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 with short ciphertexts, secret keys, and public key. Ciphertext size is logarithmic in the number of users and secret key size is independent of the number of users. Our public key size is polylogarithmic in the number of users. The recent functional encryption system of Garg, Gentry, Halevi, Raykova, Sahai, and Waters also leads to a traitor tracing scheme with similar ciphertext and secret key size, but the construction in this paper is simpler and more direct. These constructions resolve an open problem relating to differential privacy.
- Generalizing our traitor tracing system gives a private broadcast encryption scheme (where broadcast ciphertexts reveal minimal information about the recipient set) with optimal size ciphertext.

Several of our proofs of security introduce new tools for proving security using indistinguishability obfuscation.

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, but require a trusted setup phase. Prior to this work, these were the only known constructions for NIKE.

We construct new NIKE protocols from a general indistinguishability obfuscator. Our basic protocol is easy to describe: each user generates a random seed s for a pseudorandom generator G whose output is at least twice the size of the seed. The user posts $G(s)$ to the bulletin board. 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 P_{KE} (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

Inputs: public values $x_1, \ldots x_N \in \mathcal{X}^N$, an index $i \in [N]$, and a secret seed $s \in \mathcal{S}$
Embedded constant: pseudorandom function PRF with an embedded random key

1. If $x_i \neq G(s)$, output $\perp$
2. Otherwise, output $\mathsf{PRF}(x_1, x_2, \ldots, x_N)$

Figure 1: The program P_{KE}.

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 a fully adaptively secure NIKE (in the sense of [FHKP13]) from iO as a fascinating open problem.

In Section 8, 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, our iO-based protocol has a number of advantages:

- No trusted setup. Existing constructions [GGH13a, CLT13] require a trusted setup to publish public parameters: whoever generates the parameters can expose the secret keys for all groups just from the public values posted by members of the group. A variant of our iO-based construction requires no trusted setup, and in fact, requires no setup at all. We simply have user number 1 generate the obfuscated program P_{KE} and publish it along with her public values. The resulting scheme is the first statically secure NIKE protocol with no setup requirements. In Section 4 we enhance the construction and present a NIKE protocol with no setup that is secure in the stronger semi-static model. This requires changing the scheme to defend against a potentially malicious program P_{KE} published by a corrupt user. To do so we replace the secret seed s by a digital signature generated by each user. Proving security from iO requires the signature scheme to have a special property we call *constrained public-keys*, which may be of independent interest. We construct such signatures from iO.

- Short public values. In current multilinear-based NIKE protocols, the size of the values published to the bulletin board is at least linear in the number of users N. In our basic iO-based construction with trusted setup, the size of published values is independent of N.

- 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 5.1, 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 5). 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.

Another interesting aspect of the construction is that the PRG used in the scheme (as in the program P_{KE}) can be replaced by the RSA public key encryption system where the RSA secret key plays the role of the PRG seed and the corresponding RSA public key plays the role of the PRG output. Then, our broadcast system shows that iO makes it possible to use *existing* certified RSA keys in a short-ciphertext broadcast encryption system and in a NIKE protocol. To prove security using iO we need the following property of RSA: there is a distribution of invalid RSA public-keys (e.g. products of three random large primes) that is computationally indistinguishable from a distribution of real RSA public keys (i.e. products of two random large primes). This property also holds for other public-key systems such as Regev's lattice encryption scheme, but does not hold for systems like basic ElGamal encryption.

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. Until very recently, the best recipient-private broadcast systems had 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 a recipient-private broadcast system with a broadcast size of $N + O(\lambda)$, proportional to the *sum* of the security parameter and the number of users. Since the recipient set must be semantically secure, 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 is problematic 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, 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 be 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 $O(1)$.

1.4 Traitor tracing with short ciphertexts, secret keys, and public keys

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* (PLBE).

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. However, this scheme has large public keys, polynomial in N and λ. The main reason public keys are large is that the malleable encryption scheme we need requires polynomial size circuits for encryption and decryption.

Fortunately we can reduce the public-key size to only $\mathsf{poly}(\log N, \lambda)$ without affecting secret-key or ciphertext size. We do so by adapting the authenticated encryption approach discussed in the previous section: when embedding the encrypted recipient set in the broadcast ciphertext we also embed a MAC of the encrypted set. The decryption program will reject a broadcast ciphertext with an invalid MAC. To prove security we need to puncture the MAC algorithm at all possible recipient sets. Naively, since in a PLBE there are $N+1$ recipient sets, the resulting program size would be linear in N thereby resulting in large secret keys. Instead, we step through a sequence of hybrids where at each hybrid we puncture the MAC at exactly one point. This sequence of hybrids ensures that the obfuscated decryption program remains small. Once this sequential puncturing process completes, security follows from security of an embedded PRF. We emphasize that this proof technique works for proving security of a PLBE because of the small number of possible recipient sets.

The functional encryption scheme of Garg et al. [GGH$^+$13b] can also be used to obtain collusion resistant traitor tracing, however as for private broadcast encryption, our construction is conceptually simpler and has shorter secret keys.

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?" Informally, 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 $N2^{O(\lambda)}$ queries. 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].

1.5 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].

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, FHPS13, BW13]. They also give public-key broadcast systems with short ciphertexts and short public keys (using an $O(\log N)$-linear map) [BWZ14], 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 distributed 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 be 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].

Recently, Koppula, Ramchen, and Waters [KRW13] provide counter-examples to the conjecture that all bit encryption schemes are circularly secure. Concurrently and independent of our work, they use a valid/invalid key strategy that is similar to our strategy of replacing correctly generated public parameters with incorrect parameters, but in a very different context.

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

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

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: $\mathsf{PRF.Constrain}(S)$ 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 $y_1 \overset{R}{\leftarrow} \mathcal{Y}$ 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 9, 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. Intuitively, a NIKE protocol allows a group of users to simultaneously publish a single message, and all will derive the same shared group key. The first such protocols [BS03, GGH13a, CLT13] are based on multilinear maps. Our construction, based on a generic iO obfuscator, has the following properties:

- Using a punctured pseudorandom function, our protocol achieves a *static* notion of security, similar to existing protocols.
- Using a constrained pseudorandom function for circuit predicates, our protocol achieves a stronger notion of security called *semi-static* security. We show in Section 9 who to use iO to construct constrained pseudorandom functions for circuit predicates from any secure puncturable PRF.
- While our base protocol requires a trusted setup phase, our setup phase can be run *independently* of the messages sent by users. In Section 4 we use this property to remove the

setup phase altogether, arriving at the first NIKE protocol *without trusted setup*. We provide protocols for both static and semi-static security.

We begin by first defining NIKE protocols and their security. A NIKE protocol has the following three algorithms:

$\mathsf{Setup}(\lambda, G, N)$: The setup algorithm takes a security parameter λ and two integers G and N. G is the maximum number of users that can derive a shared secret key, and N is an upper bound on the number of users in the system. 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.

$\mathsf{KeyGen}(\mathsf{params}, i, \mathsf{sk}_i, S, \{\mathsf{pv}_j\}_{j\in S})$: Finally, to derive the shared key k_S for a subset $S \subseteq [N]$ of size at most G, each party in S runs KeyGen with params, their secret sk_i, and the other parties' public values $\{\mathsf{pv}_j\}_{j\in S}$.

For correctness, we require that each user derives the same secret key. That is, for all $S \subseteq [N], |S| \le G, i, i' \in S$,

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

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, and various subsets of them are engaging in the key exchange protocol. We let the adversary adaptively corrupt users and reveal the shared secret for arbitrary subsets of users of its choice. More formally, for $b = 0, 1$ we denote by $\mathtt{EXP}(b)$ the following experiment, parameterized by the total number of parties N, the maximal group size G (where potentially G is the same as N), and an adversary $\mathcal{A}$:

$\mathsf{params} \overset{R}{\leftarrow} \mathsf{Setup}(\lambda, G, N)$

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

where

$\mathsf{Reg}(i \in [N])$ registers an honest user. It takes an index i, and runs $(\mathsf{sk}_i, \mathsf{pv}_i) \leftarrow \mathsf{Publish}(\mathsf{params}, i)$. The challenger records the tuple $(i, \mathsf{sk}_i, \mathsf{pv}_i, \mathit{honest})$, and sends pk_i to A.

$\mathsf{RegCor}(i \in [N], \mathsf{pk}_i)$ registers a corrupt user. It takes an index i and a public value pv_i. The challenger records $(i, \perp, \mathsf{pv}_i, \mathit{corrupt})$. The adversary may make multiple queries for a particular identity, in which case the challenger only uses the most recent record.

$\mathsf{Ext}(i)$ extracts the secret key for an honest registered user. The challenger looks up the tuple $(i, \mathsf{sk}_i, \mathsf{pv}_i, \mathit{honest})$, and returns sk_i to the challenger.

$\mathsf{Rev}(S, i)$ reveals the shared secret for a group $S \subseteq [N], |S| \le G$ of users, as calculated by the ith user, where $i \in S$. We require that at user i was registered as honest. The challenger uses the secret key for user i to derive the shared secret key k_S, which it returns to the adversary.

$\mathsf{Test}(S)$ takes a set $S \subseteq [N], |S| \le G$ of users, all of which were registered as honest. If $b = 0$, the challenger runs KeyGen to determine the shared secret key (arbitrarily

choosing which user to calculate the key), which it returns to the adversary.
Otherwise if $b = 1$, the challenger generates a random key k to return to the adversary.

We require that all reveal and test queries are for distinct sets, and no extract query is allowed on any user in a reveal query. We require that all register queries and register-corrupt queries are for distinct i, and that $\mathsf{pv}_i \neq \mathsf{pv}_j$ for any $i \neq j$. For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\mathtt{EXP}(b)$ and we define $\mathtt{AdvKE}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

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

We also define two weaker notions of security called *semi-static* and *static* security. The first notion, semi-static security, requires the adversary to commit to a set $\hat{S}$ of users at the beginning of the experiment. The adversary must only make test queries on subsets $S^* \subseteq \hat{S}$, and can only make register corrupt and extract queries on users $i \notin \hat{S}$:

Definition 3.2. A multiparty key exchange protocol ($\mathsf{Setup}, \mathsf{Publish}, \mathsf{KeyGen}$) is semi-statically secure if, for any polynomials G and N, and any PPT adversary $\mathcal{A}$ satisfying the following properties, $\mathtt{AdvKE}(\lambda)$ is negligible:

- $\mathcal{A}$ commits to a set $\hat{S}$ of users before seeing the public parameters.
- Each query to RegCor and Ext must be on a user i outside the set $\hat{S}$.
- Each query to Rev must have $S \not\subseteq \hat{S}$.
- Each query to Test must be on a subset S^* of $\hat{S}$.

This notion is akin to the semi-static notion of security for broadcast encryption, defined by Gentry and Waters [GW09]. We also weaken the definition further, arriving at a security notion called *static* security. Here we only allow a single test query on a set S^*, and the adversary must commit to S^* before seeing the public parameters.

Definition 3.3. A multiparty key exchange protocol ($\mathsf{Setup}, \mathsf{Publish}, \mathsf{KeyGen}$) is statically secure if, for any polynomials G and N, and any PPT adversary $\mathcal{A}$ satisfying the following conditions, the function $\mathtt{AdvKE}(\lambda)$ is negligible:

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

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 3.4. *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.*

$\mathsf{Setup}(\lambda, G, N)$ *Choose a random key to obtain an instance of a pseudorandom function* PRF*. Build the program* P_{KE} *in Figure 2, padded to the appropriate length*[1]*. Also choose a random* $x_0 \in \{0,1\}^{2\lambda}$*. Output* $P_{\mathsf{iO}} = \mathsf{iO}(P_{KE})$ *and* x_0 *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* $x_i = \mathsf{PRG}(s_i)$

$\mathsf{KeyGen}(P_{\mathsf{iO}}, x_0, i, s_i, S, \{x_i\}_{i\in S})$ *Let* $S(j)$ *denote the* j*th index in* S*, and* $S^{-1}(k)$ *for* $k \in S$ *be the inverse. Let*

$$\hat{x}_j = \begin{cases} x_{S(j)} & \text{if } j \leq |S| \\ x_0 & \text{if } j > |S| \end{cases}$$

Run P_{iO} *on* $(\hat{x}_1, ..., \hat{x}_G, S^{-1}(i), s_i)$ *to obtain the key* $k = \mathsf{PRF}(\hat{x}_1, ..., \hat{x}_G)$ *or* $\perp$*.*

Inputs: $\hat{x}_1, \ldots \hat{x}_G \in \mathcal{X}^G$, $i \in [G]$, $s \in \mathcal{S}$
Constants: PRF

1. If $\hat{x}_i \neq \mathsf{PRG}(s)$, output $\perp$
2. Otherwise, output $\mathsf{PRF}(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_G)$

Figure 2: The program P_{KE}. (same as Figure 1).

Correctness is trivial by inspection. For security, we consider two cases. If PRF is a punctured PRF, then we get static 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 3.5. *If* PRG *is a secure pseudorandom generator,* PRF *a secure punctured PRF, and* iO *a secure indistinguishability obfuscator, then Construction 3.4 is a statically secure NIKE. If, in addition,* PRF *is a secure constrained PRF for circuit predicates, then Construction 3.4 is semi-statically secure.*

Before proving Theorem 3.5, notice that if the adversary is able to learn the random coins used by Setup, he will be able to break the scheme. All prior key exchange protocols [GGH13a, CLT13] also suffer from this weakness. However, note that, unlike previous protocols, $\mathsf{Publish}$ does not depend on params. In Section 4, we show how to use this property to remove the setup phase all together, arriving at the first NIKE protocol permitting an untrusted setup.

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-negligible advantage ϵ in breaking the security of Construction 3.4 as in Definition 3.2. We prove security through a sequence of games.

[1]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.

Game 0 This is the attack game from Definition 3.2, where $\mathcal{A}$ commits to a set $\hat{S} \subseteq [N]$. In response $\mathcal{A}$ gets the obfuscation of P_{KE} and then makes the following queries:

- Register honest user queries: $\mathcal{A}$ submits an $i \in [N]$. The challenger chooses a random s_i, and sends $x_i = \mathsf{PRG}(s_i)$ to $\mathcal{A}$.
- Register corrupt user queries: $\mathcal{A}$ submits an $i \in [N] \setminus \hat{S}$ and a string x_i. We require that i was not and will not be registered as honest.
- Extract queries: $\mathcal{A}$ submits an $i \in [N] \setminus \hat{S}$ that was previously registered as honest. The challenger responds with s_i.
- Reveal queries: the adversary submits a subset $S, |S| \leq G$ of users, of which at least one is honest. The challenger uses PRF to compute the group key.
- Test queries: the adversary submits a set $S^* \subset \hat{S}, |S^*| \leq G$, 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 For each $i \in \hat{S}$, choose random $x_i \in \{0,1\}^{2\lambda}$. When answering register honest user queries for $i \in \hat{S}$, we will use these x_i values instead of generating them from PRG. 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 $x_i, i \in \hat{S}$ in **Game 1** have a pre-image under PRG. The same applies to x_0. Therefore, with overwhelming probability, there is no input to P_{KE} that will cause PRF to be evaluated on points of the form $(x_{i_1}, \ldots, x_{i_G})$ for $i_k \in \hat{S} \bigcup \{0\}$. We can now constrain the PRF so that it can only be evaluated at points $(\hat{x}_1, \ldots, \hat{x}_G)$ where the set $\{\hat{x}_1, \ldots, \hat{x}_G\}$ is not contained in the set $X^* = \{x_i\}_{i \in S} \bigcup \{x_0\}$. Formally, we construct a circuit C that takes as input $(\hat{x}_1, \cdots, \hat{x}_G)$ and accepts if and only if there is some $\hat{x}_j$ 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 3. 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: $\hat{x}_1, \ldots \hat{x}_G \in \mathcal{X}^G$, $i \in [G]$, $s \in \mathcal{S}$

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

1. If $x_i \neq \mathsf{PRG}(s)$, output $\perp$
2. Otherwise, output $\mathbf{PRF}^C(x_1, x_2, \ldots, x_N)$

Figure 3: 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). $\mathcal{B}$ runs $\mathcal{A}$, obtaining a set $\hat{S}$. $\mathcal{B}$ chooses $|\hat{S}|+1$ random values $x_i \in \{0,1\}^{2\lambda}, i \in \hat{S} \bigcup \{0\}$, 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 3, giving $\mathsf{iO}(P'_{KE})$ and x_0 to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a register honest user query for $i \in \hat{S}$, $\mathcal{B}$ responds with x_i, and for all other register honest queries, $\mathcal{A}$ responds with $x_i = G(s_i)$ for a random seed s_i. For a register corrupt user query, $\mathcal{B}$ just records the public value x_i, and for an extract query, $\mathcal{B}$ just responds with the seed s_i. 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 3.4. □

4 Multi-party Key Exchange with No Setup

We now show how to remove the trusted setup from the NIKE protocol in the previous section. Our starting point is the observation that $\mathsf{Publish}$ from Construction 3.4 does not depend on params. This means that in every group of users S, we can designate one party (say, the one with the lowest i) as the *master party*, who will run Setup for himself, and publish params along with its public value pv_i. Because the rest of the users can compute their public values without knowing params, all parties can publish simultaneously (i.e. independently of one another). This gives the first multi-party NIKE protocol with no setup.

We define two desired properties of a NIKE protocol. The first is untrusted setup:

Definition 4.1. A NIKE protocol has *untrusted setup* if the random coins used by Setup are part of the public parameters params. That is, $\mathsf{Setup}(\lambda, G, N; r) = (\mathsf{params}, r)$.

An even stronger notion is that of no setup:

Definition 4.2. A NIKE protocol has *no setup* if Setup does nothing. That is, $\mathsf{Setup}(\lambda, G, N) = (\lambda, G, N)$.

The seminal key exchange protocols of Garg, Gentry, and Halevi [GGH13a] and Coron, Lepoint, and Tibouchi [CLT13] both require a trusted setup since secrets are involved in generating the parameters for the multilinear map. The entity who runs setup can generate all group keys given only the public values from the group members.

Following the observation at the beginning of this section, we can state the following theorem:

Theorem 4.3. *Let* $(\mathsf{Setup}, \mathsf{Publish}, \mathsf{KeyGen})$ *be a statically secure NIKE protocol where* $\mathsf{Publish}$ *does not depend on* params *outputted by* Setup*, but instead just takes as input* λ, G, N, i*. Then there is a statically secure NIKE protocol* $(\mathsf{Setup}', \mathsf{Publish}', \mathsf{KeyGen}')$ *with no setup.*

Proof. As described above, $\mathsf{Publish}'$ runs the publish algorithm $(\mathsf{pv}_i, \mathsf{sk}_i) \leftarrow \mathsf{Publish}(\lambda, G, N, i)$, as well as the setup algorithm $\mathsf{params}_i \leftarrow \mathsf{Setup}(\lambda, G, N)$. The published value for user i is $\mathsf{pv}'_i = (\mathsf{pv}_i, \mathsf{params}_i)$ and the secret value is sk_i. Then user i runs KeyGen' for a set S of users, which lets i^* be the lowest i in S, and runs $k \leftarrow \mathsf{KeyGen}(\mathsf{params}_{i^*}, i, \mathsf{sk}_i, S, \{\mathsf{pv}_i\}_{i \in S})$. Static security immediately follows.

□

Applying the conversion of Theorem4.3 to the scheme from Section 3.1, we immediately obtain a statically secure NIKE protocol with no setup. Unfortunately, Theorem 4.3 cannot be strengthened to provide semi-static or active security with no setup. The problem lies in the fact that all users, even corrupt users, will need to generate their own set of public parameters so they can play the role of master party if needed. The adversary can ask a reveal query on a mix of corrupt and honest users, and if the public parameters used are corrupt, non-trivial information about the honest users' secrets may leak.

Concretely, if we apply the conversion to the scheme from Section 3.1, we require all parties to publish an obfuscated program $P_{\mathsf{iO},i}$. A corrupt user may publish a malicious program $\hat{P}_{\mathsf{iO},i}$ that instead of outputting the shared group key simply outputs the input seed. Then, the adversary may ask a reveal query on a mix of honest and corrupt users, so that a corrupt $\hat{P}_{\mathsf{iO},i_c}$ is used, but an honest user i_h is asked to produce the key. The resulting key is just the user's secret seed s_{i_h}, which the adversary now learns. With this seed, the adversary can learn the shared key for any group containing user i_h, even groups of all honest users, thus breaking the security of the scheme. This attack applies even if the original NIKE is semi-statically or actively secure.

We now describe how to overcome the above difficulties and achieve semi-static security with no setup. Instead of using the secret itself as input to the obfuscated program $P_{\mathsf{iO},i}$, we give the program a signature derived from the secret. Then, even if the adversary learns this signature it will not learn the user's secret, and will not learn the group keys for other groups. Proving security from iO requires a special type of signatures we call constrained signatures.

Construction 4.4. *Let* PRF *be a pseudorandom function, and let* $(\mathsf{G}, \mathsf{Sign}, \mathsf{Ver})$ *be a signature scheme. Let* iO *be a program indistinguishability obfuscator.*

$\mathsf{Setup}(\lambda, G, N)$ *Simply output* $\mathsf{params} = (\lambda, G, N)$.

$\mathsf{Publish}(\mathsf{params}, i)$ *Party* i *runs* $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{G}(\lambda)$. *Party* i *also chooses a pseudorandom function* PRF, *and computes the obfuscation* $P_{\mathsf{iO},i}$ *of the program* P_{KE} *in Figure 4. The public value is* $\mathsf{pv}_i = (\mathsf{pk}_i, P_{\mathsf{iO},i})$ *and the secret value is* sk_i.

$\mathsf{KeyGen}(i, \mathsf{sk}_i, S, \{\mathsf{pv}_j\}_{j \in S})$ *Let* i^* *be the smallest element in* S. *Compute the signature* $\sigma = \mathsf{Sign}(\mathsf{sk}_i, S)$, *interpreting* S *as an* N*-bit string. Then run* P_{iO,i^*} *on the inputs* $S, \{\mathsf{pk}_j\}_{j \in S}, i, \sigma$ *to obtain the key* k_S.

Inputs: $S \subset [N], |S| \leq G$, $\{\mathsf{pk}_j\}_{j \in S}$, i, σ
Constants: PRF

1. Interpret S as an N-bit string.
2. If $\mathsf{Ver}(\mathsf{pk}_i, S, \sigma)$ fails, abort and output $\perp$.
3. Otherwise, output $\mathsf{PRF}(S, \{\mathsf{pk}_j\}_{j \in S})$.

Figure 4: The program P_{KE}.

Correctness is straightforward by inspection. For security, we will need a new type of signatures, which we call *constrained* signatures.

4.1 Constrained Signatures

We put forth a new type of signature scheme, which we call a *constrained* signature. Informally, a constrained signature scheme allows the computation, given a circuit C, of a constrained public key pk_C. This public key has the property that for any x where $C(x) = 0$, there are no valid signatures on x relative to pk_C. For security, we require that pk_C is indistinguishable from a valid public key by any adversary that gets to make signature queries on x such that $C(x) = 1$. More formally, a constrained signature scheme is tuple of PPT algorithms $(\mathsf{G}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{ConstrainGen})$ where:

$\mathsf{G}(\lambda, t)$ Takes as input the security parameter λ and a upper bound t on the size of circuits C handled by the scheme. It outputs a valid secret/public key pair $(\mathsf{sk}, \mathsf{pk})$. We will often omit the input t when the upper bound on C is obvious from the use case.

$\mathsf{Sign}(\mathsf{sk}, m)$ and $\mathsf{Ver}(\mathsf{sk}, m, \sigma)$ These are the usual signing and verification algorithms for a signature scheme.

$\mathsf{ConstrainGen}(\lambda, t, C)$ Takes as λ and t, and a circuit C such that $|C| \leq t$. It outputs an *invalid* secret and pubic key pair $(\mathsf{sk}_C, \mathsf{pk}_C)$. We require that $\mathsf{Sign}(\mathsf{sk}_C, x)$ produce a valid signature relative to pk_C for all x such that $C(x) = 1$. Meanwhile, for any x where $C(x) = 0$, we require that there are *no* valid signatures — that is, $\mathsf{Ver}(\mathsf{pk}_C, x, \sigma)$ rejects for all σ.

For security, we allow an adversary to make signing queries on all x where $C(x) = 1$, and ask for the adversary to distinguish a valid public key pk generated by G from an invalid public key pk_C generated by $\mathsf{ConstrainGen}$. More formally, let $\mathtt{EXP}(b)$ denote the following experiment, parameterized by security parameter λ, size bound t, and adversary $\mathcal{A}$:

```
$C \overset{R}{\leftarrow} \mathcal{A}(\lambda, t)$
    where $|C| \leq t$
$(\mathsf{pk}_0, \mathsf{sk}_0) \overset{R}{\leftarrow} \mathsf{G}(\lambda, t)$
$(\mathsf{pk}_1, \mathsf{sk}_1) \overset{R}{\leftarrow} \mathsf{ConstrainGen}(\lambda, t, C)$
$b' \leftarrow \mathcal{A}^{\mathsf{Sign}_b(\cdot)}(\mathsf{pk}_b, \lambda, t, C)$
where
    $\mathsf{Sign}_b(x)$ takes as input an $x$ such that $C(x) = 1$ and outputs $\mathsf{Sign}(\mathsf{sk}_b, x)$
```

For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\mathtt{EXP}(b)$ and define $\mathtt{CSig}^{(\mathrm{adv})}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 4.5. A constrained signature scheme $(\mathsf{G}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{ConstrainGen})$ is secure if, for all polynomials λ, t and any PPT adversary $\mathcal{A}$, the function $\mathtt{CSig}^{(\mathrm{adv})}(\lambda)$ is negligible.

We note that this definition of security implies a targeted form of unforgeability where an adversary commits to a circuit C, and can then make signing queries on inputs x where $C(x) = 1$. The adversary must then forge a signature on a message x^* where $C(x^*) = 0$. If such an adversary exists, then the forgery on x^* can be used to distinguish a valid public key from a public key pk_C generated from $\mathsf{ConstrainGen}(\lambda, C)$. This targeted form of unforgeability is the basis for the security of our key exchange protocol, though we need the full power of constrained signatures in order to prove security.

Though not critical to the security of our key exchange protocol, we note that constrained signatures actually imply existential unforgeability. The intuition is as follows: Suppose an adversary makes q chosen message queries and is able to produce an existential forgery. Let $C(x)$ be a 1-bit PRF that outputs 1 with probability $1-\epsilon$ and 0 with probability ϵ. By setting $\epsilon \approx 1/q$, with non negligible probability, all of the adversary's signing queries will be on inputs x where $C(x) = 1$ but the forgery will be on x with $C(x) = 0$. This forgery can then be used to distinguish the valid public key from the invalid public key.

For now, we assume the existence of such a constrained signature scheme. In Section 9, we show how to build such a signature scheme from indistinguishability obfuscation.

4.2 Security of our No-Setup Key Exchange Protocol

The following theorem proves the security of Construction 4.4:

Theorem 4.6. *Let* PRF *be a constrained pseudorandom function for polynomial size circuit predicates and* $(\mathsf{G}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{ConstrainGen})$ *is a constrained signature scheme for circuit predicates. Let* iO *be an indistinguishability obfuscator. Then Construction 4.4 is a semi-statically secure NIKE with no setup.*

Proof. We start with an adversary $\mathcal{A}$ breaking the semi-static security game as in Definition 3.2. We prove security through a sequence of games:

Game 0. This is the standard semi-static attack game. Assume $\mathcal{A}$ has advantage ϵ.

Game 1.i. For $i \in [N]$ we define Game 1.i. These games are identical to **Game 0**, except for the following modification. When the adversary commits to a set $\hat{S}$, we compute the circuit C_1 which takes an N-bit string, interprets the string as a set $S \subseteq [N]$, and outputs 1 if and only if $S \nsubseteq \hat{S}$. Note that t (the implicit input to G), is set to some upper bound on $|C_1|$, which is polynomial in N and G. Then, for every user $j \in \hat{S}, j \leq i$, we run $\mathsf{ConstrainGen}$ on C_1 to obtain a constrained public key pk_j. When answering a register honest user query on $j \in \hat{S}, j \leq i$, we will use this pk_j value. All other queries are answered as in **Game 0**. Observe that **Game 1.**0 is the same as **Game 0**. Define Game 1 as Game 1.N. In this Game 1 all public keys are replaced by constrained public keys, constrained by C_1.

Suppose $\mathcal{A}$ can distinguish Game 1.$(i-1)$ from Game 1.i. Then we construct an adversary $\mathcal{B}_{1,i}$ that breaks the constrained signature security. $\mathcal{B}_{1,i}$ asks its challenge for the constrained verification key pk corresponding to the circuit C_1. For an honest register query on $j \in \hat{S}, j \leq i-1$, $\mathcal{B}_{1,i}$ generates $(\mathsf{sk}_j, \mathsf{pk}_j) = \mathsf{ConstrainGen}(C_1)$ and responds with pk_j. For an honest register query on i (if $i \in \hat{S}$), $\mathcal{B}_{1,i}$ responds with pk. For all other honest register queries, $\mathcal{B}$ responds as in **Game 0**. Now $\mathcal{B}_{1,i}$ can answer all of $\mathcal{A}$'s queries on its own, except for reveal queries where user i is asked to compute the shared key. For these queries, $\mathcal{B}_{1,i}$ queries the constrained signature challenger for the appropriate signature, and then computes the shared key. Note that since reveal queries on a set S require $S \nsubseteq \hat{S}$, we have that $C_1(X) = 1$. Therefore, the query $\mathcal{B}_{1,i}$ makes to the constrained signature challenger will always be a legal query. If pk is the constrained key pk_{C_1}, then $\mathcal{B}_{1,i}$ correctly simulates **Game 1.i**, whereas if pk is a correctly generated key, then $\mathcal{B}$ simulates **Game 1.**$(i-1)$. Thus $\mathcal{B}_{1,i}$ has advantage equal to the probability $\mathcal{A}$ distinguishes **Game 1.**$(i-1)$ from **Game 1.i**.

We can combine all of the non-uniform adversaries $\mathcal{B}_{1,i}$ into a single uniform adversary $\mathcal{B}_1$ that operates by first choosing a random i in $[N]$ and then runs $\mathcal{B}_{1,i}$. $\mathcal{B}_1$ will have advantage equal to the average of the advantages of $\mathcal{B}_{1,i}$, which is equal to the probability $\mathcal{A}$ distinguishes **Game 0** from

Game 1, divided by N. The security of the signature scheme shows that this probability must be negligible, and hence $\mathcal{A}$ has advantage $\epsilon - \mathsf{negl}$ in breaking **Game 1**

Game 2.i For $i \in [N]$ these games are identical to **Game 1**, except for the following modification. When the adversary commits to a set $\hat{S}$, generate the public keys pk_j for $j \in \hat{S}$ as in **Game 1**, and then we compute the circuit C_2 which takes an N-bit string interprets as a set S, and a list of public keys of size at most G. If $S \subseteq \hat{S}$, and the list of public keys is exactly $\{\mathsf{pk}_i\}_{i\in S}$, then C_2 outputs 0. Otherwise, C_2 outputs 1. Then, for $j \in \hat{S}, j \leq i$, the challenger will generate $P_{\mathsf{iO},j}$ as the obfuscation of the program P'_{KE} in Figure 5. All other users will have the correct public values. Notice that **Game 2.**0 is the same as **Game 1**. We define **Game 2** to be **Game 2.N**. Because pk_j for $j \in \hat{S}$ have no valid signatures relative to sets $S \subseteq \hat{S}$, the programs P'_{KE} and P_{KE} have identical behaviors. Thus, their obfuscations are indistinguishable, and it is therefore straightforward to show that **Game 2.**$(i-1)$ is indistinguishable from **Game 2.**i for all i. Therefore, we can conclude that in **Game 2**=**Game 2.**N, $\mathcal{A}$ has advantage $\epsilon - \mathsf{negl}$.

Inputs: $S \subset [N], |S| \leq G$, $\{\mathsf{pk}_j\}_{j\in S}$, i, σ

Constants: $\mathbf{PRF}^{C_2}$ (replaces PRF in program P_{KE})

1. Interpret S as an N-bit string.
2. If $\mathsf{Ver}(\mathsf{pk}_i, S, \sigma)$ fails, abort and output $\perp$.
3. Otherwise, output $\mathbf{PRF}^{C_2}(S, \{\mathsf{pk}_j\}_{j\in S})$.

Figure 5: The program P'_{KE}.

Game 3.i For $i \in [N]$ these games are identical to **Game 2**, except for the following modification. During a test query on a set S, if the lowest user in S is j (which means $P_{\mathsf{iO},j}$ is used to generate the key) and $j \leq i$, then respond with a random key instead of the properly generated key. For all other test queries, respond with the correctly generated key. Notice that **Game 3.**0 is the same as **Game 2** in the case where the challenger chooses the correct key, and **Game 3.**N is **Game 2** but where the challenger responds with a random key. If $\mathcal{A}$ can distinguish between **Game 3.**$(i-1)$ and **Game 3-**i, then it is straightforward to build a constrained pseudorandom function adversary for PRF. Thus, we can conclude that $\mathcal{A}$ has negligible advantage at distinguishing **Game 3.**0 from **Game 3.**N, meaning $\mathcal{A}$ also has negligible advantage in **Game 2**. Finally, this means $\mathcal{A}$ had negligible advantage in **Game 0**, completing the proof.

□

5 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 $\mathsf{sk}_1, \dots, \mathsf{sk}_N$ and public parameters params. For $i = 1, \dots, N$, recipient number i is given the private key sk_i.

$\mathsf{Enc}(\mathsf{params}, S)$ Takes as input a subset $S \subseteq \{1, \dots, N\}$, 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 $S \subseteq \{1, \dots, N\}$, a user id $i \in \{1, \dots, N\}$ 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 $S \subseteq \{1, \dots, n\}$ and all $i \in S$ if $(\mathsf{params}, (\mathsf{sk}_1, \dots, \mathsf{sk}_N)) \overset{R}{\leftarrow} \mathsf{Setup}(\lambda, N)$ and $(\mathsf{Hdr}, k) \overset{R}{\leftarrow} \mathsf{Enc}(\mathsf{params}, S)$ 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 $i = 1, \dots, N$ 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 5.1. 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 5.2. 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 5.3 ([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 5.4. 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^*.

5.1 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:

$\mathsf{Setup}(\lambda, N)$ Set $G = N$, and run the key exchange setup algorithm to obtain public parameters params'.

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

$\mathsf{Enc}(\mathsf{params}, \mathsf{bk}, S)$ Let i^* be some user in S. Let $k \leftarrow \mathsf{KeyGen}(\mathsf{params}', i^*, \mathsf{sk}_{i^*}, S, \{\mathsf{pv}_i\}_{i\in[S]})$. Output $(\mathsf{Hdr} = \emptyset, k)$.

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

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 5.5. *Given a statically, 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:

$\mathsf{Setup}(\lambda, N)$ Set $G = N + 1$ and run the key exchange setup for algorithm for $N + 1$ users to obtain public parameters params'.

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

$\mathsf{Enc}(\mathsf{params}, S)$ Run $(\mathsf{sk}_{N+1}, \mathsf{pv}_{N+1}) \leftarrow \mathsf{Publish}(\lambda, G = N + 1, N + 1, N + 1)$. Compute $k \leftarrow \mathsf{KeyGen}(\mathsf{params}', N + 1, \mathsf{sk}_{N+1}, S \bigcup \{N + 1\}, \{\mathsf{pv}_i\}_{i\in S \bigcup \{N+1\}})$. Output $(\mathsf{Hdr} = \mathsf{pv}_{N+1}, k)$.

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

We obtain the following theorem:

Theorem 5.6. *Given a statically, 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 5.6 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 5.6 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 5.6, which is very similar to the proof of Theorem 5.5:

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.

Using a simple hybrid argument, we can assume the adversary makes only a single challenge query.

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 N register honest user queries for identities $i \in [N]$, receiving public values pv_i in return. $\mathcal{B}$ sends to $\mathcal{A}$ the public parameters $\mathsf{params} = (\mathsf{params}', \{pv_i\}_{i\in[N]})$. 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 i. When $\mathcal{A}$ makes the challenge query on set S, $\mathcal{B}$ registers a new honest user $i = N+1$ (obtaining public value pv_{N+1}), and then makes a test query on the identities $S \bigcup \{N+1\}$. $\mathcal{B}$ returns pv_{N+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.

□

6 Traitor Tracing With Small Parameters

In this section, we present a private linear broadcast encryption (PLBE) scheme, which has short ciphertexts, secret keys, and public keys. Boneh, Sahai, and Waters [BSW06] show that this implies a fully collusion resistant traitor tracing system with the same parameters.

Our approach gives a more general primitive called a *recipient-private* broadcast encryption system: a broadcast system where ciphertexts reveal no information about the intended recipient set (beyond what is explicitly allowed) [BBW06]. The system is made up of three algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ as in the public recipient 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.
- $\binom{[N]}{r}$, 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.
- $\texttt{Lin}_N = \{\emptyset = [0], [1], \ldots, [N]\}$. Privacy with respect to this set system gives private linear broadcast encryption.

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 $\texttt{EXP}(b)$ on an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ parameterized by the total number of recipients N and by a bit $b \in \{0, 1\}$:

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

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

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 $\texttt{EXP}(b)$. Define $\texttt{PBE}^{(\text{adv})}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 6.1. We say that a broadcast encryption system is $\mathcal{S}$-recipient-private selectively (resp. semi-statically, adaptively) semantically secure if it is a selectively (resp. semi-statically, adaptively) semantically secure broadcast system and for all probabilistic polynomial time adversaries $\mathcal{A}$ the function $\texttt{PBE}^{(\text{adv})}(\lambda)$ is negligible.

Definition 6.1 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 $\texttt{EXP}(b)$. Security for a public-key recipient-private system is defined analogously with bk removed from the definition of $\texttt{EXP}(b)$.

Remark 6.2. We note that for private linear broadcast, there are only a polynomial number of recipient sets, meaning selective and adaptive security are equivalent.

6.1 Private broadcast encryption: first construction

Construction overview. Since a broadcast ciphertext should reveal as little as possible about the recipient set S our plan is to embed 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 is non-trivial, and requires a certain type of encryption system.

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.

We immediately see a problem with the description above: the obfuscated program must, at a minimum, have each of the x_i embedded in it, making the program and hence the public key linear in size. To keep the public key short, we instead generate the seeds s_i using a pseudorandom function PRF_{sk}: $s_i = \mathsf{PRF}_{sk}(i)$. We then have the program compute the x_i on the fly as $x_i = \mathsf{PRG}(\mathsf{PRF}_{sk}(i))$.

Another problem with the above description is that 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.

To solve this problem, we authenticate the encrypted recipient set using a message authentication code (MAC). However, proving security is a bit challenging because the decryption program must include the secret MAC key, and we need to ensure that this key does not leak to the attacker. We do so by implementing the MAC using a constrained PRF that supports interval constraints. We then prove that this is sufficient to thwart the aforementioned malleability attacks and allows us to prove security of the scheme.

We will need a slightly stronger notion of constrained PRF, which we will call an *interval* constrained PRF:

Definition 6.3. $\mathsf{PRF} : [L] \to \mathcal{Y}$ is a *interval constrained* PRF if it has the following properties:

- PRF is a constrained PRF for the class of intervals $[A, B]$ for $A \leq B$ and $A, B \in [L]$
- The constrained programs $\mathsf{PRF}^{[A,B]}$ each have size at most $O(\log L)$.

The GGM construction for pseudorandom functions (or any constrained PRF for prefixes) satisfies this notion. For our construction, we will only need to support intervals of the form $[1, \ell]$ and $[\ell', L]$.

We next present our private linear broadcast construction (i.e. the case where $\mathcal{S} = \mathtt{Lin}_N$). We first present a private-key variant, where a secret broadcast key is required to encrypt. We then show how to make the scheme public-key. We discuss extending this construction to other set systems at the end of the section.

Construction 6.4. *Our traitor tracing scheme consists of three algorithms* ($\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}$) *defined as follows:*

$\mathsf{Setup}(\lambda, N)$ *Let* $\mathsf{PRF}_{enc} : \{0,1\}^{2\lambda} \to [N]$ *and* $\mathsf{PRF}_{key} : \{0,1\}^{2\lambda} \times \{0,\dots,N\} \to \{0,1\}^{\lambda}$ *be punctured PRFs and* $\mathsf{PRF}_{mac} : \{0,1\}^{2\lambda} \times \{0,\dots,N\} \to \{0,1\}^{\lambda}$ *and* $\mathsf{PRF}_{sk} : [N] \to \{0,1\}^{\lambda}$ *be interval constrained PRFs. Let* $s_i \leftarrow \mathsf{PRF}_{sk}(i)$ *for each* $i \in [N]$*. Let* $P_{TT-\mathsf{Dec}}$ *be the program in Figure 6, padded to the appropriate length. User* i*'s secret key is* s_i*, and the public parameters are* $\mathsf{params} = P_{\mathsf{Dec}} = \mathsf{iO}(P_{TT-\mathsf{Dec}})$.

$\mathsf{Enc}((\mathsf{PRF}_{enc}, \mathsf{PRF}_{mac}, \mathsf{PRF}_{key}), [j])$ *Pick a random* $r \in \{0,1\}^{2\lambda}$*, and let* $c_1 \leftarrow \mathsf{PRF}_{enc}(r) + j \mod (N+1)$*. Let* $c_2 \leftarrow \mathsf{PRF}_{mac}(r, c_1)$*. Finally, let* $k \leftarrow \mathsf{PRF}_{key}(r, c_1)$*. Output* $(\mathsf{Hdr} = (r, c_1, c_2), k)$.

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

Inputs: r, c_1, c_2, s, i
Constants: $\mathsf{PRF}_{enc}, \mathsf{PRF}_{mac}, \mathsf{PRF}_{key}, \mathsf{PRF}_{sk}$

1. Let $j \leftarrow c_1 - \mathsf{PRF}_1(r) \mod (N+1)$
2. Let $x \leftarrow \mathsf{PRG}(\mathsf{PRF}_{sk}(i))$
3. Let $y \leftarrow \mathsf{PRG}(\mathsf{PRF}_{mac}(r, c_1))$
4. Check that $\mathsf{PRG}(s) = x$, $\mathsf{PRG}(c_2) = y$, and $i \le j$. If check fails, output $\perp$ and stop
5. Otherwise, output $\mathsf{PRF}_{key}(r, c_1)$

Figure 6: The program $P_{TT-\mathsf{Dec}}$.

A public-key system. As described, our scheme requires a 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_{enc} 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 let P_{Enc} be the obfuscation of the program $P_{TT-\mathsf{Enc}}$ in Figure 7 (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 $P_{TT-\mathsf{Enc}}$ does not help the adversary learn anything about PRF_{enc}, PRF_{mac}, or PRF_{key} at the point r^*. We get the following encryption algorithm:

$\mathsf{Enc}(\mathsf{params}, [j])$ Let $t \leftarrow \{0,1\}^{\lambda}$. Run $(r, c, k) \leftarrow P_{\mathsf{Enc}}(j, t)$. Output (r, c) as the header and k as the message encryption key.

```
Inputs: $j, t$
Constants: $\mathsf{PRF}_{enc}, \mathsf{PRF}_{mac}, \mathsf{PRF}_{key}$

  1. $r \leftarrow \mathsf{PRG}(t)$
  2. $c_1 \leftarrow \mathsf{PRF}_{enc}(r) + j \mod (N+1)$
  3. $c_2 \leftarrow \mathsf{PRF}_{mac}(r, c_1)$
  4. $k \leftarrow \mathsf{PRF}_{key}(r, c_1)$
  5. Output $(\mathsf{Hdr} = (r, c_1, c_2), k)$
```

Figure 7: The program $P_{TT-\mathsf{Enc}}$.

Secret keys have length λ, and ciphertexts have size $3\lambda + \log(N+1)$. The public key consists of two obfuscated programs. The size of these programs is only dependent polylogarithmically on the number of users, so the obfuscated programs will have size $\mathsf{poly}(\log N, \lambda)$. Therefore, we simultaneously achieve small ciphertexts, secret keys, and public keys. Security is given by the following theorem:

Theorem 6.5. *If* PRF_{enc} *and* PRF_{key} *are secure puncture PRFs,* PRF_{mac} *and* PRF_{sk} *are secure interval constrained PRFs, and* PRG *is a secure pseudorandom generator, then* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *in Construction 6.4 is an adaptively secure private linear broadcast encryption scheme.*

Proof. We first prove semantic security. Since there are only a polynomial number of sets in the set system, we can assume the adversary commits to a target set $[j^*]$ before seeing any public parameters or secret keys. The adversary then receives the public parameters, as well as the secret keys for all $i > j^*$. The adversary's challenge will be on the set $[j^*]$. Therefore, we can also construct the challenge ciphertext before giving the adversary the public parameters and secret keys. The challenge header will be (r^*, c_1^*, c_2^*) where r^* is a random PRG sample, $c_1^* = \mathsf{PRF}_{enc}(r^*) + j^*$ is an encryption of j^*, and $c_2^* = \mathsf{PRF}_{mac}(r^*, c^*)$ is a MAC.

First, we slightly change the game above by choosing the challenge r^* as a truly random value in $\{0,1\}^{2\lambda}$. This game is indistinguishable from the original game by the security of PRG. Now, with high probability, r^* is not in the image of PRG. Since we generate (r^*, c_1^*, c_2^*) before giving the adversary the public key, we can puncture PRF_{key} at the point (r^*, c_1^*) in the encryption program $P_{TT-\mathsf{Enc}}$ without changing its functionality. Indistinguishability of obfuscations shows that the adversary cannot detect this change.

Next, our goal is to puncture PRF_{key} at the point (r^*, c_1^*) in the decryption program $P_{TT-\mathsf{Dec}}$. The naive way to accomplish this is to puncture PRF_{sk} at all $i \le j^*$, and hard-code $x_i = \mathsf{PRG}(\mathsf{PRF}_{sk}(i))$ into the decryption program. Then we can replace each of the x_i for $i \le j^*$ embedded in the decryption program with a truly random value in $\{0,1\}^{2\lambda}$, and with high probability none of these will be in the image of PRG. This will allow us to add a check that $i > j^*$ at the beginning of the obfuscated decryption program and stop the program if the check fails. This check does not change the functionality of the program. Since in step (4) the program checks that $i \le j$ and aborts if not, we know that on the challenge ciphertext, where $j = j^*$, one of these checks always fails so that the program will never reach step (5). Thus, we can puncture PRF_{key} at the challenge, as desired.

The problem is that we hard-coded j^* different x_i values in the decryption program, making the program size potentially linear in N. Therefore, to apply iO, we would then need the original $P_{TT-\mathsf{Dec}}$ to also be linear in size, making our public keys large.

Our solution is to puncture PRF_{sk} at the relevant points, one point at a time. This will let us keep the size of the program small. Define hybrid decryption programs $P^{(\ell)}_{TT-\mathsf{Dec}}$ and $P^{(\ell+1/2)}_{TT-\mathsf{Dec}}$ as in Figures 8 and 9. Note that $P^{(\ell+1/2)}_{TT-\mathsf{Dec}}$ includes a *single* hard-coded value x_ℓ used in step (3).

```
Inputs: $r, c_1, c_2, s, i$
Constants: $\mathsf{PRF}_{enc}, \mathsf{PRF}_{mac}, \mathsf{PRF}_{key}, \mathsf{PRF}_{sk}^{\{\ell,\dots,N\}}$
  1. Let $j \leftarrow c_1 - \mathsf{PRF}_1(r) \mod (N+1)$
  2. if $i < \ell$ output $\perp$ and stop        // this added test is the only change from $P_{TT-\mathsf{Dec}}$
  3. Let $x \leftarrow \mathsf{PRG}(\mathsf{PRF}_{sk}(i))$
  4. Let $y \leftarrow \mathsf{PRG}(\mathsf{PRF}_{mac}(r, c_1))$
  5. Check that $\mathsf{PRG}(s) = x$, $\mathsf{PRG}(c_2) = y$, and $i \le j$. If check fails, output $\perp$ and stop
  6. Otherwise, output $\mathsf{PRF}_{key}(r, c_1)$
```

Figure 8: The program $P^{(\ell)}_{TT-\mathsf{Dec}}$ for $\ell = 1, \dots, j^* + 1$.

```
Inputs: $r, c_1, c_2, s, i$
Constants: $\mathsf{PRF}_{enc}, \mathsf{PRF}_{mac}, \mathsf{PRF}_{key}, \mathsf{PRF}_{sk}^{\{\ell+1,\dots,N\}}, x_\ell$
  1. Let $j \leftarrow c_1 - \mathsf{PRF}_1(r) \mod (N+1)$
  2. if $i < \ell$ output $\perp$ and stop
  3. Let $x \leftarrow \begin{cases} \mathsf{PRG}(\mathsf{PRF}_{sk}(i)) & \text{if } i > \ell \\ x_\ell & \text{if } i = \ell \end{cases}$        // this special case is only change from $P^{(\ell)}_{TT-\mathsf{Dec}}$
  4. Let $y \leftarrow \mathsf{PRG}(\mathsf{PRF}_{mac}(r, c_1))$
  5. Check that $\mathsf{PRG}(s) = x$, $\mathsf{PRG}(c_2) = y$, and $i \le j$. If check fails, output $\perp$ and stop
  6. Otherwise, output $\mathsf{PRF}_{key}(r, c_1)$
```

Figure 9: The program $P^{(\ell+1/2)}_{TT-\mathsf{Dec}}$ for $\ell = 1, \dots, j^*$.

Each of these programs is at most $\mathsf{poly}(\log N, \lambda)$ is size. Therefore, to apply iO we only need the original program to be $\mathsf{poly}(\log N, \lambda)$ size, keeping the public key short.

Now, observe that $P_{TT-\mathsf{Dec}}$ is equivalent to $P^{(1)}_{TT-\mathsf{Dec}}$. Moreover, for all $\ell \in [j^* + 1]$ when $x_\ell = \mathsf{PRG}(\mathsf{PRF}_{key}(\ell))$ the program $P^{(\ell)}_{TT-\mathsf{Dec}}$ is functionally equivalent to $P^{(\ell+1/2)}_{TT-\mathsf{Dec}}$. Then indistinguishability of obfuscations allows us move from ℓ to $\ell + 1/2$ without the adversary detecting the change. Now, PRF_{sk} can be punctured at the point ℓ so that $s_\ell = \mathsf{PRF}_{sk}(\ell)$ can be replaced by a random value $s_\ell \in \{0,1\}^\lambda$ by the security of PRF_{sk}. Next $x_\ell = \mathsf{PRG}(s_\ell)$ can be exchanged for a completely random $x_\ell \in \{0,1\}^{2\lambda}$ by the security of PRG, so long as $\ell \leq j^*$ (so that the adversary does not get to see s_ℓ). A truly random x_ℓ is then, with overwhelming probability, not in the image of PRG, in which case $P^{(\ell+1/2)}_{TT-\mathsf{Dec}}$ will always output $\perp$ when $i = \ell$, so we can modify the check in step (2) to abort if $i < \ell + 1$, obtaining the program $P^{(\ell+1)}_{TT_{\mathsf{Dec}}}$. Thus, so long as $\ell \leq j^*$, indistinguishability of obfuscations lets us to move from the program $(\ell + 1/2)$ to the program $(\ell + 1)$ without the adversary detecting the change.

In summary, we can define a sequence of hybrids, the first of which is the original game, and the last of which we give the adversary the decryption program $P^{(j^*+1)}_{TT-\mathsf{Dec}}$, and each hybrid is indistinguishable from the previous hybrid.

Now, the program $P^{(j^*+1)}_{TT-\mathsf{Dec}}$ outputs $\perp$ on encryptions to the set $[j^*]$ since it checks that $i \leq j^*$ and $i \geq j^* + 1$. Therefore, since we generate the challenge ciphertext (r^*, c_1^*, c_2^*) before giving the adversary the public key, we can puncture PRF_{key} at (r^*, c_1^*) without changing the functionality of the program. As a result we can simulate the entire view of the adversary with only the punctured program $\mathsf{PRF}^{\{(r^*,c_1^*)\}}_{key}$. But then, by the security of PRF_{key}, the challenge key $k^* = \mathsf{PRF}_{key}(r^*, c_1^*)$ is indistinguishable from a truly random key, thus proving the semantic security of the scheme.

Recipient privacy. It remains to prove recipient privacy. The proof is similar to semantic security and we only sketch the proof. We can again assume the adversary commits to a value j^*, but now he receives secret keys for every user other than j^*. In the challenge, we encrypt either to $[j^*]$ or $[j^* - 1]$. Our goal is to show that the adversary cannot tell the difference. First, we puncture PRF_{sk} at j^* and hard-code $x_{j^*} = \mathsf{PRG}(s_{j^*}) = \mathsf{PRG}(\mathsf{PRF}_{sk}(i))$ into $P_{TT-\mathsf{Dec}}$. Then we replace s_{j^*} with a truly random value, and then replace x_{j^*} itself with a truly random value. The indistinguishability of obfuscations, the security of PRF_{sk}, and the security of PRG show that these changes are not detectable. Now, with overwhelming probability, x_{j^*} is not in the image of PRG, so we can modify $P_{TT-\mathsf{Dec}}$ to abort when $i = j^*$ without affecting the program.

Next, we need to make sure our challenge ciphertext (r^*, c_1^*, c_2^*) is non-malleable so that the adversary cannot use $P_{TT-\mathsf{Dec}}$ to learn the set we encrypted to. To accomplish this, pick a point (r^*, c_1'), puncture PRF_{mac} at (r^*, c_1'), and hard-code $y_{c_1'} = \mathsf{PRG}(\mathsf{PRF}_{mac}(r^*, c_1'))$ into the program. If $c_1' \neq c_1^*$, we can replace $y_{c_1'}$ with a truly random value, so that with high probability there will be no c_2 with $\mathsf{PRG}(c_2) = y_{c_1'}$. Thus, without changing the functionality of the program, we can include the following check: if $r = r^*$ and $c_1 = c_1'$, output $\perp$ and stop. We can also remove the value $y_{c_1'}$ from the program. If $c_1' = c_1^*$, then we can instead hard-code $y^* = \mathsf{PRG}(c_2^*)$ into the program and add the check: if $r = r^*$, $c_1 = c_1^*$, and $\mathsf{PRG}(c_2) \neq y^*$, output $\perp$ and stop.

By doing these changes for each of the $\log(N+1)$ different c_1' values, we move to a setting where the only (r^*, c_1, c_2) that $P_{TT-\mathsf{Dec}}$ will run on is (r^*, c_1^*, c_2^*). We need to make sure we can iterate through the c_1' in a way that keeps the programs short. By iterating through the c_1' in order, we ensure that we only puncture PRF_{mac} at an interval: $(r^*, 0), \ldots (r^*, \ell)$ for some ℓ. In otherwords, we constrain PRF_{mac} to two intervals: all elements less that $(r^*, 0)$ or greater than (r^*, ℓ). Moreover, we can combine the $c_1 \neq c_1'$ checks together into at most two inequality checks (one for the c_1' less than c_1^*, and one for the c_1' greater than c_1^*). Therefore, using the indistinguishability of obfuscations, the security of PRG, and the security of PRF_{mac}, we move to a game where the adversary sees the

```
Inputs: $r, c_1, c_2, s, i$
Constants: $\mathsf{PRF}_{enc}, \mathsf{PRF}_{mac}^{\overline{\{(r^*,c_1):c_1\in\{0,\dots,N\}\}}}, \mathsf{PRF}_{key}, \mathsf{PRF}_{sk}^{\overline{\{j^*\}}}, y^*$

1. Let $j \leftarrow c_1 - \mathsf{PRF}_1(r) \mod (N+1)$
2. Check that $i \neq j^*$. If $r = r^*$ check that $c_1 = c_1^*$. If check fails, output $\perp$.
3. Let $x \leftarrow \mathsf{PRG}(\mathsf{PRF}_{sk}(i))$
4. Let $y \leftarrow \begin{cases} \mathsf{PRG}(\mathsf{PRF}_{mac}(r, c_1)) & \text{if } r \neq r^* \\ y^* & \text{if } r = r^* \text{ and } c_1 = c_1^* \end{cases}$
5. Check that $\mathsf{PRG}(s) = x$, $\mathsf{PRG}(c_2) = y$, and $i \leq j$. If check fails, output $\perp$
6. Otherwise, output $\mathsf{PRF}_{key}(r, c_1)$
```

Figure 10: The program $P'_{TT-\mathsf{Dec}}$.

obfuscation of the program in Figure 10.

Now we can puncture PRF_{enc} at r^* and hard-code $z^* = \mathsf{PRF}_{enc}(r^*)$ into the obfuscated program. Next, replace z^* with a truly random value, which is not noticeable by the security of PRF_{enc}. Notice that we can change from an encryption to $[j^*]$ to an encryption to $[j^* - 1]$ by simply adding 1 to z^* (mod $N+1$). Since z^* is a random value mod $N+1$, so is z^*+1. But moving from z^* when encrypting $[j^*]$ to z^*+1 when encrypting to $[j^*-1]$ does not affect the functionality of $P'_{TT-\mathsf{Dec}}$, meaning that the adversary cannot tell an encryption to $[j^*]$ from an encryption to $[j^*-1]$. This proves the recipient privacy of our scheme. □

6.2 Extension to Other Set Systems

We can easily extend Construction 6.4 to the other set systems of interest: $\mathcal{S} = 2^{[N]}$ and $\mathcal{S} = \binom{[N]}{r}$. The only difference is that we represent sets $S \in \mathcal{S}$ as integers in $\{0, \dots, L-1\}$ where $L = |\mathcal{S}|$, and let PRF_{enc} output integers in $\{0, \dots, L-1\}$. We obtain recipient private broadcast systems for these collections of sets where secret keys have size λ, independent of N. Ciphertexts will have size 3λ plus the size of the representation of elements in the collection. For $\mathcal{S} = 2^{[N]}$, this is $3\lambda + N$, and for $\mathcal{S} = \binom{[N]}{r}$, this is $3\lambda + \log\binom{N}{r} \approx 3\lambda + r\log N$. These are essentially optimal. The public keys will be polynomial in the security parameter and the size of representations.

However, the proof of security is problematic, even for the simpler case of selective security. In the recipient private proof, we gradually puncture PRF_{mac} at all ciphertexts with the same randomness r^* as the challenge ciphertext, and each step requires several hybrids. The number of hybrids is therefore proportional to the number of sets in our set system. Therefore, for either $\mathcal{S} = 2^{[N]}$ or $\mathcal{S} = \binom{[N]}{r}$, this is exponential, meaning we take an exponential hit in the security of the construction.

One solution is to use complexity leveraging, assuming that the underlying primitives have exponential security. In the next section, we show that we can obtain security without complexity leveraging using a different technique. The drawback to our next scheme is the public key size, which is $N^{O(1)}$.

7 Recipient-Private Broadcast Encryption

In this section, we give an alternate construction of recipient-private broadcast encryption with short ciphertexts. There are two variants of our scheme, depending on the security of the underlying PRF. Using a punctured PRF, we obtain selective security without complexity leveraging, which was not possible with our previous scheme. Using a constrained PRF for circuit predicates, we obtain *semi-static* security. While the known constructions of such PRFs ([BW13] and Section 9) all require complexity leveraging, our scheme does not require any additional complexity leveraging. 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. The main drawback is that our public keys will not be as compact, as they will be polynomial in N and λ.

7.1 Our Recipient-Private Broadcast Scheme

Preventing malleability by authenticating caused our security proof to require an exponential number of hybrids. Our idea in this section is to allow malleability, but prevent the attacker from learning anything by changing the ciphertext. We call our solution the *malleable-key* method.

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 $\texttt{Lin}_N$. 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 $\mathcal{S} = \texttt{Lin}_N$: the key space $\mathcal{K}$ is the set of all $(N + 1)!$ permutations σ on $\{0, 1, \dots N\}$. Encryption and decryption are defined as:

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

  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. Similar to Section 6, our public parameters will consist of two obfuscated programs: one for encryption and one for decryption. If the encryption program is not included, our scheme becomes a secret key scheme.

Construction 7.1. *Let $\mathcal{S} = 2^{[N]}, \texttt{Lin}_N$, 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 $\mathsf{PRF}_{enc} : \{0,1\}^{2\lambda} \to \mathcal{K}$ and $\mathsf{PRF}_{key} : \{0,1\}^{2\lambda} \times \mathcal{S} \to \{0,1\}^\lambda$. Let $P_{PBE-\mathsf{Enc}}$ and $P_{PBE-\mathsf{Dec}}$ be the programs given in Figures 11 and 12, padded to the appropriate size. Player i receives the private key s_i, and the public key is* $\mathsf{params} = (P_{\mathsf{Enc}} = \mathsf{iO}(P_{PBE-\mathsf{Enc}}), P_{\mathsf{Dec}} = \mathsf{iO}(P_{PBE-\mathsf{Dec}}))$.

$\mathsf{Enc}(\mathsf{params}, S)$ *Generate a random seed t and output* $(\mathsf{Hdr} = (r, c), k) \leftarrow P_{\mathsf{Enc}}(S, t)$

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

```
Inputs: $S, t$
Constants: $\mathsf{PRF}_{enc}, \mathsf{PRF}_{key}$

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

Figure 11: The program $P_{PBE-\mathsf{Enc}}$.

```
Inputs: $r, c, s, i$
Constants: $x_1, \dots x_N, \mathsf{PRF}_{enc}, \mathsf{PRF}_{key}$

  1. Let $k \leftarrow \mathsf{PRF}_{enc}(r)$
  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 $\mathsf{PRF}_{key}(r, c)$
```

Figure 12: The program $P_{PBE-\mathsf{Dec}}$.

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:

Theorem 7.2. *For $\mathcal{S} = 2^{[N]}, \mathtt{Lin}_N$, or $\binom{[N]}{r}$, if iO is a secure indistinguishability obfuscator, PRF_{enc} is a secure punctured PRF, and PRF_{key} is a secure constrained PRF for circuit predicates (resp. punctured PRF), then Construction 7.1 is a semi-statically (resp. selectively) secure recipient-private broadcast encryption scheme.*

We prove Theorem 7.2 in a more general form in Section 7.4. For now we sketch its proof for the linear set system $\mathtt{Lin}_N = \{\emptyset, [1], \dots, [N]\}$.

Proof sketch for $\mathcal{S} = \mathtt{Lin}_N$. We prove that the scheme satisfies Definition 6.1. 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_{enc} at the challenge randomness r^*, and include the "challenge key" $k^* = \mathsf{PRF}_{enc}(r^*)$ hard-coded in the program $P_{PBE-\mathsf{Dec}}$. The resulting program $P'_{PBE-\mathsf{Dec}}$ is shown in Figure 14. 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_{enc} implies that we can actually choose k^* at random, independent of PRF_{enc}. Recall that k^* is just a random permutation on $\{0, \dots, N\}$. 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. □

Remark 7.3. Similar to the traitor tracing scheme from Section 6, we can avoid having all of the x_i values hard-coded into $P_{PBE-\mathsf{Dec}}$: instead have s_i be generated by a PRF and the x_i computed on the fly as needed. We did this in Section 6 to shrink our public keys. However, $P_{PBE-\mathsf{Dec}}$ in this section will still be quite large because the encryption scheme for sets (E, D) has polynomial-sized circuits. Therefore, generating s_i using a pseudorandom function will have only a minimal impact on public key size.

```
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 13: The algorithm Int2Set.

7.2 Applications

7.3 Private Broadcast Encryption for All Sets

Instantiating our scheme with $\mathcal{S} = 2^{[N]}$, we can encrypt to an arbitrary subset of users while maintaining recipient privacy. Representing subsets $S \in \mathcal{S}$ by their incidence vectors $v_S \in \{0,1\}^N$, the ciphertexts will be of size $2\lambda + N$, which is essentially optimal. This scheme is very similar to the private broadcast scheme obtained at the end of Section 6, except that there is no authentication. The main difference is the proof of security, the proof from this section giving better security guarantees.

7.3.1 Traitor Tracing with Short Ciphertexts and Secret Keys

Instantiating our scheme with $\mathcal{S} = \mathtt{Lin}_N$, we get another construction of 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. This scheme will have marginally smaller ciphertexts than our previous construction from Section 6, since authentication is no longer necessary. However, the public keys in this scheme will be quite long: polynomial in N and λ.

7.3.2 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, only 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 13, 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 $2\lambda + \log \binom{N}{r}$.

7.4 Proving security for general set systems

We now turn to proving Theorem 7.2. 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 7.4 ($\mathcal{E}$-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 7.5 ($\mathcal{E}$-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 7.6 (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 7.2 that encompasses the three collections of sets we are interested in:

Theorem 7.7. *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_{enc} is a secure punctured PRF, PRF_{key} is a secure constrained PRF (resp. punctured PRF), and iO is an indistinguishability obfuscator, then Construction 7.1 instantiated with $\mathcal{K}, E, D$ is $\mathcal{S}$-recipient-private semi-statically (resp. selectively) semantically secure.

Before proving Theorem 7.7, we explain why it generalizes Theorem 7.2. 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 $\mathcal{E} = \{\{1\}, \dots, \{N\}\}$, 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 $\{0, \dots, N\}$.
- $\mathcal{S} = \binom{[N]}{r}$. 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 prove the semi-static case, the selective case being simpler. We need to prove that Construction 7.1 is both semi-static semantically secure and has recipient set privacy. We prove security for the private-key setting (without the program $P_{PBE-\mathsf{Enc}}$), the public-key setting being similar.

Semantic security follows a similar argument as in the proof of Theorem 5.6. 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_{enc} at r^* and include the key $\sigma^* \leftarrow \mathsf{PRF}_{enc}(r^*)$ hard-wired in the decryption program. Then we can replace σ^* with a truly random key. Next, we puncture PRF_{key} 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 7.1. $\mathcal{A}'$ 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 $\sigma^* = \mathsf{PRF}_{enc}(r^*)$. 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 14, puncturing PRF_{enc} at r^*, and including $\sigma^* = \mathsf{PRF}_{enc}(r^*)$ 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 $\mathsf{PRF}_{enc}^{\{r^*\}}$.

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_{enc} 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^*

```
Inputs: $r, c, s, i$
Constants: $x_1, \dots x_N$, $\mathsf{PRF}_{enc}^{\overline{\{r^*\}}}$, $\sigma^*$, $\mathsf{PRF}_{key}$

  1. Let $\boldsymbol{\sigma} \leftarrow \begin{cases} \mathsf{PRF}_1^{\overline{\{r^*\}}}(\boldsymbol{r}) & \text{if } \boldsymbol{r} \neq \boldsymbol{r}^* \\ \boldsymbol{\sigma}^* & \text{if } \boldsymbol{r} = \boldsymbol{r}^* \end{cases}$
  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 $\mathsf{PRF}_{key}(r, c)$
```

Figure 14: The program $P'_{PBE-\mathsf{Dec}}$.

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 7.1 is recipient-private.

□

8 Extensions

8.1 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. Our first traitor tracing scheme is CCA-secure, even if the adversary is able to learn the recipient set *and* key for a header of its choice. The reason is that the simulator is always able to answer decryption queries for ciphertexts with a different randomness than the challenge ciphertext, and we showed that all ciphertexts with the same randomness as the challenge can be rejected.

Our private broadcast scheme, including the second traitor tracing scheme, are also secure if the adversary is restricted to only learning the *key* for headers of his choice. 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.

8.2 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 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 8.1. *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 15 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* $k_S = P_{\mathsf{iO}}(S, s_{\mathsf{id}}, \mathsf{id})$.

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 15: The program P_{IBKE}.

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 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.

8.3 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 5 since the encryption step would require an extract query. Instead, we present a direct construction:

Construction 8.2. *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 16 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* $\mathsf{PRF}_{key}(\mathsf{id})$.

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 16: The program P_{IBBE}.

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

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.

8.4 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 traitor tracing and private broadcast schemes do not meet the notion of distributed broadcast encryption, since the decryption algorithm depends on each party's public keys. Nonetheless, for our private broadcast scheme, 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.

9 Constrained PRFs and Signatures from Indistinguishability Obfuscation

In this section, we explain how to realize constrained pseudorandom functions and signatures for circuit predicates from indistinguishability obfuscation.

9.1 Constrained PRFs

We start with constrained PRFs. 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 9.1. *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 17. Output $\mathsf{iO}(P_C)$.*

```
Inputs: $x$
Constants: $\mathsf{PRF}, C$

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

Figure 17: 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 9.2. *For any adversary $\mathcal{A}$ breaking the security of Construction 9.1, 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 $\mathtt{IO}^{(\text{adv})}{}_{\mathcal{B}}(\lambda) = \gamma/q_{cons}$. $\mathcal{A}$ thus has advantage at least $\epsilon/2^n q_{chal} - q_{cons}\mathtt{IO}^{(\text{adv})}{}_{\mathcal{B}}(\lambda)$ 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,

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

Rearranging gives the desired inequality, completing the proof.

□

9.2 Constrained Signatures

Now we move on to constrained signatures. Recall that these signatures come with an algorithm $\mathsf{ConstrainGen}(C)$, which outputs a public key pk_C which rejects all signatures for messages $x \in \{0,1\}^n$ with $C(x) = 0$. For security, we require that pk_C is indistinguishable from a valid pk by any adversary making a polynomial number of signature queries to x such that $C(x) = 1$. We give two constructions. Our first construction relies on witness indistinguishable proofs and perfectly binding commitments. Our second construction uses only obfuscation and one-way functions, but requires complexity leveraging.

Construction from Witness Indistinguishable Proofs. Our first construction is built from perfectly binding commitments and non-interactive witness indistinguishable proofs (called zaps). A very similar construction is implicit in the attribute-based encryption scheme of Garg et al. [GGSW13].

A non-interactive zap [DN00] for a relation R consists of an efficient prover and verifier P, V. The prover, on input an instance x and witness w such that $R(x, w) = 1$, produces a proof π. The verifier, on instance x and proof π, either accepts or rejects. We require the following properties:

- Perfect Completeness: for a true statement, the honest prover convinces the honest verfier with probability 1. That is, for any (x, w) where $R(x, w) = 1$, if $\pi \leftarrow P(x, w)$, then $V(x, \pi)$ accepts.

- Perfect Soundness: for a false statement, it is impossible to convince an honest verifier. That is, for any x and any (possibly unbounded) adversary $\mathcal{A}$, if $\pi \leftarrow \mathcal{A}(x)$ and $V(x, \pi)$ accepts, then there must exist a w such that $R(x, w) = 1$.

- Witness Indistinguishability: given the proof, it is computationally infeasible to tell which witness was used. In other words, for all PPT adversaries $\mathcal{A}$ in the following experiment, $\mathcal{A}$ has negligible advantage in guessing b: $\mathcal{A}$ produces (x, w_0, w_1) such that $R(x, w_0) = R(x, w_1) = 1$. The challenger responds with $\pi \leftarrow P(x, w_b)$.

Constructions of non-interactive zaps are given in [BOV03] and [GOS06].

Construction 9.3. *Let (P, V) be a non-interactive zap and* Com *a perfectly binding commitment scheme.*

$\mathsf{G}(\lambda, n, t)$ *Sets up a signature scheme for n-bit messages that can be constrained to circuits C of description size at most t. Choose random r, s, and let $c_1 = \mathsf{Com}(0; r)$ and $c_2 = \mathsf{Com}(0^t; s)$. The secret key is $\mathsf{sk} = r$ and the public key is $\mathsf{pk} = (c_1, c_2)$.*

$\mathsf{Sign}(r, m)$ *Compute the proof $\pi_m \leftarrow P(x_m, (r, \perp, \perp))$ where r is a witness for following NP statement x_m:*

$$\exists w_1, w_2, C \text{ such that } c_1 = \mathsf{Com}(0; w_1) \text{ or } (c_2 = \mathsf{Com}(C; w_2) \text{ and } C(m) = 1)$$

$\mathsf{Ver}((c_1, c_2), m, \pi_m)$ *Run the verification procedure $V(x_m, \pi_m)$ where x_m is defined as above.*

$\mathsf{ConstrainGen}(\lambda, t, C)$ *On input a circuit C of size at most t, choose a random r, s and let $c_1 = \mathsf{Com}(1; r)$ and $c_2 = \mathsf{Com}(C; s)$. The secret key is $\mathsf{sk} = s$ and the public key is $\mathsf{pk} = (c_1, c_2)$. To sign a message m such that $C(m) = 1$, run $P(x_m, (\perp, s, C))$. Observe that $(\perp, s, C)$ is a valid witness for the statement x_m.*

Theorem 9.4. *If* Com *is a perfectly binding computational hiding commitment scheme and (P, V) is a non-interactive zap, then* $(\mathsf{G}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{ConstrainGen})$ *is a secure constrained signature scheme*

Proof. Fix a circuit C. We prove security through a sequence of hybrid games.

Game 0. This is the game where the adversary is given a correct public key. That is, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{G}(\lambda, n, t)$, where $\mathsf{pk} = (c_0, c_1) = (\mathsf{Com}(0; r), \mathsf{Com}(0^n; s))$ and $\mathsf{sk} = r$. The adversary is allowed to make signature queries on messages m such that $C(m) = 1$; in response the adversary receives the proof $\pi_m = P(x_m, (r, \perp, \perp))$.

Game 1. This is identical to **Game 0**, except that $c_2 = \mathsf{Com}(C; s)$. Since s is never used after generating c_2, the computational hiding property of Com shows that **Game 0** and **Game 1** are indistinguishable.

Game 2. This is identical to **Game 1**, except that now signature queries are answered using the witness $(\perp, s, C)$. The witness indistinguishability of (P, V) and a simple hybrid across signature queries shows that **Game 1** and **Game 2** are indistinguishable.

Game 3. This is identical to **Game 2**, except that $c_1 = \mathsf{Com}(1; r)$. Since r is never used after generating c_1, the computational hiding property of Com shows that **Game 2** and **Game 3** are indistinguishable. Observe that **Game 3** is exactly the setting where the adversary is given a constrained public key for the circuit C. Thus, the constrained public key is indistinguishable from an unconstrained public key.

It remains to show that given a constrained key for a circuit C, there are no valid signatures relative to messages m where $C(m) = 0$. Indeed, such a signature exists, it implies that the statement x_m is true. Since Com is perfectly binding, $c_1 = \mathsf{Com}(1)$ and $c_2 = \mathsf{Com}(C)$ and $C(m) = 0$, there is no witness (w_1, w_2, C') such that $c_1 = \mathsf{Com}(0; w_1)$ or $(c_2 = \mathsf{Com}(C'; w_2)$ and $C'(m) = 1)$. Therefore, the statement x_m is false, meaning no signature exists. This completes the proof. □

Construction from Obfuscation. We now give our second construction from indistinguishability obfuscaiton and any one-way function. Our construction is very similar to the signature scheme of Sahai and Waters [SW13].

Construction 9.5. *Let* PRF *be a pseudorandom function,* $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *a pseudorandom generator, and* iO *and indistinguishability obfuscator.*

$\mathsf{G}(\lambda, n, t)$ *Sets up a signature scheme for n-bit messages that can be constrained to circuits C of description size at most t. Choose a random instance of* PRF*. The secret key is* $\mathsf{sk} = \mathsf{PRF}$*. The public key will be the obfuscation of* P_{Ver} *in Figure 18, padded to the appropriate length.*

$\mathsf{Sign}(\mathsf{PRF}, x)$ *Output* $\mathsf{PRF}(x)$*.*

$\mathsf{Ver}(P_{\mathsf{Ver}}, x, \sigma)$ *Output* $P_{\mathsf{Ver}}(x, \sigma)$*.*

$\mathsf{ConstrainGen}(\lambda, t, C)$ *On input a circuit C of size at most t, choose a random instance of* PRF*, and set the secret key to be* $\mathsf{sk} = \mathsf{PRF}$*. The public key will be the obfuscation of* $P_{\mathsf{Ver},C}$ *in Figure 19, padded to the appropriate length.*

```
Inputs: $x, \sigma$
Constants: PRF

  1. Output 1 if $\mathsf{PRG}(\mathsf{PRF}(x)) = \mathsf{PRG}(\sigma)$. Otherwise output 0.
```

Figure 18: The program P_{Ver}.

```
Inputs: $x, \sigma$
Constants: PRF, $C$

  1. If $C(x) = 0$, output 0 and stop.
  2. Output 1 if $\mathsf{PRG}(\mathsf{PRF}(x)) = \mathsf{PRG}(\sigma)$. Otherwise output 0.
```

Figure 19: The program $P_{\mathsf{Ver},C}$.

Theorem 9.6. *For any adversary $\mathcal{A}$ breaking the security of Construction 9.5, there is an adversary $\mathcal{B}$ for* iO*,an adversary $\mathcal{C}$ for* PRF *such that, and an adversary $\mathcal{D}$ for* PRG*:*

$$\mathtt{PRF}^{(adv)}{}_{\mathcal{A}}(\lambda) \leq 2^n(2^{-\lambda} + \mathtt{IO}^{(adv)}{}_{\mathcal{B}}(\lambda) + \mathtt{PRF}^{(adv)}{}_{\mathcal{C}}(\lambda) + \mathtt{PRG}^{(adv)}{}_{\mathcal{D}}(\lambda))$$

Proof. Let $\epsilon_{\mathcal{A}}$ be the advantage of $\mathcal{A}$ in distinguishing an obfuscation of P_{Ver} from an obfuscation of $P_{\mathsf{Ver},C}$, for a circuit C of $\mathcal{A}$'s choice, and where $\mathcal{A}$ can make a polynomial number of signature queries on messages x satisfying $C(x) = 1$. We define security though a sequence of games:

Game 0. In this game, $\mathcal{A}$ is given a correct public key $\mathsf{iO}(P_{\mathsf{Ver}})$.

Game i.0. For each $i \in [0, 2^n - 1]$, we define **Game i.0** as follows: when the challenger receives a circuit C, it constructs a circuit C_i where $C_i(x) = C(x)$ for $x \leq i$ (treating x as an integer in $[0, 2^n - 1]$), and $C_i(x) = 1$ for $x > i$. It then runs $\mathsf{ConstrainGen}$ on C_i and gives the resulting public key pk_{C_i} to the adversary. Notice that **Game 0.0** is identical to **Game 0**.

Game i.1. In this game, if $C(i+1) = 1$, the challenger behaves exactly as in **Game i.0**. However, if $C(i+1) = 0$, the challenger computes the constrained PRF $\mathsf{PRF}^{\overline{\{i+1\}}}$ as well as $y = \mathsf{PRG}(\mathsf{PRF}(i+1))$. Then the challenger gives an obfuscation of the program $P^{(1)}_{\mathsf{Ver},C_i}$ in Figure 20 to the adversary. Notice that, P_{Ver,C_i} and $P^{(1)}_{\mathsf{Ver},C_i}$ have the same functionality.

Inputs: x, σ
Constants: $\mathsf{PRF}^{\overline{\{i+1\}}}, C_i, y, i$

1. If $C_i(x) = 0$, output 0 and stop.
2. If $x = i + 1$: Output 1 if $y = \mathsf{PRG}(\sigma)$, and otherwise output 0.
3. If $x \neq i + 1$: Output 1 if $\mathsf{PRG}(\mathsf{PRF}(x)) = \mathsf{PRG}(\sigma)$ and otherwise output 0.

Figure 20: The program $P^{(1)}_{\mathsf{Ver},C_i}$.

Game i.2. In this game, if $C(i + 1) = 1$, the challenger still behaves exactly as in **Game i.0**. However, if $C(i + 1) = 0$, the challenger behaves as in **Game i.1**, except that it sets $y = \mathsf{PRG}(s)$ for a random seed $s \in \{0, 1\}^\lambda$. Notice that the obfuscated programs in **Game i.2** and **Game i.1** can both be constructed knowing only the constrained function $\mathsf{PRF}^{\overline{\{i+1\}}}$, as can all answers to signing queries asked by $\mathcal{A}$. Therefore, if $\mathcal{A}$ can distinguish **Game i.2** form **Game i.1**, then we can construct an adversary $\mathcal{C}_i$ for PRF with the same advantage as $\mathcal{A}$.

Game i.3. In this game, if $C(i + 1) = 1$, the challenger still behaves exactly as in **Game i.0**. However, if $C(i + 1) = 0$, the challenger behaves as in **Game i.2**, except that it chooses y uniformly at random from $\{0, 1\}^{2\lambda}$. If $\mathcal{A}$ can detect this change, then we can construct a adversary $\mathcal{D}_i$ for PRG. Now y is only in the range of PRG with probability at most $1/2^\lambda$. Notice that if $C(i + 1) = 0$ and y is not in the range of PRG, the programs $P^{(1)}_{\mathsf{Ver},C_i}$ and $P_{\mathsf{Ver},C_{i+1}}$ have identical functionality. Similarly, if $C(i + 1) = 1$, then the programs P_{Ver,C_i} and $P_{\mathsf{Ver},C_{i+1}}$ have the same functionality. Hence, for both $C(i + 1) = 0$ and $C(i + 1) = 1$, the obfuscated programs in **Game i.3** and **Game $(i + 1)$.0** compute the same functionality. Hence, so do the programs in **Game i.3** and **Game $(i + 1)$.1**. Therefore, if $\mathcal{A}$ can distinguish these two with probability ϵ, we can construct an adversary $\mathcal{B}_i$ that breaks the security of iO with probability $\epsilon - 1/2^\lambda$.

Finally, notice that **Game 2^n.0** is identical to the case where the adversary is given the output of $\mathsf{ConstrainGen}$ on C, where n is the number of bits of input to C. Combining all the $\mathcal{B}_i$, $\mathcal{C}_i$, and $\mathcal{D}_i$ adversaries into single adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}$ where i is chosen at random, we get the claim of the theorem. □

10 Conclusion and Open Problems

We give the first construction of multiparty key exchange requiring no trusted setup and prove security in the static and semi-static models. Building on these ideas we get the first *distributed* broadcast encryption scheme, that is, every party generates its own secret key. We also construct a short-ciphertext recipient-private broadcast encryption scheme from which we obtain traitor tracing with very short parameters. Our constructions extend to give identity-based broadcast encryption. Along the way we develop new techniques for proving security using indistinguishability obfuscation.

We leave several open problems. We obtain adaptive security for our broadcast encryption and recipient private schemes using the conversion of Gentry and Waters [GW09]. No such conversion is known for multiparty key exchange, and achieving adaptive security for key exchange from iO is an interesting open problem. While our traitor tracing system has short public keys, our general recipient private broadcast system has long public keys, and we leave open the problem of reducing the public key size of our construction.

Acknowledgments

We thank Jonathan Ullman for his comments on the connection to differential privacy. We thank Brent Waters for suggesting adding capabilities to existing systems such as RSA, and for comments on the definitions of security for key exchange protocols. 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.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In *Proc. of Crypto*, 2003.

[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.

[BWZ14] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In *Proceedings of CRYPTO*, 2014.

[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.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *FOCS*, pages 283 – 293, 2000.

[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.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 467–476, New York, NY, USA, 2013. ACM.

[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.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *Proc. of Eurocrypt*, 2006.

[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.

[KRW13] Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. Cryptology ePrint Archive, Report 2013/683, 2013. http://eprint.iacr.org/.

[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 $\mathrm{n}^{\{2+\mathrm{o}(1)\}}$ 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.