenger begins by sampling* $\mathsf{pp} \leftarrow \mathsf{PrmsGen}(1^\lambda)$, $\mathsf{msk} \leftarrow \mathsf{MSKGen}(\mathsf{pp})$*, and gives* pp *to $\mathcal{A}$. The adversary is then given access to the following oracles:*
  • ***Key-generation oracle:*** *On input $x \in \mathcal{X}$, the challenger replies with* $\mathsf{sk}_x \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x)$.
  • ***Encryption oracle:*** *On input $f \in \mathcal{F}$, the challenger replies with* $\mathsf{ct}_f \leftarrow \mathsf{SKEnc}(\mathsf{msk}, f)$.

  - ***Challenge oracle:*** *On input two functions* $f_0, f_1 \in \mathcal{F}$*, the challenger replies with* $\mathsf{ct} \leftarrow \mathsf{SKEnc}(\mathsf{msk}, f_b)$.

  At the end of the game, the adversary outputs a bit $b' \in \{0,1\}$*, which is also the output of the experiment. An adversary* $\mathcal{A}$ *is admissible for the mixed FE semantic security game if it makes one challenge query* (f_0, f_1)*, and for all inputs* $x \in \mathcal{X}$ *the adversary submits to the key-generation oracle,* $f_0(x) = f_1(x)$*. The mixed FE scheme satisfies (adaptive) semantic security if for all efficient and admissible adversaries* $\mathcal{A}$*,*

$$|\Pr[\mathsf{ExptMFE}_{\mathsf{SS}}[\lambda, \mathcal{A}, 0] = 1] - \Pr[\mathsf{ExptMFE}_{\mathsf{SS}}[\lambda, \mathcal{A}, 1] = 1]| = \mathsf{negl}(\lambda).$$

– ***Public/secret key indistinguishability:*** *For a bit* $b \in \{0,1\}$*, we define the security experiment* $\mathsf{ExptMFE}_{\mathsf{PK/SK}}[\lambda, \mathcal{A}, b]$ *between a challenger and an adversary* $\mathcal{A}$*. The challenger begins by sampling* $\mathsf{pp} \leftarrow \mathsf{PrmsGen}(1^\lambda)$*,* $\mathsf{msk} \leftarrow \mathsf{MSKGen}(\mathsf{pp})$*, and gives* pp *to* $\mathcal{A}$*. The adversary is then given access to the following oracles:*
  - ***Key-generation oracle:*** *On input* $x \in \mathcal{X}$*, the challenger replies with* $\mathsf{sk}_x \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x)$.
  - ***Encryption oracle:*** *On input* $f \in \mathcal{F}$*, the challenger replies with* $\mathsf{ct}_f \leftarrow \mathsf{SKEnc}(\mathsf{msk}, f)$.
  - ***Challenge oracle:*** *On input a function* $f \in \mathcal{F}$*, the challenger computes* $\mathsf{ct}_0 \leftarrow \mathsf{PKEnc}(\mathsf{pp})$ *and* $\mathsf{ct}_1 \leftarrow \mathsf{SKEnc}(\mathsf{msk}, f)$ *and gives* sk_b *to the adversary.*

  At the end of the game, the adversary outputs a bit $b' \in \{0,1\}$*, which is also the output of the experiment. An adversary* $\mathcal{A}$ *is admissible for the public/secret key indistinguishability game if it makes a single challenge query* $f \in \mathcal{F}$ *and for all inputs* $x \in \mathcal{X}$ *the adversary submits to the key-generation oracle,* $f(x) = 1$*. The mixed FE scheme satisfies (adaptive) public/secret key indistinguishability if for all efficient and admissible adversaries* $\mathcal{A}$*, it holds that*

$$\left|\Pr[\mathsf{ExptMFE}_{\mathsf{PK/SK}}[\lambda, \mathcal{A}, 0] = 1] - \Pr[\mathsf{ExptMFE}_{\mathsf{PK/SK}}[\lambda, \mathcal{A}, 1] = 1]\right| = \mathsf{negl}(\lambda).$$

We include additional preliminaries and discussion about mixed FE (e.g., imposing a bound on the number of encryption oracle queries the adversary can make in the security games) in the full version of this paper [KW19a].

3 Revocable Predicate Encryption

In this section, we introduce our notion of a secret-key revocable predicate encryption scheme that supports a public broadcast functionality (i.e., a public-key encryption algorithm that outputs ciphertexts that can be decrypted by all secret keys in the system). This will be the primary primitive we use to construct our identity-based trace-and-revoke scheme (described in Section 4). Our definitions can be viewed as a special case of the more general notion of (public-key) revocable functional encryption from [NWZ16]. The advantage of considering this relaxed notion is that it enables constructions from standard assumptions (whereas we only know how to construct fully secure revocable functional encryption from indistinguishability obfuscation). We introduce our notion below and then show how to construct it by combining mixed FE, ABE, and a subset-cover set system in Section 3.1.

Definition 3.1 (Secret-Key Revocable Predicate Encryption with Broadcast). *A* secret-key revocable predicate encryption scheme (RPE) scheme with broadcast *for an identity space $\mathcal{ID}$, an attribute space $\mathcal{X}$, a function family $\mathcal{F} = \{f : \mathcal{X} \to \{0,1\}\}$, and a message space $\mathcal{M}$ is a tuple of algorithms $\Pi_{\mathsf{RPE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Broadcast}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$*: On input the security parameter λ, the setup algorithm outputs the public parameters* pp *and the master secret key* msk.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, x) \to \mathsf{sk}_{\mathsf{id},x}$*: On input the master secret key* msk*, an identity* $\mathsf{id} \in \mathcal{ID}$*, and an attribute $x \in \mathcal{X}$, the key-generation algorithm outputs a decryption key* $\mathsf{sk}_{\mathsf{id},x}$.
- $\mathsf{Broadcast}(\mathsf{pp}, m, \mathcal{L}) \to \mathsf{ct}_{m,\mathcal{L}}$*: On input the public key, a message m, and a revocation list $\mathcal{L} \subseteq \mathcal{ID}$, the broadcast algorithm outputs a ciphertext* $\mathsf{ct}_{m,\mathcal{L}}$.
- $\mathsf{Enc}(\mathsf{msk}, f, m, \mathcal{L}) \to \mathsf{ct}_{f,m,\mathcal{L}}$*: On input the master secret key* msk*, a function $f \in \mathcal{F}$, a message $m \in \mathcal{M}$, and a revocation list $\mathcal{L} \subseteq \mathcal{ID}$, the encryption algorithm outputs a ciphertext* $\mathsf{ct}_{f,m,\mathcal{L}}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m/\bot$*: On input a decryption key* sk *and a ciphertext* ct*, the decryption algorithm either outputs a message $m \in \mathcal{M}$ or a special symbol $\bot$.*

A secret-key RPE scheme with broadcast should satisfy the following properties:

- ***Correctness:*** *For all functions $f \in \mathcal{F}$, all identities* $\mathsf{id} \in \mathcal{ID}$*, all attributes $x \in \mathcal{X}$ where $f(x) = 1$, all messages $m \in \mathcal{M}$, and all revocation lists $\mathcal{L} \subseteq \mathcal{ID}$ where* $\mathsf{id} \notin \mathcal{L}$*, if we set* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$*,* $\mathsf{sk}_{\mathsf{id},x} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, x)$*, the following holds:*
  - ***Broadcast correctness:*** *If* $\mathsf{ct}_{m,\mathcal{L}} \leftarrow \mathsf{Broadcast}(\mathsf{pp}, m, \mathcal{L})$*, then*
  $$\Pr[\mathsf{Dec}(\mathsf{sk}_{\mathsf{id},x}, \mathsf{ct}_{m,\mathcal{L}}) = m] = 1 - \mathsf{negl}(\lambda).$$
  - ***Encryption correctness:*** *If* $\mathsf{ct}_{f,m,\mathcal{L}} \leftarrow \mathsf{Enc}(\mathsf{msk}, f, m, \mathcal{L})$*, then*
  $$\Pr[\mathsf{Dec}(\mathsf{sk}_{\mathsf{id},x}, \mathsf{ct}_{f,m,\mathcal{L}}) = m] = 1 - \mathsf{negl}(\lambda).$$
- ***Message hiding:*** *For a bit $b \in \{0,1\}$, we define the experiment* $\mathsf{ExptRPE}_{\mathsf{MH}}[\lambda, \mathcal{A}, b]$ *between a challenger and an adversary $\mathcal{A}$. The challenger begins by sampling* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and gives* pp *to $\mathcal{A}$. The adversary is then given access to the following oracles:*
  - ***Key-generation oracle:*** *On input an identity* $\mathsf{id} \in \mathcal{ID}$ *and an attribute $x \in \mathcal{X}$, the challenger replies with* $\mathsf{sk}_{\mathsf{id},x} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, x)$.
  - ***Encryption oracle:*** *On input a function $f \in \mathcal{F}$, a message $m \in \mathcal{M}$, and a revocation list $\mathcal{L} \subseteq \mathcal{ID}$, the challenger replies with* $\mathsf{ct}_{f,m,\mathcal{L}} \leftarrow \mathsf{Enc}(\mathsf{msk}, f, m, \mathcal{L})$.

- ***Challenge oracle:*** *On input a function* $f \in \mathcal{F}$*, two messages* $m_0, m_1 \in \mathcal{M}$*, and a revocation list* $\mathcal{L} \subseteq \mathcal{ID}$*, the challenger computes* $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{msk}, f, m_b, \mathcal{L})$ *and gives* ct_b *to the adversary.*

At the end of the game, the adversary outputs a bit $b' \in \{0,1\}$*, which is the output of the experiment. An adversary* $\mathcal{A}$ *is admissible for the message hiding game if it makes a single challenge query* $(f, m_0, m_1, \mathcal{L})$ *such that for all pairs* (id, x) *the adversary submitted to the key-generation oracle, it holds that* $f(x) = 0$ *or* $\mathsf{id} \in \mathcal{L}$*. We say that* Π_{RPE} *satisfies (adaptive)* message hiding *if for all efficient and admissible adversaries* $\mathcal{A}$*,*

$$|\Pr[\mathsf{ExptRPE}_{\mathsf{MH}}[\lambda, \mathcal{A}, 0] = 1] - \Pr[\mathsf{ExptRPE}_{\mathsf{MH}}[\lambda, \mathcal{A}, 1] = 1]| = \mathsf{negl}(\lambda).$$

– ***Function hiding:*** *For a bit* $b \in \{0,1\}$*, we define the experiment* $\mathsf{ExptRPE}_{\mathsf{FH}}[\lambda, \mathcal{A}, b]$ *between a challenger and an adversary* $\mathcal{A}$ *exactly as* $\mathsf{ExptRPE}_{\mathsf{MH}}[\lambda, \mathcal{A}, b]$*, except the challenge oracle is replaced with the following:*
  - ***Challenge oracle:*** *On input two functions* $f_0, f_1 \in \mathcal{F}$*, a message* $m \in \mathcal{M}$*, and a revocation list* $\mathcal{L} \subseteq \mathcal{ID}$*, the challenger computes* $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{msk}, f_b, m, \mathcal{L})$ *and gives* ct_b *to the adversary.*

  We say an adversary $\mathcal{A}$ *is admissible for the function-hiding game if it makes a single challenge query* $(f_0, f_1, m, \mathcal{L})$ *such that for all pairs* (id, x) *the adversary submitted to the key-generation oracle, either* $f_0(x) = f_1(x)$ *or* $\mathsf{id} \in \mathcal{L}$*. We say that* Π_{RPE} *satisfies (adaptive)* function hiding *if for all efficient and admissible adversaries* $\mathcal{A}$*,*

$$|\Pr[\mathsf{ExptRPE}_{\mathsf{FH}}[\lambda, \mathcal{A}, 0] = 1] - \Pr[\mathsf{ExptRPE}_{\mathsf{FH}}[\lambda, \mathcal{A}, 1] = 1]| = \mathsf{negl}(\lambda).$$

– ***Broadcast security:*** *For a bit* $b \in \{0,1\}$*, we define the security experiment* $\mathsf{ExptRPE}_{\mathsf{BC}}[\lambda, \mathcal{A}, b]$ *between a challenger and an adversary* $\mathcal{A}$ *exactly as* $\mathsf{ExptRPE}_{\mathsf{MH}}[\lambda, \mathcal{A}, b]$*, except the challenge oracle is replaced with the following:*
  - ***Challenge oracle:*** *On input a message* $m \in \mathcal{M}$ *and a revocation list* $\mathcal{L} \subseteq \mathcal{ID}$*, the challenger computes* $\mathsf{ct}_0 \leftarrow \mathsf{Broadcast}(\mathsf{pp}, m, \mathcal{L})$ *and* $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{msk}, f, m, \mathcal{L})$ *where* f_{accept} *is the "always-accept" function (i.e.,* $f_{\mathsf{accept}}(x) = 1$ *for all* $x \in \mathcal{X}$*). It gives* ct_b *to the adversary.*

  At the end of the game, the adversary outputs a bit $b' \in \{0,1\}$*, which is the output of the experiment. We say that* Π_{RPE} *satisfies (adaptive)* broadcast security *if for all efficient adversaries* $\mathcal{A}$ *that make at most one challenge query,*

$$|\Pr[\mathsf{ExptRPE}_{\mathsf{BC}}[\lambda, \mathcal{A}, b] = 1] - \Pr[\mathsf{ExptRPE}_{\mathsf{BC}}[\lambda, \mathcal{A}, 1]]| = \mathsf{negl}(\lambda).$$

Remark 3.2 (Non-Adaptive q-Query Security). For each of the security notions in Definition 3.1 (message hiding, function hiding, and broadcast security), we define a notion of *non-adaptive q-query security* where the corresponding security notion only holds against all adversaries that make at most $q \in \mathbb{N}$ queries to the encryption oracle, and moreover, all of the non-encryption queries occur *before* the encryption queries. Achieving this notion is easier and suffices for our main construction (*adaptively-secure* trace-and-revoke).

Remark 3.3 (Embedding the Revocation List in the Attribute). A natural approach for constructing a revocable predicate encryption scheme from any vanilla predicate encryption scheme is to include the revocation list $\mathcal{L}$ as part of the function in the predicate encryption scheme. A decryption key for an identity id would then check that id is not contained in the revocation list $\mathcal{L}$ associated with the ciphertext. This is the approach suggested in [NWZ16, Remark 6.2] in the context of constructing a revocable functional encryption scheme. While this approach may seem straightforward, it has a significant drawback in most settings. In existing predicate encryption schemes schemes from standard assumptions, the decryption functionality is represented as a *circuit*, which takes *fixed-size* inputs. Thus, if the revocation list is embedded as part of the ciphertext, then a predicate encryption scheme for circuit-based predicates would only be able to support an *a priori* bounded number of revocations. In contrast, the our construction allows for revoking an *arbitrary* polynomial number of users (Section 3.1). Of course, if we can construct predicate or functional encryption for Turing machine or RAM computations, then this natural revocation approach would suffice. Existing constructions of functional encryption for Turing machine computations all rely on indistinguishability obfuscation [KLW15, AJS17, AS16, GS18].

3.1 Constructing Secret-Key Revocable Predicate Encryption with Broadcast

In this section, we describe our construction of a secret-key revocable predicate encryption with broadcast scheme for general predicates by combining a mixed FE scheme, an ABE scheme, and a subset-cover set system. As discussed in 1.1, our core construction (without revocation) can be viewed as a direct generalization of the construction of private linear broadcast encryption from mixed FE and ABE from [GKW18]. We next augment our construction with a subset cover set system to support revocation. Our techniques allow revoking an arbitrary number of users (in contrast to previous trace-and-revoke schemes from standard assumptions that could only handle bounded revocations [NWZ16, ABP+17]). We give our full construction and its analysis below:

Construction 3.4 (Secret-Key Revocable Predicate Encryption with Broadcast). Fix an identity space $\mathcal{ID} = \{0,1\}^n$, attribute space $\mathcal{X}$, function family $\mathcal{F} = \{f \colon \mathcal{X} \to \{0,1\}\}$ and message space $\mathcal{M}$, where $n = n(\lambda)$.

– Let $[K]$ be the subset-cover set system for the set $\mathcal{ID} = \{0,1\}^n$. Let $\Pi_{\mathsf{SC}} = (\mathsf{Encode}, \mathsf{ComputeCover})$ be the algorithms associated with the set system.
– Let $\Pi_{\mathsf{MFE}} = (\mathsf{MFE.PrmsGen}, \mathsf{MFE.MSKGen}, \mathsf{MFE.KeyGen}, \mathsf{MFE.PKEnc}, \mathsf{MFE.SKEnc}, \mathsf{MFE.Dec})$ be a mixed FE scheme with domain $\mathcal{X}$ and function family $\mathcal{F}$. Let $\rho = \rho(\lambda)$ be the randomness complexity of the master secret key generation algorithm $\mathsf{MFE.MSKGen}$, let $\mathcal{CT}$ denote the ciphertext space of Π_{MFE} (i.e., the range of $\mathsf{MFE.PKEnc}$ and $\mathsf{MFE.SKEnc}$), and let $\mathcal{SK}$ denote the secret key space of Π_{MFE} (i.e., the range of $\mathsf{MFE.KeyGen}$). We will require that Π_{MFE} be sub-exponentially secure, so let $\varepsilon > 0$ be a constant such that $2^{-\Omega(\lambda^\varepsilon)}$ bounds the advantage of any efficient adversary $\mathcal{A}$ for the security of Π_{MFE}.

- For a secret key $\mathsf{mfe.sk} \in \mathcal{SK}$ and an index $i^* \in [K]$, define the function $g_{\mathsf{mfe.sk},i^*} : \mathcal{CT} \times [K] \to \{0,1\}$ to be the function

$$g_{\mathsf{mfe.sk},i^*}(\mathsf{ct}, i) = \begin{cases} 1 & \mathsf{MFE.Dec}(\mathsf{mfe.sk}, \mathsf{ct}) = 1 \text{ and } i = i^* \\ 0 & \text{otherwise.} \end{cases}$$

- Let $\Pi_{\mathsf{ABE}} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ be an attribute-based encryption scheme over message space $\mathcal{M}$, attribute space $\mathcal{X}' = \mathcal{CT} \times [K]$ and function family $\mathcal{F}' = \{\mathsf{mfe.sk} \in \mathcal{SK}, i^* \in [K] : g_{\mathsf{mfe.sk},i^*}\}$.
- Let $F : \mathcal{K} \times [K] \to \{0,1\}^\rho$ be a pseudorandom function.

We construct a secret-key revocable predicate encryption scheme as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm sets $\lambda' = \max(\lambda, (\log K)^{2/\varepsilon})$. It then generates mixed FE public parameters $\mathsf{mfe.pp} \leftarrow \mathsf{MFE.PrmsGen}(1^{\lambda'})$. It also instantiates an attribute-based encryption scheme $(\mathsf{abe.pp}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, samples a PRF key $k \xleftarrow{\text{R}} \mathcal{K}$, and outputs

$$\mathsf{pp} = (\mathsf{mfe.pp}, \mathsf{abe.pp}) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \mathsf{abe.msk}, k).$$

- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, x)$: On input a master secret key msk, an identity $\mathsf{id} \in \mathcal{ID}$, and an attribute $x \in \mathcal{X}$, the key-generation algorithm does the following:
  1. Compute a subset-cover encoding of the identity $\mathcal{I}_{\mathsf{id}} \leftarrow \mathsf{Encode}(\mathsf{id})$.
  2. For each index $i \in \mathcal{I}_{\mathsf{id}}$, the algorithm samples randomness $r_i \leftarrow F(k, i)$. It then generates a mixed FE master secret key $\mathsf{mfe.msk}_i \leftarrow \mathsf{MFE.MSKGen}(\mathsf{mfe.pp}; r_i)$ and a mixed FE decryption key $\mathsf{mfe.sk}_{i,x} \leftarrow \mathsf{MFE.KeyGen}(\mathsf{mfe.msk}_i, x)$.
  3. Finally, for each $i \in \mathcal{I}_{\mathsf{id}}$, it constructs an ABE decryption key with respect to the function $g_{\mathsf{mfe.msk}_{i,x},i}$ as follows: $\mathsf{abe.sk}_{i,x} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, g_{\mathsf{mfe.sk}_{i,x},i})$.
  4. It outputs the collection of keys $\mathsf{sk}_{\mathsf{id},x} = \{(i, \mathsf{abe.sk}_{i,x})\}_{i \in \mathcal{I}_{\mathsf{id}}}$.
- $\mathsf{Broadcast}(\mathsf{pp}, m, \mathcal{L})$: On input the public parameters $\mathsf{pp} = (\mathsf{mfe.pp}, \mathsf{abe.pp})$, a message m, and a revocation list $\mathcal{L} \subseteq \mathcal{ID}$, the broadcast algorithm does the following:
  1. Obtain a cover for $\mathcal{ID} \setminus \mathcal{L}$ by computing $\mathcal{J}_{\mathcal{L}} \leftarrow \mathsf{ComputeCover}(\mathcal{L})$.
  2. For each $i \in \mathcal{J}_{\mathcal{L}}$, it generates a mixed FE ciphertext $\mathsf{mfe.ct}_i \leftarrow \mathsf{MFE.PKEnc}(\mathsf{mfe.pp})$ and an ABE ciphertext $\mathsf{abe.ct}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, (\mathsf{mfe.ct}_i, i), m)$.
  3. It outputs the ciphertext $\mathsf{ct}_{m,\mathcal{L}} = \{(i, \mathsf{abe.ct}_i)\}_{i \in \mathcal{J}_{\mathcal{L}}}$.
- $\mathsf{Enc}(\mathsf{msk}, f, m, \mathcal{L})$: On input the master secret key $\mathsf{msk} = (\mathsf{pp}, \mathsf{abe.msk}, k)$, a function $f \in \mathcal{F}$, a message $m \in \mathcal{M}$, and a revocation list $\mathcal{L} \subseteq \mathcal{ID}$, where $\mathsf{pp} = (\mathsf{mfe.pp}, \mathsf{abe.pp})$, the encryption algorithm does the following:
  1. Obtain a cover for $\mathcal{ID} \setminus \mathcal{L}$ by computing $\mathcal{J}_{\mathcal{L}} \leftarrow \mathsf{ComputeCover}(\mathcal{L})$.
  2. Then, for each $i \in \mathcal{J}_{\mathcal{L}}$, it computes $r_i \leftarrow F(k, i)$ and derives the corresponding mixed FE master secret key $\mathsf{mfe.msk}_i \leftarrow \mathsf{MFE.MSKGen}(\mathsf{mfe.pp}; r_i)$. It then encrypts $\mathsf{mfe.ct}_i \leftarrow \mathsf{MFE.SKEnc}(\mathsf{mfe.msk}_i, f)$.
  3. For each $i \in \mathcal{J}_{\mathcal{L}}$, it computes $\mathsf{abe.ct}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, (\mathsf{mfe.ct}_i, i), m)$, and outputs the ciphertext $\mathsf{ct}_{f,m,\mathcal{L}} = \{(i, \mathsf{abe.ct}_i)\}_{i \in \mathcal{J}_{\mathcal{L}}}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: On input a key $\mathsf{sk} = \{(i, \mathsf{abe.sk}_i)\}_{i \in \mathcal{I}}$ and a ciphertext $\mathsf{ct} = \{(i, \mathsf{abe.ct}_i)\}_{i \in \mathcal{J}}$, the decryption algorithm first checks if $\mathcal{I} \cap \mathcal{J} = \varnothing$. If so, it outputs $\perp$. Otherwise, it chooses an arbitrary index $i \in \mathcal{I} \cap \mathcal{J}$ and outputs $m \leftarrow \mathsf{ABE.Dec}(\mathsf{abe.sk}_i, \mathsf{abe.ct}_i)$.

Correctness and security analysis. We state our main theorems on the properties of Construction 3.4 below, but defer their analysis to the full version of this paper [KW19a].

Theorem 3.5 (Correctness). *Suppose that Π_{MFE}, Π_{ABE}, and Π_{SC} are correct. Then, the predicate encryption scheme Π_{RPE} from Construction 3.4 is correct.*

Theorem 3.6 (Message Hiding). *Suppose that Π_{MFE} and Π_{SC} are correct, and Π_{ABE} satisfies semantic security. Then, the predicate encryption scheme Π_{RPE} from Construction 3.4 satisfies message hiding.*

Theorem 3.7 (Function Hiding). *Suppose that Π_{MFE} satisfies sub-exponential non-adaptive q-query (resp., adaptive) semantic security. Specifically, suppose that the advantage of any adversary running in time $\mathsf{poly}(\lambda)$ in the semantic security game is bounded by $2^{-\Omega(\lambda^{\varepsilon})}$. In addition, suppose that Π_{ABE} is secure, F is a secure PRF, and Π_{SC} is correct. Then, the predicate encryption scheme in Construction 3.4 satisfies non-adaptive q-query (resp., adaptive) function hiding security.*

Theorem 3.8 (Broadcast Security). *Suppose that Π_{MFE} satisfies sub-exponential non-adaptive q-query (resp., adaptive) public/secret key indistinguishability. Specifically, suppose that the advantage of any adversary running in time $\mathsf{poly}(\lambda)$ in the public/secret key indistinguishability game is bounded by $2^{-\Omega(\lambda^{\varepsilon})}$. In addition, suppose that F is a secure PRF. Then the predicate encryption scheme Π_{RPE} in Construction 3.4 satisfies non-adaptive q-query (resp., adaptive) broadcast security.*

3.2 Instantiating Secret-Key Revocable Predicate Encryption with Broadcast

In this section, we describe one possible instantiation of secret-key revocable predicate encryption with broadcast from Construction 3.4. In particular, combining Construction 3.4 with Theorems 3.5 through 3.8 yields the following corollary:

Corollary 3.9 (Secret-Key Revocable Predicate Encryption from LWE). *Take an identity-space $\mathcal{ID} = \{0,1\}^n$, attribute space $\mathcal{X} = \{0,1\}^{\ell}$, and message space $\mathcal{M} = \{0,1\}^t$ where $n = n(\lambda)$, $\ell = \ell(\lambda)$, and $t = t(\lambda)$. Let $\mathcal{F} = \{f : \mathcal{X} \to \{0,1\}\}$ be a function family where every function $f \in \mathcal{F}$ can be specified by a string of length $z = z(\lambda)$ and computed by a Boolean circuit of depth $d = d(\lambda)$. Then, assuming sub-exponential hardness of LWE (with a super-polynomial modulus-to-noise ratio), there exists a non-adaptive 1-key secure secret-key revocable predicate encryption scheme with broadcast Π_{RPE} over the identity space $\mathcal{ID}$, attribute space $\mathcal{X}$, and function family $\mathcal{F}$. Moreover, Π_{RPE} satisfies the following properties:*

- ***Public parameter size:*** $|\mathsf{pp}| = \ell \cdot \mathsf{poly}(\lambda, d, n, z)$.
- ***Secret key size:*** *The secret key $\mathsf{sk}_{\mathsf{id},x}$ for an identity $\mathsf{id} \in \{0,1\}^n$ and an attribute $x \in \{0,1\}^{\ell}$ has size $|\mathsf{sk}_{\mathsf{id},x}| = \ell + \mathsf{poly}(\lambda, d, n, z)$.*

- ***Ciphertext size:*** *An encryption* $\mathsf{ct}_{m,\mathcal{L}}$ *of a message* $m \in \{0,1\}^t$ *with revocation list* $\mathcal{L}$ *has size* $|\mathsf{ct}_{m,\mathcal{L}}| = t + |\mathcal{L}| \cdot \mathsf{poly}(\lambda, d, n, z)$.

Proof. We instantiate Construction 3.4 using the subset-cover set system from Fact 2.4, the mixed FE scheme using the construction of Chen et al. [CVW$^+$18], the ABE scheme using the construction of Boneh et al. [BGG$^+$14], and the PRF from any one-way function [GGM84]. We describe the exact instantiations in greater detail in the full version of this paper [KW19a]. The mixed FE scheme is instantiated with domain $\mathcal{X} = \{0,1\}^\ell$ and function family $\mathcal{F}$, while the ABE scheme is instantiated with message space $\mathcal{M}$, attribute space $\mathcal{X}' = \mathcal{CT} \times [K]$ and function family $\mathcal{F}' = \{\mathsf{mfe.sk} \in \mathcal{SK}, i^* \in [K] : g_{\mathsf{mfe.sk},i^*}\}$. We will use the following bounds in our analysis:

- From Fact 2.4, we have that $K = O(N)$, and correspondingly, $\log K = O(\log N) = O(n)$.
- We have that the length of a mixed FE ciphertext $\mathsf{mfe.ct} \in \mathcal{CT}$ is bounded by $|\mathsf{mfe.ct}| = \mathsf{poly}(\lambda, d, z)$. Correspondingly, this means that the length ℓ_{ABE} of an ABE attribute is bounded by $\ell_{\mathsf{ABE}} = \mathsf{poly}(\lambda, d, z) + \log K = \mathsf{poly}(\lambda, d, n, z)$.
- Each function $g_{\mathsf{mfe.sk},i^*}$ can be implemented by a circuit with depth at most $\mathsf{poly}(\lambda, d) + \log\log K = \mathsf{poly}(\lambda, d, \log n)$. Specifically, the mixed FE decryption circuit can be evaluated by a circuit of depth $\mathsf{poly}(\lambda', d) = \mathsf{poly}(\lambda, d, n, z)$ and the equality-check circuit can be evaluated by a circuit of depth $\log\log K$ (since each input to the equality-check circuit is a $(\log K)$-bit value). Thus, the functions in $\mathcal{F}'$ can be computed by Boolean circuits with depth at most $d_{\mathsf{ABE}} \le \mathsf{poly}(\lambda, d, n, z)$. The description length of functions in $\mathcal{F}'$ is $|\mathsf{mfe.sk}| + \log K = \ell + \mathsf{poly}(\lambda, n, z)$.

Putting all the pieces together, we now have the following:

- **Public parameter size:** The public parameters pp consist of the ABE public parameters $\mathsf{abe.pp}$ and the mixed FE public parameters $\mathsf{mfe.pp}$. Then,
$$|\mathsf{abe.pp}| = \mathsf{poly}(\lambda, d_{\mathsf{ABE}}, \ell_{\mathsf{ABE}}) = \mathsf{poly}(\lambda, d, n, z),$$
and correspondingly,
$$|\mathsf{mfe.pp}| = \ell \cdot \mathsf{poly}(\lambda', d, z) = \ell \cdot \mathsf{poly}(\lambda, d, n, z),$$
since $\lambda' = \mathsf{poly}(\lambda, \log K) = \mathsf{poly}(\lambda, n)$. Thus, $|\mathsf{pp}| = \ell \cdot \mathsf{poly}(\lambda, d, n, z)$.
- **Secret key size:** The secret key $\mathsf{sk}_{\mathsf{id},x} = \{(i, \mathsf{abe.sk}_{i,x})\}_{i \in \mathcal{I}_{\mathsf{id}}}$ for an identity id and attribute x consists of $|\mathcal{I}_{\mathsf{id}}|$ ABE secret keys, where $|\mathcal{I}_{\mathsf{id}}| \leftarrow \mathsf{Encode}(\mathsf{id})$. By Fact 2.4, $|\mathcal{I}_{id}| = \log N + 1 = \mathsf{poly}(n)$. Finally,
$$|\mathsf{abe.sk}_{i,x}| = \left|g_{\mathsf{mfe.sk}_{i,x},i}\right| + \mathsf{poly}(\lambda, d_{\mathsf{ABE}}, \ell_{\mathsf{ABE}}) = \ell + \mathsf{poly}(\lambda, d, n, z).$$
Thus, $|\mathsf{sk}_{\mathsf{id},x}| = |\mathcal{I}_{\mathsf{id}}| \cdot |\mathsf{abe.sk}_{i,x}| = \ell + \mathsf{poly}(\lambda, d, n, z)$.
- **Ciphertext size:** Without loss of generality, we can always use hybrid encryption for the ciphertexts. Namely, the encryption algorithm samples a symmetric key k to encrypt the message and then encrypts k using the

secret-key revocable predicate encryption scheme. The final ciphertext $\mathsf{ct}_{m,\mathcal{L}}$ then consists of a symmetric encryption of the message m (which has size $|m| + \mathsf{poly}(\lambda)$) and a revocable predicate encryption ciphertext $\widehat{\mathsf{ct}}$ of the key k. In this case, $|k| = \mathsf{poly}(\lambda)$, and the overall ciphertext size is $|\mathsf{ct}| = |m| + \mathsf{poly}(\lambda) + |\widehat{\mathsf{ct}}|$, where $\widehat{\mathsf{ct}} = \{(i, \mathsf{abe.ct}_i)\}_{i \in \mathcal{J}_\mathcal{L}}$ is an encryption of k using Π_{RPE}. By construction, $\widehat{\mathsf{ct}}$ consists of $|\mathcal{J}_\mathcal{L}|$ ABE ciphertexts, where $\mathcal{J}_\mathcal{L} \leftarrow \mathsf{ComputeCover}(\mathcal{L})$. By Fact 2.4, $|\mathcal{L}| = O(|\mathcal{L}| \log(N/|\mathcal{L}|)) = |\mathcal{L}| \cdot \mathsf{poly}(n)$. Finally, $|\mathsf{abe.ct}_i| = |k| + \ell_{\mathsf{ABE}} \cdot \mathsf{poly}(\lambda, d_{\mathsf{ABE}}, \ell_{\mathsf{ABE}}) = \mathsf{poly}(\lambda, d, n, z)$, and so

$$|\mathsf{ct}_{m,\mathcal{L}}| = |m| + \mathsf{poly}(\lambda) + |\widehat{\mathsf{ct}}| = t + |\mathcal{L}| \cdot \mathsf{poly}(\lambda, d, n, z). \square$$

Remark 3.10 (Handling More General Revocation Policies). Construction 3.4 naturally supports any revocation policy that can be described by a polynomial-size cover in the underlying subset-cover set system. In particular, the prefix-based subset-cover set system by Naor et al. [NNL01] from Fact 2.4 can compute a cover that excludes any polynomial number of *prefixes* (in addition to full identities). For instance, we can use the set system to revoke all users whose identities start with "000" or "01" (i.e., revoke all identities of the form 000*** and 01****). This way, the number of revoked users in the set $\mathcal{L}$ can be *exponential*, as long as they can be described by a polynomial-number of prefix-based clusters. Correspondingly, the traitor tracing scheme we construct in Section 4 will also support these types of revocation policies.

4 Identity-Based Trace-and-Revoke

In this section, we describe how to construct an identity-based trace-and-revoke scheme using a secret-key revocable predicate encryption scheme with broadcast (Definition 3.1). We begin by recalling the formal definition of a trace-and-revoke scheme. Our definitions are adapted from the corresponding ones in [BW06, NWZ16]. As we discuss in greater detail in Remark 4.2, our definition combines aspects of both definitions and is strictly stronger than both of the previous notions.

Definition 4.1 (Trace-and-Revoke [NWZ16, adapted]). *A trace-and-revoke scheme for a set of identities $\mathcal{ID}$ and a message space $\mathcal{M}$ is a tuple of algorithms $\Pi_{\mathsf{TR}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ defined as follows:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$*: On input the security parameter λ, the setup algorithm outputs the public parameters* pp *and the master secret key* msk.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$*: On input the master secret key* msk *and an identity* $\mathsf{id} \in \mathcal{ID}$*, the key-generation algorithm outputs a secret key* $\mathsf{sk}_{\mathsf{id}}$.
- $\mathsf{Enc}(\mathsf{pp}, m, \mathcal{L}) \to \mathsf{ct}_{m,\mathcal{L}}$*: On input the public parameters* pp*, a message $m \in \mathcal{M}$, and a list of revoked users $\mathcal{L} \subseteq \mathcal{ID}$, the encryption algorithm outputs a ciphertext* $\mathsf{ct}_{m,\mathcal{L}}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m/\bot$*: On input a decryption key* sk *and a ciphertext* ct*, the decryption algorithm either outputs a message $m \in \mathcal{M}$ or a special symbol $\bot$.*

– $\mathsf{Trace}^{\mathcal{D}}(\mathsf{msk}, m_0, m_1, \mathcal{L}, \varepsilon) \to \mathsf{id}/\perp$*: On input the master secret key* msk*, two messages* $m_0, m_1 \in \mathcal{M}$*, a revocation list* $\mathcal{L} \subseteq \mathcal{ID}$*, a decoder-success parameter* $\varepsilon > 0$*, and assuming oracle access to a decoder algorithm* $\mathcal{D}$*, the tracing algorithm either outputs an identity* $\mathsf{id} \in \mathcal{ID}$ *or* $\perp$*.*

Moreover, a trace-and-revoke scheme should satisfy the following properties:

– ***Correctness:*** *For all messages* $m \in \mathcal{M}$*, all identities* $\mathsf{id} \in \mathcal{ID}$*, and all revocation lists* $\mathcal{L} \subseteq \mathcal{ID}$ *where* $\mathsf{id} \notin \mathcal{L}$*, if we set* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$*,* $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$*, and* $\mathsf{ct}_{m,\mathcal{L}} \leftarrow \mathsf{Enc}(\mathsf{pp}, m, \mathcal{L})$*, then*

$$\Pr[\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}_{m,\mathcal{L}}) = m] = 1 - \mathsf{negl}(\lambda).$$

– ***Semantic Security:*** *For a bit* $b \in \{0,1\}$*, we define the security experiment* $\mathsf{ExptTR}_{\mathsf{SS}}[\lambda, \mathcal{A}, b]$ *between a challenger and an adversary* $\mathcal{A}$*. The challenger begins by sampling* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and gives* pp *to* $\mathcal{A}$*. The adversary is then given access to the following oracles:*
  - ***Key-generation oracle.*** *On input an identity* $\mathsf{id} \in \mathcal{ID}$*, the challenger replies with* $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$*.*
  - ***Challenge oracle.*** *On input two messages* $m_0, m_1 \in \mathcal{M}$ *and a revocation list* $\mathcal{L} \subseteq \mathcal{ID}$*, the challenger replies with* $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pp}, m_b, \mathcal{L})$*.*

  At the end of the game, the adversary outputs a bit $b' \in \{0,1\}$*, which is the output of the experiment. An adversary* $\mathcal{A}$ *is admissible for the semantic security game if it makes a single challenge query* $(m_0, m_1, \mathcal{L})$*, and moreover, for all key-generation queries* id *the adversary makes,* $\mathsf{id} \in \mathcal{L}$*. We say that* Π_{TR} *is semantically secure if for all efficient and admissible adversaries* $\mathcal{A}$*,*

$$|\Pr[\mathsf{ExptTR}_{\mathsf{SS}}[\lambda, \mathcal{A}, 0] = 1] - \Pr[\mathsf{ExptTR}_{\mathsf{SS}}[\lambda, \mathcal{A}, 1] = 1]| = \mathsf{negl}(\lambda).$$

– ***Traceability:*** *We define the experiment* $\mathsf{ExptTR}_{\mathsf{TR}}[\lambda, \mathcal{A}]$ *between a challenger and an adversary* $\mathcal{A}$*. The challenger begins by sampling* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and gives* pp *to* $\mathcal{A}$*. The adversary is then given access to the key-generation oracle:*
  - ***Key-generation oracle.*** *On input an identity* $\mathsf{id} \in \mathcal{ID}$*, the challenger replies with* $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$*.*

  At the end of the game, the adversary outputs a decoder algorithm $\mathcal{D}$*, two messages* $m_0, m_1 \in \mathcal{M}$*, a revocation list* $\mathcal{L} \subseteq \mathcal{ID}$*, and a non-negligible decoder-success probability* $\varepsilon > 0$*. Let* $\mathcal{R} \subseteq \mathcal{ID}$ *be the set of identities the adversary submitted to the key-generation oracle and let* $\mathsf{id}^* \leftarrow \mathsf{Trace}^{\mathcal{D}}(\mathsf{msk}, m_0, m_1, \mathcal{L}, \varepsilon)$*. Then the output of the experiment is* 1 *if* $\mathsf{id}^* \notin \mathcal{R} \setminus \mathcal{L}$ *and* 0 *otherwise. We say that an adversary* $\mathcal{A}$ *is admissible for the traceability game if the decoder algorithm output by* $\mathcal{A}$ *satisfies*

$$\Pr[b \xleftarrow{\text{R}} \{0,1\} : \mathcal{D}(\mathsf{Enc}(\mathsf{pp}, m_b, \mathcal{L})) = b] \geq 1/2 + \varepsilon.$$

  Finally, we say that Π_{TR} *satisfies traceability security if for all efficient and admissible adversaries* $\mathcal{A}$*,*

$$\Pr[\mathsf{ExptTR}_{\mathsf{TR}}[\lambda, \mathcal{A}] = 1] = \mathsf{negl}(\lambda).$$

Remark 4.2 (Comparison to Previous Traceability Notions). Our notion of traceability in Definition 4.1 combines aspects of the notions considered in [BW06] and [NWZ16] and is stronger than both of these previous definitions. First, similar to [NWZ16], we only require that the decoder $\mathcal{D}$ output by $\mathcal{A}$ to be able to distinguish *the encryptions of two adversarially-chosen messages. The previous notion in [BW06] made the more stringent requirement that the adversary's decoder must correctly decrypt a noticeable fraction of ciphertexts. Thus, our definitions enable tracing for much weaker decoders. Next, and similar to [BW06], our tracing definition naturally incorporates revocation. Namely, if an adversary constructs a decoder that is able to distinguish encryptions of two messages with respect to a revocation list $\mathcal{L}$, then the tracing algorithm must identify a compromised key that is outside $\mathcal{L}$. In contrast, the definition in [NWZ16] only considered tracing in a standalone setting: namely, while the scheme supports revocation, the tracing definition only considered decoders that can decrypt ciphertexts encrypted to an* empty *revocation list. Overall, our definition is stronger than the previous definitions and we believe provides a more realistic modeling of the security demands in applications of trace-and-revoke systems.*

Remark 4.3 (Adaptive Security). We note that all of the security requirements in Definition 4.1 are adaptive: namely, the adversary chooses its challenge messages and revocation list after seeing the public parameters and (adaptively-chosen) secret decryption keys. Our final construction is fully adaptive (Construction 4.4, Corollary 4.8), but we do rely on complexity leveraging and sub-exponential hardness assumptions. We remark here that a selective notion of security where the adversary commits to its revocation list ahead of time does not seem to directly imply adaptive security by the usual complexity leveraging technique [BB04] unless we additionally impose an a priori bound on the size of the revocation list (which we do not require in our analysis). It is an interesting problem to construct a fully collusion resistant trace-and-revoke scheme for arbitrary identities from standard polynomial hardness assumptions.

4.1 Constructing an Identity-Based Trace-and-Revoke Scheme

Our construction follows the general high-level schema as that by Nishimaki et al. [NWZ16], except our construction is secretly-traceable (but will provide *full* collusion resistance). Very briefly, we use a secret-key revocable predicate encryption scheme to embed an instance of the generalized jump-finding problem (Definition 2.5) where the position of the "jumps" correspond to non-revoked keys. The tracing algorithm relies on the generalized jump-finding algorithm (Theorem 2.6) to identify the compromised keys. We give our construction below.

Construction 4.4 (Identity-Based Trace-and-Revoke). Let $\mathcal{ID} = \{0,1\}^n$ be the identity space and let $\mathcal{M}$ be a message space. We additionally rely on the following primitives:

– Let $H \colon \mathcal{K} \times \mathcal{ID} \to [2^\ell]$ be a keyed collision-resistant hash function.

- Let $\mathcal{ID}_0 = [2^{\ell+1}]$. For a pair $(i, u) \in [n] \times [0, 2^{\ell+1}]$, define the function $f_{i,u} \colon \mathcal{ID}_0^n \to \{0,1\}$ to be the function that takes as input $v = (v_1, \ldots, v_n)$, where each $v_i \in \mathcal{ID}_0$, and outputs 1 if $v_i \le u$ and 0 otherwise. When $u = 0$, $f_{i,u}(v) = 0$ for all $i \in [n]$ and $v \in \mathcal{ID}_0^n$. Similarly, when $u = 2^{\ell+1}$, $f_{i,u}(v) = 1$ for all $i \in [n]$ and $v \in \mathcal{ID}_0^n$. We will use a canonical "all-zeroes" function to represent $f_{i,0}$ and a canonical "all-ones" function to represent $f_{i,2^{\ell+1}}$ for all $i \in [n]$.
- Let $\Pi_{\mathsf{RPE}} = (\mathsf{RPE.Setup}, \mathsf{RPE.KeyGen}, \mathsf{RPE.Broadcast}, \mathsf{RPE.Enc}, \mathsf{RPE.Dec})$ be a secret-key revocable predicate encryption scheme with broadcast with attribute space $\mathcal{ID}_0^n$, label space $[2^\ell]$, message space $\mathcal{M}$, and function space $\mathcal{F} = \{i \in [n], u \in [0, 2^{\ell+1}] : f_{i,u}\}$.

We construct a trace-and-revoke scheme $\Pi_{\mathsf{TR}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ with identity space $\mathcal{ID}$ and message space $\mathcal{M}$ as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm samples a key $\mathsf{hk} \xleftarrow{\text{R}} \mathcal{K}$, parameters $(\mathsf{rpe.pp}, \mathsf{rpe.msk}) \leftarrow \mathsf{RPE.Setup}(1^\lambda)$, and outputs

$$\mathsf{pp} = (\mathsf{hk}, \mathsf{rpe.pp}) \quad \text{and} \quad \mathsf{msk} = (\mathsf{hk}, \mathsf{rpe.msk}).$$

- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: On input the master secret key $\mathsf{msk} = (\mathsf{hk}, \mathsf{rpe.msk})$ and an identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_n) \in \mathcal{ID}$, the key-generation algorithm computes $s_{\mathsf{id}} \leftarrow H(\mathsf{hk}, \mathsf{id})$ and defines the vector $v_{\mathsf{id}} = (2s_{\mathsf{id}} - \mathsf{id}_1, \ldots, 2s_{\mathsf{id}} - \mathsf{id}_n) \in \mathcal{ID}_0^n$. It outputs $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{RPE.KeyGen}(\mathsf{rpe.msk}, s_{\mathsf{id}}, v_{\mathsf{id}})$.
- $\mathsf{Enc}(\mathsf{pp}, m, \mathcal{L})$: On input the public parameters $\mathsf{pp} = (\mathsf{hk}, \mathsf{rpe.pp})$, a message m, and a revocation list $\mathcal{L} \subseteq \mathcal{ID}$, the encryption algorithm first constructs a new list $\mathcal{L}' \subseteq \{0,1\}^\ell$ where $\mathcal{L}' = \{\mathsf{id} \in \mathcal{L} : H(\mathsf{hk}, \mathsf{id})\}$. Then, it outputs $\mathsf{ct}_{m,\mathcal{L}} \leftarrow \mathsf{RPE.Broadcast}(\mathsf{rpe.pp}, m, \mathcal{L}')$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: On input a secret key sk and a ciphertext ct, the decryption algorithm outputs $m \leftarrow \mathsf{RPE.Dec}(\mathsf{sk}, \mathsf{ct})$.
- $\mathsf{Trace}^{\mathcal{D}}(\mathsf{msk}, m_0, m_1, \mathcal{L}, \varepsilon)$: On input the decryption oracle $\mathcal{D}$, the master secret key $\mathsf{msk} = (\mathsf{hk}, \mathsf{rpe.msk})$, messages $m_0, m_1 \in \mathcal{M}$, a revocation list $\mathcal{L} \subseteq \mathcal{ID}$, and a success probability ε, the tracing algorithm begins by constructing the set $\mathcal{L}' \subseteq \{0,1\}^\ell$ where $\mathcal{L}' = \{\mathsf{id} \in \mathcal{L} : H(\mathsf{hk}, \mathsf{id})\}$. It then defines the following *randomized* oracle Q:

On input a pair $(i, u) \in [n] \times [0, 2^{\ell+1}]$:

1. Sample a random bit $b \xleftarrow{\text{R}} \{0,1\}$, and construct the ciphertext $\mathsf{ct}_b \leftarrow \mathsf{RPE.Enc}(\mathsf{rpe.msk}, f_{i,u}, m_b, \mathcal{L}')$.
2. Run the decoder algorithm $\mathcal{D}$ on the ciphertext ct_b to obtain a bit $b' \leftarrow \mathcal{D}(\mathsf{ct}_b)$.
3. Output 1 if $b = b'$ and 0 otherwise.

Fig. 1: The randomized oracle Q used for tracing.

Let $q = 1$, set $\delta_q = \varepsilon/(9+4(\ell-1)q)$, and compute $\mathcal{T}_q \leftarrow \mathsf{QTrace}^Q(\lambda, 2^\ell, n, q, \delta_q, \varepsilon)$. If $\mathcal{T}_q$ is non-empty, take any element $(s_{\mathsf{id}}, \mathsf{id}_1, \ldots, \mathsf{id}_n) \in \mathcal{T}_q$, and output

$\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_n) \in \mathcal{ID}$. Otherwise, update $q \leftarrow 2q$ and repeat this procedure.[6]

Correctness and security analysis. We now show that Π_{TR} from Construction 4.4 satisfies correctness, semantic security, and traceability. We state the main theorems below, but defer their formal proofs to the full version of this paper [KW19a]. The analysis proceeds similarly to the corresponding analysis from [NWZ16], except we operate in the secret-traceability setting. The main challenge in the secret-key setting is that when the adversary in the traceability game outputs a pirate decoder, the reduction algorithm cannot easily tell whether the decoder is "useful" or not (where a "useful" decoder is one that can be leveraged to break the security of the underlying secret-key revocable predicate encryption scheme). The analysis in [NWZ16] solves this problem by having the reduction algorithm sample ciphertexts of its own and observe the decoder's behavior on those ciphertexts. In this way, the reduction is able to estimate the decoder's distinguishing advantage and identify whether the adversary produced a good decoder or not. In the secret-key setting, the reduction *cannot* sample ciphertexts of its own and as such, it cannot estimate the decoder's success probability. To solve this problem, we adopt the approach taken in [GKW18] and allow the reduction algorithm to make a *single* encryption query to the secret-key predicate encryption scheme. Using the same type of analysis as in [GKW18], we then show that with just a single encryption query, the reduction can leverage the decoder output by the traceability adversary to break security of the underlying predicate encryption scheme. The full analysis is provided in the full version of this paper [KW19a].

Theorem 4.5 (Correctness). *If H is collision-resistant and Π_{RPE} is correct, then Π_{TR} from Construction 4.4 is correct.*

Theorem 4.6 (Semantic Security). *If Π_{RPE} satisfies broadcast security and message hiding (without encryption queries), then Π_{TR} from Construction 4.4 is semantically secure.*

Theorem 4.7 (Traceability). *If H is collision-resistant and Π_{RPE} satisfies non-adaptive 1-query message hiding security, non-adaptive 1-query function hiding, and non-adaptive 1-query broadcast security, then Π_{TR} is traceable. In particular, the tracing algorithm* Trace *is efficient.*

[6]We will argue in the proof of Theorem 4.7 that this algorithm will terminate with overwhelming probability. Alternatively, we can set an upper bound on the maximum number of iterations q_{max}. In this case, the tracing algorithm succeeds as long as the total number of keys issued is bounded by $2^{q_{\mathsf{max}}}$. Note that this is not an *a priori* bound on the number of keys that can be issued, just a bound on the number of iterations on which to run the tracing algorithm, which can be a flexible parameter (independent of other scheme parameters).

4.2 Instantiating the Trace-and-Revoke Scheme

In this section, we describe our instantiation of our resulting trace-and-revoke scheme using the secret-key revocable predicate encryption scheme from Section 3.1 (Construction 3.4, Corollary 3.9). In particular, combining Construction 4.4 with Theorems 4.5 through 4.7 yields the following corollary:

Corollary 4.8 (Identity-Based Trace-and-Revoke from LWE). *Assuming sub-exponential hardness of LWE (with a super-polynomial modulus-to-noise ratio), there exists a fully secure identity-based trace-and-revoke scheme with identity space $\mathcal{ID} = \{0,1\}^n$ and message space $\mathcal{M} = \{0,1\}^t$ with the following properties:*

- ***Public parameter size:*** $|\mathsf{pp}| = n \cdot \mathsf{poly}(\lambda, \log n)$.
- ***Secret key size:*** *The secret key* $\mathsf{sk}_{\mathsf{id}}$ *for an identity* $\mathsf{id} \in \{0,1\}^n$ *has size* $\mathsf{sk}_{\mathsf{id}} = n \cdot \mathsf{poly}(\lambda, \log n)$.
- ***Ciphertext size:*** *An encryption* $\mathsf{ct}_{m,\mathcal{L}}$ *of a message* $m \in \{0,1\}^t$ *with respect to a revocation list* $\mathcal{L}$ *has size* $\mathsf{ct}_{m,\mathcal{L}} = t + |\mathcal{L}| \cdot \mathsf{poly}(\lambda, \log n)$.

Proof. The claim follows by instantiating Construction 4.4 with the following primitives:

- We can instantiate the collision-resistant hash function H with the standard SIS-based collision-resistant hash function [Ajt96, GGH96]. In this case, the hash key hk has size $|\mathsf{hk}| = \mathsf{poly}(\lambda)$ and the output length of the hash function is also $\ell = \mathsf{poly}(\lambda)$.
- We instantiate the secret-key revocable predicate encryption scheme with broadcast Π_{RPE} with the construction from Corollary 3.9. For $i \in [n]$ and $u \in [0, 2^{\ell+1}]$, the description length z of the functions $f_{i,u} \in \mathcal{F}$ satisfies

$$z = |i| + |u| \leq \log n + \ell + 3 = \mathsf{poly}(\lambda, \log n).$$

  Moreover, each function $f_{i,u}$ is computing a comparison on ℓ-bit values and selecting one out of the n components of the vector. This can be computed by a Boolean circuit with depth $d = \mathsf{poly}(\lambda, \log n)$—$\mathsf{poly}(\lambda)$ for the comparison and $\mathsf{poly}(\log n)$ to select the element to compare. Finally, the identity-space for the underlying revocable predicate encryption scheme is $\mathcal{ID}_0 = [2^{\ell+1}]$ and the attribute space is $\mathcal{ID}_0^n$.

We now verify the parameter sizes for the resulting construction:

- **Public parameters size:** The public parameters pp consists of the hash key hk and the public parameters $\mathsf{rpe.pp}$ for the revocable predicate encryption scheme. Thus,

$$|\mathsf{pp}| = |\mathsf{hk}| + |\mathsf{rpe.pp}| = \mathsf{poly}(\lambda) + n\ell \cdot \mathsf{poly}(\lambda, d, \ell, z) = n \cdot \mathsf{poly}(\lambda, \log n).$$

- **Secret key size:** The secret key $\mathsf{sk}_{\mathsf{id}}$ for an identity $\mathsf{id} \in \{0,1\}^n$ consists of a secret key for the underlying revocable predicate encryption scheme. By Corollary 3.9, we have that $|\mathsf{sk}_{\mathsf{id}}| = n\ell + \mathsf{poly}(\lambda, d, \ell, z) = n \cdot \mathsf{poly}(\lambda, \log n)$.

– **Ciphertext size:** The ciphertext $\mathsf{ct}_{m,\mathcal{L}}$ for a message $m \in \{0,1\}^t$ with respect to a revocation list $\mathcal{L}$ consists of a ciphertext for the underlying revocable predicate encryption scheme. By Corollary 3.9,

$$|\mathsf{ct}_{m,\mathcal{L}}| = t + |\mathcal{L}| \cdot \mathsf{poly}(\lambda, d, \ell, z) = t + |\mathcal{L}| \cdot \mathsf{poly}(\lambda, \log n). \square$$

Acknowledgments

We thank Ahmadreza Rahimi for helpful discussions on this work and the anonymous reviewers for useful suggestions on improving the exposition. S. Kim is supported by NSF, DARPA, a grant from ONR, and the Simons Foundation. D. J. Wu is supported by NSF CNS-1917414 and a University of Virginia SEAS Research Innovation Award. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

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 *ACM CCS*, pages 2277–2293, 2017.

AJS17. Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation for turing machines: Constant overhead and amortization. In *CRYPTO*, pages 252–279, 2017.

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108, 1996.

AS16. Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In *TCC*, pages 125–153, 2016.

BB04. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

BGG+14. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, pages 533–556, 2014.

BGI+12. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.

BN08. Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *ACM CCS*, pages 501–510, 2008.

BP09. Olivier Billet and Duong Hieu Phan. Traitors collaborating in public: Pirates 2.0. In *EUROCRYPT*, pages 189–205, 2009.

BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, pages 573–592, 2006.

BW06. Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *ACM CCS*, pages 211–220, 2006.

BW07. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, pages 257–270, 1994.

CFNP00. Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Trans. Information Theory*, 46(3):893–910, 2000.

CHN$^+$16. Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In *STOC*, pages 1115–1127, 2016.

CVW$^+$18. Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from LWE made simple and attribute-based. In *TCC*, pages 341–369, 2018.

DF02. Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Security and Privacy in Digital Rights Management*, pages 61–80, 2002.

FN93. Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, pages 480–491, 1993.

GGH96. Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. *IACR Cryptology ePrint Archive*, 1996:9, 1996.

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 *FOCS*, pages 40–49, 2013.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *FOCS*, pages 464–479, 1984.

GKM$^+$19. Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J. Wu. Watermarking public-key cryptographic primitives. In *CRYPTO*, 2019.

GKP$^+$13. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.

GKSW10. Sanjam Garg, Abishek Kumarasubramanian, Amit Sahai, and Brent Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In *ACM CCS*, pages 121–130, 2010.

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *STOC*, pages 660–670, 2018.

GKW19. Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In *TCC*, 2019.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, pages 89–98, 2006.

GQWW19. Rishab Goyal, Willy Quach, Brent Waters, and Daniel Wichs. Broadcast and trace with n$\hat{}\epsilon$ ciphertext size from standard assumptions. In *CRYPTO*, pages 826–855, 2019.

GS18. Sanjam Garg and Akshayaram Srinivasan. A simple construction of io for turing machines. In *TCC*, pages 425–454, 2018.

GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012.

GVW19. Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. Collusion resistant broadcast and trace from positional witness encryption. In *PKC*, pages 3–33, 2019.

HS02. Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In *CRYPTO*, pages 47–60, 2002.

KD98. Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *EUROCRYPT*, pages 145–157, 1998.

KLW15. Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *STOC*, pages 419–428, 2015.

KP07. Aggelos Kiayias and Serdar Pehlivanoglu. Pirate evolution: How to make the most of your traitor keys. In *CRYPTO*, pages 448–465, 2007.

KSW08. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.

KT15. Aggelos Kiayias and Qiang Tang. Traitor deterring schemes: Using bitcoin as collateral for digital content. In *ACM CCS*, pages 231–242, 2015.

KW17. Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *CRYPTO*, pages 503–536, 2017.

KW19a. Sam Kim and David J. Wu. Collusion resistant trace-and-revoke for arbitrary identities from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2019:984, 2019.

KW19b. Sam Kim and David J. Wu. Watermarking PRFs from lattices: Stronger security via extractable PRFs. In *CRYPTO*, 2019.

LPSS14. San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-lwe and applications in traitor tracing. In *CRYPTO*, pages 315–334, 2014.

NNL01. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, pages 41–62, 2001.

NP98. Moni Naor and Benny Pinkas. Threshold traitor tracing. In *CRYPTO*, pages 502–517, 1998.

NP00. Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography*, pages 1–20, 2000.

NWZ16. Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In *EUROCRYPT*, pages 388–419, 2016.

QWZ18. Willy Quach, Daniel Wichs, and Giorgos Zirdelis. Watermarking PRFs under standard assumptions: Public marking and security with extraction queries. In *TCC*, pages 669–698, 2018.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

SBC$^+$07. Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE (S&P*, pages 350–364, 2007.

SS10. Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM CCS*, pages 463–472, 2010.

SSW01. Jessica Staddon, Douglas R. Stinson, and Ruizhong Wei. Combinatorial properties of frameproof and traceability codes. *IEEE Trans. Information Theory*, 47(3):1042–1049, 2001.

SW05. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.