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

Dan Boneh and 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

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 $\mathsf{PRF}(x_1, x_2, \ldots, x_N)$

Fig. 1: The program P_{KE}.

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

In the full version [BZ14], 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 the full version [BZ14] 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 indpendent 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, 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 the full version [BZ14], 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 by Gentry and Waters [GW09]). 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 a 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 $O(\lambda + N)$, proportional to the *sum* of the security parameter and the number of users. This is the best possible broadcast size. If one is allowed to leak the size k of the recipient set (and nothing else) then we construct a system where the broadcast size is proportional to $O(\lambda + k \log N)$, which is again the best possible. Building such systems has been open for some time [BBW06] and is now resolved using iO.

Our approach to building a recipient-private broadcast system is to embed an encryption of the intended recipient set in the broadcast header. We publish an obfuscated program in the public key that begins by decrypting the encrypted recipient set in the broadcast header. It then decrypts the message body only if the recipient can provide a proof that it is one of the intended recipients. Interestingly, encrypting the recipient set in a way that lets us prove security using iO is non-trivial. The problem is that using a generic CPA-secure scheme is insecure due to potential malleability attacks on the encrypted recipient set that can break recipient privacy. Using an authenticated encryption scheme to prevent the malleability attack 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 arises because 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

In this section, we briefly discuss notation and the building blocks for our constructions: indistinguishability obfuscation and constrained pseudorandom functions. A more complete description of these primitives appears in the full version [BZ14].

Notation We let $[N] = \{1, \cdots, N\}$ denote the positive integers from 1 to N. 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.

Indistinguishability Obfuscation. An indistinguishability obfuscatior iO is a probabilistic polynomial time algorithm that takes a circuit C and produces an obfuscated circuit $C' = \mathsf{iO}(C)$. We require that $C'(x) = C(x)$ for all inputs x. For security, we require that, for any two circuits C_1 and C_2 that agree on all inputs, no probabilistic polynomial time adversary can distinguish the obfuscation $C'_1 = \mathsf{iO}(C_1)$ from $C'_2 = \mathsf{iO}(C_2)$. The first candidate construction of such obfuscators is due to Garg et al. [GGH+13b].

Constrained Pseudorandom Functions. A constrained pseudorandom function (PRF) [BW13,KPTZ13,BGI13] PRF for a class of subsets $\mathcal{S}$ is a pseudorandom function for which there is an efficient algorithm that takes the secret key k for PRF and a set $S \in \mathcal{S}$, and outputs a circuit PRF_k^S which satisfies

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

For security, we require that the circuit PRF_k^S reveals no information about $\mathsf{PRF}_k(x)$ for points $x \notin S$. From this point forward, we will omit reference to the secret key k. We are interested in several classes of subsets $\mathcal{S}$. A *punctured* PRF is a constrained PRF for all sets whose complements are polynomial in size. The PRF construction of Goldreich, Goldwasser, and Micali [GGM86] satisfies this notion. An *inverval* constrained PRF allows sets of the form $\{1, \ldots, \ell\}$ and $\{\ell', \ldots, D\}$ where the domain is $\{1, \ldots, D\}$. The construction of [GGM86] also satisfies this stronger notion. We also consider constrained PRFs for circuit predicates, where $\mathcal{S}$ consists of all sets accepted by polynomial-sized circuits. Boneh and Waters [BW13] show how to build a constrained PRF for circuit predicates using multilinear maps. In the full version [BZ14], we show that such PRFs can also be built from indistinguishability obfuscation and any punctured PRF.

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 the full version [BZ14] how 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 the full version [BZ14]

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. To setup the NIKE protocol for N users, run a procedure $\mathsf{Setup}(\lambda, N)$, which outputs public parameters params. Then each party $i \in [N]$ runs a publish algorithm $\mathsf{Publish}(\mathsf{params}, i)$, which 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. Finally, each user runs a key generation algorithm $\mathsf{KeyGen}(\mathsf{params}, i, \mathsf{sk}_i, \{\mathsf{pv}_j\}_{j\in[N]})$ using their secret and all other user's public values, which outputs a shared key k.

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

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

For security, here we only consider a static notion of security. In the full version [BZ14], we also consider a stronger semi-static security notion. Fix a bit b, and consider the following experiment. The challenger runs $\mathsf{params} \leftarrow \mathsf{Setup}(\lambda, N)$. For $i \in [N]$, the challenger also runs $(\mathsf{sk}_i, \mathsf{pv}_i) \leftarrow \mathsf{Publish}(\mathsf{params}, i)$. Set $k_0 = \mathsf{KeyGen}(\mathsf{params}, 1, \mathsf{sk}_i, \{\mathsf{pv}_j\}_{j\in[N]})$ and $k_1 \leftarrow \mathcal{K}$. Give the adversary $\{\mathsf{pv}_j\}_{j\in[N]}, k_b$. For $b = 0, 1$ let W_b be the event that $b' = 1$ and define $\mathtt{AdvKE}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 1. *A multiparty key exchange protocol* $(\mathsf{Setup}, \mathsf{Publish}, \mathsf{KeyGen})$ *is statically secure if, for any polynomials N, and any PPT adversary $\mathcal{A}$, the function* $\mathtt{AdvKE}(\lambda)$ *is negligible.*

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 1 *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.*

– $\mathsf{Setup}(\lambda, G, N)$: *Sets up the key exchange protocol supporting at most N users and allowing any group of at most G users to compute a shared secret key. 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 $x_i = \mathsf{PRG}(s_i)$*
– $\mathsf{KeyGen}(P_{\mathsf{iO}}, x_0, i, s_i, S, \{x_i\}_{i \in S})$: *Abort if $|S| > G$ or $i \notin S$. Let $S(j)$ denote the jth 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 $k = \mathsf{PRF}(\hat{x}_1, ..., \hat{x}_G)$ 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 $\mathsf{PRF}(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_G)$

Fig. 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 a semi-static notion of security (as defined in the full version [BZ14]). Security is summarized by the following theorem:

Theorem 2. *If* PRG *is a secure pseudorandom generator,* PRF *a secure punctured PRF, and* iO *a secure indistinguishability obfuscator, then Construction 1 is a statically secure NIKE. If, in addition,* PRF *is a secure constrained PRF for circuit predicates, then Construction 1 is semi-statically secure.*

Removing trusted setup. Before proving Theorem 2, 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, Publish does not depend on params. This allows us to remove trusted setup as follows: each user runs both Setup and Publish, and publishes their own public parameters params_i along with x_i. Then

[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.

in KeyGen, choose some params_i in a canonical way (say, corresponding smallest x_i when treated as an integer), and run the original key exchange protocol using $\mathsf{params} = \mathsf{params}_i$. Now there is no setup, and static security follows from the static security of the original scheme. However, in the full version [BZ14] we show that semi-static does not follow. Instead, in the full version we give a modified construction that achieves semi-static security.

The proof of Theorem 2 is given in the full version [BZ14]. Here we sketch the main idea:

Proof sketch. For simplicity, assume $G = N$, though it is easy to generalize to $N > G$. In the static security game, the challenger draws N random seeds s_i^*, and lets $x_i^* = \mathsf{PRG}(s_i^*)$. It also constructs an obfuscation of the program P_{KE} in Figure 2. Then it gives this obfuscation, all of the x_i^*, and a challenge key k^* to the adversary $\mathcal{A}$. $\mathcal{A}$ then outputs its guess for whether $k^* = \mathsf{PRF}(x_1^*, \ldots, x_N^*)$ or not. We first slightly change the game by choosing the x_i^* uniformly at random in $\{0,1\}^{2\lambda}$. The security of PRG shows that this modification at most negligibly changes $\mathcal{A}$'s advantage. Because the image of PRG is so much bigger than its domain, with high probability, none of the x_i^* are in the image of PRG. Thus, we can modify P_{KE} to obtain a new program P'_{KE} that aborts whenever an input x_i equals x_i^* for some i and this does not change the functionality of P_{KE}. Now P'_{KE} never evaluates PRF on the point $(x_1^*, \ldots, x_N^*)$, so we can puncture PRF at that point, and include only the punctured program in P'_{KE}. The indistinguishability of obfuscations shows that these modifications are undetectable by $\mathcal{A}$. We can simulate the view of $\mathcal{A}$ using only the punctured PRF, and $\mathcal{A}$ still succeeds with non-negligible probability. However, $\mathcal{A}$ now distinguishes the correct value of PRF at the puncture point from a truly random value, violating the security of PRF. For semi-static security, we need to puncture PRF at all points corresponding to the various subsets the adversary may challenge on, which we do using a PRF for circuit predicates. □

4 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 system. Informally, a recipient private broadcast system allows the broadcaster to broadcast a message to a subset of N users. For security, we require that any user outside of the recipient set cannot learn the message, and each user only learns one bit of information about the recipient set: whether or not they are in it. We give a formal definition in the full version [BZ14]. Private linear broadcast encryption is recipient broadcast encryption where the only recipient sets allowed are $\emptyset = [0], [1], \ldots, [N]$.

To setup a PLBE scheme for N users, run a setup procedure $\mathsf{Setup}(\lambda, N)$, which outputs public parameters params and user secret keys $\{\mathsf{sk}_i\}_{i\in[N]}$ for each user. Distribute sk_i to user i. To encrypt to a set $[j]$, run an encryption algorithm $\mathsf{Enc}(\mathsf{params}, j)$ to obtain a header Hdr and message encryption key k. Use k to encrypt the message, and broadcast Hdr along with the resulting ciphertext. A user $i \leq j$ decrypts by running $\mathsf{Dec}(\mathsf{params}, \mathsf{sk}_i, \mathsf{Hdr})$, which outputs the message encryption key that can be used to actually decrypt the ciphertext.

For correctness, we require that any encryption to a set $[i]$ can be decrypted by any user in $[j]$. In other words, if $(\mathsf{Hdr}, k) \leftarrow \mathsf{Enc}(\mathsf{params}, j)$, then $\mathsf{Dec}(\mathsf{params}, \mathsf{sk}_i, \mathsf{Hdr}) = k$ for $i \leq j$.

For security, we have two experiments: semantic security and recipient privacy. For semantic security, fix a bit b and consider the following experiment. The adversary commits to a set $[j]$. The challenger then runs $\mathsf{params}, \{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{G}(\lambda, N)$, and then gives params as well as the secret keys $\{\mathsf{sk}_i\}_{i>j}$ for users not in $[j]$ to the adversary. The challenger also runs $(\mathsf{Hdr}, k_0) \leftarrow \mathsf{Enc}(\mathsf{params}, j)$, and generates $k_1 \leftarrow \mathcal{K}$, and gives Hdr, k_b. The adversary outputs a guess b' for b. For $b = 0, 1$ let W_b be the event that $b' = 1$ and $\mathtt{PLBE}_{\mathtt{SS}}^{(\mathrm{adv})}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

For recipient privacy, also fix a bit b. The adversary commits to a user i^*. The challenger runs $\mathsf{params}, \{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{G}(\lambda, N)$, and gives params as well as the secret keys $\{\mathsf{sk}_i\}_{i\neq i^*}$ for all users except i^* to the adversary. The challenger also runs $(\mathsf{Hdr}, k) \leftarrow \mathsf{Enc}(\mathsf{params}, i^* - b)$ and gives Hdr to the adversary. The adversary outputs a guess b' for b. For $b = 0, 1$ let W_b be the event that $b' = 1$ and $\mathtt{PLBE}_{\mathtt{RP}}^{(\mathrm{adv})}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 2. *A private linear broadcast (PLBE) scheme* $(\mathsf{Setup}, \mathsf{Publish}, \mathsf{KeyGen})$ *is secure if, for any polynomials N, and any PPT adversary $\mathcal{A}$, the functions* $\mathtt{PLBE}_{\mathtt{SS}}^{(adv)}(\lambda)$ *and* $\mathtt{PLBE}_{\mathtt{RP}}^{(adv)}(\lambda)$ *are negligible.*

4.1 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 now 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. In the full version [BZ14], we show how to make the system public-key. We discuss extending this construction to other set systems at the end of the section.

Construction 3 *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* $\mathsf{PRF}_{key} : \{0,1\}^{2\lambda} \times \{0,\ldots,N\} \to \{0,1\}^{\lambda}$ *be punctured PRFs and* $\mathsf{PRF}_{mac} : \{0,1\}^{2\lambda} \times \{0,\ldots,N\} \to \{0,1\}^{\lambda}$ *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 3, 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* $k \leftarrow P_{\mathsf{Dec}}(r, c, s_i, i)$.

A public-key system. As described, our scheme requires a secret broadcast key in order to encrypt. However, using the trick of Sahai and Waters [SW13], we show

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 $i \leq j$. If check fails, output $\perp$ and stop
5. Otherwise, output $\mathsf{PRF}_{key}(r, c_1)$

Fig. 3: The program $P_{TT-\mathsf{Dec}}$.

in the full version [BZ14] how to include in the public parameters an obfuscated program that allows anyone to encrypt.

In our public key scheme, 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 4. *If* PRF_{enc} *and* PRF_{key} *are secure punctured 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 3 is an adaptively secure private linear broadcast encryption scheme.*

The proof is given in the full version [BZ14]. Here we sketch the main ideas:

Proof sketch. We must prove that our scheme is both semantically secure and has recipient privacy. For semantic security, the adversary $\mathcal{A}$ commits to a set $[j^*]$, and receives the secret keys s_i for all $i > j^*$. $\mathcal{A}$ also receives the obfuscation of $P_{TT-\mathsf{Dec}}$, as well as a challenge (r^*, c_1^*, c_2^*) that is an encryption to the set $[j^*]$, and a key k^*. $\mathcal{A}$ must distinguish the correct k^* from random. Our first step is to modify $P_{TT-\mathsf{Dec}}$ by puncturing PRF_{sk} at each key that $\mathcal{A}$ does not receive, and hard-code the values $x_i = \mathsf{PRG}(\mathsf{PRF}_{sk}(i))$ into $P_{TT-\mathsf{Dec}}$ so that is correctly decrypts all valid ciphertexts. This does not change the functionality of $P_{TT-\mathsf{Dec}}$. We then replace these x_i with random values in $\{0,1\}^{2\lambda}$, and the security of PRF_{sk} and PRG shows that this change is undetectable by $\mathcal{A}$. Now, with overwhelming probability, none of the x_i for $i \leq j^*$ are in the image of PRG, so we can modify $P_{TT-\mathsf{Dec}}$ to abort if $i \leq j^*$. On the challenge ciphertext, $j = j^*$, the $P_{TT-\mathsf{Dec}}$ will also abort if $i > j^*$, meaning $P_{TT-\mathsf{Dec}}$ will always abort. Therefore, we can puncture PRF_{key} at (r^*, c_1^*) without modifying the functionality of $P_{TT-\mathsf{Dec}}$. The indistinguishability of obfuscations shows that these changes are undetectable. However, $\mathcal{A}$ now distinguishes the correct value of PRF_{key} ad (r^*, c_1^*) from random, violating the security of PRF_{key}. One problem with the above proof is that hard-coding all x_i into $P_{TT-\mathsf{Dec}}$ expands its size considerably. We show in the full version [BZ14] how to puncture PRF_{sk}, one user at a time, using a sequence of hybrids while keeping $P_{TT-\mathsf{Dec}}$ small.

For recipient privacy, the proof is more complicated but similar. Here, $\mathcal{A}$ commits to a j^*, and receives all secret keys except those for user j^* and and encryption to the set $[j^* - b]$ for some $b \in \{0,1\}$. $\mathcal{A}$ must determine b. Similar to the semantic security case, we puncture PRF_{sk} at j^* and replace $x_{j^*} = \mathsf{PRG}(\mathsf{PRF}_{sk}(j^*))$ with a truly random value in $\{0,1\}^{2\lambda}$, and then modify $P_{TT-\mathsf{Dec}}$ so that it aborts if $i = j^*$. Now we puncture PRF_{mac} at all points (r^*, c_1), and hard-code $y_{c_1} = \mathsf{PRG}(\mathsf{PRF}_{mac}(r^*, c_1))$ into $P_{TT-\mathsf{Dec}}$. For each $c_1 \neq c_1^*$, we replace y_{c_1} with a truly random value in $\{0,1\}^{2\lambda}$. Similar to the semantic security proof, we have to puncture iteratively in order to keep the program size small. At this point, the only (r^*, c_1, c_2) that authenticates is the challenge ciphertext itself. This means we can puncture PRF_{enc} at r^*, and hard-code the necessary values to decrypt the challenge ciphertext, including $z^* = \mathsf{PRF}_{enc}(r^*)$. The security of PRF_{enc} shows that we can replace z^* with a truly random value. At this point, $c_1^* = j^* - b + z^*$. As we show in the full version [BZ14], in the $b = 0$ case, we can replace z^* with $z^* - 1$ without changing the functionality of $P_{TT-\mathsf{Dec}}$. However, moving to $z^* - 1$ also moves us to the $b = 1$ case, meaning that $\mathcal{A}$ actually breaks the indistinguishability of obfuscations. □

4.2 Extension to Other Set Systems

Construction 3 easily extends the other classes of recipient sets — for example, the set of all subsets of $[N]$, or the subsets of size exactly r. Ciphertexts will simply be an encryption of (the description of) the recipient set, and the obfuscated program will output the PRF applied to the ciphertext only if the user can supply a valid seed for one of the users in the set. However, now the number of possible recipient sets is exponential, and consequently our security reduction becomes non-polynomial. In the full version [BZ14], we give a different construction that has a polynomial security proof for these classes of recipient sets. However, the public key size becomes $N^{O(1)}$.

Acknowledgments. This work is supported by NSF, DARPA, IARPA, and others, as listed in the full version.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

BZ14. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Proceedings of CRYPTO*, 2014. Full version available at the Cryptology ePrint Archives http://eprint.iacr.org/2013/642.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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