Multiparty Key Exchange, Efficient Traitor Tracing, and More from Indistinguishability Obfuscation

Dan Boneh Mark Zhandry

Stanford University
{dabo, zhandry}@cs.stanford.edu

Abstract

In this work, we show how to use indistinguishability obfuscation (iO) to build multiparty key exchange, efficient broadcast encryption, and efficient traitor tracing. Our schemes enjoy several interesting properties that have not been achievable before:

- Our multiparty non-interactive key exchange protocol does not require a trusted setup. Moreover, the size of the published value from each user is independent of the total number of users.
- Our broadcast encryption schemes support *distributed* setup, where users choose their own secret keys rather than be given secret keys by a trusted entity. The broadcast ciphertext size is *independent* of the number of users.
- Our traitor tracing system is fully collusion resistant and provides ciphertexts that are logarithmic in the number of users and constant-sized secret keys. The recent functional encryption system of Garg, Gentry, Halevi, Raykova, Sahai, and Waters also leads to a traitor tracing system with short ciphertexts and secret keys, but the construction in this paper is simpler and more direct. These constructions resolve an open problem relating to differential privacy.

Our proof of security for traitor tracing introduces a new tool for iO proofs: the construction makes use of a key-homomorphic symmetric cipher which plays a crucial role in the proof of security.

1 Introduction

An obfuscator is a machine that takes as input a program, and produces a second program with identical functionality that in some sense hides how the original program works. An important notion of obfuscation called *indistinguishability obfuscation* (iO) was proposed by Barak et al. [BGI+01] and further studied by Goldwasser and Rothblum [GR07]. Indistinguishability obfuscation asks that obfuscations of any two (equal-size) programs that compute the same function are computationally indistinguishable. The reason iO has become so important is a recent breakthrough result of Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH+13b] that put forward the first candidate construction for an efficient iO obfuscator for general boolean circuits. The construction builds upon the multilinear map candidates of Garg, Gentry, and Halevi [GGH13a] and Coron, Lepoint, and Tibouchi [CLT13].

In subsequent work, Sahai and Waters [SW13] showed that indistinguishability obfuscation is a powerful cryptographic primitive: it can be used to build public-key encryption from pseudorandom functions, selectively-secure short signatures, deniable encryption, and much more. Hohenberger, Sahai, and Waters [HSW13] showed that iO can be used to securely instantiate the random oracle in several random-oracle cryptographic systems.

Our results. In this paper, we show further powerful applications for indistinguishability obfuscation. While the recent iO constructions make use of multilinear maps, the converse does not seem to hold: we do not yet know how to build multilinear maps from iO. Nevertheless, we show that iO *can* be used to construct many of the powerful applications that follow from multilinear maps. The resulting iO-based constructions have surprising features that could not be previously achieved, not even using the current candidate multilinear maps. All of our constructions employ the punctured PRF technique introduced by Sahai and Waters [SW13].

1.1 Multiparty non-interactive key exchange

Our first construction uses iO to construct a multiparty non-interactive key exchange protocol (NIKE) from a pseudorandom generator. Recall that in a NIKE protocol, N parties each post a single message to a public bulletin board. All parties then read the board and agree on a shared key k that is secret from any eavesdropper who only sees the bulletin board. The classic Diffie-Hellman protocol solves the two-party case $N = 2$. The first three-party protocol was proposed by Joux [Jou04] using bilinear maps. Boneh and Silverberg [BS03] gave a protocol for general N using multilinear maps. The candidate multilinear map constructions by Garg, Gentry, and Halevi [GGH13a] using ideal lattices, and by Coron, Lepoint, and Tibouchi [CLT13] over the integers, provide the first implementations for N parties. Prior to this work, these were the only known constructions for NIKE.

We construct a new NIKE from a general indistinguishability obfuscator. The protocol is easy to describe: each user generates a random seed s to a pseudorandom generator G whose output is at least twice the size of the seed. The user posts $G(s)$ to the bulletin board. Now, when N users wish to generate a shared group key, they each collect all the public values from the bulletin board and run a certain *public* obfuscated program (shown in Figure 1) on the public values along with their secret seed. The program outputs the group key.

We show that this protocol is secure in a semi-static model [FHKP13]: an adversary that is allowed to (non-adaptively) corrupt participants of its choice cannot learn the shared group key of a group of uncorrupt users of its choice. The proof uses the punctured PRF technique of Sahai and Waters, but interestingly requires the full power of the constrained PRFs of Boneh and Waters [BW13] for arbitrary circuit constraints. In addition, we show that the point-wise punctured PRFs used by Sahai and Waters are sufficient to prove security, but only in a weaker static security model where the adversary cannot corrupt users. We leave the construction of an iO-based fully adaptively secure NIKE (in the sense of [FHKP13]) as a fascinating open problem.

In Section 6, we observe that our iO-based NIKE can be easily extended to an identity-based multiparty key exchange. Existing ID-NIKE protocols are based on multilinear maps [FHPS13].

Comparison to existing constructions. While NIKE can be built directly from multilinear maps, the iO-based protocol has a number of advantages:

- First, existing constructions [GGH13a, CLT13] require a trusted setup to publish public parameters: whoever publishes the parameters can recover the secret keys for all groups from the public values. A simple variant of our iO-based construction requires no trusted setup — in fact, the protocol requires no setup at all.
- Second, in current multilinear-based NIKE protocols, the size of the values published to the bulletin board is linear in the number of users N. In our iO-based construction, the size of published values is independent of N.
- Third, since the published values are independent of any public parameters, the same published values can be used in multiple NIKE environments setup by different organizations.

It is also worth noting that since our NIKE is built from a generic iO mechanism, it may eventually depend on a weaker complexity assumption than those needed for secure multilinear maps.

1.2 Broadcast encryption

Broadcast encryption [FN94] lets an encryptor broadcast a message to a subset of recipients. The system is said to be collusion resistant if no set of non-recipients can learn information about the plaintext. The efficiency of a broadcast system is measured in the ciphertext overhead: the number of bits in the ciphertext beyond what is needed to describe the recipient set and encrypt the payload message using a symmetric cipher. The shorter the overhead, the better (an overhead of zero is optimal). We survey some existing constructions in related work below.

Using a generic conversion from NIKE to broadcast encryption described in Section 4.1, we obtain two collusion-resistant broadcast systems. The first is a secret-key broadcast system with *optimal* broadcast size. The second is a public-key broadcast system with constant overhead, namely *independent* of the number of recipients. In both systems, decryption keys are constant size (i.e. independent of the number of users). The encryption key, however, is linear in the number of users as in several other broadcast systems [BGW05, GW09, DPP07, BW13].

By starting from our semi-static secure NIKE, we obtain a semi-static secure broadcast encryption (as defined in Section 4). Then applying a generic conversion due to Gentry and Waters [GW09], we obtain a fully adaptively secure public-key broadcast encryption system with the shortest known ciphertext overhead.

Our public-key broadcast encryption has a remarkable property that has so far not been possible, not even using the candidate multilinear maps. The system is the first public-key *distributed* broadcast system: users generate secret keys on their own and simply append their corresponding public values to the broadcast public key. In contrast, in existing low-overhead public-key broadcast systems surveyed below, users are assigned their secret key by a trusted authority who has the power to decrypt all broadcasts. In our iO-based public-key system, there is no trusted authority.

1.3 Recipient-private broadcast encryption

A broadcast encryption system is said to be recipient-private if broadcast ciphertexts reveal nothing about the intended set of recipients [BBW06, LPQ12, FP12]. Valid recipients will learn that they are members of the recipient set (by successfully decrypting the ciphertext), but should learn nothing else about the set. Currently, the best recipient-private broadcast systems have a broadcast size of $O(\lambda \cdot N)$, proportional to the product of the security parameter λ and the number of users N.

Using iO, we construct the first recipient-private broadcast system with a broadcast size of $O(\lambda + N)$, proportional to the *sum* of the security parameter and the number of users. This is the best possible broadcast size. If one is allowed to leak the size k of the recipient set (and nothing else) then we construct a system where the broadcast size is proportional to $O(\lambda + k \log N)$, which is again the best possible. Building such systems has been open for some time [BBW06] and is now resolved using iO.

Our approach to building a recipient-private broadcast system is to embed an encryption of the intended recipient set in the broadcast header. We publish an obfuscated program in the public key that begins by decrypting the encrypted recipient set in the broadcast header. It then decrypts the message body only if the recipient can provide a proof that it is one of the intended recipients. Interestingly, encrypting the recipient set in a way that lets us prove security using iO is non-trivial. The problem is that using a generic CPA-secure scheme is insecure due to potential malleability attacks on the encrypted recipient set that can break recipient privacy. Using an authenticated encryption scheme to prevent the malleability attack does not work either because forged valid ciphertexts exist (even though they may be difficult to construct), and this prevents us from proving security using iO. The difficulty stems from the fact that iO can only be applied to two programs that agree on *all* inputs, including hard-to-compute ones.

Instead of using authenticated encryption, which does not seem to work, we encrypt the recipient set using a certain malleable encryption scheme that lets us translate an encryption of a recipient set S to an encryption of some other recipient set S'. We use indistinguishability of obfuscations to argue that an attacker cannot detect this change, thereby proving recipient privacy.

The recent succinct functional encryption scheme of Garg et al. [GGH+13b] can also used to build recipient-private broadcast encryption from iO. However, our construction is quite different and is simpler and more direct. For example, it does not use non-interactive zero-knowledge proofs. Moreover, our scheme has shorter secret keys: $O(1)$ as a function of N compared to $N^{O(1)}$. The main drawback of our scheme is the larger public key: $N^{O(1)}$ compared to $O(\log N)$. It is worth noting however that the system of [GGH+13b] requires public-key encryption. In order to instantiate the system using only iO and one-way functions like we do, the resulting public key size will be comparable to ours.

A traitor tracing system with short ciphertexts. Private broadcast-encryption is further motivated by its application to traitor tracing systems [CFN94]. Recall that traitor tracing systems, introduced by Chor, Fiat, and Naor, help content distributors identify the origin of pirate decryption boxes, such as pirate cable-TV set top decoders. Boneh, Sahai, and Waters [BSW06] showed that a private broadcast encryption system that can broadcast privately to any of the $N + 1$ sets $\emptyset, \{1\}, \{1, 2\}, \ldots, \{1, \ldots, N\}$ is sufficient for building an N-user traitor tracing system. The ciphertext used in the traitor tracing system under normal operation is simply a broadcast to the full set $\{1, \ldots, N\}$, allowing all decoders to decrypt. Therefore, the goal is, as before, to build a private broadcast system for this specific set system where ciphertext overhead is minimized. Such systems are called *private linear broadcast encryption*.

Adapting our iO-based private broadcast system to the linear set system above, we obtain a collusion resistant traitor tracing system where ciphertext size is $O(\lambda + \log N)$ where λ is the security parameter and N is the total number of users in the system. Moreover, secret keys are short: their length is λ, independent of N. Our iO-based system is the first collusion resistant system to simultaneously achieve such short ciphertexts and secret keys.

The functional encryption scheme of Garg et al. [GGH+13b] can also be used to obtain collusion resistant traitor tracing. Similar to the general private broadcast case, our construction is conceptually simpler and has slightly shorter secret keys. As before, public keys in our scheme are larger: $N^{O(1)}$ compared to $O(\log N)$.

Connection to Differential Privacy Dwork et al. [DNR+09] show that efficient traitor tracing schemes imply the impossibility of any differentially private data release mechanism. A data release mechanism is a procedure that outputs a data structure that supports approximations to queries of the form "what fraction of records have property P?" A data release mechanism is differentially private if it does not reveal whether any individual record is in the database.

Applying the counter-example of [DNR+09] to our traitor tracing scheme, we obtain a database of N records of size λ and $O(N2^\lambda)$ queries.[1] Moreover, the records are just independent uniform bit strings. Even with these small and simple records and relatively few queries, no polynomial time (in λ and N) differentially private data release mechanism is possible, so long as our construction is secure. The first scheme this counter example was applied to is the traitor tracing scheme of Boneh, Sahai, and Waters [BSW06], giving records of size $O(\lambda)$, but with a query set of size $2^{\tilde{O}(\sqrt{N})}$, exponential in N.

Ullman [Ull13] shows that, assuming one-way functions exist, there is no algorithm that takes a database of N records of size λ and an arbitrary set of approximately $O(N^2)$ queries, and approximately answers each query in time $\mathsf{poly}(N, \lambda)$ while preserving differential privacy. This result also uses the connection between traitor tracing and differential privacy, but is qualitatively different from ours. Their result applies to algorithms answering any *arbitrary* set of $O(N^2)$ queries while maintaining differential privacy, whereas we demonstrate a *fixed* set of $O(N2^\lambda)$ queries that are impossible to answer efficiently.

Constrained PRFs. Recall that constrained PRFs, needed in iO proofs of security, are PRFs for which there are constrained keys than enable the evaluation of the PRF at a subset of the PRF domain and nowhere else [BW13, KPTZ13, BGI13]. The next section gives a precise definition. Our last construction shows that iO, together with a one-way function, are sufficient to build a constrained PRF for arbitrary circuit constraints. Consequently, all our constructions that utilize circuit constrained PRFs can be directly built from iO and a one-way function without additional assumptions. In fact, Moran and Rosen [MR13] show, under the assumption that NP is not solvable in probabilistic polynomial time in the worst case, that indistinguishability obfuscation implies one-way functions. Previously, constrained PRFs for arbitrary circuit constraints were built using multilinear maps [BW13].

1.4 Related work

While some works have shown how to obfuscate simple functionalities such as point functions [Can97, CMR98, LPS04, Wee05], inner products [CRV10], and d-CNFs [BR13a], it is only recently that obfuscation for poly-size circuits became possible [GGH+13b, BR13b, BGK+13] and was applied to building higher level cryptographic primitives [SW13, HSW13].

[1] Our scheme has large public keys, which affects the complexity of the queries, but this does not affect the number of queries or the size of the database records.

Broadcast encryption. Fully collusion resistant broadcast encryption has been widely studied. Revocation systems [NNL01, HS02, GST04, DF02, LSW10] can encrypt to $N - r$ users with ciphertext size of $O(r)$. Further combinatorial solutions [NP00, DF03] achieve similar parameters. Algebraic constructions [BGW05, GW09, DPP07] using bilinear maps achieve constant (but non-zero) ciphertext overhead and some are even identity-based [GW09, Del07, SF07]. Multilinear maps give secret-key broadcast systems with optimal ciphertext size and short private keys [BS03, BW13]. They also give public-key broadcast systems with short ciphertexts and short public keys (using an $O(\log N)$-linear map), but using the existing multilinear candidates, those systems are not distributed: users must be given their private keys by a central authority. The difficulty with using existing N-linear maps for public-key broadcast encryption is that the encoding of a single element requires $\Omega(N)$ bits, and therefore a short ciphertext cannot include even a single element.

Recipient-private broadcast encryption. The first constructions for private broadcast encryption [BBW06, LPQ12] required a ciphertext header whose size is proportional to the product of the security parameter and the number of recipients. More recently, Fazio and Perera [FP12] presented a system with a weaker privacy guarantee called *outsider anonymity*, but where the header size is proportional to the number of revoked users. Kiayias and Samari [KS13] even provide lower bounds showing that certain types of natural constructions cannot improve on these bounds.

The functional encryption scheme of Garg et al. [GGH+13b] can also used to build recipient-private broadcast encryption from iO. Our scheme is conceptually simpler, and avoids the need for non-interactive zero-knowledge proofs. Moreover, our scheme has shorter secret keys: $O(1)$ in N compared to $N^{O(1)}$ — though for private *linear* broadcast, their secret keys are $\mathsf{polylog}(N)$. The main drawback of our scheme is the large public key size: $N^{O(1)}$ compared to $O(\log N)$.

Traitor tracing. The literature on traitor tracing is vast and here we only discuss results on fully collusion resistant systems. Since the trivial fully-collusion resistant system has ciphertext size that is linear in the number of users, we are only interested in fully collusion resistant systems that achieve sub-linear size ciphertext. The first such system [BSW06, BW06], using bilinear maps, achieved $\sqrt{n}$ size ciphertexts with constant size keys. Other schemes based on different assumptions achieve similar parameters [GKSW10, Fre10]. Combinatorial constructions can achieve constant size ciphertexts [BN08, Sir07], but require secret keys whose size is quadratic (or worse) in the number of users. In most traitor tracing systems, the tracing key must be kept secret. Some systems, including ours, allow anyone to run the tracing algorithm [Pfi96, PW97, WHI01, KY02, CPP05, BW06].

2 Preliminaries: Definitions and Notation

Here we give the necessary background, including notation and the definitions cryptographic primitives we will be using.

Notation We let $[N] = \{1, \cdots, N\}$ denote the positive integers from 1 to N. Given a set S, we let 2^S denote the power set of S: the set of all subsets of S. Given two sets S and T, we denote by $S \Delta T$ the symmetric difference between the sets: the set of all points in exactly one of S and T. Given a permutation $\sigma : \mathcal{X} \to \mathcal{X}$, and given a subset $S \subseteq \mathcal{X}$, denote by $\sigma(S)$ the set where each element $i \in S$ is replaced by $\sigma(i)$. That is, $\sigma(S) = \{\sigma(i) : i \in S\}$. For a set S we denote by $x \leftarrow S$

the uniform random variable on S. For a randomized algorithm $\mathcal{A}$, we denote by $x \leftarrow \mathcal{A}(y)$ the random variable defined by the output of $\mathcal{A}$ on input y.

2.1 Indistinguishability Obfuscation

The following formulation of indistinguishability obfuscation is due to Garg et al. [GGH+13b]:

Definition 2.1. (Indistinguishability Obfuscation) An *indistinguiability obfuscator* iO for a circuit class $\{\mathcal{C}_\lambda\}$ is a PPT uniform algorithm satisfying the following conditions:

- $\mathsf{iO}(\lambda, C)$ preserves the functionality of C. That is, for any $C \in \mathcal{C}_\lambda$, if we compute $C' = \mathsf{iO}(\lambda, C)$, then $C'(x) = C(x)$ for all inputs x.
- For any λ and any two circuits $C_0, C_1 \in \mathcal{C}_\lambda$ with the same functionality, the circuits $\mathsf{iO}(\lambda, C)$ and $\mathsf{iO}(\lambda, C')$ are indistinguishable. More precisely, for all pairs of PPT adversaries (Samp, D) there exists a negligible function α such that, if

$$\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, \sigma) \leftarrow \mathrm{Samp}(\lambda)] > 1 - \alpha(\lambda)$$

  then

$$\left|\Pr[D(\sigma, \mathsf{iO}(\lambda, C_0)) = 1] - \Pr[D(\sigma, \mathsf{iO}(\lambda, C_1)) = 1]\right| < \alpha(\lambda)$$

The circuit classes we are interested in are polynomial-size circuits — that is, when $\mathcal{C}_\lambda$ is the collection of all circuits of size at most λ. We call an obfuscator for this class an *indistinguishability obfuscator for P/poly*. The first candidate construction of such obfuscators is due to Garg et al. [GGH+13b].

When clear from context, we will often drop λ as an input to iO and as a subscript for $\mathcal{C}$.

2.2 Constrained Pseudorandom Functions

A pseudorandom function (PRF) is a function $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ where $\mathsf{PRF}(k, \cdot)$ is indistinguishable from a random function for a randomly chosen key k [GGM86]. We will generally omit reference to the key k, and just write $\mathsf{PRF}(\cdot)$ to refer to an instance of the function $\mathsf{PRF}(k, \cdot)$ for a random key k.

Following Boneh and Waters [BW13], we define constrained pseudorandom functions for a collection $\mathcal{S} \subseteq 2^{\mathcal{X}}$ of subsets as a PRF with the following added functionality: $\mathsf{PRF}.\mathsf{Constrain}(S)$ for $S \in \mathcal{S}$ outputs an efficient program for the function

$$\mathsf{PRF}^S(x) = \begin{cases} \mathsf{PRF}(x) & \text{if } x \in S \\ \perp & \text{if } x \notin \mathcal{S} \end{cases}.$$

That is, the program $\mathsf{PRF}^S(x)$ enables the evaluation of PRF at $x \in S$ and nowhere else. Similar notions to constraint PRFs were presented by Kiayias et al. [KPTZ13], where they were called delegatable PRFs, and Boyle et al. [BGI13], where they were called functional PRFs.

Security We adopt a weaker notion of security for constrained PRFs than [BW13] that is sufficient for our purposes: the adversary is allowed to request a *single* constraint key and should be unable to distinguish PRF from random at any point outside S. We use the following experiment, denoted $\mathtt{EXP}(b)$, parameterized by a bit $b \in \{0, 1\}$ on an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

$k \leftarrow \mathcal{K}$, $\mathsf{PRF}(\cdot) := \mathsf{PRF}(k, \cdot)$
$(S, \mathsf{state}) \leftarrow \mathcal{A}_1(\lambda)$ // $\mathcal{A}$ generates a single constraint S
$\mathsf{PRF}^S \leftarrow \mathsf{PRF.Constrain}(S)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{PRF}(\cdot),\mathsf{RoR}(b,\cdot)}(\lambda, \mathsf{PRF}^S, \mathsf{state})$
where
$\mathsf{PRF}(x)$ is just the oracle for $\mathsf{PRF}(x) = \mathsf{PRF}(k, x)$
$\mathsf{RoR}(b, x)$ is a real-or-random oracle: it takes as input $b \in \{0,1\}$ and $x \in \mathcal{X}$,
computes $y_0 \leftarrow \mathsf{PRF}(x)$ and $y_1 \xleftarrow{R} \mathcal{Y}$ and returns y_b

We require that each x given to RoR are distinct, lie outside of S, and are distinct from all of the x given to the PRF oracle. For $b = 0, 1$, let W_b be the event that $b' = 1$ in $\mathtt{EXP}(b)$, and define $\mathtt{PRF}^{(\mathrm{adv})}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$

Definition 2.2. We say that a constrained PRF is secure if for all probabilistic polynomial time adversaries $\mathcal{A}$, the function $\mathtt{PRF}^{(\mathrm{adv})}(\lambda)$ is negligible.

Example 2.3 (Prefix Constrained PRFs)**.** The PRF construction of Goldreich, Goldwasser, and Micali [GGM86] is a constrained PRF for sets of the form $\{x \in \{0,1\}^n : x_i = y_i \ \forall i \in [k]\}$ for some fixed $y_1 \dots y_k \in \{0,1\}^k$. In other words, the GGM PRF can be constrained to sets of a common prefix, as shown in [BW13, KPTZ13, BGI13]. This PRF can be built from any one-way function.

Example 2.4 (Punctured PRFs)**.** As defined by Sahai and Waters [SW13], a *punctured* PRF is a constrained PRF that can be constrained on the complement of any polynomial sized set $S \subseteq \mathcal{X}$. These can easily be realized using the prefix constrained PRFs above. We will write $\mathsf{PRF}^{\overline{S}}$ for the PRF punctured on the set S.

Example 2.5 (Constrained PRFs for Circuit Predicates)**.** Boneh and Waters [BW13] construct PRFs that support constraining to sets S accepted by a polynomial size circuit C. We will write PRF^C for the PRF constrained to the set accepted by C. In Section 7, we show how to realize such PRFs from indistinguishability obfuscators and one-way functions. Boneh-Waters give a realization from certain multi-linear maps.

3 Key Exchange from Indistinguishability Obfuscation

In this section, we show how to realize multiparty non-interactive key exchange (NIKE) from general indistinguishability obfuscation. A NIKE protocol consists of the following three algorithms:

$\mathsf{Setup}(\lambda, N)$: The setup algorithm takes a security parameter λ and a number N of users. It outputs public parameters params.

$\mathsf{Publish}(\mathsf{params}, i)$: Each party executes the publishing algorithm, which takes as input the public parameters and the index of the party, and generates two values: a user secret key sk_i and a user public value pv_i. User i keeps sk_i as his secret, and publishes pv_i to the other users.

$\mathsf{KeyGen}(\mathsf{params}, i, \mathsf{sk}_i, \{\mathsf{pv}_j\}_{j\in[N]})$: Finally, each party derives the shared key k using the public parameters params, their secret sk_i, and the other parties' public values $\{\mathsf{pv}_j\}_{j\in[N]}$.

For correctness, we require that each user derives the same secret key. That is,

$$\mathsf{KeyGen}(\mathsf{params}, i, \mathsf{sk}_i, \{\mathsf{pv}_j\}_{j\in[N]}) = \mathsf{KeyGen}(\mathsf{params}, i', \mathsf{sk}_{i'}, \{\mathsf{pv}_j\}_{j\in[N]}) \ \ \forall i, i' \in [N]$$

We consider several security notions for multiparty key exchange, starting with *adaptive* security, a generalization of the m-CKS-heavy security notion of Freire et al. [FHKP13]. In this notion of security, there are many users (labelled by identities in $\mathcal{ID}$), and various subsets of them are engaging in the key exchange protocol. We let the adversary adaptively corrupt users to learn the shared secret for arbitrary subsets of users. We define security for *symmetric* key exchange protocols, where $\mathsf{Publish}$ does not depend on i — generalizing to $\mathsf{Publish}$ that depend on i is straighforward. More formally, denote by $\texttt{EXP}(b)$ the following experiment, parameterized by the total number of parties N and a bit $b \in \{0, 1\}$ on an adversary $\mathcal{A}$:

$\mathsf{params} \xleftarrow{R} \mathsf{Setup}(\lambda, N)$

$b' \leftarrow \mathcal{A}^{\mathsf{Reg}(\cdot),\mathsf{RegCor}(\cdot,\cdot),\mathsf{Ext}(\cdot),\mathsf{Rev}(\cdots),\mathsf{Test}(\cdots)}(\lambda, N, \mathsf{params})$

where

- $\mathsf{Reg}(\mathsf{id} \in \mathcal{ID})$ registers an honest user. It takes an identity id, and runs $(\mathsf{sk}, \mathsf{pv}) \leftarrow \mathsf{Publish}(\mathsf{params})$. The challenger records the tuple $(\mathsf{id}, \mathsf{sk}, \mathsf{pv}, \mathit{honest})$, and sends pk to A.
- $\mathsf{RegCor}(\mathsf{id} \in \mathcal{ID}, \mathsf{pk})$ registers a corrupt user. It takes an identity id and a public value pv. The challenger records $(\mathsf{id}, \perp, \mathsf{pv}, \mathit{corrupt})$. The adversary may make multiple queries for a particular identity, in which case the challenger only uses the most recent record.
- $\mathsf{Ext}(\mathsf{id} \in \mathcal{ID})$ extracts the secret key for an honest registered user. The challenger looks up the tuple $(\mathsf{id}, \mathsf{sk}, \mathsf{pv}, \mathit{honest})$, and returns sk to the challenger.
- $\mathsf{Rev}(\mathsf{id}_1, \ldots, \mathsf{id}_N)$ reveals the shared secret for a group of users. We require that at least one of the identities was registered as honest, and that all the identities are distinct. The challenger uses the secret key for the honest user to derive the shared secret key k, which it returns to the adversary.
- $\mathsf{Test}(\mathsf{id}_1, \ldots, \mathsf{id}_N)$ takes in N distinct identities, all of which were registered as honest. If $b = 0$, the challenger runs KeyGen to determine the shared secret key, which it returns to the adversary. Otherwise if $b = 1$, the challenger generates a random key k to return to the adversary. In this case, it will record k to use in future Test queries for the same users in a different order.

We require that all reveal and test queries are distinct, and no extract query is allowed on any identity in a reveal query. Lastly, we require that all register queries and register-corrupt queries are for distinct id. For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\texttt{EXP}(b)$ and we define $\texttt{AdvKE}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 3.1. A multiparty key exchange protocol $(\mathsf{Setup}, \mathsf{Publish}, \mathsf{KeyGen})$ is adaptively secure if, for any PPT adversary $\mathcal{A}$ and any integer N, the function $\texttt{AdvKE}(\lambda)$ is negligible.

Unfortunately, we will not be able to meet this strong notion of security using our techniques. Instead, we will define two relaxations. In the first, we will not allow any extraction queries, a notion we call *semi-static* security:

Definition 3.2. A multiparty key exchange protocol $(\mathsf{Setup}, \mathsf{Publish}, \mathsf{KeyGen})$ is semi-statically secure if, for any PPT adversary $\mathcal{A}$ which does not make extract queries, and any integer N, $\texttt{AdvKE}(\lambda)$ is negligible.

We also weaken the definition further, arriving at a security notion called selective security. Basically, we only allow N honest register queries, no register corrupt queries, no extract queries, no reveal queries, and a single test query.

Definition 3.3. A multiparty key exchange protocol $(\mathsf{Setup}, \mathsf{Publish}, \mathsf{KeyGen})$ is selectively secure if, for any PPT adversary $\mathcal{A}$ that makes exactly N honest register queries and a single test query, the function $\mathtt{AdvKE}(\lambda)$ is negligible.

The above formulations have several limitations. For example, in order to guarantee security, either one of the parties must run setup *before* the protocol is carried out (making the protocol not truly single round), or the users must agree to use a system that has already been set up by a third party. That third party may be able to learn the secret, meaning the setup must be trusted. Another limitation is that once Setup is run, the maximum number of parties engaging in the exchange is fixed. Below, we give a few variants of multiparty key exchange that avoid some of these difficulties:

Untrusted setup In this formulation, we change the security game so that Setup is run by the *adversary*. Existing schemes [GGH13a, CLT13] do not maintain security in this setting.

Universal setup Here, we require that the setup algorithm does not depend on N. That is, $\mathsf{Setup}(\lambda, N) = \mathsf{Setup}(\lambda)$. We now require that N be given explicitly to $\mathsf{Publish}$ and KeyGen.

No setup Here, we require that setup just outputs $\mathsf{params} := \lambda$. Similarly, we make N explicit in $\mathsf{Publish}$ and KeyGen.

Independent publication Here, we require that $\mathsf{Publish}$ does not depend on the public parameters, but only on λ and potentially N. That is, $\mathsf{Publish}(\mathsf{params}, i) = \mathsf{Publish}(\lambda, N, i)$.

Symmetry Here, we require that each party behaves identically. This means that $\mathsf{Publish}$ and KeyGen cannot not depend on i. That is, $\mathsf{Publish}(\mathsf{params}, i) = \mathsf{Publish}(\mathsf{params})$ and $\mathsf{KeyGen}(\mathsf{params}, i, \mathsf{sk}_i, \{\mathsf{pv}_j\}_{j\in[N]}) = \mathsf{KeyGen}(\mathsf{params}, \mathsf{sk}_i, \{\mathsf{pv}_j\}_{j\in[N]})$.

Remark 3.4. Any NIKE with no setup also trivially has universal and untrusted setup. We also observe that any NIKE can be made symmetric by having every party run $\mathsf{Publish}$ for all $i \in [N]$. Once this is done, an ordering of the parties is computed based on their public values (say, the lexicographic order of their public values). The KeyGen is then run using only the values corresponding to the ordering.

Remark 3.5. Any NIKE with independent publication can be converted into a NIKE with no setup by designating one of the parties as the master party, who runs the setup algorithm and publishes the public parameters along with his public value. If the original NIKE is symmetric, we can make the new NIKE symmetric by having every party publish their own public parameters, and in KeyGen use the lexicographically minimum parameters.

The seminal key exchange protocols of Garg, Gentry, and Halevi [GGH13a] and Coron, Lepoint, and Tibouchi [CLT13] are both symmetric, but require a trusted setup since secrets are involved in generating the parameters for the multilinear map. These schemes make use of a "zero test parameter" to extract keys, and this parameter relies on the number of users who are exchanging keys. Therefore, these schemes do not have a universal setup. In the next section, we use iO to side-step some of these limitations.

3.1 Our Construction

We now build a multiparty non-interactive key exchange (NIKE) from indistinguishability obfuscation and pseudorandom generators. The idea is the following: each party generates a seed s_i as their secret key, and publishes $x_i = \mathsf{PRG}(s_i)$ as their public value, where PRG is a pseudorandom generator. In the setup-phase, a key k is chosen for a punctured pseudorandom function PRF. The shared secret key will be the function PRF evaluated at the concatenation of the samples x_i. To allow the parties to compute the key, the setup will publish an obfuscated program for PRF which requires knowledge of a seed to operate. In this way, each of the parties can compute the key, but anyone else will not know any of the seeds, and will therefore be unable to compute the key.

The construction is as follows:

Construction 3.6. *Let* PRF *be a constrained pseudorandom function, and let* $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *be a pseudorandom generator. Let* iO *be a program indistinguishability obfuscator.*

$\mathsf{Setup}(\lambda, N)$ *Choose a random key to obtain an instance of a pseudorandom function* PRF*. Build the program* P_{KE} *in Figure 1, padded to the appropriate length*[2]*. Output* $P_{\mathsf{iO}} = \mathsf{iO}(P_{KE})$ *as the public parameters.*

$\mathsf{Publish}(\lambda)$ *Party* i *chooses a random seed* $s_i \in \{0,1\}^\lambda$ *as a secret key, and publish* $x_i = \mathsf{PRG}(s_i)$

$\mathsf{KeyGen}(P_{\mathsf{iO}}, i, s_i, \{x_i\}_{i\in[N]})$ *Sort the* x_i *lexicographically, and run* P_{iO} *on* $(x_1, ..., x_N, s_i)$ *to obtain the key* $k = \mathsf{PRF}(x_1, ..., x_N)$ *or* $\perp$*.*

Inputs: $x_1, \ldots x_N \in \mathcal{X}^N$, $s \in \mathcal{S}$
Constants: PRF

1. If $x_i \neq \mathsf{PRG}(s)$ for all $i \in [N]$, output $\perp$
2. Otherwise, output $\mathsf{PRF}(x_1, x_2, \ldots, x_N)$

Figure 1: The program P_{KE}.

Correctness is trivial by inspection. For security, we consider two cases. If PRF is a punctured PRF, then we get selective security. If PRF is a constrained PRF for circuit predicates, then our construction actually achieves the semi-static notion of security (as in Definition 3.2). Security is summarized by the following theorem:

Theorem 3.7. *If* PRG *is a secure pseudorandom generator,* PRF *a secure punctured PRF, and* iO *a secure indistinguishability obfuscator, then Construction 3.6 is a selectively secure NIKE. If, in addition,* PRF *is a secure constrained PRF for circuit predicates, then Construction 3.6 is semi-statically secure.*

Before proving Theorem 3.7, we make several remarks:

[2]To prove security, we will replace P_{KE} with the obfuscation of another program P'_{KE}, which may be larger than P_{KE}. In order for the obfuscations to be indistinguishable, both programs must have the same size.

- $\mathsf{Publish}$ does not depend on any public parameters (aside from the security level). Therefore, Setup can be run independently of $\mathsf{Publish}$. Using Remark 3.5, we designate party 1 as the master party who runs Setup and publishes params along with x_1. We thus obtain the first multiparty key exchange protocol with no setup.

- The system is symmetric. To make this system have no setup while preserving symmetry, we can have each party generate their own PRF_i, and each build obfuscated programs $P_{\mathsf{iO},i}$. In KeyGen, the program corresponding to the user with the smallest x_i will be used to generate the key.

We now prove Theorem 3.7:

Proof. We prove the case where PRF is a constrained PRF for circuit predicates, the other case being simpler. Assume towards contradiction that an adversary $\mathcal{A}$ has non-nelgigible advantage ϵ in breaking the security of Construction 3.6 as in Definition 3.2. We prove security through a sequence of games.

Game 0 This is the attack game from Definition 3.2, where $\mathcal{A}$ gets the obfuscation of P_{KE}, and makes the following queries:

- Register honest user queries: $\mathcal{A}$ submits an $\mathsf{id} \in \mathcal{ID}$. The challenger chooses a random s_{id}, and sends $x_{\mathsf{id}} = G(s_{\mathsf{id}})$ to $\mathcal{A}$.

- Register corrupt user queries: $\mathcal{A}$ submits an $\mathsf{id} \in \mathcal{ID}$ and a string x_{id}. We require that id was not and will not be registered as honest.

- Extract queries: in the semi-static game, no extract queries are allowed.

- Reveal queries: the adversary submits N registered identities id_i, of which at least one is honest. The challenger uses PRF to compute the group key.

- Test queries: the adversary submits N honest users, and receives either the correct group key (if $b = 0$) or a random key (if $b = 1$).

After these queries, the adversary must make a guess b' for b.

Game 1 Let q_{RH} be an upper bound on the number of register honest queries. Before the game begins, choose q_{RH} random values $x_i^* \in \{0,1\}^{2\lambda}$. We will use these x_i^* values as the x_{id} values to answer honest user queries. The security of PRG shows that this game is indistinguishable from **Game 0**, so $\mathcal{A}$ still wins with advantage at least $\epsilon - \mathsf{negl}$.

Game 2 Notice that with overwhelming probability, none of the x_i^* for honest users in **Game 1** have a pre-image under PRG. Therefore, with overwhelming probability, there is no input to P_{KE} that will cause PRF to be evaluated on points of the form $(x_{i_1}^*, \ldots, x_{i_N}^*)$. We can now constrain the PRF so that it can only be evaluted at points $(x_1, \ldots, x_N)$ where the set $\{x_1, \ldots, x_N\}$ is not contained in the set $X^* = \{x_1^*, \ldots, x_{q_{RH}}^*\}$. Formally, we construct a circuit C that takes as input $(x_1, \cdots, x_N)$ and accepts if and only if there is some x_j that is not contained in X^*. We then construct the constrained function PRF^C.

Next, replace PRF with PRF^C in the program P_{KE}, arriving at the program P'_{KE} given in Figure 2. During Setup, give the adversary $P_{\mathsf{iO}} = \mathsf{iO}(P'_{KE})$ as the public parameters.

Since, with overwhelming probability, P_{KE} and P'_{KE} have the same functionality, security of iO implies that $\mathcal{A}$ still has advantage $\epsilon - \mathsf{negl}$ when it is given the obfuscation of P'_{KE} instead of the obfuscation of P_{KE}. Therefore $\mathcal{A}$ has non-negligible advantage in this Game 2.

Inputs: $x_1, \ldots x_N \in \mathcal{X}^N$, $s \in \mathcal{S}$

Constants: $\mathbf{PRF}^C$ (replaces PRF in program P_{KE})

1. If $x_i \neq \mathsf{PRG}(s)$ for all $i \in [N]$, output $\perp$
2. Otherwise, output $\mathbf{PRF}^C(x_1, x_2, \ldots, x_N)$

Figure 2: The program P'_{KE}.

An adversary $\mathcal{A}$ that has non-negligible advantage in Game 2 can be used to build a PRF adversary $\mathcal{B}$ that breaks the security of PRF as a constrained PRF (as in Definition 2.2). $\mathcal{B}$ chooses q_{RH} random values $x_i^* \in \{0,1\}^{2\lambda}$, and asks the PRF challenger for the constrained function PRF^C for C as defined above. $\mathcal{B}$ then builds the obfuscation of P'_{KE} in Figure 2, giving $\mathsf{iO}(P'_{KE})$ to $\mathcal{A}$. $\mathcal{B}$ then runs $\mathcal{A}$. Whenever $\mathcal{A}$ makes a register honest user query, $\mathcal{B}$ chooses one of the x_i^* that hasn't been used, and gives x_i^* to $\mathcal{A}$. For a register corrupt user query, $\mathcal{B}$ just records the public value x_{id}. For a reveal query, $\mathcal{B}$ asks its PRF oracle for the correct key and thus always reveals the correct key. Finally, for a test query, $\mathcal{B}$ makes a real-or-random challenge query for PRF. $\mathcal{B}$ thus perfectly simulates the view of $\mathcal{A}$ in **Game 2** and therefore $\mathcal{B}$ breaks the security of PRF with advantage $\epsilon - \mathsf{negl}$. It follows that ϵ must be negligible, proving the security of Construction 3.6. □

4 Adaptively Secure Public-key Broadcast Encryption With Optimal Ciphertext Size

In this section, we build broadcast encryption based on our key exchange mechanism. We begin by defining a broadcast encryption scheme and what it means to be secure. A (public-key) broadcast encryption system [FN94] is made up of three randomized algorithms:

$\mathsf{Setup}(\lambda, N)$ Given the security parameter λ and the number of receivers N, output N private keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ and public parameters params. For $i = 1, \ldots, N$, recipient number i is given the private key sk_i.

$\mathsf{Enc}(\mathsf{params}, S)$ Takes as input a subset $S \subseteq \{1, \ldots, N\}$, and the public parameters params. It outputs a pair (Hdr, k) where Hdr is called the header and $k \in \mathcal{K}$ is a message encryption key chosen from a key space $\mathcal{K}$. We will often refer to Hdr as the broadcast ciphertext.

Let m be a message to be broadcast that should be decipherable precisely by the receivers in S. Let c_m be the encryption of m under the symmetric key k. The broadcast data consists of (S, Hdr, c_m). The pair (S, Hdr) is often called the full header and c_m is often called the broadcast body.

$\mathsf{Dec}(\mathsf{params}, i, \mathsf{sk}_i, S, \mathsf{Hdr})$ Takes as input a subset $S \subseteq \{1, \ldots, N\}$, a user id $i \in \{1, \ldots, N\}$ and the private key sk_i for user i, and a header Hdr. If $i \in S$ the algorithm outputs a key $k \in \mathcal{K}$. Intuitively, user i can then use k to decrypt the broadcast body c_m and obtain the message m.

The above definition describes a public-key broadcast encryption scheme. In a secret-key broadcast system, the encryption algorithm Enc requires as an additional input a private broadcast key bk that is only known to the broadcaster.

The **length efficiency** of a broadcast encryption system is measured in the length of the header Hdr. The shorter the header, the more efficient the system. Some systems such as [BGW05, Del07, DPP07, BS03, SF07] achieve a fixed size header that depends only on the security parameter and is independent of the size of the recipient set S.

As usual, we require that the system be correct, namely that for all subsets $S \subseteq \{1, \ldots, n\}$ and all $i \in S$ if $(\mathsf{params}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N)) \xleftarrow{R} \mathsf{Setup}(\lambda, N)$ and $(\mathsf{Hdr}, k) \xleftarrow{R} \mathsf{Enc}(\mathsf{params}, S)$ then $\mathsf{Dec}(\mathsf{params}, i, \mathsf{sk}_i, S, \mathsf{Hdr}) = k$.

Distributed broadcast encryption. Existing public-key broadcast encryption systems with short ciphertexts [BGW05, DPP07, Del07, GW09, BS03, SF07] require that key generation is done by a central setup algorithm. Participants are given their secret keys by a central authority and this central authority can decrypt all broadcasts.

The broadcast encryption systems we present are the first short-ciphertext systems to support *distributed* key generation, where each user generates a secret key for itself and there is no central authority. In such systems, the Setup algorithm is divided into two randomized algorithms:

- $\mathsf{Setup}'(\lambda, N)$: given the maximum number of users N outputs system parameters params, and
- $\mathsf{Join}(\mathsf{params}, i)$: outputs a pair $(\mathsf{sk}_i, \mathsf{pv}_i)$.

Algorithm $\mathsf{Setup}(\lambda)$ initializes the system. Then every recipient $i = 1, \ldots, N$ generates a public/private key pair for itself by running $\mathsf{Join}(\mathsf{params}, i)$. The overall system's public key consists of params and all the public values generated by Join.

Security. We consider several notions of security for broadcast encryption systems. The strongest is adaptive security, where an adversary $\mathcal{A}$ that adaptively obtains recipient keys sk_i of its choice cannot break the semantic security of a broadcast ciphertext intended for a recipient set S^* for which $\mathcal{A}$ has no secret keys. More precisely, security is defined using the following experiment, denoted $\mathtt{EXP}(b)$, parameterized by the total number of recipients N and by a bit $b \in \{0, 1\}$:

$(\mathsf{params}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N)) \xleftarrow{R} \mathsf{Setup}(\lambda, N)$

$b' \leftarrow \mathcal{A}^{\mathsf{RK}(\cdot), \mathsf{SK}(\cdot), \mathsf{RoR}(b, \cdot)}(\lambda, N)$

where

$\mathsf{RK}(i)$ is a recipient key oracle that takes as input $i \in [N]$ and returns sk_i,

$\mathsf{SK}(S)$ is an oracle that takes as input $S \subseteq [N]$ and returns $\mathsf{Enc}(\mathsf{params}, S)$, and

$\mathsf{RoR}(b, S^*)$ is a real-or-random oracle: it takes as input $b \in \{0, 1\}$ and $S^* \subseteq [n]$, computes $(\mathsf{Hdr}, k_0) \xleftarrow{R} \mathsf{Enc}(\mathsf{params}, S^*)$ and $k_1 \xleftarrow{R} \mathcal{K}$, and returns (Hdr, k_b).

We require that all sets S^* given as input to oracle RoR are distinct from all sets S given as input to SK and that S^* does not contain any index i given as input to RK. For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\mathtt{EXP}(b)$ and as usual define $\mathtt{BE}^{(\mathrm{adv})}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 4.1. We say that a broadcast encryption system is adaptively secure if for all probabilistic polynomial time adversaries $\mathcal{A}$ the function $\mathtt{BE}^{(\mathrm{adv})}(\lambda)$ is negligible.

Next, we consider two weaker versions of security. The first is *semi-static* security, where $\mathcal{A}$ is required to commit to a set $\hat{S}$ of users before seeing the public parameters. All recipient key queries are required to be outside of $\hat{S}$, and all real-or-random queries must be for recipient sets S^* that are a subset of $\hat{S}$.

Definition 4.2. A broadcast encryption system is semi-statically secure if, for all probabilistic polynomial times adversaries $\mathcal{A}$ meeting the following conditions, the function $\mathtt{BE}^{(\mathrm{adv})}(\lambda)$ is negligible:

- $\mathcal{A}$ commits to a set $\hat{S}$ of users before seeing the public parameters.
- Each query to RK must be on a user i outside the set $\hat{S}$.
- Each query to RoR must be on a set S^* that is a subset of $\hat{S}$.

Gentry and Waters [GW09] give a simple conversion from any semi-static broadcast encryption scheme on $2N$ users to an adaptively secure broadcast encryption scheme on N users. This gives the following theorem:

Theorem 4.3 ([GW09])**.** *Given any broadcast encryption scheme that is semi-statically secure for $2N$ users, it is possible to construct an adaptively secure broadcast encryption scheme for N users. If h is the header size of the original encryption scheme, then a broadcast to a set S in the new scheme will have a header will have size $|S| + 2h + O(\lambda)$.*

The final security notion we consider is *selective* security, where $\mathcal{A}$ must commit to the set S^* itself before seeing the public parameters:

Definition 4.4. A broadcast encryption system is selectively secure if, for all probabilistic polynomial times adversaries $\mathcal{A}$ meeting the following conditions, the function $\mathtt{BE}^{(\mathrm{adv})}(\lambda)$ is negligible:

- $\mathcal{A}$ commits to a set S^* before seeing the public parameters.
- $\mathcal{A}$ makes only a single query to RoR, and this query is on the set S^*.

4.1 Broadcast Encryption From Key Exchange

Any multiparty non-interactive key exchange (NIKE) protocol gives a broadcast encryption scheme. Moreover, if the $\mathsf{Publish}$ step is independent of the public parameters, the resulting scheme is distributed. We give two variants: a distributed secret-key scheme and a distributed public-key scheme. We start with the secret-key scheme:

$\mathsf{Setup}(\lambda, N)$ Run the key exchange setup algorithm to obtain public parameters params'. Also run $(\mathsf{sk}_{i,0}, \mathsf{pv}_{i,0}) \leftarrow \mathsf{Publish}(\lambda, N, i)$ for $i \in [N]$. The public key is $(\mathsf{params}', \{\mathsf{pv}_{i,1}\}_{i \in [N]})$.

$\mathsf{Join}(\lambda, N, i)$ Run $(\mathsf{sk}_{i,1}, \mathsf{pv}_{i,1}) \leftarrow \mathsf{Publish}(\lambda, N, i)$. User i publishes $\mathsf{pv}_{i,1}$ and keeps $\mathsf{sk}_{i,1}$ as its secret key. The overall public key is $\mathsf{params} = (\mathsf{params}', \{\mathsf{pv}_{i,b}\}_{i\in[N],b\in\{0,1\}})$.

$\mathsf{Enc}(\mathsf{params}, \mathsf{bk}, S)$ Let $S_i = 1$ for $i \in S$ and $S_i = 0$ for $i \notin S$. Let $k \leftarrow \mathsf{KeyGen}(\mathsf{params}', 1, \mathsf{sk}_{1,1}, \{\mathsf{pv}_{i,S_i}\}_{i\in[N]})$. Output $(\mathsf{Hdr} = \emptyset, k)$.

$\mathsf{Dec}(\mathsf{params}, i, \mathsf{sk}_i, S)$ If $i \notin S$, abort. Otherwise, let $k \leftarrow \mathsf{KeyGen}(\mathsf{params}', i, \mathsf{sk}_{i,1}, \{\mathsf{pv}_{i,S_i}\}_{i\in[N]})$.

This construction is reminiscent of the construction of Boneh and Silverberg [BS03] — indeed, building key exchange from multilinear maps and then applying this conversion gives a variant of their scheme. We obtain the following theorem:

Theorem 4.5. *Given a selectively, semi-statically, or adaptively secure non-interactive key exchange protocol for N users, it is possible to construct a selectively, semi-statically, or adaptively secure distributed secret-key broadcast encryption for N users, respectively. The header size will be 0.*

We can also easily produce a public key scheme by having the sender pretend to be one of the parties in the key exchange:

$\mathsf{Setup}(\lambda, N)$ Run the key exchange setup for algorithm for $N + 1$ users to obtain public parameters params'. Also run $(\mathsf{sk}_{i,0}, \mathsf{pv}_{i,0}) \leftarrow \mathsf{Publish}(\lambda, N, i)$ for $i \in [N]$. The public key is $\mathsf{params} = (\mathsf{params}', \{\mathsf{pv}_{i,0}\}_{i\in[N]})$.

$\mathsf{Join}(\lambda, N, i)$ Run $(\mathsf{sk}_{i,1}, \mathsf{pv}_{i,1}) \leftarrow \mathsf{Publish}(\lambda, N, i)$. User i publishes $\mathsf{pv}_{i,1}$ and keys $\mathsf{sk}_{i,1}$ as its secret key. The overall public key is $\mathsf{params} = (\mathsf{params}', \{\mathsf{pv}_{i,b}\}_{i\in[N],b\in\{0,1\}})$.

$\mathsf{Enc}(\mathsf{params}, S)$ Let $S_i = 1$ for $i \in S$, $S_i = 0$ for $i \notin S$, and $S_{N+1} = 1$. Run $(\mathsf{sk}_{N+1,1}, \mathsf{pv}_{N+1,1}) \leftarrow \mathsf{Publish}(\lambda, N, N + 1)$. Compute $k \leftarrow \mathsf{KeyGen}(\mathsf{params}', N + 1, \mathsf{sk}_{N+1,1}, \{\mathsf{pv}_{i,S_i}\}_{i\in[N+1]})$. Output $(\mathsf{Hdr} = \mathsf{pk}_{N+1,1}, k)$.

$\mathsf{Dec}(\mathsf{pk}, i, \mathsf{sk}_i, S, \mathsf{Hdr} = \mathsf{pv}_{N+1,1})$ If $i \notin S$, abort. Otherwise, let $k \leftarrow \mathsf{KeyGen}(\mathsf{params}', i, \mathsf{sk}_{i,1}, \{\mathsf{pv}_{i,S_i}\}_{i\in[N+1]})$.

We obtain the following theorem:

Theorem 4.6. *Given a selectively, semi-statically, or adaptively secure non-interactive key exchange protocol for $N + 1$ users it is possible to construct a selectively, semi-statically, or adaptively secure distributed public-key broadcast encryption for N users. The header size will be the length of a single user's public value.*

Instantiating Theorem 4.6 with the NIKE from Section 3.1 gives a header size that is constant with respect to the number of users. The resulting semi-static broadcast system can be converted to an adaptively-secure scheme using the generic conversion of Gentry and Waters [GW09]. Therefore, using indistinguishability obfuscation and constrained PRFs, it is possible to build a public-key adaptively secure distributed broadcast with constant size ciphertext.

We note that applying Theorem 4.6 to the existing N-user NIKE protocols from an $(N - 1)$ linear map [GGH13a, CLT13] results in a non-interesting broadcast system because the resulting broadcast system will have $\Omega(N)$ size headers. This makes it worse than the trivial broadcast encryption scheme. The reason for the large header size is that existing N-linear maps require $\Omega(N)$ bits to encode a single element.

To conclude this section, we prove Theorem 4.6, which is very similar to the proof of Theorem 4.5:

Proof. We prove security for the adaptive case, the other proofs being nearly identical. Suppose we have an adversary $\mathcal{A}$ that breaks the broadcast encryption scheme. $\mathcal{A}$ receives the public parameters, and makes the following queries:

- Secret key queries: $\mathcal{A}$ submits a user i, and receives the secret key sk_i for that user.
- Challenge query: $\mathcal{A}$ submits a set S of users for which he does not have any secret keys. If $b = 0$, the challenger responds with the correct header. Otherwise if $b = 1$, the challenger responds with a random key.

We create a simple adversary $\mathcal{B}$ that breaks the adaptive security of the underlying key exchange protocol. $\mathcal{B}$ receives public parameters params' from the key exchange challenger. $\mathcal{B}$ also makes $2N$ register honest user queries for identities $\mathsf{id} = (i, b), i \in [N], b \in \{0, 1\}$, receiving public values $\mathsf{pv}_{i,b}$ in return. $\mathcal{B}$ sends to $\mathcal{A}$ the public parameters $\mathsf{params} = (\mathsf{params}', \{pv_{i,b}\}_{i\in[N],b\in\{0,1\}})$. Now $\mathcal{B}$ simulates $\mathcal{A}$. When $\mathcal{A}$ makes a secret key query for user i, $\mathcal{B}$ makes an extract query for the identity $\mathsf{id} = (i, 1)$. When $\mathcal{A}$ makes a challenge query on set S, $\mathcal{B}$ registers a new user $\mathsf{id} = (N + 1, 1)$ (with public value $\mathsf{pv}_{N+1,1}$), and then makes a test query on the identities (i, S_i) where $i \in [N + 1]$ and S_i is 1 for all $i \in S$, 0 for $i \notin S$, and 1 for $i = N + 1$. $\mathcal{B}$ returns $\mathsf{pv}_{N+1,1}$ as the header, and the response from the test query as the key. At the end of the game, when $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs the same bit. $\mathcal{B}$ correctly simulates $\mathcal{A}$ in both the $b = 0$ and $b = 1$ case, so $\mathcal{B}$ has the same success probability as $\mathcal{A}$. Therefore, since the key exchange protocol is secure, so is the broadcast encryption scheme. □

5 Recipient-Private Broadcast Encryption

In this section, we build recipient-private broadcast encryption with short ciphertexts [BBW06]. As in the previous section, our schemes will be secure in the *semi-static* model. The generic transformation of Gentry and Waters [GW09] from semi-static to adaptive security also applies to private broadcast, and we therefore obtain an adaptively-secure recipient-private broadcast system.

A recipient-private broadcast encryption system is a broadcast system in which the ciphertext reveals no information about the intended recipient set (beyond what is explicitly allowed). The system is made up of three algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ as in the public-set settings, except that the input to Dec does not include the intended recipient set S. That is, Dec takes as input $(\mathsf{params}, i, \mathsf{sk}_i, \mathsf{Hdr})$ and outputs a key $k \in \mathcal{K}$ or $\perp$.

Security. Recipient-private broadcast systems often need only broadcast to a specific collection of user sets $\mathcal{S} \subset 2^{[N]}$ and security is defined with respect to this collection $\mathcal{S}$. The attacker should be unable to distinguish a broadcast to one set $S_0 \in \mathcal{S}$ from a broadcast to another set $S_1 \in \mathcal{S}$ (subject to some natural constraints on the choice of S_0 and S_1 explained below). The set systems of interest to us are:

- $2^{[N]}$, the set of all subsets. Since the ciphertext should reveal nothing about which $S \in \mathcal{S}$ is the target set, a system capable of broadcasting to any subset of $[N]$ must reveal nothing about the recipient set, not even its size. Our constructions achieve header size $O(\lambda + N)$.
- $\binom{[N]}{r}$, the collection of subsets of size exactly r. A system capable of broadcasting to any set in $\binom{[N]}{r}$ may reveal the size r of the recipient set, but should reveal nothing else about the set. Our constructions achieve header size $O(\lambda + r \log N)$.

- $\mathtt{Lin}_N = \{\emptyset = [0], [1], \ldots, [N]\}$. Privacy with respect to this set system is needed for traitor tracing. Our constructions achieve header size $O(\lambda + \log N)$.

Recipient privacy with respect to a given set system $\mathcal{S}$ states that an attacker who specifies two recipient sets $S_0, S_1 \in \mathcal{S}$ should be unable to distinguish a broadcast encryption to S_0 from a broadcast to S_1, even if the attacker is given private keys for all users in $S_0 \cap S_1$ and $[N] \setminus (S_0 \cup S_1)$. This is the maximum number of keys we can give the attacker since any other key trivially lets the attacker distinguish a broadcast to S_0 from a broadcast to S_1. More precisely, security is defined using the following experiment $\mathtt{EXP}(b)$ on an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ parameterized by the total number of recipients N and by a bit $b \in \{0, 1\}$:

$(\mathsf{params}, \mathsf{bk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N)) \xleftarrow{R} \mathsf{Setup}(\lambda, N)$

$b' \leftarrow \mathcal{A}^{\mathsf{RK}(\cdot), \mathsf{SK}(\cdot), \mathsf{Ch}(b, \cdot, \cdot)}(\lambda, n)$

where

$\mathsf{RK}(i)$ is the recipient key oracle that takes as input at index $i \in [N]$, and returns the secret key sk_i for user i.

$\mathsf{SK}(S)$ is an oracle that takes as input $S \in \mathcal{S}$ and returns $\mathsf{Enc}(\mathsf{bk}, S)$, and

$\mathsf{Ch}(b, S_0, S_1)$ takes as input a bit $b \in \{0, 1\}$ and two sets $S_0, S_1 \in \mathcal{S}$ and returns a challenge ciphertext $\mathsf{Enc}(\mathsf{bk}, S_b)$.

We require that for each challenge on (S_0, S_1) and each recipient key query for $i \in [N]$, that i is not in the symmetric difference of S_0 and S_1, namely $i \notin S_0 \Delta S_1$. In other words, the adversary's secret keys cannot trivially allow it to break the scheme. For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\mathtt{EXP}(b)$. Define $\mathtt{PBE}^{(\mathrm{adv})}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 5.1. We say that a broadcast encryption system is $\mathcal{S}$-recipient-private (semi-static) semantically secure if it is a (semi-static) semantically secure broadcast system and for all probabilistic polynomial time adversaries $\mathcal{A}$ the function $\mathtt{PBE}^{(\mathrm{adv})}(\lambda)$ is negligible.

Definition 5.1 captures recipient privacy for a secret-key broadcast system: the encryption key bk is kept secret and not given to the adversary in the security experiment $\mathtt{EXP}(b)$. Security for a public-key recipient-private system is defined analogously with bk removed from the definition of $\mathtt{EXP}(b)$.

5.1 A Construction With Optimal-Size Ciphertexts

In recipient-private broadcast encryption, the set of users is no longer transmitted in the clear, and must instead be kept secret. Our plan is to broadcast to a recipient set S by embedding an encryption of the set S in the broadcast ciphertext. The public-key will contain an obfuscated program that decrypts the encrypted recipient set S and then outputs a message decryption key only if the recipient can prove it is a member of S. However, encrypting the set S so that we can prove security using iO requires some work and in particular, requires that S be encrypted using a symmetric cipher that supports a certain key-malleability property.

In more detail, each user's private key will be a random seed s_i, and we let $x_i = \mathsf{PRG}(s_i)$ as in the previous section. We need to allow user i to learn the message decryption key for all sets S containing i. To that end, we include in the public key an obfuscated program that takes three inputs: an encrypted recipient set, an index i, and a seed s_i. The program decrypts the

encrypted set, checks that the index i is in the set, and that the seed s_i is correct for that index (i.e. $x_i = \mathsf{PRG}(s_i)$). If all the checks pass, the program evaluates some pseudorandom function on the ciphertext to obtain the message decryption key and outputs that key.

Unfortunately, encrypting the recipient set S using a generic CPA-secure encryption scheme is insufficient for providing recipient privacy. The problem is that ciphertexts may be malleable: an attacker may be able to transform an encryption of a set S containing user i into an encryption of a set S' containing user j instead (that is, j is in S' if and only if i is in S). Now the attacker can use user j's secret key to decrypt the broadcast ciphertext. If decryption succeeds the attacker learns that user i is in the original ciphertext's recipient set, despite not having user i's secret key. This violates recipient privacy.

The obvious approach to preventing this malleability attack is to authenticate the ciphertext using a MAC or a signature. This would avoid the potential attack from the previous paragraph, but unfortunately is not sufficient to prove security using iO. First, if a MAC is used, then since verification is performed by the obfuscated program, it is not clear that obfuscating the MAC verification program prevents the attacker from modifying authenticated ciphertexts. Second, for authentication using either MACs or signatures, even if the adversary cannot construct forged valid ciphertexts by itself, we know that they do exist. Therefore, there may still be inputs that reveal whether $i \in S$, even if a secret key for user i does not exist. This presents a problem in the iO security proof, because we would like to first change the parameters so that there is no secret key for user i, and then replace the obfuscated program with the obfuscation of another program that contains no information about whether $i \in S$. However, to apply indistinguishability obfuscation, we need this new program to have *identical* functionality to the original, but that will be false: there exist forged valid encryptions of the recipient set that will distinguish the modified program from the original as done in the malleability attack above. In other words, because iO can only be applied to two programs that agree on every input — including on hard-to-compute inputs — authenticating the encrypted recipient set with a MAC or a signature does not resolve the malleability problem.

The malleable-key method. To solve the difficulty discussed above, we develop a solution based on specialized encryption schemes tailored to each of the three set systems $\mathcal{S}$ discussed above: $2^{[N]}$, $\binom{[N]}{r}$, and $\mathtt{Lin}_N$. These encryption systems will be malleable with respect to the key, which will enable us to prove security.

We start by defining an encryption scheme for each of the three set systems of interest. Let $E : \mathcal{K} \times \mathcal{S} \to \mathcal{S}$ and $D : \mathcal{K} \times \mathcal{S} \to \mathcal{S}$ be an "encryption scheme for sets." That is, the plaintext space for these schemes are sets $S \in \mathcal{S}$. The encryption schemes are defined as follows:

- For $\mathcal{S} = \mathtt{Lin}_N$: the key space $\mathcal{K}$ is the set of all $N!$ permutations σ on $\{0, 1, \ldots N\}$. Encryption and decryption are defined as:

$$E(\sigma, [r]) = [\sigma(r)] = \{1, 2, \ldots, \sigma(r)\}$$
$$D(\sigma, [r]) = [\sigma^{-1}(r)] = \{1, 2, \ldots, \sigma^{-1}(r)\}$$

  For $i \in [N]$, this scheme has the following malleability property: for $\sigma \in \mathcal{K}$ let $\sigma' \in \mathcal{K}$ be the same permutation as σ except that it swaps the image of i and $i - 1$. Then

$$\text{if } D(\sigma, c) = [i] \text{ then } D(\sigma', c) = [i - 1] \ .$$

By changing the key from σ to σ', we changed the decrypted set from $[i]$ to $[i-1]$. In the construction below, an observer will not be able to detect this change to the key, making it impossible to tell whether the broadcast ciphertext is encrypted for the set $[i]$ or $[i-1]$. We use this to prove recipient privacy.

- For $\mathcal{S} = 2^{[N]}$: the key space is $\mathcal{K} = 2^{[N]}$ and encryption of a set S is defined as $E(T, S) = T \Delta S$. Similarly, $D(T, c) = T \Delta c$. (the Δ operator denotes the symmetric difference of two sets)

  For $i \in [N]$ this scheme has the following malleability property: let $T \in \mathcal{K}$ and $T' \in \mathcal{K}$ be the same set as T, but with element i added or removed: $T' = T \Delta \{i\}$. Then

  $$\text{if } i \in D(T, S) \text{ then } i \notin D(T', S) \ .$$

  By changing the key from T to T', we flip whether or not i is in the decrypted set.

- For $\mathcal{S} = \binom{[N]}{r}$: the key space is $\mathcal{K} = S_N$ (the set of all permutations σ on $[N]$). We define:

  $$E(\sigma, S) = \sigma(S) = \{\sigma(i) : i \in S\}$$
  $$D(\sigma, c) = \sigma^{-1}(c) = \{\sigma^{-1}(i) : i \in c\}$$

  For $i, j \in [N]$ with $i \neq j$, this scheme has the following malleability property: let $\sigma \in \mathcal{K}$ and let $\sigma' \in \mathcal{K}$ be the permutation $\sigma \circ \langle i \ \ j \rangle$. That is, σ' is σ composed with the 2-cycle exchanging i and j. Then

  $$\text{if } i \in D(\sigma, S) \text{ and } j \notin D(\sigma, S) \text{ then } i \notin D(\sigma', S) \text{ and } j \in D(\sigma', S) \ .$$

  By changing σ to σ', we remove i from the decrypted set and add j.

The construction. Using the encryption schemes above, we can now define our recipient-private broadcast construction. For now, we describe the secret-key variant of the scheme. We show at the end of this section how to make the scheme public-key.

Construction 5.2. *Let $\mathcal{S} = 2^{[N]}, \mathtt{Lin}_N$, or $\binom{[N]}{r}$ be a collection of subsets. Let $E : \mathcal{K} \times \mathcal{S} \to \mathcal{S}$ and $D : \mathcal{K} \times \mathcal{S} \to \mathcal{S}$ be the encryption scheme for $\mathcal{S}$ described above. Let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$. The recipient-private broadcast encryption scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ works as follows:*

$\mathsf{Setup}(\lambda, N)$ *Sample s_i at random from $\{0,1\}^\lambda$ for each $i \in [N]$, and let $x_i = \mathsf{PRG}(s_i)$. Sample constrained pseudorandom functions $\mathsf{PRF}_1 : \{0,1\}^\lambda \to \mathcal{K}$ and $\mathsf{PRF}_2 : \{0,1\}^\lambda \times \mathcal{S} \to \{0,1\}^\lambda$. Let $P_{PBE-\mathsf{Dec}}$ be the program given in Figure 3, padded to the appropriate size. Player i receives the private key s_i, and the public key is $\mathsf{iO}(P_{PBE-\mathsf{Dec}})$.*

$\mathsf{Enc}((\mathsf{PRF}_1, \mathsf{PRF}_2), S)$ *Generate a random seed r and compute $k \leftarrow \mathsf{PRF}_1(r)$. Next compute $c \leftarrow E(k, S)$ and $k_S \leftarrow \mathsf{PRF}_2(r, c)$. Output $(\mathsf{Hdr} = (r, c), k)$*

$\mathsf{Dec}(\mathsf{params}, r, c, s_i, i)$ *Run $k_S \leftarrow P_{PBE-\mathsf{Dec}}(r, c, s_i, i)$.*

Correctness is trivial by inspection, given that $D(k, E(k, S)) = S$ for each of the examples. Security is summarized by the following theorem:

Inputs: r, c, s, i
Constants: $x_1, \ldots x_N, \mathsf{PRF}_1, \mathsf{PRF}_2$

1. Let $k \leftarrow \mathsf{PRF}_1(r)$
2. Let $S \leftarrow D(k, c)$ // decrypt c to obtain the set S
3. Check that $\mathsf{PRG}(s) = x_i$ and $i \in S$
4. If check fails, output $\perp$
5. Otherwise, output $\mathsf{PRF}_2(r, c)$

Figure 3: The program $P_{PBE-\mathsf{Dec}}$.

Theorem 5.3. *For $\mathcal{S} = 2^{[N]}, \mathtt{Lin}_N$, or $\binom{[N]}{r}$, if iO is a secure indistinguishability obfuscator, PRF_1 is a secure punctured PRF, and PRF_2 is a secure constrained PRF for circuit predicates, then Construction 5.2 is a secure recipient-private broadcast encryption scheme.*

We prove Theorem 5.3 in a more general form in Section 5.3. For now we sketch its proof for the linear set system $\mathtt{Lin}_N = \{\emptyset, [1], \ldots, [N]\}$.

Proof sketch for $\mathcal{S} = \mathtt{Lin}_N$. We prove that the scheme satisfies Definition 5.1. Incurring only a polynomial loss in security, we can require the adversary $\mathcal{A}$ to commit to a challenge index i at the beginning of the experiment. Then $\mathcal{A}$'s goal is to distinguish encryptions to the set $[i]$ from encryptions to the set $[i-1]$, given the secret keys to all $j \neq i$. Next, we observe that, by security of PRG, we can replace user i's public value x_i with a uniformly random string. This means the index i can never be used to decrypt. The next step is to puncture PRF_1 at the challenge randomness r^*, and include the "challenge key" $k^* = \mathsf{PRF}_1(r^*)$ hard-coded in the program $P_{PBE-\mathsf{Dec}}$. The resulting program $P'_{PBE-\mathsf{Dec}}$ is shown in Figure 6. This does not change the functionality, and therefore the indistinguishability of iO shows that $\mathcal{A}$ cannot tell the difference. Now, the security of PRF_1 implies that we can actually choose k^* at random, independent of PRF_1. Recall that k^* is just a random permutation on $\{0, \ldots, N\}$. In particular, given k^*, there is another permutation k' that is identical to k^*, except that it flips the image of i and $i-1$. Since i can never be used to decrypt, exchanging k^* for k' does not change the functionality of the decryption algorithm. So the adversary cannot tell whether k^* or k' is encrypting S. However, exchanging k^* for k' flips an encryption of $[i]$ for $[i-1]$, meaning the adversary cannot tell which set we encrypted to. □

A public-key system. As described, our scheme requires the secret broadcast key in order to encrypt. However, it is straightforward to allow anyone to encrypt. The idea is to include an obfuscated program for encryption. This does not quite work, as it would give everyone the ability to query PRF_1 directly. Instead, we use the trick of Sahai and Waters [SW13] and obfuscate the program that takes the randomness, applies a pseudorandom generator, and then proceeds to encrypt using the output of the pseudorandom generator as randomness. In particular, we obfuscate the program $P_{PBE-\mathsf{Enc}}$ in Figure 4 (as usual, padded to the appropriate size), and include it in the public key. The idea is that the r^* created by the challenger is (with overwhelming probability) not

a PRG sample, so giving out the program $P_{PBE-\mathsf{Enc}}$ does not help the adversary learn anything about PRF_1 or PRF_2 at the point r^*.

Inputs: S, t
Constants: $\mathsf{PRF}_1, \mathsf{PRF}_2$

1. Let $r \leftarrow \mathsf{PRG}(t)$
2. Let $\sigma \leftarrow \mathsf{PRF}_1(r)$
3. Let $c \leftarrow E(\sigma, S)$
4. Let $k \leftarrow \mathsf{PRF}_2(r, c)$
5. Output $(\mathsf{Hdr} = (r, c), k)$

Figure 4: The program $P_{PBE-\mathsf{Enc}}$.

5.2 Applications

5.2.1 Traitor Tracing with Short Ciphertexts and Secret Keys

Instantiating our scheme with $\mathcal{S} = \mathtt{Lin}_N$, we get a secure private linear broadcast encryption (PLBE) scheme. We can then appeal to the generic conversion of Boneh, Sahai, and Waters [BSW06] to get a traitor tracing scheme with the same parameters. The resulting traitor tracing scheme has the following characteristics:

- **Short ciphertexts**: The header consists of a nonce $r \in \{0,1\}^\lambda$ and a set $c = [r]$ for $r \in \{0, \ldots, N\}$. We can therefore represent c by the integer r. The header thus has length $\lambda + \log(N+1)$. If we are encrypting a message of length m, the ciphertext size will then be $\lambda + \log(N+1) + m$, which is optimal in N and m.
- **Short secret keys**: The secret keys in our system are just pseudorandom generator seeds, and have length λ, independent of N.
- **Public traceability**: In our scheme, anyone can run the tracing algorithm. If this is undesirable, the program $P_{PBE-\mathsf{Enc}}$ can be modified so it only accepts the set $S = [N]$.
- **Long public keys**: The public key in our scheme consists of the obfuscation of two programs, both programs being linear in size. Therefore, the public keys of our system are quite large, and there is room for improvement.

We note that, since the collection of possible recipient sets is linear in size, static and adaptive security are equivalent. This means that we can actually use a *punctured* PRF instead of a constrained PRF in our scheme, and still achieve full security, without any complexity leveraging arguments. In particular, this means our traitor tracing scheme can be built from any indistinguishability obfuscator and any one-way function.

Inputs: $N, r \in \{0, \ldots, N\}, i \in [\binom{N}{r}]$
Output: $S \subseteq [N]$ where $|S| = r$

1. If $r = 0$, then $i = 1$. Return $\emptyset$.
2. If $r = N$, then $i = 1$. Return $[N]$.
3. Otherwise, if $i \leq \binom{N-1}{r-1}$, let $S' \leftarrow \mathsf{Int2Set}(N-1, r-1, i)$, and output $S = S' \bigcup \{N\}$.
4. Otherwise, $i > \binom{N-1}{r-1}$. Let $i' = i - \binom{N-1}{r-1}$. Then $i' \leq \binom{N-1}{r}$.
   Output $S \leftarrow \mathsf{Int2Set}(N-1, r, i')$

Figure 5: The algorithm Int2Set.

5.2.2 Private Broadcast Encryption with Optimal Ciphertext Length

Instantiating our scheme with $\mathcal{S} = \binom{[N]}{r}$, we can encrypt to an arbitrary set of size r without revealing r. To analyze the ciphertext size, we need to determine a compact representation for an arbitrary set of size r. Using the algorithm Int2Set in Figure 5, we can map integers in the range $\{1, \ldots, \binom{N}{r}\}$ into distinct subsets of size r. This map is bijective, and the inverse is efficiently computable. Thus, we can represent a set of size r using $\log \binom{N}{r}$ bits, which is optimal. The broadcast header size will therefore be $\lambda + \log \binom{N}{r}$.

5.3 Proving security for general set systems

We now turn to proving Theorem 5.3. Instead of proving the theorem for each of the three set systems $\mathcal{S}$ of interest, we prove a more general theorem. First we need some terminology to capture the key malleability properties needed for the proof.

Definition 5.4 ($\mathcal{E}$-path)**.** Let $\mathcal{S}$ and $\mathcal{E}$ be collections of subsets of $2^{[N]}$. Let P be a sequence of sets in $\mathcal{S}$:

$$S_0 \to S_1 \to \cdots \to S_k$$

We say that P is an $\mathcal{E}$-path from S_0 to S_k if:

- $S_i \Delta S_{i+1} \in \mathcal{E}$ for all i
- $S_i \Delta S \subsetneq S_{i+1} \Delta S$ for all i

An $\mathcal{E}$-path from S to T is therefore a sequence of sets where the symmetric difference between two adjacent sets is an element of $\mathcal{E}$ and each set in the sequence is "closer" to T than the previous set.

Definition 5.5 ($\mathcal{E}$-Connected Set)**.** Let $\mathcal{S}$ and $\mathcal{E}$ be collections of subsets of $2^{[N]}$. We say that $\mathcal{S}$ is $\mathcal{E}$-connected if, between any two sets $S, T \in \mathcal{S}$, there is an $\mathcal{E}$-path starting at S and ending at T. We say that $\mathcal{S}$ is $\mathcal{E}$-efficiently connected if such a path can be found in polynomial time.

By restricting to efficiently-connected collections, a simple argument allows us to assume that the adversary's challenge must consist of two sets that differ by a set $e^* \in \mathcal{E}$. We can also require the adversary to commit to the set e^* at the beginning of the game, incurring only a polynomial loss in security.

For any collection $\mathcal{S} \subseteq 2^{[N]}$ that is $\mathcal{E}$-efficiently connected, let Flip_e for $e \in \mathcal{E}$ be the following permutation on $\mathcal{S}$:

$$\mathsf{Flip}_e(S) = \begin{cases} S & \text{if } S \Delta e \notin \mathcal{S} \\ S \Delta e & \text{if } S \Delta e \in \mathcal{S} \end{cases}$$

Flip_e takes the exclusive difference with e if it can, and otherwise leaves its input unaffected. Notice that Flip_e has order 2: $\mathsf{Flip}_e(\mathsf{Flip}_e(S)) = S$.

We can now simplify the game even further: the adversary commits to a set $e^* \in \mathcal{S}$ at the beginning of the game. In his challenge, he submits just a single set S, and we encrypt to the set S or the set $\mathsf{Flip}_{e^*}(S)$. The adversary must tell which set we encrypted to.

Now we can discuss what kind of encryption scheme E we need to securely encrypt S. We note that for a fixed key k, $E(k, \cdot)$ is just a permutation on sets. So we will associate keys k with permutations $\sigma(\cdot) = E(k, \cdot)$. The question then becomes: What class of permutations do we need for our scheme to be secure?

Recall that, together with the secret key for i, the public key can be used as a membership oracle answering questions of the form "is i in S?" We need that, even with such an oracle for each $i \notin e^*$, the adversary still cannot determine which set the message is encrypted to. Intuitively, we need that, for every permutation σ that may be used to encrypt S, there is another permutation σ' that preserves the adversary's membership oracle, but exchanges encryptions of S with encryptions of $\mathsf{Flip}_{e^*}(S)$. In other words, $\sigma' = \sigma \circ \mathsf{Flip}_{e^*}$. Therefore, at a minimum, we need the set of permutations to include the group generated by the Flip_e permutations:

Definition 5.6 (Flip Group)**.** Let $\mathcal{S} \subseteq 2^{[N]}$ be $\mathcal{E}$-efficiently connected. Let $G_{\mathsf{Flip}}(\mathcal{S}, \mathcal{E})$, called the *flip group* of $\mathcal{S}$, be the group generated by the Flip_e permutations. When the collections are clear, we will often write G_{Flip}. We say that the flip group is *efficiently represented* if all group elements can be efficiently represented, a random group element efficiently generated, and all group operations and the application of a group element to elements in $\mathcal{S}$ can be computed efficiently.

We can now state a generalization of Theorem 5.3 that encompasses the three collections of sets we are interested in:

Theorem 5.7. *Let $\mathcal{S}$ and $\mathcal{E}$ be collections of sets such that:*

- *$\mathcal{S}$ is $\mathcal{E}$-efficiently connected*
- *$G_{\mathsf{Flip}}(\mathcal{S}, \mathcal{E})$ is efficiently represented*

Let $\mathcal{K} = G_{\mathsf{Flip}}(\mathcal{S}, \mathcal{E})$, $E(\sigma, S) = \sigma(S)$ and $D(\sigma, S) = \sigma^{-1}(S)$. If PRF_1 is a secure punctured PRF, PRF_2 is a secure constrained PRF, and iO is an indistinguishability obfuscator, then Construction 5.2 instantiated with $\mathcal{K}, E, D$ is $\mathcal{S}$-recipient-private semi-static semantically secure.

Before proving Theorem 5.7, we explain why it generalizes Theorem 5.3. In particular, for $\mathcal{S} = 2^{[N]}, \mathtt{Lin}_N$, and $\binom{[N]}{r}$, we give a set $\mathcal{E}$ that efficiently connects $\mathcal{S}$ and determined the flip group $G_{\mathsf{Flip}}(\mathcal{S}, \mathcal{E})$:

- $\mathcal{S} = 2^{[N]}$. Here we let $\mathcal{E} = \{\{1\}, \ldots, \{N\}\}$, the set of singletons. Then $\mathsf{Flip}_{\{i\}}(S) = S\Delta\{i\}$. These flip permutations generate the group $2^{[N]}$, with composition and action on $\mathcal{S}$ given by symmetric difference.

- $\mathcal{S} = \mathtt{Lin}_N$. We again use the set of singletons as $\mathcal{E}$. We note that we can represent an element $[r]$ by the integer r. Under this representation, $\mathsf{Flip}_{\{i\}}(i) = i - 1$, $\mathsf{Flip}_{\{i\}}(i-1) = i$, and $\mathsf{Flip}_{\{i\}}(r) = r$ for $r \neq i, i-1$. Thus $\mathsf{Flip}_{\{i\}}$ is just the 2-cycle $\langle i-1 \;\; i \rangle$. These 2-cycles generate the set of all permutations on $\{0, \ldots, N\}$.

- $\mathcal{S} = \binom{[N]}{r}$. This time we let $\mathcal{E}$ be the set of all distinct pairs on integers in $[N]$. Then $\mathsf{Flip}_{\{i,j\}}(S)$ exchanges i for j and vice versa. Thus, we can associate $\mathsf{Flip}_{\{i,j\}}$ with the 2-cycle $\langle i \;\; j \rangle$ in the sense that $\mathsf{Flip}_{\{i,j\}} = \{\langle i \;\; j \rangle(r) : r \in S\}$. These 2-cycles generate S_N.

Proof. We need to prove that Construction 5.2 is both semi-static semantically secure and has recipient set privacy. We prove security for the private-key setting, the public-key setting being similar.

Semantic security follows a similar argument as in the proof of Theorem 4.6. The main difference is that now the secret key is derived by applying a PRF to the set of users, rather than the public values from the set of users. If we let r^* be the randomness used for the challenge ciphertext, we first puncture PRF_1 at r^* and include the key $\sigma^* \leftarrow \mathsf{PRF}_1(r^*)$ hard-wired in the decryption program. Then we can replace σ^* with a truly random key. Next, we puncture PRF_2 at all "encryptions" of subsets $S \subseteq \hat{S}$ using randomness r^*: that is, at all points $(r^*, E(\sigma^*, S))$ for $S \subseteq \hat{S}$. The derived key for any challenge query the adversary gives us is then independent of the program, and therefore can be replaced with a random key. Therefore, the adversary can not distinguish a random key from the correct key.

Now we turn our focus to proving recipient set privacy. To that end, we assume toward contradiction that there is a polynomial time adversary $\mathcal{A}'$ breaking the privacy of Construction 5.2. $\mathcal{A}'$ succeeds at the following task: First, $\mathcal{A}'$ receives the public parameters. Then $\mathcal{A}'$ then makes several kinds of queries:

- Recipient key queries for index i, for which $\mathcal{A}'$ receives the secret key s_i for user i.

- Message encryption queries for a set S, for which $\mathcal{A}'$ receives an encryption (r, c) to S.

- Challenge queries for sets S_0, S_1, for which $\mathcal{A}'$ receives an encryption (r^*, c^*) to S_b.

We require that each challenge (S_0, S_1) and each recipient key query i satisfies $i \notin S_0 \Delta S_1$. A simple hybrid argument allows us to assume that $\mathcal{A}'$ only makes a single challenge query, receiving a single challenge ciphertext (r^*, c^*) in return. Let $\sigma^* = \mathsf{PRF}_1(r^*)$. Then $c^* = \sigma^*(S_b)$. A' then outputs a guess b' for b. By assumption, $\mathcal{A}'$ distinguishes $b = 0$ from $b = 1$ with non-negligible probability ϵ'.

Our first step is to make two simplifying assumptions: First, assume that $S_0 \Delta S_1 \in \mathcal{E}$. Since it is always possible to compute an $\mathcal{E}$ path from S_1 to S_2 of polynomial length p in polynomial time, it is straightforward to turn an adversary winning in the original game with advantage ϵ' to an adversary winning in this restricted setting with advantage ϵ'/p. We may also require the adversary to commit to $e^* = S_1 \Delta S_2$ at the beginning of the game. Since $\mathcal{E}$ is polynomial q in size, we can just guess e at the beginning, and abort if our guess is incorrect. We can simplify things even further by having the adversary make a challenge on a single set S^*, and then letting $S_0 = S^*$ and $S_1 = \mathsf{Flip}_{e^*}(S^*)$.

Therefore, we obtain an adversary $\mathcal{A}$ winning in this even more restricted setting with probability $\epsilon := \epsilon'/pq$.

Now we prove that no such efficient adversary can exist. We prove security through a sequence of games:

Game 0 This game is the game described above, where $\mathcal{A}$ has advantage ϵ. We can assume that r^* is chosen at the beginning of the game.

Game 1 Here, we answer encryption queries by generating a random r not equal to r^*. Since r is chosen at random from an exponential-sized set, **Game 1** is indistinguishable from **Game 0**.

Game 2 Now, for every $i \in e^*$, we choose x_i uniformly at random. By the security of PRG, this game is indistinguishable from **Game 1**, meaning $\mathcal{A}$ still has advantage $\epsilon - \mathsf{negl}$.

Game 3 Here, we construct a modified program $P'_{PBE-\mathsf{Dec}}$ as in Figure 6, puncturing PRF_1 at r^*, and including $\sigma^* = \mathsf{PRF}_1(r^*)$ explicitly in the constants for the program. We also add a check that $i \notin e^*$. With overwhelming probability, $\mathsf{PRG}(s) \neq x_i$ for any s and x_i for $i \in e^*$, so this check is redundant. The security of iO implies that $\mathcal{A}$ still has advantage at least $\epsilon - \mathsf{negl}$. Note that, since r^* is different from any r used in an encryption query, all encryption queries can be answered using the punctured program $\mathsf{PRF}_1^{\overline{\{r^*\}}}$.

Inputs: r, c, s, i

Constants: $x_1, \ldots x_N$,$\mathbf{PRF}_{\mathbf{1}}^{\overline{\{r^*\}}}, \sigma^*, \mathsf{PRF}_2$

1. Let $\boldsymbol{\sigma} \leftarrow \begin{cases} \mathbf{PRF}_{\mathbf{1}}^{\overline{\{r^*\}}}(\boldsymbol{r}) & \text{if } \boldsymbol{r \neq r^*} \\ \boldsymbol{\sigma^*} & \text{if } \boldsymbol{r = r^*} \end{cases}$
2. Let $S \leftarrow \sigma^{-1}(c)$
3. Check that $\mathsf{PRG}(s) = x_i$ and $i \in S$ and that $\boldsymbol{i \notin e^*}$.
4. If check fails, about $\perp$
5. Otherwise, output $\mathsf{PRF}_2(r, c)$

Figure 6: The program $P'_{PBE-\mathsf{Dec}}$.

Game 4 Sample σ_0^* at random from G_{Flip}, and let $\sigma_1^* = \sigma \circ \mathsf{Flip}_{e^*}$. Let $\sigma^* = \sigma_b^*$. Then σ^* is chosen uniformly at random. The security of PRF_1 implies that $\mathcal{A}$ still has advantage at least $\epsilon - \mathsf{negl}$.

Game 5 In the $b = 1$ case, instead of encrypting $S_1 = \mathsf{Flip}_{e^*}(S^*)$ using $\sigma^* = \sigma_1^* = \sigma_0^* \circ \mathsf{Flip}_{e^*}$, we encrypt S^* using σ_0^*. This does not change the ciphertext.

Now the only difference between the $b = 0$ and $b = 1$ case is the value of σ^* used to build $P'_{PBE-\mathsf{Dec}}$: when $b = 0$, $\sigma^* = \sigma_0^*$, and when $b = 1$, $\sigma^* = \sigma_1^* = \sigma_0^* \circ \mathsf{Flip}_{e^*}$. Changing from σ_0^* to σ_1^*

has the effect of changing the plaintext S^* to $\mathsf{Flip}_{e^*}(S^*)$, and thus membership is not affected for each $i \notin e^*$. Since $P'_{PBE-\mathsf{Dec}}$ always aborts on $i \in e^*$, this change does not affect the functionality of the program. Thus, the indistinguishability of iO implies that the obfuscated program in each case is indistinguishable. Thus, the advantage of $\mathcal{A}$ in **Game 4** is negligible. This means ϵ is negligible, and therefore, Construction 5.2 is recipient-private.

□

6 Extensions

6.1 CCA-secure Broadcast Encryption

Our key exchange to broadcast conversion actually gives a CCA scheme. For CCA security, the adversary is allowed decryption queries, where the adversary submits a header, and receives in response the corresponding message encryption key. In the proof of security, these decryption queries can easily be handled by making reveal queries to the key exchange challenger. Applying to our key exchange protocol, decryption queries then correspond to making PRF queries to PRF.

A similar statement applies to our recipient-private broadcast scheme, though there are some subtleties. If the adversary is restricted to only learning the key for headers of his choice, then the scheme is secure by the same reasoning as above. However, if the adversary may learn the set to which a message is encrypted, he will easily be able to break our scheme. The reason is that the "encryption scheme for sets" that we use is malleable. By authenticating the encryption scheme, it may be possible to acheive CCA security even for this strong attack — we leave this as an open problem.

6.2 Identity-Based Multiparty Key Exchange

It is straightforward to turn our scheme into an identity-based key exchange. In identity-based multiparty non-interactive key exchange (ID-NIKE), there is no more $\mathsf{Publish}$ step. Instead, KeyGen takes as input a list of identities, as well as a secret key for one of them, and outputs a group key for those identities. To give a user id his secret key, the authority runs an additional algorithm Ext which outputs the secret key.

Let $\mathcal{ID}$ be the identity space. Our basic idea is to have an additional PRF PRF_{key} which takes an identity $\mathsf{id} \in \mathcal{ID}$ and outputs a seed s_i that wil be the secret key. How the PRF_{key} is incorporated into our basic scheme is described below:

Construction 6.1. *Let* PRG *be a pseudorandom generator, let* PRF *and* PRF_{key} *be constrained PRFs for circuit predicates.*

$\mathsf{Setup}(\lambda, N)$ *Pick a random instance of* PRF *and* PRF_{key}*. Compute the program* P_{IBKE} *in Figure 7 padded to the appropriate length, and compute* $P_{\mathsf{iO}} = \mathsf{iO}(P_{IBKE})$*. Publish the public parameters* $\mathsf{params} = P_{\mathsf{iO}}$*.*

$\mathsf{Ext}(\mathsf{PRF}_{key}, \mathsf{id})$ *Outputs* $\mathsf{PRF}_{key}(\mathsf{id})$*.*

$\mathsf{KeyGen}(\mathsf{params}, S, \mathsf{id}, s_{\mathsf{id}})$ *To obtain the key* k_S*, sort* S *and compute* $k_S = P_{\mathsf{iO}}(S, s_{\mathsf{id}}, \mathsf{id})$*.*

Similar to our basic key exchange protocol, this scheme does not meet the strongest notion of security, where the adversary may adaptively choose the set of identities it gets secret keys for

Inputs: $S = \{\mathsf{id}_1, \ldots, \mathsf{id}_N\}, s, \mathsf{id}$
Constants: $\mathsf{PRF}, \mathsf{PRF}_{key}$

1. If $\mathsf{id} \notin S$ or $G(s) \neq G(\mathsf{PRF}_{key}(\mathsf{id}))$, output $\perp$
2. Otherwise, output $\mathsf{PRF}(\mathsf{id}_1, \ldots, \mathsf{id}_N)$

Figure 7: The program P_{IBKE}.

and makes challenges on. Nonetheless, it meets a *semi-static* security notion, where the adversary commits to a set of identities for which he will not receive the secret key, and must challenge on a subset of these identities. We leave achieving full adaptive security as an open question.

6.3 Identity-Based Broadcast Encryption

Using the same ideas as for identity-based key exchange, we can get identity-based broadcast encryption. We cannot quite go through the key exchange to broadcast conversion from Section 4 since the encryption step would require an extract query. Instead, we present a direct construction:

Construction 6.2. *Let* PRG *be a pseudorandom generator, let* PRF *and* PRF_{key} *be constrained PRFs for circuit predicates.*

$\mathsf{Setup}(\lambda, N)$ *Pick a random instance of* PRF *and* PRF_{key}*. Compute the program* P_{IBBE} *in Figure 8 padded to the appropriate length, and compute* $P_{\mathsf{iO}} = \mathsf{iO}(P_{IBBE})$*. Publish the public parameters* $\mathsf{params} = P_{\mathsf{iO}}$*.*

$\mathsf{Enc}(\mathsf{params}, S)$ *Pick a random seed* $s_\perp$ *and let* $x = \mathsf{PRG}(s_\perp)$*. Sort* S*, and let* $k_S = P_{\mathsf{iO}}(x, S, s_\perp, \perp)$*. Output* $(\mathsf{Hdr} = x, k_S)$*.*

$\mathsf{Ext}(\mathsf{PRF}_{key}, \mathsf{id})$ *Outputs* $\mathsf{PRF}_{key}(\mathsf{id})$*.*

$\mathsf{Dec}(\mathsf{params}, S, \mathsf{id}, s_{\mathsf{id}}, x_0)$ *To obtain the key* k_S*, sort* S *and compute* $k_S = P_{\mathsf{iO}}(x, S, s_{\mathsf{id}}, \mathsf{id})$*.*

Inputs: $x, \mathsf{id}_1, \ldots, \mathsf{id}_N, s, \mathsf{id}$
Constants: $\mathsf{PRF}, \mathsf{PRF}_{key}$

1. If $\mathsf{id} \notin S \bigcup \{\perp\}$, output $\perp$
2. If $\mathsf{id} = \perp$ and $G(s) \neq x$, output $\perp$
3. If $\mathsf{id} \neq \perp$ and $G(s) \neq G(\mathsf{PRF}_{key}(\mathsf{id}))$, output $\perp$
4. Otherwise, output $\mathsf{PRF}(x, \mathsf{id}_1, \ldots, \mathsf{id}_N)$

Figure 8: The program P_{IBBE}.

Again, this scheme does not meet the adaptive notion of security, but instead meets a *semi-static* security notion, where the adversary commits to a set of identities for which he will not receive the secret key, and must challenge on a subset of these identities. We leave achieving full adaptive security as an open question.

6.4 Distributed Broadcast Encryption

Our public-set broadcast encryption scheme is distributed — the sender sets up the system, but each user generates their own key when joining the system. Our recipient-private broadcast scheme does not meet the notion of distributed broadcast encryption, since the algorithm $P_{PBE-\mathsf{Dec}}$ depends on each party's public keys. Nonetheless, each party can generate their own secret and public values *before* the setup algorithm is run. Then, each party can send their public values to the broadcaster, who will generate the public parameters. In this way, our scheme satisfies a weaker notion of distributed broadcast encryption.

7 Constrained PRFs from Indistinguishability Obfuscation

In this section, we explain how to realize constrained pseudorandom functions for circuit predicates from indistinguishability obfuscation. Such PRFs were already constructed by Boneh and Waters [BW13] directly from multilinear maps. We give an alternative construction that uses only indistinguishability obfuscation and punctured PRFs, which can in turn be built from one-way functions. The idea is the following: starting with a punctured PRF, to constrain to circuit C, we obfuscate the program that takes an input x, checks that $C(x) = 1$, and then outputs $\mathsf{PRF}(x)$. The exact construction is the following:

Construction 7.1. *Let* PRF *be a punctured PRF with domain* $\{0,1\}^n$*, and* iO *an indistinguishability obfuscator. To constrain* PRF *to a circuit* C*, we run the following procedure:*

$\mathsf{PRF.Constrain}(C)$*: Build the program* P_C *in Figure 9. Output* $\mathsf{iO}(P_C)$*.*

Inputs: x
Constants: PRF, C

1. Check that $C(x) = 1$.
2. If check fails, output $\perp$
3. Otherwise, output $\mathsf{PRF}(x)$

Figure 9: The program P_C.

To prove selective security, we puncture PRF at the adversary's challenge x^*. Since $C(x^*) = 0$, this new program is identical, so the security of iO shows that no efficient adversary can tell the difference. But now the security of PRF implies that the adversary cannot distinguish the challenge value from random. Unfortunately, this achieves only selective security — we need to know x^* in order to compute the new program. For adaptive security, we must guess x^* at the beginning of the game, incurring an exponential loss in security. Then we must strengthen the security requirements of PRF and iO and apply a complexity leveraging argument to achieve security. This problem is also present in the construction of Boneh and Waters [BW13].

We note that our construction actually achieves the stronger notion of security of Boneh and Waters [BW13], where the adversary may adaptively ask for the constrained programs for many

circuits C, and may make many challenge queries. Let q_{cons} be the number of constrain queries and q_{chal} be the number of challenge queries. The following theorem states the security of the scheme:

Theorem 7.2. *For any adversary $\mathcal{A}$ breaking the security of Construction 7.1, there is an adversary $\mathcal{B}$ for iO and an adversary $\mathcal{C}$ for PRF such that:*

$$\mathtt{PRF}^{(adv)}{}_{\mathcal{A}}(\lambda) \leq q_{chal} 2^n (q_{cons} \mathtt{IO}^{(adv)}{}_{\mathcal{B}}(\lambda) + \mathtt{PRF}^{(adv)}{}_{\mathcal{C}}(\lambda))$$

Proof. A simple hybrid argument shows that any adversary with advantage ϵ making q_{chal} challenge queries can be turned into an adversary with advantage ϵ/q_{chal} making a single challenge query. We will therefore start with an adversary $\mathcal{A}$ that makes a single challenge on x^*, and has advantage ϵ/q_{chal}. We prove security through a sequence of games:

Game 0 This is the standard constrained PRF game, where $\mathcal{A}$ makes a polynomial number q_{PRF} of PRF queries on inputs $x_1, \ldots, x_{q_{PRF}}$, and a polynomial number q_{cons} of constrain queries on circuits $C_1, \ldots, C_{q_{cons}}$, and a single challenge query on x^*. We require that $x^* \neq x_i$ for any i, and that $C_j(x^*) = 0$ for all j. If $b = 0$, the response to the challenge query is $\mathsf{PRF}(x^*)$. If $b = 1$, the response is chosen at random. By assumption, $\mathcal{A}$ has advantage ϵ/q_{chal} in distinguishing $b = 0$ from $b = 1$.

Game 1 Here, we guess x^* at the beginning of the game, and abort if our guess was wrong. The advantage of A is now $\epsilon/2^n q_{chal}$.

Game 2 Now we puncture PRF at x^*, obtaining $\mathsf{PRF}^{\overline{\{x^*\}}}$. Let $y^* = \mathsf{PRF}(x^*)$. When responding to a constrain query on a circuit C_j, we replace PRF with $\mathsf{PRF}^{\overline{\{x^*\}}}$ in the program P_{C_j}. Since $C_j(x^*) = 0$ by assumption, this new program has the same functionality as the old program. If $\mathcal{A}$ can distinguish **Game 2** from **Game 1** with probability γ, then it is straightforward to build an adversary $\mathcal{B}$ that breaks the indistinguishability of iO with probability $\mathtt{IO}^{(\mathrm{adv})}{}_{\mathcal{B}}(\lambda) = \gamma/q_{cons}$. $\mathcal{A}$ thus has advantage at least $\epsilon/2^n q_{chal} - q_{cons} \mathtt{IO}^{(\mathrm{adv})}{}_{\mathcal{B}}(\lambda)$ in **Game 2**.

Now we describe an adversary $\mathcal{C}$ that breaks the security of PRF as a punctured PRF. Guess x^* at the beginning of the game, and query the punctured PRF challenger on x^*, obtaining y^*. Also ask for the punctured PRF $\mathsf{PRF}^{\overline{\{x^*\}}}$. When $\mathcal{A}$ makes a PRF query, forward the query to the punctured PRF challenger. When $\mathcal{A}$ makes a constrain query on circuit C_j, build the modified program P_{C_j} as described above and give the obfuscation of this program to $\mathcal{A}$. When $\mathcal{A}$ makes its challenge, abort and output a random bit if the challenge is not x^*. Otherwise, respond with y^*. This new adversary perfectly simulates the view of $\mathcal{A}$ in **Game 3**, and therefore has the same advantage of $\mathcal{A}$ in this game. In other words,

$$\epsilon/2^n q_{chal} - q_{cons} \mathtt{IO}^{(\mathrm{adv})}{}_{\mathcal{B}}(\lambda) \leq \mathtt{PRF}^{(\mathrm{adv})}{}_{\mathcal{C}}(\lambda)$$

Rearranging gives the desired inequality, completing the proof.

□

8 Conclusion and Open Problems

We give the first construction of multiparty key exchange requiring no trusted setup from indistinguishability obfuscation and constrained PRFs. Using our ideas, we give the first distributed broadcast encryption scheme, where each party generates their own secret keys. We also construct a recipient-private broadcast encryption scheme. From this construction, we obtained traitor tracing

with very short ciphertexts and secret keys. We also give several extensions, such as identity-based broadcast encryption. All our constructions can be built from any indistinguishability obfuscator and a one-way function.

We leave several open problems. One interesting direction is achieving semi-static or adaptive security without complexity leveraging. Many schemes built from indistinguishability obfuscation, including our schemes, as well as the short signature scheme of Sahai and Waters [SW13], achieve only selective security. We obtain semi-static security by using constrained PRFs for circuit predicates, but the only known constructions of these require complexity leveraging to achieve adaptive security — in essence, we have offloaded the complexity leveraging step to the constrained PRF.

Another direction is to reduce public key size — all of our schemes have public keys that are at least linear in the number of users. New techniques seem to be required to shrink the public key.

Acknowledgments

We thank Jonathan Ullman for his comments on the connection to differential privacy. This work was supported by NSF, the DARPA PROCEED program, an AFO SR MURI award, a grant from ONR, an IARPA project provided via DoI/NBC, and by a Google faculty scholarship. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or IARPA.

References

[BBW06] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography*, pages 52–64, 2006.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (Im)possibility of obfuscating programs. In *Advances in Cryptology — CRYPTO 2001*, number Im, 2001.

[BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. Cryptology ePrint Archive, Report 2013/401, 2013.

[BGK+13] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. Cryptology ePrint Archive, Report 2013/631, 2013. http://eprint.iacr.org/.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. *Advances in Cryptology — CRYPTO 2005*, pages 1–19, 2005.

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *ACM Conference on Computer and Communications Security*, pages 501–510, 2008.

[BR13a] Zvika Brakerski and Guy N. Rothblum. Black-box obfuscation for d-cnfs. Cryptology ePrint Archive, Report 2013/557, 2013. http://eprint.iacr.org/.

[BR13b] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. Cryptology ePrint Archive, Report 2013/563, 2013. http://eprint.iacr.org/.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. *Advances in Cryptology – EUROCRYPT 2006*, pages 573–592, 2006.

[BW06] Dan Boneh and Brent Waters. A fully collusion resistant broadcast trace and revoke system with public traceability. In *ACM Conference on Computer and Communication Security (CCS)*, 2006.

[BW13] Dan Boneh and Brent Waters. Constrained Pseudorandom Functions and Their Applications. *Advances in Cryptology — AsiaCrypt 2013*, pages 1–23, 2013.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. *Advances in Cryptology — CRYPTO 1997*, pages 455–469, 1997.

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

[CLT13] Jéan-Sebastien Coron, Tancède Lepoint, and Mehdi Tibouchi. Practical Multilinear Maps over the Integers. *Advances in Cryptology — CRYPTO 2013*, pages 1–22, 2013.

[CMR98] Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly One-Way Probabilistic Hash Functions. *Proc. of STOC 1998*, pages 131–140, 1998.

[CPP05] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT'05*, pages 542–558, 2005.

[CRV10] Ran Canetti, Guy N Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. *Theory of Cryptography Conference 2010*, 5978:72–89, 2010.

[Del07] Cécile Delerablée. Identity-Based Broadcast Encryption with Constant Size Ciphertexts and Private Keys. 2:200–215, 2007.

[DF02] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Proceedings of the Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.

[DF03] Y. Dodis and N. Fazio. Public key broadcast encryption secure against adaptive chosen ciphertext attack. In *Workshop on Public Key Cryptography (PKC)*, 2003.

[DNR+09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of STOC 2009*, 2009.

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. *PAIRING 2007*, (July), 2007.

[FHKP13] Eduarda S.V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenny Paterson. Non-interactive key exchange. In *Public-Key Cryptography*, pages 1–28, 2013.

[FHPS13] Eduarda S.V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In *CRYPTO 2103*, pages 513–530, 2013.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. *Advances in Cryptology — CRYPTO 1993*, 773:480–491, 1994.

[FP12] Nelly Fazio and IrippugeMilinda Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In *Public Key Cryptography — PKC 2012*, volume 7293 of *LNCS*, pages 225–242, 2012.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010.

[GGH13a] S Garg, Craig Gentry, and S Halevi. Candidate multilinear maps from ideal lattices. *Advances in Cryptology — EUROCRYPT 2013*, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *Proc. of FOCS 2013*, 2013.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[GKSW10] Sanjam Garg, Abishek Kumarasubramanian, Amit Sahai, and Brent Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2010.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *TCC*, pages 194–213, 2007.

[GST04] M. T. Goodrich, J. Z. Sun, , and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Proceedings of Crypto '04*, volume 2204 of *LNCS*, 2004.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). *Advances in Cryptology — EUROCRYPT 2009*, pages 1–18, 2009.

[HS02] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In *Proceedings of Crypto '02*, volume 2442 of *LNCS*, pages 47–60, 2002.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/509, 2013.

[Jou04] Antoine Joux. A One Round Protocol for Tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, June 2004.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings ACM CCS*, 2013.

[KS13] Aggelos Kiayias and Katerina Samari. Lower bounds for private broadcast encryption. In *Information Hiding*, pages 176–190. Springer, 2013.

[KY02] Aggelos Kiayias and Moti Yung. Breaking and repairing asymmetric public-key traitor tracing. In Joan Feigenbaum, editor, *ACM Workshop in Digital Rights Management – DRM 2002*, volume 2696 of *Lecture Notes in Computer Science*, pages pp. 32–50. Springer, 2002.

[LPQ12] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In *Public Key Cryptography*, pages 206–224, 2012.

[LPS04] Benjamin Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. *Advances in Cryptology — EUROCRYPT 2004*, pages 1–18, 2004.

[LSW10] Allison B. Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *IEEE Symposium on Security and Privacy*, pages 273–285, 2010.

[MR13] Tal Moran and Alon Rosen. There is no indistinguishability obfuscation in pessiland. Cryptology ePrint Archive, Report 2013/643, 2013. http://eprint.iacr.org/.

[NNL01] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Proceedings of Crypto '01*, volume 2139 of *LNCS*, pages 41–62, 2001.

[NP00] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial cryptography 2000*, volume 1962 of *LNCS*, pages 1–20. Springer, 2000.

[Pfi96] B. Pfitzmann. Trials of traced traitors. In *Proceedings of Information Hiding Workshop*, pages 49–64, 1996.

[PW97] B. Pfitzmann and M. Waidner. Asymmetric fingerprinting for larger collusions. In *Proceedings of the ACM Conference on Computer and Communication Security*, pages 151–160, 1997.

[SF07] Ryuichi Sakai and Jun Furukawa. Identity-Based Broadcast Encryption. *IACR Cryptology ePrint Archive*, 2007.

[Sir07] Thomas Sirvent. Traitor tracing scheme with constant ciphertext rate against powerful pirates. In *Workshop on Coding and Cryptography*, 2007.

[SW13] Amit Sahai and Brent Waters. How to Use Indistinguishability Obfuscation: Deniable Encryption, and More. Cryptology ePrint Archive, Report 2013/454, 2013. http://eprint.iacr.org/.

[Ull13] Jonathan Ullman. Answering $n^{\{2+o(1)\}}$ counting queries with differential privacy is hard. In *STOC*, pages 361–370, 2013.

[Wee05] Hoeteck Wee. On obfuscating point functions. *Proc. of STOC 2005*, page 523, 2005.

[WHI01] Yuji Watanabe, Goichiro Hanaoka, and Hideki Imai. Efficient asymmetric public-key traitor tracing without trusted agents. In *Proceedings CT-RSA '01*, volume 2020 of *LNCS*, pages 392–407, 2001.