Multi-party Key Exchange for Unbounded Parties from Indistinguishability Obfuscation

Dakshita Khurana[1](✉), Vanishree Rao[2], and Amit Sahai[1]

[1] Department of Computer Science, Center for Encrypted Functionalities, UCLA, Los Angeles, CA, USA
{dakshita,sahai}@cs.ucla.edu
[2] PARC, a Xerox Company, Palo Alto, CA, USA
Vanishree.Rao@parc.com

Abstract. Existing protocols for non-interactive multi-party key exchange either (1) support a bounded number of users, (2) require a trusted setup, or (3) rely on knowledge-type assumptions.

We construct the first non-interactive key exchange protocols which support an unbounded number of parties and have a security proof that does not rely on knowledge assumptions. Our non-interactive key-exchange protocol does not require a trusted setup and extends easily to the identity-based setting. Our protocols suffer only a polynomial loss to the underlying hardness assumptions.

1 Introduction

Non-interactive key exchange (NIKE) enables a group of parties to derive a shared secret key without any interaction. In a NIKE protocol, all parties simultaneously broadcast a message to all other parties. After this broadcast phase, each party should be able to locally compute a shared secret key for any group of which he is a member. All members of a group should generate an identical shared key, and the shared key for a group should look random to a non-member.

This notion was introduced by Diffie and Hellman [16], who also gave a protocol for non-interactive key exchange in the two-party setting. More than two decades later, Joux [32] constructed the first non-interactive key exchange protocol for three parties. Given a set of N parties (where N is a polynomial in

D. Khurana and A. Sahai—Research supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.
V. Rao—Work done while studying at UCLA.

T. Iwata and J.H. Cheon (Eds.): ASIACRYPT 2015, Part I, LNCS 9452, pp. 52–75, 2015.
DOI: 10.1007/978-3-662-48797-6_3

the security parameter), Boneh and Silverberg [4] obtained a multiparty NIKE protocol based on multilinear maps. The recent candidates for multilinear maps given by [14,15,22,27] can be used to instantiate the scheme of [4], assuming a trusted setup. After the advent of candidate constructions for indistinguishability obfuscation (iO) starting with the result of Garg et. al. [23], Boneh and Zhandry [6] demonstrated how to obtain static secure NIKE and ID-NIKE based on indistinguishability obfuscation, without relying on a trusted setup.

However, all these constructions require an a-priori bound on the number of parties. The only known protocols which can possibly handle an unbounded number of parties are the ones by Ananth et. al. and Abusalah et. al. [1,2], but their solutions rely on *differing-inputs obfuscation* (diO)[2,3,7]. Unfortunately, diO is a knowledge-type assumption, and recent work [9,25,31] demonstrates that it may suffer from implausibility results. In this paper, we address the following question:

Can we obtain NIKE supporting an a-priori unbounded number of parties and not requiring any setup, based on indistinguishability obfuscation?

We give a positive answer to this question, by demonstrating non-interactive key exchange protocols that achieve static security and support an a-priori unbounded number of parties, based on indistinguishability obfuscation.

1.1 Our Contributions

We consider a setting where an a-priori unbounded number of parties may broadcast or publish messages, such that later any party can derive a shared secret key for a group of which it is a member. In our setting, parameters do not grow with the number of parties. Our results can be summarized as follows.

Theorem 1. *Assuming indistinguishability obfuscation and fully homomorphic encryption, it is possible to obtain static-secure non-interactive multi-party key exchange for an a-priori unbounded number of parties, without any setup.*

Theorem 2. *Assuming indistinguishability obfuscation and fully homomorphic encryption, it is possible to obtain static-secure identity-based non-interactive multi-party key exchange for an a-priori unbounded number of parties.*

Fully homomorphic encryption was first constructed by Gentry [26] and subsequently Brakerski and Vaikuntanathan [10] constructed it under the learning with errors assumption. Alternatively, it can be constructed based on sub-exponentially secure iO for circuits and sub-exponential one-way functions [12].

1.2 Technical Overview

Bottlenecks in known constructions. Our starting point the static-secure NIKE protocol of Boneh and Zhandry [6] based on indistinguishability obfuscation, in the simplest case where parties have access to a trusted setup. The adversary fixes a set of parties to corrupt, independent of the system parameters. Then the setup generates public parameters, and each party broadcasts its public values. We require that the shared group key for all the honest parties should look indistinguishable from random, from the point of view of the adversary.

The basic Boneh-Zhandry construction uses a trusted setup to generate an obfuscated program with a secret PRF key. Parties pick a secret value uniformly at random, and publish the output of a length-doubling PRG applied to this value, as their public value. To derive the shared secret key for a group of users, a member of the group inputs public values of all users in the group (including himself) according to some fixed ordering, as well as his own secret value into this obfuscated program. The program checks if the PRG applied to the secret value corresponds to one of the public values in the group, and if the check passes, outputs a shared secret key by applying a PRF to the public value of all parties in the group.

To prove security, we note that the adversary never corrupts any party in the challenge set, so we never need to reveal the secret value for any party in this set. Thus, (by security of the PRG) we can set the public values of parties in the challenge set, to uniformly random values in the co-domain of the PRG. Then with overwhelming probability, there exists no pre-image for any of the public values in the challenge set, and therefore there exists no secret value for which the PRG check in the program would go through. This allows us to puncture the program at the challenge set, and then replace the shared secret key for this set with random. However, in this construction it is necessary to set an a-priori bound on the number of participants, since the size of inputs to the setup circuit must be bounded.

The construction of Ananth et. al. [2] works without this a-priori bound, by making use of differing-inputs obfuscation and collision resistant hashing. To obtain a shared group key, parties first hash down the group public values and then generate a short proof of membership in the group. The program takes the hashed value and proof, and outputs a shared secret key for the group if and only if the proof verifies. However, because the hash is compressing, there exist collisions and thus there exist false proofs of membership. The only guarantee is that these proofs are 'hard' to find in a computational sense. Unfortunately, proving security in such a situation requires the use of differing-inputs obfuscation, since we must argue that any non-member which distinguishes from random the shared key of a group, could actually have generated false proofs. Such an argument inherently involves an extractable assumption.

First attempt. Now, it seems that we may benefit from using an iO-friendly tool for hashing, such that the proof of membership is unconditionally sound for a select piece of the input to the hash. Moreover, the description of the hash should computationally hide which part of the input it is sound for. Then, like in the previous proof, we can set the public values of parties in the challenge set to uniformly random values in the co-domain of the PRG, and try to puncture the program at each position one by one. This resembles the "selective enforcement" techniques of Koppula et. al. [34] who construct accumulators (from iO) as objects enabling bounded commitments to an unbounded storage, which are unconditionally binding for a select piece of the storage.

At this point, it may seem that we can use the accumulator hash function and we should be done. Parties can hash down an unbounded number of public values to a bounded size input using the accumulator, and generate a short proof of correctness. On input the hashed value and proof, the program can be set to output a shared key if the proof verifies. To prove security, we could begin by generating the public values for parties in the challenge set, uniformly at random in the co-domain of the PRG. Then, it should be possible to make the accumulator binding at the first index, and consequently puncture out the first index from the program. Then we can continue across indices and finally puncture out the hash value at the challenge set for all indices.

Even though this proof seems straightforward, the afore-mentioned approach fails. This is because an accumulator can be made unconditionally sound at a particular index, *only conditioned on the previous indices being equal to an a-priori fixed sequence.* In the setting of obfuscating unbounded-storage Turing Machines, for which such accumulators were first introduced [34], there was indeed a well-defined "correct" path that was generated by the machine itself, and consequently there was a way to enforce correct behaviour on all previous indices. However, in our setting the adversary is allowed to hash completely arbitrary values for all indices. Very roughly, we require a tool that enables selective enforcing even when the adversary is allowed to behave arbitrarily on all other indices.

Our solution. At this point, we require a hash function which can enforce soundness at hidden indices, while allowing arbitrary behaviour on all other indices. Such a hash function was introduced recently in the beautiful work of Hubacek and Wichs [30], in the context of studying the communication complexity of secure function evaluation with long outputs. They call it a *somewhere statistically binding* (SSB) hash and give a construction based on fully homomorphic encryption. An SSB hash can be used like other hash functions, to hash an a-priori unbounded number of chunks of input, each of bounded size, onto a bounded space. Moreover, this hash can operate in various *modes*, where each mode is statistically binding on some fixed index i determined at setup; yet, the description of the hash function computationally hides this index.

Equipped with this tool, it is possible to argue security via a selective enforcing hybrid argument. As usual, we begin by generating the public values of all parties in the challenge set, uniformly at random in the co-domain of the PRG. With overwhelming probability, this ensures that there exist no secret values that could generate the public values in the challenge set. Now, we zoom into each index (of the hash) one by one, and make the hash function statistically binding at that particular index. Specifically, we know that a specific output of the hash h^* can only be achieved via a single fixed public value at the enforcing index (say i). Moreover, with overwhelming probability, this public value lies outside the range of the PRG, and thus there exist no false proofs for hash value h^* at the enforcing index i.

This allows us to alter the obfuscated circuit to always ignore the value h^* at index i. Once we have changed the program, we generate the hash function to be statistically binding at index $(i+1)$, and repeat the argument. Note that once we are at this next index, there may exist false proofs for previous indices – however, at this point, we have already programmed the obfuscation to eliminate the value h^* for all previous indices.

In the identity-based NIKE setting, we generate secret keys for identities as PRF outputs on the identity (in a manner similar to [6]). In addition to using the enforce-and-move technique detailed above, we need to avoid simultaneously programming in an unbounded number of public values. We handle this using Sahai-Waters [37] punctured programming techniques(using PRGs) to puncture and then *un-puncture* the PRF keys before moving on to the next value.

1.3 Other Related Work

Cash, Kiltz, and Shoup [13] and Freire, Hofheinz, Kiltz, and Paterson [19] formalized various security models in the two-party NIKE setting. Bones and Zhandry [6] first resolved NIKE for bounded $N > 3$ parties without relying on a trusted setup, assuming indistinguishability obfuscation. However, their security proofs worked only for the static and semi-static scenarios. Hofheinz et. al. [29] realized adaptive secure bounded N-party NIKE in the random oracle model without setup, and Rao [36] realized bounded N-party NIKE with adaptive security and without setup based on assumptions over multilinear maps. A recent independent work of Yamakawa et. al. [39] gives multilinear maps where the multilinearity levels need not be bounded during setup, and the size of the representations of elements is independent of the level of multi-linearity. In the same paper, these maps are used to construct multiparty NIKE with unbounded parties, however, requiring a trusted setup. Furthermore, their scheme does not seem to extend directly to the identity-based NIKE setting.

In the identity-based NIKE setting, there is a trusted master party that generates secret values for identities using a master secret key. This (seemingly weaker) setting has been extensively studied both in the standard and the random oracle models, and under various static and adaptive notions of security [18,20,35,38], but again for an a-priori bounded number of parties.

Zhandry [40] uses somewhere statistically binding hash along with obfuscation to obtain adaptively secure broadcast encryption with small parameters.

2 Preliminaries

2.1 Indistinguishability Obfuscation and PRFs

Definition 1 (Indistinguishability Obfuscator *(iO)).* *A uniform PPT machine* iO *is called an indistinguishability obfuscator for circuits if the following conditions are satisfied:*

- *For all security parameters $\kappa \in \mathbb{N}$, for all circuits C, for all inputs x, we have that*
$$\Pr[C'(x) = C(x) : C' \leftarrow \mathsf{iO}(\kappa, C)] = 1$$
- *For any (not necessarily uniform) PPT adversaries Samp, D, there exists a negligible function α such that the following holds: if $\Pr[|C_0| = |C_1|$ and $\forall x, C_0(x) = C_1(x) : (C_0, C_1, \sigma) \leftarrow Samp(1^\kappa)] > 1 - \alpha(\kappa)$, then we have:*
$$\Big| \Pr\left[D(\sigma, \mathsf{iO}(\kappa, C_0)) = 1 : (C_0, C_1, \sigma) \leftarrow Samp(1^\kappa)\right]$$
$$- \Pr\left[D(\sigma, \mathsf{iO}(\kappa, C_1)) = 1 : (C_0, C_1, \sigma) \leftarrow Samp(1^\kappa)\right]\Big| \leq \alpha(\kappa)$$

Such indistinguishability obfuscators for circuits were constructed under novel algebraic hardness assumptions in [24].

Definition 2. *A puncturable family of PRFs F is given by a triple of Turing Machines Key_F, $\mathrm{Puncture}_F$, and Eval_F, and a pair of computable functions $n(\cdot)$ and $m(\cdot)$, satisfying the following conditions:*

- ***[Functionality preserved under puncturing]*** *For every PPT adversary A such that $A(1^\kappa)$ outputs a set $S \subseteq \{0,1\}^{n(\kappa)}$, then for all $x \in \{0,1\}^{n(\kappa)}$ where $x \notin S$, we have that:*
$$\Pr\left[\mathrm{Eval}_F(K, x) = \mathrm{Eval}_F(K_S, x) : K \leftarrow \mathrm{Key}_F(1^\kappa), K_S = \mathrm{Puncture}_F(K, S)\right] = 1$$
- ***[Pseudorandom at punctured points]*** *For every PPT adversary (A_1, A_2) such that $A_1(1^\kappa)$ outputs a set $S \subseteq \{0,1\}^{n(\kappa)}$ and state σ, consider an experiment where $K \leftarrow \mathrm{Key}_F(1^\kappa)$ and $K_S = \mathrm{Puncture}_F(K, S)$. Then we have*
$$\Big| \Pr\left[A_2(\sigma, K_S, S, \mathrm{Eval}_F(K, S)) = 1\right] - \Pr\left[A_2(\sigma, K_S, S, U_{m(\kappa)\cdot|S|}) = 1\right]\Big| = negl(\kappa)$$
where $\mathrm{Eval}_F(K, S)$ denotes the concatenation of $\mathrm{Eval}_F(K, x_1)), \ldots, \mathrm{Eval}_F(K, x_k))$ where $S = \{x_1, \ldots, x_k\}$ is the enumeration of the elements of S in lexicographic order, $negl(\cdot)$ is a negligible function, and U_ℓ denotes the uniform distribution over ℓ bits.

For ease of notation, we write $\mathsf{PRF}(K, x)$ to represent $\mathrm{Eval}_F(K, x)$. We also represent the punctured key $\mathrm{Puncture}_F(K, S)$ by $K\{S\}$.

The GGM tree-based construction of PRFs [28] from one-way functions are easily seen to yield puncturable PRFs, as recently observed by [5,8,33]. Thus,

Imported Theorem 1. *[5,8,28,33] If one-way functions exist, then for all efficiently computable functions $n(\kappa)$ and $m(\kappa)$, there exists a puncturable PRF family that maps $n(\kappa)$ bits to $m(\kappa)$ bits.*

2.2 Somewhere Statistically Binding Hash

Definition 3. *We use the primitive somewhere statistically binding hash (SSB hash), constructed by Hubacek and Wichs [30]. Intuitively, these are a special type of collision resistant hash function that is binding on a hidden index, and can be used with indistinguishability obfuscation. An SSB hash is a tripe of algorithms* $(\mathsf{Gen}, \mathsf{Open}, \mathsf{Ver})$ *where:*

- $\mathsf{Gen}(s, i)$ *takes as input two integers* s *and* i*, where* $s \leq 2^\kappa$ *denotes the number of blocks that will be hashed, and* $i \in [s]$ *indexes a particular block. The output is a function* $H : \Sigma^s \to \mathcal{Z}$*. The size of the description of* H *is independent of* s *and* i *(though it will depend on the security parameter).*
- $\mathsf{Open}(H, x = \{x_\ell\}_{\ell \in [s]}, j)$ *for* $x_\ell \in \Sigma$ *and* $j \in [s]$ *produces an "opening"* π *that proves that the* j^{th} *element in* x *is* x_j*.*
- $\mathsf{Ver}(H, h \in \mathcal{Z}, j \in [s], u \in \Sigma, \pi)$ *either accepts or rejects. The idea is that Ver should only accept when* $h = H(x)$ *where* $x_j = u$*.*
- *Correctness:* $\mathsf{Ver}(H, H(x), j, x_j, \mathsf{Open}(H, x, j))$ *accepts.*
- *Index hiding:* $\mathsf{Gen}(s, i_0)$ *is computationally indistinguishable from* $\mathsf{Gen}(s, i_1)$ *for any* i_0, i_1*.*
- *Somewhere Statistically Binding: If* $H \leftarrow \mathsf{Gen}(s, i)$ *then if* $\mathsf{Ver}(H, h, i, u, \pi)$ *and* $\mathsf{Ver}(H, h, i, u', \pi')$ *accept, it must be that* $u = u'$*.*

Remark. Note that using SSB hash functions, one can efficiently hash down an a-priori unbounded (polynomial) number of values in the security parameter κ.

Imported Theorem 2. *Assuming the existence of FHE, there exists a somewhere statistically binding hash function family mapping unbounded polynomial size inputs to outputs of size* κ *bits (where* κ *denotes the security parameter), according to Definition 3.*

3 Definitions

Definition 4 (Multiparty Non-interactive Key Exchange). *An adaptive multiparty NIKE protocol has the following three algorithms:*

- $\mathsf{Setup}(1^\kappa)$: *The setup algorithm takes a security parameter* κ*, and outputs public parameters* params.
- $\mathsf{Publish}(1^\kappa, i)$: *Each party executes the publishing algorithm, which takes as input the index of the party, and outputs two values: a secret key* sv_i *and a public value* pv_i*. Party* P_i *keeps* sv_i *as his secret value, and publishes* pv_i *to the other parties.*
- $\mathsf{KeyGen}(\mathsf{params}, \mathcal{S}, (pv_i)_{i \in \mathcal{S}}, j, sv_j)$: *To derive the common key* $k_\mathcal{S}$ *for a subset* $\mathcal{S}$*, each party in* $\mathcal{S}$ *runs* KeyGen *with* params*, its secret value* sv_j *and the public values* $(pv_i)i \in \mathcal{S}$ *of the parties in* $\mathcal{S}$*.*

Then, these algorithms should satisfy the following properties:

– *Correctness: For all $\mathcal{S}, i, i' \in \mathcal{S}$,*

$$\mathsf{KeyGen}(\mathsf{params}, \mathcal{S}, (pv_j)_{j\in\mathcal{S}}, i, sv_i) = \mathsf{KeyGen}(\mathsf{params}, \mathcal{S}, (pv_j)_{j\in\mathcal{S}}, i', sv_{i'}).$$

– *Security: The adversary is allowed to (statically) corrupt any subset of users of his choice. More formally, for $b \in \{0,1\}$, we denote by* $\mathsf{expmt}(b)$ *the following experiment, parameterized only by the security parameter κ and an adversary $\mathcal{A}$, and* $\mathsf{params} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$ *and* $b' \leftarrow \mathcal{A}^{\mathsf{Reg}(\cdot),\mathsf{RegCor}(\cdot,\cdot),\mathsf{Ext}(\cdot),\mathsf{Rev}(\cdots),\mathsf{Test}(\cdots)}(1^\kappa, \mathsf{params})$ *where:*
  - $\mathsf{Reg}(i \in [2^\kappa])$ *registers an honest party P_i. It takes an index i, and runs $(sv_i, pv_i) \leftarrow$* $\mathsf{Publish}(\mathsf{params}, i)$. *The challenger then records the tuple $(i, sk_i, pv_i, honest)$ and sends pv_i to A.*
  - $\mathsf{RegCor}(i \in [2^\kappa], pk_i)$ *registers a corrupt party P_i^*. It takes an index i and a public value pv_i. The challenger records $(i, \bot, pv_i, corrupt)$. 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 party. The challenger looks up the tuple $(i, sv_i, pv_i, honest)$ and returns sv_i to $\mathcal{A}$.*
  - $\mathsf{Rev}(\mathcal{S}, i)$ *reveals the shared secret for a group $\mathcal{S}$ of parties, as calculated by the ith party, where $i \in \mathcal{S}$. We require that party P_i was registered as honest. The challenger uses the secret key for party P_i to derive the shared secret key $k_\mathcal{S}$, which it returns to the adversary.*
  - $\mathsf{Test}(\mathcal{S})$: *Takes a set $\mathcal{S}$ of users, all of which were registered as honest.*

  Next, 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. Else if $b = 1$, the challenger generates a random key k to return to the adversary.*

A static adversary $\mathcal{A}$ must have the following restrictions:

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

We require that all register queries and register-corrupt queries are for distinct i, and that $pv_i \neq pv_j$ for any $i \neq j$. For $b = 0$, let W_b be the event that $b' = 1$ in $\mathsf{expmt}(b)$ *and we define* $\mathsf{Adv}_{\mathsf{NIKE}}(\kappa) = |\Pr[W_0] - \Pr[W_1]|$.

Then, a multi-party key exchange protocol $(\mathsf{Setup}, \mathsf{Publish}, \mathsf{KeyGen})$ *is statically secure if* $\mathsf{Adv}_{\mathsf{NIKE}}(\kappa)$ *is negl(κ) for any static PPT adversary $\mathcal{A}$.*

4 Static Secure NIKE for Unbounded Parties

4.1 Construction

Let PRF denote a puncturable PRF mapping κ bits to κ bits, and PRG denote a length-doubling pseudorandom generator with inputs of size κ bits. Let $(\mathsf{Gen}, \mathsf{Open}, \mathsf{Ver})$ denote the algorithms of a somewhere statistically binding hash

```
NIKE- Public Parameters

Constants: $H$, PRF key $K$.
Input: $h, i, pv, sv, \pi, t$.

1. If $i > t$, output $\perp$ and abort.
2. Set $K_t = \mathsf{PRF}(K, t)$.
3. If $\mathsf{Ver}(H, h, i, pv, \pi) = 1$ and $\mathsf{PRG}(sv) = pv$, output $\mathsf{PRF}(K_t, h)$.
4. Else output $\perp$.
```

Fig. 1. Static Secure NIKE Parameters P_{KE}

scheme as per Definition 3. Then the static secure NIKE algorithms are constructed as follows.

$\mathsf{Setup}(1^\kappa)$: Pick puncturable PRF key $K \xleftarrow{\$} \{0,1\}^\kappa$. Run $\mathsf{Gen}(1^\kappa, 2^\kappa, 0)$ to obtain H. Obfuscate using iO the program P_{KE} in Fig. 1, padded to the appropriate length. Output $P_{\mathsf{iO}} = \mathsf{iO}(P_{KE})$ as public parameters.

$\mathsf{Publish}$: Party i chooses a random seed $s_i \in \{0,1\}^\lambda$ as a secret value, and publishes $x_i = \mathsf{PRG}(s_i)$.

$\mathsf{KeyGen}(P_{KE}, i, s_i, S, \{pv_j\}_{j\in S})$: Compute $h = H(\{pv_j\}_{j\in S})$. Compute $\pi = \mathsf{Open}(H, \{pv_j\}_{j\in S}, i)$. Run program P_{KE} on input $(h, i, pv, sv, \pi, |S|)$ to obtain shared key K_S.

4.2 Security Game and Hybrids

Hybrid_0: This is the real world attack game, where $\mathcal{A}$ commits to a set $\hat{S}$. In response $\mathcal{A}$ gets the public parameters from the setup, and then makes the following queries.

- Register honest user queries: $\mathcal{A}$ submits an index i. 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 index i such that $i \notin \hat{S}$, along with a string x_i as the public value for party 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 shared key queries: The adversary submits a subset $\mathcal{S} \neq \hat{S}$ of users, of which at least one is honest. The challenger uses PRF to compute and send the group key.
- Finally, for set $\hat{S}$, the adversary receives either the correct group key (if $b = 0$) or a random key (if $b = 1$). The adversary outputs a bit b' and wins if $\Pr[b' = b] > \frac{1}{2} + 1/\mathsf{poly}(\kappa)$ for some polynomial $\mathsf{poly}(\cdot)$.

We now demonstrate a sequence of hybrids, via which we argue that the advantage of the adversary in guessing the bit b is $\mathsf{negl}(\kappa)$, where $\mathsf{negl}(\cdot)$ is a function that is asymptotically smaller than $1/\mathsf{poly}(\kappa)$ for all polynomials $\mathsf{poly}(\cdot)$. We give

```
NIKE- Public Parameters

Constants: $H$, $\mathsf{PRF}$ key $K, i^*, h^*$.
Input: $h, i, pv, sv, \pi, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K, t)$.
3. $\underline{\text{If } i \leq i^*,\ h = h^*, \text{ output } \perp \text{ and abort.}}$
4. If $\mathsf{Ver}(H, h, i, pv, \pi) = 1$ and $\mathsf{PRG}(sv) = pv$, output $\mathsf{PRF}(K_t, h)$.
5. Else output $\perp$.
```

Fig. 2. Static Secure NIKE Parameters P_{KE,i^*}

short overviews of indistinguishability between the hybrids, with full proofs in Appendix A. We use $\underline{\text{underline}}$ changes between subsequent hybrids.

Hybrid_1: $\underline{\text{For each } i \in \hat{S}, \text{ choose } x_i \xleftarrow{\$} \{0,1\}^{2\lambda}}$. When answering register honest user queries for $i \in \hat{S}$, use these x_i values instead of generating them from PRG. Follow the rest of the game same as Hybrid_0. This hybrid is indistinguishable from Hybrid_0, by security of the PRG.

Let $t^* = |S|$. We start with $\mathsf{Hybrid}_{2,1,a}$, and go across hybrids in the following sequence: $\mathsf{Hybrid}_{2,1,a}, \mathsf{Hybrid}_{2,1,b}, \mathsf{Hybrid}_{2,2,a}, \mathsf{Hybrid}_{2,2,b} \ldots \mathsf{Hybrid}_{2,t^*,a}$, $\mathsf{Hybrid}_{2,t^*,b}$. The hybrids $\mathsf{Hybrid}_{2,i^*,a}$ and $\mathsf{Hybrid}_{2,i^*,b}$ are described below, for $i^* \in [t^*]$.

$\mathsf{Hybrid}_{2,1,a}$: $\underline{\text{Generate hash function } H \xleftarrow{\$} \mathsf{Gen}(1^\kappa, 2^\kappa, 1)}$. Follow the rest of the game same as Hybrid_1. This is indistinguishable from Hybrid_1 because of indistinguishability of statistical binding index.

$\mathsf{Hybrid}_{2,i^*,a}$: $\underline{\text{Generate hash function } H \xleftarrow{\$} \mathsf{Gen}(1^\kappa, 2^\kappa, i^*)}$. Follow the rest of the game same as $\mathsf{Hybrid}_{2,i^*-1,b}$. This is indistinguishable from $\mathsf{Hybrid}_{2,i^*-1,b}$ because of indistinguishability of statistical binding index.

$\mathsf{Hybrid}_{2,i^*,b}$: Generate hash function $H \xleftarrow{\$} \mathsf{Gen}(1^\kappa, 2^\kappa, i^*)$. $\underline{\text{Let } h^* = H(\{pv_j\}_{j \in S})}$. $\underline{\text{Set } P_{KE} \text{ to } C_{i^*}, \text{ an obfuscation of the circuit in Fig. 2.}}$ This is indistinguishable from $\mathsf{Hybrid}_{2,i^*,a}$ because of iO between functionally equivalent circuits P_{KE,i^*} and P_{KE,i^*-1} when the hash is statistically binding at index i^*.

$\mathsf{Hybrid}_{2,t^*,b}$: $\underline{\text{Generate hash function } H \xleftarrow{\$} \mathsf{Gen}(1^\kappa, 2^\kappa, t^*)}$. Let $h^* = H(\{pv_j\}_{j \in S})$. Set P_{KE} to C_i which is an obfuscation of the circuit in Fig. 3.

Hybrid_3 : Generate hash function $H \xleftarrow{\$} \mathsf{Gen}(1^\kappa, 2^\kappa, t^*)$. Let $h^* = H(\{pv_j\}_{j \in S})$. $\underline{\text{Set } k^* = \mathsf{PRF}(K, t^*)}$. Set P_{KE} to C_i which is an obfuscation of the circuit in Fig. 4, using punctured key $K\{t^*\}$. Follow the rest of the game the same as in $\mathsf{Hybrid}_{2,t^*,b}$. This program is functionally equivalent to the program in Hybrid_3, and thus the hybrid is indistinguishable by iO.

Hybrid_4 : Generate hash function $H \xleftarrow{\$} \mathsf{Gen}(1^\kappa, 2^\kappa, t^*)$. Let $h^* = H(\{pv_j\}_{j \in S})$. $\underline{\text{Set } k^* \xleftarrow{\$} \{0,1\}^\kappa}$. Follow the rest of the game honestly according to Hybrid_3, with

```
NIKE- Public Parameters

Constants: $H$, PRF key $K, t^*, h^*$.
Input: $h, i, pv, sv, \pi, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K, t)$.
3. $\underline{\text{If } i \le t^*,\ h = h^*, \text{ output } \perp \text{ and abort.}}$
4. If $\mathsf{Ver}(H, h, i, pv, \pi) = 1$ and $\mathsf{PRG}(sv) = pv$, output $\mathsf{PRF}(K_t, h)$.
5. Else output $\perp$.
```

Fig. 3. Static Secure NIKE Parameters P_{KE,t^*}

```
NIKE- Public Parameters

Constants: $H$, PRF key $K\{t^*\}, t^*, h^*, k^*$.
Input: $h, i, pv, sv, \pi, t$.

1. If $i > t$, output $\perp$ and abort.
2. $\underline{\text{If } t = t^*, \text{ set } K_t = k^*.}$ Else compute $K_t = \mathsf{PRF}(K\{t^*\}, t)$.
3. If $i \le t^*$, $h = h^*$, output $\perp$ and abort.
4. $\underline{\text{If } t = t^*,\ h = h^*, \text{ output } \perp \text{ and abort.}}$
5. If $\mathsf{Ver}(H, h, i, pv, \pi) = 1$ and $\mathsf{PRG}(sv) = pv$, output $\mathsf{PRF}(K_t, h)$.
6. Else output $\perp$.
```

Fig. 4. Static Secure NIKE Parameters $P_{KE'}$

this value of k^*. This hybrid is indistinguishable from Hybrid_3 because of security of the punctured PRF.

Hybrid_5 : Generate hash function $H \xleftarrow{\$} \mathsf{Gen}(1^\kappa, 2^\kappa, t^*)$. Set $h^* = H(\{pv_j\}_{j \in S})$. Set $k^* \xleftarrow{\$} \{0,1\}^\kappa$. Set P_{KE} to C_i which is an obfuscation of the circuit in Fig. 5, using punctured key $k^*\{h^*\}$. This program is functionally equivalent to the program in Hybrid_4, thus the hybrids are indistinguishable by iO.

Finally, by security of punctured PRF, $\mathcal{A}$'s advantage in Hybrid_5 is $\mathsf{negl}(\kappa)$.

4.3 Removing the Setup

We note that in our protocol, the Publish algorithm is independent of the Setup algorithm. In such a scenario, [6] gave the following theorem, which can be used to remove setup from our scheme in the case of static corruptions.

Imported Theorem 3. *[6] Let* (Setup, Publish, KeyGen) *be a statically secure NIKE protocol where* Publish *does not depend on* params *output by* Setup*, but instead just takes as input* (λ, i)*. Then there is a statically secure NIKE protocol* (Setup′, Publish′, KeyGen′) *with no setup.*

```
NIKE- Public Parameters

Constants: $H$, $\mathsf{PRF}$ key $K\{t^*\}, t^*, h^*, k^*\{h^*\}$.
Input: $h, i, pv, sv, \pi, t$.

1. If $i > t$, output $\perp$ and abort.
2. $\underline{\text{If } t = t^*\text{, set } K_t = k^*\{h^*\}}$. Else compute $K_t = \mathsf{PRF}(K, t)$.
3. If $i \leq t^*$, $h = h^*$, output $\perp$ and abort.
4. If $t = t^*$, $h = h^*$, output $\perp$ and abort.
5. If $\mathsf{Ver}(H, h, i, pv, \pi) = 1$ and $\mathsf{PRG}(sv) = pv$, output $\mathsf{PRF}(K_t, h)$.
6. Else output $\perp$.
```

Fig. 5. Static Secure NIKE Parameters P_{KE}

5 ID-NIKE for Unbounded Parties

In the identity-based NIKE setting, there is a trusted setup that outputs public parameters, and generates secret keys for parties based on their identity. These parties then run another key-generating algorithm on their secret keys and setup, to compute shared group keys.

Our NIKE scheme can be extended to obtain identity-based NIKE for unbounded parties with a polynomial reduction to the security of indistinguishability obfuscation and fully homomorphic encryption. In this section, we describe our protocol for ID-NIKE for a-priori unbounded parties, and give an overview of the hybrid arguments.

5.1 Construction

Let κ denote the security parameter, PRF denote a puncturable pseudo-random function family mapping κ bits to κ bits and PRG denote a length-doubling pseudo-random generator with inputs of size κ bits. ID-NIKE consists of the following algorithms.

- $\mathsf{Setup}(1^\kappa)$: Sample random PRF keys K_1 and K_2. Compute the program P_{IBKE} in Fig. 6 padded to the appropriate length, and compute $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$. Sample SSB hash $H \leftarrow \mathsf{Gen}(1^\kappa)$. Publish the public parameters $\mathsf{params} = P_{\mathsf{iO}}, H$.
- $\mathsf{Extract}(K_2, \mathsf{id})$: Output $sk_{\mathsf{id}} = \mathsf{PRF}(K_2, \mathsf{id})$.
- $\mathsf{KeyGen}(\mathsf{params}, S, \mathsf{id}, s_{\mathsf{id}})$: To compute shared key k_S for lexicographically ordered set S, compute $h = H(S)$ and $\pi = \mathsf{Open}(h, S, i)$; where i denotes the index of id in sorted S. Then obtain output $k_S = P_{\mathsf{iO}}(h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, |S|)$.

5.2 Security Game and Hybrids

Hybrid_0: This is the real world attack game, where $\mathcal{A}$ commits to a set $\hat{S}$. In response $\mathcal{A}$ gets the public parameters as hash function H and the obfuscation of P_{IBKE} and then makes the following queries.

```
NIKE- Public Parameters

Constants: $H, \mathsf{PRF}$ keys $K_1, K_2$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K_1, t)$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$ or $\mathsf{PRG}(s_{\mathsf{id}}) \neq \mathsf{PRG}(\mathsf{PRF}(K_2, \mathsf{id}))$, output
   $\perp$ and abort.
4. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 6. Static Secure ID-NIKE Parameters P_{IBKE}

- Obtain secret keys: $\mathcal{A}$ submits an identity id such that $\mathsf{id} \leq \mathsf{poly}(\kappa)$ and $\mathsf{id} \notin \hat{S}$. The challenger outputs $\mathsf{Extract}(K_2, \mathsf{id})$ to $\mathcal{A}$.
- Reveal shared keys: The adversary submits a subset $\mathcal{S} \neq \hat{S}$ of users, of which at least one is honest. The challenger uses the public parameters to compute and send the group key.
- Finally, for set $\hat{S}$, the adversary receives either the correct group key (if $b = 0$) or a random key (if $b = 1$). The adversary outputs a bit b' and wins if $\Pr[b' = b] > \frac{1}{2} + 1/\mathsf{poly}(\kappa)$ for some polynomial $\mathsf{poly}(\cdot)$.

We now demonstrate a sequence of hybrids, via which we argue that the advantage of the adversary in guessing the bit b is $\mathsf{negl}(\kappa)$, where $\mathsf{negl}(\cdot)$ is a function that is asymptotically smaller than $1/\mathsf{poly}(\kappa)$ for all polynomials $\mathsf{poly}(\cdot)$. We give short arguments for indistinguishability between the hybrids, with complete proofs in the full version.

Let $\hat{S} = \{\mathsf{id}^*_1, \mathsf{id}^*_2, \mathsf{id}^*_3, \ldots \mathsf{id}^*_{|\hat{S}|}\}$. Then, for $p \in [1, |\hat{S}|]$, we have the sequence of hybrids:
$\mathsf{Hybrid}_{p-1,j}, \mathsf{Hybrid}_{p,a}, \mathsf{Hybrid}_{p,b}, \ \ldots \mathsf{Hybrid}_{p,j}, \mathsf{Hybrid}_{p+1,a}, \mathsf{Hybrid}_{p+1,b}, \ldots$. Here, $\mathsf{Hybrid}_0 \equiv \mathsf{Hybrid}_{0,j}$. We now write out the experiments and demonstrate the sequence of changes between $\mathsf{Hybrid}_{p-1,j}$ and $\mathsf{Hybrid}_{p,j}$ for any $p \in [1, |\hat{S}|]$.

$\mathsf{Hybrid}_{p-1,j}$: This is the same as Hybrid_0 except that the hash is generated as $\underline{H \leftarrow \mathsf{Gen}(1^\kappa, 2^\kappa, p-1)}$. The challenger computes $h^* = H(\hat{S})$ and outputs the program P_{IBKE} in Fig. 7 padded to the appropriate length. He publishes $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$.

$\mathsf{Hybrid}_{p,a}$: This is the same as $\mathsf{Hybrid}_{p-1,j}$ except that the challenger computes $\underline{r^*_p = \mathsf{PRF}(K_2, \mathsf{id}^*_p)}$, $\underline{z^*_p = \mathsf{PRG}(r^*_p)}$. He computes punctured PRF key $K_2\{id^*_p\}$ and using the program P_{IBKE} in Fig. 8 padded to the appropriate length, computes $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$. This is indistinguishable from $\mathsf{Hybrid}_{p-1,j}$ because of iO between functionally equivalent circuits.

$\mathsf{Hybrid}_{p,b}$: This is the same as $\mathsf{Hybrid}_{p,a}$ except that the challenger picks $\underline{r^*_p \xleftarrow{\$} \{0,1\}^\kappa}$ and sets $z^*_p = \mathsf{PRG}(r^*_p)$. This is indistinguishable from $\mathsf{Hybrid}_{p,a}$ by security of the puncturable PRF.

```
NIKE- Public Parameters

Constants: $H, \mathsf{PRF}$ keys $K_1, K_2, h^*$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K_1, t)$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$, output $\perp$ and abort.
4. $\underline{\text{If } i \leq (p-1), h = h^*, \text{ then output } \perp \text{ and abort.}}$
5. Else if $\mathsf{PRG}(s_{\mathsf{id}}) \neq \mathsf{PRG}(\mathsf{PRF}(K_2, \mathsf{id}))$, output $\perp$ and abort.
6. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 7. Static Secure ID-NIKE Parameters P_{IBKE}

```
NIKE- Public Parameters

Constants: $H, \mathsf{PRF}$ keys $K_1, \underline{K_2\{\mathsf{id}_p^*\}}, \mathsf{id}_p^*, z_p^*, h^*$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K_1, t)$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$, output $\perp$ and abort.
4. If $i \leq (p-1), h = h^*$, then output $\perp$ and abort.
5. $\underline{\text{If } \mathsf{id} = \mathsf{id}_p^*, \text{ then if } \mathsf{PRG}(s_{\mathsf{id}}) \neq z_p^*, \text{ output } \perp \text{ and abort.}}$
6. If $\mathsf{id} \neq \mathsf{id}_p^*$, then if $\mathsf{PRG}(s_{\mathsf{id}}) \neq \mathsf{PRG}(\mathsf{PRF}(K_2\{\mathsf{id}_p^*\}, \mathsf{id}))$, output $\perp$
   and abort.
7. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 8. Static Secure ID-NIKE Parameters P_{IBKE}

$\mathsf{Hybrid}_{p,c}$: This is the same as $\mathsf{Hybrid}_{p,b}$ except that the challenger sets $\underline{z_p^* \xleftarrow{\$} \{0,1\}^{2\kappa}}$. This is indistinguishable from $\mathsf{Hybrid}_{p,b}$ by security of the PRG.

$\mathsf{Hybrid}_{p,d}$: This is the same as $\mathsf{Hybrid}_{p,c}$ except that the challenger computes the program P_{IBKE} in Fig. 9 padded to the appropriate length, and publishes $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$.

With probability $1/2^\kappa$ over random choice of z_1^*, the value z_1^* does not lie in the co-domain of the length-doubling PRG. Then this hybrid is indistinguishable from $\mathsf{Hybrid}_{1,c}$ because of iO between functionally equivalent circuits.

$\mathsf{Hybrid}_{p,e}$: In this hybrid, the challenger generates $H \xleftarrow{\$} \mathsf{Gen}(1^\kappa, 2^\kappa, p)$ (such that it is statistically binding at index p). The rest of the game is same as $\mathsf{Hybrid}_{p,d}$. This hybrid is indistinguishable from $\mathsf{Hybrid}_{p,d}$ because of indistinguishability of statistical binding index.

$\mathsf{Hybrid}_{p,f}$: This is the same as $\mathsf{Hybrid}_{p,e}$ except that the challenger outputs the program P_{IBKE} in Fig. 10 padded to the appropriate length. He publishes $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$.

```
NIKE- Public Parameters

Constants: $H, \mathsf{PRF}$ keys $K_1, K_2\{\mathsf{id}_p^*\}, \mathsf{id}_p^*, h^*$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K_1, t)$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$, output $\perp$ and abort.
4. If $i \leq (p-1), h = h^*$, then output $\perp$ and abort.
5. $\underline{\text{If } \mathsf{id} = \mathsf{id}_p^*, \text{ then output } \perp \text{ and abort.}}$
6. If $\mathsf{id} \neq \mathsf{id}_p^*$, then if $\mathsf{PRG}(s_{\mathsf{id}}) \neq \mathsf{PRG}(\mathsf{PRF}(K_2\{\mathsf{id}_p^*\}, \mathsf{id}))$, output $\perp$
   and abort.
7. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 9. Static Secure ID-NIKE Parameters P_{IBKE}

```
NIKE- Public Parameters

Constants: $H, \mathsf{PRF}$ keys $K_1, K_2\{\mathsf{id}_p^*\}, h^*$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K_1, t)$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$, output $\perp$ and abort.
4. $\underline{\text{If } i \leq p, h = h^*, \text{ then output } \perp \text{ and abort.}}$
5. Else if $\mathsf{PRG}(s_{\mathsf{id}}) \neq \mathsf{PRG}(\mathsf{PRF}(K_2\{\mathsf{id}_p^*\}, \mathsf{id}))$, output $\perp$ and abort.
6. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 10. Static Secure ID-NIKE Parameters P_{IBKE}

This is indistinguishable from $\mathsf{Hybrid}_{p,e}$ because of iO between functionally equivalent circuits. The circuits are functionally equivalent because the hash is statistically binding at index p.

$\mathsf{Hybrid}_{p,g}$: This is the same as $\mathsf{Hybrid}_{p,f}$, except that the challenger picks $\underline{z_p^* \xleftarrow{\$} \{0,1\}^{2\kappa}}$, and then outputs the program P_{IBKE} in Fig. 11 padded to the appropriate length. He publishes $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$.

With probability $1/2^{\kappa}$ over the randomness of choice of z_p^*, the value z_p^* lies outside the co-domain of the PRG. Thus, with over whelming probability, the extra statement is never activated and the circuit is functionally equivalent to the one in $\mathsf{Hybrid}_{p,f}$. Then this is indistinguishable from $\mathsf{Hybrid}_{p,f}$ because of iO between functionally equivalent circuits.

$\mathsf{Hybrid}_{p,h}$: This is the same as $\mathsf{Hybrid}_{1,g}$ except that the challenger picks $\underline{r_p^* \xleftarrow{\$} \{0,1\}^{\kappa}}$ and sets $\underline{z_p^* = \mathsf{PRG}(r_p^*)}$. It follows the rest of the game same as $\mathsf{Hybrid}_{p,g}$ with this value of z_p^*. This hybrid is indistinguishable from $\mathsf{Hybrid}_{p,g}$ because of security of length-doubling PRGs.

```
NIKE- Public Parameters

Constants: $H$, PRF keys $K_1, K_2\{\mathsf{id}_p^*\}, h^*, \underline{z_p^*}$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K_1, t)$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$, output $\perp$ and abort.
4. If $i \leq p, h = h^*$, then output $\perp$ and abort.
5. $\underline{\text{If } \mathsf{id} = \mathsf{id}_p^* \text{ then if } \mathsf{PRG}(s_{\mathsf{id}}) \neq z_p^* \text{ output } \perp \text{ and abort.}}$
6. Else if $\mathsf{PRG}(s_{\mathsf{id}}) \neq \mathsf{PRG}(\mathsf{PRF}(K_2\{\mathsf{id}_p^*\}, \mathsf{id}))$, output $\perp$ and abort.
7. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 11. Static Secure ID-NIKE Parameters P_{IBKE}

```
NIKE- Public Parameters

Constants: $H$, PRF keys $K_1, K_2, h^*$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K_1, t)$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$, output $\perp$ and abort.
4. If $i \leq p, h = h^*$, then output $\perp$ and abort.
5. Else if $\mathsf{PRG}(s) \neq \mathsf{PRG}(\mathsf{PRF}(K_2, \mathsf{id}))$, output $\perp$ and abort.
6. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 12. Static Secure ID-NIKE Parameters P_{IBKE}

$\mathsf{Hybrid}_{p,i}$: This is the same as $\mathsf{Hybrid}_{p,h}$ except that the challenger sets $\underline{r_p^* = \mathsf{PRF}(K_2, \mathsf{id}_p^*)}$ and $z_p^* = \mathsf{PRG}(r_p^*)$. It follows the rest of the game same as $\mathsf{Hybrid}_{p,h}$ with this value of z_p^*. This hybrid is indistinguishable from $\mathsf{Hybrid}_{p,h}$ because of security of the puncturable PRF.

$\mathsf{Hybrid}_{p,j}$: This is the same as $\mathsf{Hybrid}_{p,i}$ except that the challenger outputs the program P_{IBKE} in Fig. 12 padded to the appropriate length. He publishes $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$.

This is indistinguishable from $\mathsf{Hybrid}_{p,i}$ by iO between functionally equivalent circuits. Note that at this stage, we have un-punctured the PRF at value id_p^*. This is crucial for our hybrid arguments to go through, because we will eventually have to program in an a-priori un-bounded number of identities.

$\mathsf{Hybrid}_{|\hat{S}|,j}$: This is the final hybrid in the sequence, where the hash is generated as $\underline{H \leftarrow \mathsf{Gen}(1^\kappa, 2^\kappa, |\hat{S}|)}$. The challenger computes $h^* = H(\hat{S})$ and outputs the program P_{IBKE} in Fig. 7 padded to the appropriate length. He publishes $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$.

Finally, we have the following two hybrids

```
NIKE- Public Parameters

Constants: $H, \mathsf{PRF}$ keys $K_1, K_2, h^*$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

1. If $i > t$, output $\perp$ and abort.
2. Compute $K_t = \mathsf{PRF}(K_1, t)$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$, output $\perp$ and abort.
4. $\underline{\text{If } i \leq \hat{S}, h = h^*, \text{ then output } \perp \text{ and abort.}}$
5. Else if $\mathsf{PRG}(s) \neq \mathsf{PRG}(\mathsf{PRF}(K_2, \mathsf{id}))$, output $\perp$ and abort.
6. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 13. Static Secure ID-NIKE Parameters P_{IBKE}

```
NIKE- Public Parameters

Constants: $H, \mathsf{PRF}$ keys $\underline{K_1\{t^*\}}, K_2, h^*, \underline{k^*}$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

1. If $i > t$, output $\perp$ and abort.
2. $\underline{\text{If } t = t^*, \text{ set } K_t = k^*, \text{ else compute } K_t = \mathsf{PRF}(K_1, t)}$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$, output $\perp$ and abort.
4. If $i \leq |\hat{S}|, h = h^*$, then output $\perp$ and abort.
5. Else if $\mathsf{PRG}(s) \neq \mathsf{PRG}(\mathsf{PRF}(K_2, \mathsf{id}))$, output $\perp$ and abort.
6. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 14. Static Secure ID-NIKE Parameters P_{IBKE}

$\mathsf{Hybrid}_{\mathsf{ante-penultimate}}$: In this hybrid, the challenger generates the hash function as $H \leftarrow \mathsf{Gen}(1^\kappa, 2^\kappa, |\hat{S}|)$. The challenger computes $h^* = H(\hat{S})$ and sets $\underline{k^* = \mathsf{PRF}(K_1, t^*)}$. He punctures the PRF key K_1 on input t^* to obtain $\underline{K\{t^*\}}$. He outputs the program P_{IBKE} in Fig. 14 padded to the appropriate length. He publishes $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$.

This is indistinguishable from $\mathsf{Hybrid}_{|\hat{S}|,j}$ because of indistinguishability between functionally equivalent circuits.

$\mathsf{Hybrid}_{\mathsf{penultimate:}}$ This is the same as $\mathsf{Hybrid}_{\mathsf{ante\text{-}penultimate}}$, except that the challenger sets $\underline{k^* \xleftarrow{\$} \{0,1\}^\kappa}$. This is indistinguishable from $\mathsf{Hybrid}_{\mathsf{ante\text{-}penultimate}}$ because of security of the puncturable PRF.

$\mathsf{Hybrid}_{\mathsf{ultimate:}}$ This is the same as $\mathsf{Hybrid}_{\mathsf{penultimate}}$, except that the challenger punctures PRF key k^* on value h^*. Then, he sets the program P_{IBKE} in Fig. 15 using punctured key $\underline{k^*\{h^*\}}$ padded to the appropriate length. He publishes $P_{\mathsf{iO}} = \mathsf{iO}(P_{\mathsf{IBKE}})$.

This hybrid is indistinguishable from $\mathsf{Hybrid}_{\mathsf{penultimate}}$ because of iO between functionally equivalent programs. Finally, the distinguishing advantage of the adversary in this hybrid is at most $\mathsf{negl}(\kappa)$, by security of the puncturable PRF.

NIKE- Public Parameters

Constants: H, PRF keys $K_1\{t^*\}, K_2, h^*, \underline{k^*\{h^*\}}$.
Input: $h, i, \mathsf{id}, \pi, s_{\mathsf{id}}, t$.

```
1. If $i > t$, output $\perp$ and abort.
2. If $t = t^*$, set $K_t = k^*\{h^*\}$, else compute $K_t = \mathsf{PRF}(K_1, t)$.
3. If $\mathsf{Ver}(H, h, i, \mathsf{id}, \pi) = 0$, output $\perp$ and abort.
4. If $i \leq |\hat{S}|, h = h^*$, output $\perp$ and abort.
5. $\underline{\text{If } t = t^*, h = h^*, \text{ output } \perp \text{ and abort.}}$
6. Else if $\mathsf{PRG}(s) \neq \mathsf{PRG}(\mathsf{PRF}(K_2, \mathsf{id}))$, output $\perp$ and abort.
7. Else output $\mathsf{PRF}(K_t, h)$.
```

Fig. 15. Static Secure ID-NIKE Parameters P_{IBKE}

6 Conclusion

We construct static-secure protocols that allow NIKE and ID-NIKE between an unbounded number of parties, relying on more feasible assumptions such as indistinguishability obfuscation and fully homomorphic encryption; as opposed to ‘knowledge-type’ assumptions such as differing-inputs obfuscation. It would be interesting to design protocols that tolerate more active attacks by adversaries, for an unbounded number of parties.

A NIKE: Proofs of Indistinguishability of the Hybrids

Lemma 1. *For all (non-uniform) PPT adversaries $\mathcal{D}$, $\mathcal{D}(\mathsf{Hybrid}_0) \approx_c \mathcal{D}(\mathsf{Hybrid}_1)$.*

Proof. We define a sub-sequence of polynomially many (concretely, $|\hat{S}|$) sub-hybrids $\mathsf{Hybrid}_{0,j}$ for $j \in |\hat{S}|$, where $\mathsf{Hybrid}_{0,j}$ is the same as Hybrid_0 except that the challenger samples the first j public values for parties in the set $\hat{S}$ at random in $\{0,1\}^{2\kappa}$ instead of generating them as the output of the PRG. Note that $\mathsf{Hybrid}_{0,0} \equiv \mathsf{Hybrid}_0$ and $\mathsf{Hybrid}_{0,|\hat{S}|} \equiv \mathsf{Hybrid}_1$.

We show that hybrids $\mathsf{Hybrid}_{0,j}$ and $\mathsf{Hybrid}_{0,j+1}$ are computationally indistinguishable. We will prove this by contradiction.

Suppose there exists a distinguisher $\mathcal{D}$ which distinguishes between $\mathsf{Hybrid}_{0,j}$ and $\mathsf{Hybrid}_{0,j+1}$ with advantage $1/\mathsf{poly}(\kappa)$ for some polynomial $\mathsf{poly}(\cdot)$. We construct a reduction that uses this distinguisher to break security of the underlying PRG. The reduction obtains a challenge value x, which may either be the output of the PRG on a uniform input, or may be chosen uniformly at random in $\{0,1\}^{2\kappa}$. It picks the first j public values for parties in the set $\hat{S}$ at random in $\{0,1\}^{2\kappa}$ instead of generating them as the output of the PRG. It sets the $(j+1)^{th}$ public value to the challenge value x. It samples the remaining public values for parties in the set $\hat{S}$ by picking a uniform $s_i \in \{0,1\}^{\kappa}$ and computing $x_i = \mathsf{PRG}(s_i)$.

If x is the output of a PRG this is the experiment in $\mathsf{Hybrid}_{0,j}$ else it is $\mathsf{Hybrid}_{0,j+1}$. Therefore, the reduction can mimic the output of the distinguisher between $\mathsf{Hybrid}_{0,j}$ and $\mathsf{Hybrid}_{0,j+1}$, thereby breaking the security of the PRG with advantage $1/\mathsf{poly}(\kappa)$.

Lemma 2. *For all (non-uniform) PPT adversaries* $\mathcal{D}$, $\mathcal{D}(\mathsf{Hybrid}_1) \approx_c \mathcal{D}(\mathsf{Hybrid}_{2,1,a})$.

Proof. Suppose there exists a distinguisher which distinguishes between Hybrid_1 and $\mathsf{Hybrid}_{2,1,a}$ with advantage $1/\mathsf{poly}(\kappa)$ for some polynomial $\mathsf{poly}(\cdot)$. We construct a reduction that uses this distinguisher to break index-hiding security of the somewhere statistically hiding hash.

The reduction gives indices $\{0,1\}$ to the hash challenger. The challenger then generates hash $H = \mathsf{Gen}(s,b)$ for $b \xleftarrow{\$} \{0,1\}$ and sends them to the reduction. The reduction uses this function H as the hash (instead of generating the hash itself), and continues the game with the distinguisher. If $b=0$, this corresponds to Hybrid_1, and if $b=1$ the game corresponds to $\mathsf{Hybrid}_{2,1,a}$. Therefore, the reduction can mimic the output of the distinguisher between $\mathsf{Hybrid}_{2,i^*-1,b}$ and $\mathsf{Hybrid}_{2,i^*,a}$, thereby breaking index-hiding security of the hash function with advantage $1/\mathsf{poly}(\kappa)$.

Lemma 3. *For all (non-uniform) PPT adversaries* $\mathcal{D}$, $\mathcal{D}(\mathsf{Hybrid}_{2,i^*-1,b}) \approx_c \mathcal{D}(\mathsf{Hybrid}_{2,i^*,a})$.

Proof. Suppose there exists a distinguisher which distinguishes between $\mathsf{Hybrid}_{2,i^*-1,b}$ and $\mathsf{Hybrid}_{2,i^*,a}$ with advantage $1/\mathsf{poly}(\kappa)$ for some polynomial $\mathsf{poly}(\cdot)$. We construct a reduction that uses this distinguisher to break index-hiding security of the somewhere statistically hiding hash.

The reduction gives indices $\{i^*-1, i^*\}$ to the hash challenger. The challenger then generates hash $H = \mathsf{Gen}(s,b)$ for $b \xleftarrow{\$} \{i^*-1, i^*\}$ and sends them to the reduction. The reduction uses this function H as the hash (instead of generating the hash itself), and continues the game with the distinguisher. If $b = i^*-1$, this corresponds to $\mathsf{Hybrid}_{2,i^*-1,b}$, and if $b=i^*$ the game corresponds to $\mathsf{Hybrid}_{2,i^*,a}$. Therefore, the reduction can mimic the output of the distinguisher between Hybrid_1 and $\mathsf{Hybrid}_{2,1,a}$, thereby breaking index-hiding security of the hash function with advantage $1/\mathsf{poly}(\kappa)$.

Lemma 4. *For all (non-uniform) PPT adversaries* $\mathcal{D}$, $\mathcal{D}(\mathsf{Hybrid}_{2,i^*,a}) \approx_c \mathcal{D}(\mathsf{Hybrid}_{2,i^*,b})$.

Proof. Note that in the experiment of $\mathsf{Hybrid}_{2,1,a/b}$, $H \leftarrow Gen(1^\kappa, 2^\kappa, i^*)$. Thus, by the statistical binding property of the hash function, if $\mathsf{Ver}(H,h,i^*,u,\pi)$ and $\mathsf{Ver}(H,h,i^*,u',\pi')$ accept, then it must be that $u=u'$.

Consider the programs P_{KE,i^*-1} and P_{KE,i^*}. Note that the only place where the two programs may differ is on inputs of the form $(h^*, i^*, pv, sv, \pi, t)$, where $h^* = H(\hat{S})$. Denote the public values $\{pv_j\}_{j\in S}$ by $pv_{x_1}, pv_{x_2}, \ldots pv_{x_{|S|}}$. In this case, P_{KE,i^*-1} (in $\mathsf{Hybrid}_{2,i^*,a}$) checks if $\mathsf{Ver}(H,h^*,i^*,pv,\pi) = 1$ and if $\mathsf{PRG}(sv) = pv$, then outputs $\mathsf{PRF}(K, h)$ else outputs $\perp$. On the other hand, P_{KE,i^*} (in $\mathsf{Hybrid}_{2,i^*,b}$) always outputs $\perp$. Because of the statistical binding property of the hash at index i, if $\mathsf{Ver}(H, h^*, i^*, pv, \pi)$ accepts for any value of (pv, π), then $pv = pv_{x_{i^*}}$. Moreover, since $pv_{x_{i^*}}$ is uniformly chosen in the range of the PRG, then with overwhelming probability, there does not exist any value sv such that $\mathsf{PRG}(sv) = pv_{x_{i^*}}$. Thus, the 'if' condition in P_{KE,i^*-1} in $\mathsf{Hybrid}_{2,i^*,a}$ will never be activated, and the two programs are functionally equivalent.

Therefore, the obfuscated circuits $iO(P_{KE,i^*-1})$ and $iO(P_{KE,i^*})$ must be indistinguishable by security of the iO. Suppose they are not, then consider a distinguisher $\mathcal{D}$ which distinguishes between these hybrids with non-negligible advantage. $\mathcal{D}$ can be used to break selective security of the indistinguishability obfuscation (according to Definition 1) via the following reduction to iO. The reduction acts as challenger in the experiment of $\mathsf{Hybrid}_{2,i^*,a}$.

The iO challenger $\mathsf{Samp}(1^\kappa)$ first activates the reduction, which samples the two circuits P_{KE,i^*-1}, P_{KE,i^*} and gives them to $\mathsf{Samp}(1^\kappa)$. The challenger then samples challenge circuit $C \xleftarrow{\$} \{P_{KE,i^*-1}, P_{KE,i^*}\}$, and sends $C' = iO(C)$ to the reduction. The reduction continues the game of $\mathsf{Hybrid}_{2,i^*,a}$ with C' in place of the obfuscation of program P_{KE,i^*-1}.

If $C = P_{KE,i^*-1}$, this corresponds to $\mathsf{Hybrid}_{2,i^*-1,b}$, and if $C = P_{KE,i^*}$ the game corresponds to $\mathsf{Hybrid}_{2,i^*,a}$. Therefore, the reduction can mimic the output of the distinguisher between $\mathsf{Hybrid}_{2,i^*,a}$ and $\mathsf{Hybrid}_{2,i^*,b}$, thereby breaking security of the iO with advantage $1/\mathsf{poly}(\kappa)$.

Lemma 5. *For all (non-uniform) PPT adversaries $\mathcal{D}$, $\mathcal{D}(\mathsf{Hybrid}_{2,t^*,b}) \approx_c \mathcal{D}(\mathsf{Hybrid}_3)$.*

Proof. Consider the programs P_{KE,t^*} ($\mathsf{Hybrid}_{2,t^*,b}$) and $P_{KE'}$ (Hybrid_3). Note that the only place where the two programs may differ is on inputs where $t = t^*$.

In this case, if $i \leq t^*$, then for all $h = h^*$, both programs output $\perp$ and abort. If $i > t^*$ and $t = t^*$, then $i > t$ and both programs output $\perp$ and abort in Step 1. Moreover, for $t = t^*$, $k^* = \mathsf{PRF}(K, t^*)$ and thus the programs are functionally equivalent.

Therefore, the obfuscated circuits $iO(P_{KE,t^*})$ and $iO(P_{KE'})$ must be indistinguishable by security of the iO. Suppose they are not, then consider a distinguisher $\mathcal{D}$ which distinguishes between these hybrids with non-negligible advantage. $\mathcal{D}$ can be used to break security of the indistinguishability obfuscation (according to Definition 1) via the following reduction to iO. The reduction acts as challenger in the experiment of Hybrid_3.

The iO challenger $\mathsf{Samp}(1^\kappa)$ first activates the reduction, which samples the two circuits $P_{KE,t^*}, P_{KE'}$ and gives them to $\mathsf{Samp}(1^\kappa)$. The challenger then samples challenge circuit $C \xleftarrow{\$} \{P_{KE,t^*}, P_{KE'}\}$, and sends $C' = iO(C)$ to the reduction. The reduction continues the game of Hybrid_3 with C' in place of the obfuscation of program $P_{KE'}$.

If $C = P_{KE,t^*}$, this corresponds to $\mathsf{Hybrid}_{2,t^*,b}$, and if $C = P_{KE'}$ the game corresponds to Hybrid_3. Therefore, the reduction can mimic the output of the distinguisher between $\mathsf{Hybrid}_{2,t^*,b}$ and Hybrid_3, thereby breaking security of the iO with advantage $1/\mathsf{poly}(\kappa)$.

Lemma 6. *For all (non-uniform) PPT adversaries $\mathcal{D}$, $\mathcal{D}(\mathsf{Hybrid}_3) \approx_c \mathcal{D}(\mathsf{Hybrid}_4)$.*

Proof. Suppose there exists a distinguisher $\mathcal{D}$ which distinguishes between these hybrids with non-negligible advantage. $\mathcal{D}$ can be used to break selective security of the puncturable PRF via the following reduction. The reduction acts as challenger in the experiment of Hybrid_3.

It obtains challenge set $\hat{S}$ from the distinguisher, and computes $t^* = |\hat{S}|$. Then, it gives t^* to the PRF challenger, and obtains punctured PRF key $K\{t^*\}$ and a challenge a, which is either chosen uniformly at random or is the output of the PRF at t^*. Then, the reduction continues the experiment of Hybrid_3 as challenger, except that he sets $r^* = a$.

If $a = \mathsf{PRF}(K, t^*)$ then this is the experiment of Hybrid_3, and if a is chosen uniformly at random, then this is the experiment of Hybrid_4. Therefore, the reduction can mimic the output of the distinguisher between Hybrid_3 and Hybrid_4, thereby breaking security of the puncturable PRF with advantage $1/\mathsf{poly}(\kappa)$.

Lemma 7. *For all (non-uniform) PPT adversaries $\mathcal{D}$, $\mathcal{D}(\mathsf{Hybrid}_4) \approx_c \mathcal{D}(\mathsf{Hybrid}_5)$.*

Proof. Consider the programs $P_{KE'}$ (Hybrid_4) and $P_{KE''}$ (Hybrid_5). Note that the only place where the two programs may differ is on inputs where $t = t^*$. Then, for $t = t^*, h = h^*$, both programs output $\perp$. Moreover, because of functional equivalence of the punctured key $k^*\{h^*\}$ and k^* on all points where $h \neq h^*$, the programs are equivalent.

Suppose there exists a distinguisher $\mathcal{D}$ which distinguishes between these hybrids with non-negligible advantage. $\mathcal{D}$ can be used to break security of indistinguishability obfuscation via the following reduction. The reduction acts as challenger in the experiment of Hybrid_4. It obtains challenge set $\hat{S}$ from the distinguisher, and computes $h^* = H(\hat{S}|)$. Then, it constructs gives circuits $P_{KE'}, P_{KE''}$ as input to the Samp algorithm. Samp picks $C \xleftarrow{\$} \{P_{KE'}, P_{KE''}\}$ and sends $C' = iO(C)$ to the reduction. The reduction uses C' in place of $P_{KE'}$ in the experiment of Hybrid_4.

If $C = P_{KE'}$ then this is the experiment of Hybrid_4, and if $C = P_{KE''}$, then this is the experiment of Hybrid_5. Therefore, the reduction can mimic the output of the distinguisher between Hybrid_4 and Hybrid_5, thereby breaking security of the indistinguishability obfuscation with advantage $1/\mathsf{poly}(\kappa)$.

Lemma 8. *For all (non-uniform) PPT adversaries $\mathcal{D}$, $\mathsf{Adv}_{\mathcal{D}}(\mathsf{Hybrid}_5) = \mathsf{negl}(\kappa)$.*

Proof. Suppose there exists a distinguisher $\mathcal{D}$ which has non-negligible advantage in Hybrid_5. $\mathcal{D}$ can be used to break selective security of the puncturable PRF via the following reduction. The reduction acts as challenger in the experiment of Hybrid_5.

It obtains challenge set $\hat{S}$ from the distinguisher, and computes $h^* = H(\hat{S})$. Then, it gives h^* to the PRF challenger, and obtains punctured PRF key $k^*\{t^*\}$ and a challenge a, which is either chosen uniformly at random or is the output of the PRF at h^*. Then, the reduction continues the experiment of Hybrid_3 as challenger, except that he sets the shared key to a.

If $a = \mathsf{PRF}(k^*, h^*)$ then this corresponds to the correct shared key for group $\hat{S}$, whereas if a is chosen uniformly at random, this corresponds to a random key. Therefore, the reduction can mimic the output of the distinguisher, thereby breaking security of the puncturable PRF with advantage $1/\mathsf{poly}(\kappa)$.

References

1. Abusalah, H., Fuchsbauer, G., Pietrzak, K.: Constrained prfs for unbounded inputs. IACR Cryptology ePrint Archive 2014, p. 840 (2014). http://eprint.iacr.org/2014/840
2. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. IACR Cryptology ePrint Archive 2013, p. 689 (2013). http://eprint.iacr.org/2013/689
3. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 1. Springer, Heidelberg (2001). http://dx.doi.org/10.1007/3-540-44647-8_1
4. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. IACR Cryptology ePrint Archive 2002, p. 80 (2002). http://eprint.iacr.org/2002/080
5. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. IACR Cryptology ePrint Archive 2013, p. 352 (2013)
6. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay and Gennaro [21], pp. 480–499. http://dx.doi.org/10.1007/978-3-662-44371-2_27
7. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014). http://dx.doi.org/10.1007/978-3-642-54242-8_3
8. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. IACR Cryptology ePrint Archive 2013, p. 401 (2013)
9. Boyle, E., Pass, R.: Limits of extractability assumptions with distributional auxiliary input. IACR Cryptology ePrint Archive 2013, p. 703 (2013). http://eprint.iacr.org/2013/703
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM J. Comput. **43**(2), 831–871 (2014). http://dx.doi.org/10.1137/120868669
11. Canetti, R., Garay, J.A. (eds.): CRYPTO 2013, Part I. LNCS, vol. 8042. Springer, Heidelberg (2013). http://dx.doi.org/10.1007/978-3-642-40041-4
12. Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis and Nielsen [17], pp. 468–497. http://dx.doi.org/10.1007/978-3-662-46497-7_19
13. Cash, D., Kiltz, E., Shoup, V.: The twin diffie-hellman problem and applications. J. Cryptol. **22**(4), 470–504 (2009). http://dx.doi.org/10.1007/s00145-009-9041-6

14. Coron, J., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti and Garay [11], pp. 476–493. http://dx.doi.org/10.1007/978-3-642-40041-4_26
15. Coron, J., Lepoint, T., Tibouchi, M.: New multilinear maps over the integers. IACR Cryptology ePrint Archive 2015, p. 162 (2015). http://eprint.iacr.org/2015/162
16. Diffie, W., Hellman, M.E.: New directions in cryptography. J. IEEE Trans. Inf. Theor. **22**(6), 644–654 (1976)
17. Dodis, Y., Nielsen, J.B. (eds.): TCC 2015, Part II. LNCS, vol. 9015. Springer, Heidelberg (2015). http://dx.doi.org/10.1007/978-3-662-46497-7
18. Dupont, R., Enge, A.: Provably secure non-interactive key distribution based on pairings. Discrete Appl. Math. **154**(2), 270–276 (2006). http://www.sciencedirect.com/science/article/pii/S0166218X05002337, Coding and Cryptography
19. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. IACR Cryptology ePrint Archive 2012, p. 732 (2012). http://eprint.iacr.org/2012/732
20. Freire, E.S.V., Hofheinz, D., Paterson, K.G., Striecks, C.: Programmable hash functions in the multilinear setting. In: Canetti and Garay [11], pp. 513–530. http://dx.doi.org/10.1007/978-3-642-40041-4_28
21. Garay, J.A., Gennaro, R. (eds.): CRYPTO 2014, Part I. LNCS, vol. 8616. Springer, Heidelberg (2014). http://dx.doi.org/10.1007/978-3-662-44371-2
22. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)
23. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, October 2013, Berkeley, CA, USA, pp. 40–49, 26–29. IEEE Computer Society (2013). http://dx.doi.org/10.1109/FOCS.2013.13
24. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
25. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay and Gennaro [21], pp. 518–535. http://dx.doi.org/10.1007/978-3-662-44371-2_29
26. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009, pp. 169–178. ACM (2009). http://doi.acm.org/10.1145/1536414.1536440
27. Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: Dodis and Nielsen [17], pp. 498–527. http://dx.doi.org/10.1007/978-3-662-46497-7_20
28. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: FOCS, pp. 464–479 (1984)
29. Hofheinz, D., Jager, T., Khurana, D., Sahai, A., Waters, B., Zhandry, M.: How to generate and use universal parameters. IACR Cryptology ePrint Archive 2014, p. 507 (2014). http://eprint.iacr.org/2014/507
30. Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Roughgarden, T. (ed.) Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11–13, 2015, pp. 163–172. ACM (2015). http://doi.acm.org/10.1145/2688073.2688105

31. Ishai, Y., Pandey, O., Sahai, A.: Public-coin differing-inputs obfuscation and its applications. In: Dodis and Nielsen [17], pp. 668–697. http://dx.doi.org/10.1007/978-3-662-46497-7_26
32. Joux, A.: Public-coin differing-inputs obfuscation and its applications. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838. Springer, Heidelberg (2000). http://dx.doi.org/10.1007/10722028_23
33. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. IACR Cryptology ePrint Archive 2013, p. 379 (2013)
34. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. IACR Cryptology ePrint Archive 2014, p. 925 (2014). http://eprint.iacr.org/2014/925
35. Paterson, K.G., Srinivasan, S.: On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. Des. Codes Crypt. **52**(2), 219–241 (2009). http://dx.doi.org/10.1007/s10623-009-9278-y
36. Rao, V.: Adaptive multiparty non-interactive key exchange without setup in the standard model. IACR Cryptology ePrint Archive 2014, p. 910 (2014). http://eprint.iacr.org/2014/910
37. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014, pp. 475–484. ACM (2014). http://doi.acm.org/10.1145/2591796.2591825
38. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: Symposium on Cryptography and Information Security SCIS (2000)
39. Yamakawa, T., Yamada, S., Hanaoka, G., Kunihiro, N.: Self-bilinear map on unknown order groups from indistinguishability obfuscation and its applications. Cryptology ePrint Archive, Report 2015/128 (2015). http://eprint.iacr.org/
40. Zhandry, M.: Adaptively secure broadcast encryption with small system parameters. IACR Cryptology ePrint Archive 2014, p.757 (2014). http://eprint.iacr.org/2014/757