SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Attribute-based non-interactive key exchange

TANG Fei[1*], ZHANG Rui[2*] & LI Hongda[2*]

[1]*College of Computer Science and Technology*
Chongqing University of Posts and Telecommunications, Chongqing 400065*, China;*
[2]*State Key Laboratory of Information Security*
Institute of Information Engineering of Chinese Academy of Sciences, Beijing 100093*, China*

Appendix A Introduction

Non-interactive key exchange (NIKE) [11] is an important cryptographic primitive which allows two or more users in network environments to non-interactively agree on a common shared key. Identity-based non-interactive key exchange (IBNIKE) [9,21] is an extension of the notion of NIKE in the identity-based setting. In an identity-based cryptosystem [20], each user can be identified by some unique identity id, e.g., email address. Any authorized user obtains a secret key, sk_{id}, from a trusted key generator center (KGC). IBNIKE has an advantage that, with respect to the traditional NIKE in the PKI setting, it reduces communication costs to check the validity of the public keys.

In our increasingly complex network environments, it is often convenient for the users to communicate with the others using attributes that, rather than an unique identity, describe their roles or responsibilities. The notion of attribute-based cryptography [22] was introduced for such environments. The first attempt of attribute-based cryptosystem is encryption, that is attribute-based encryption (ABE). According to Goyal et al.'s idea in [18], the notion of ABE is divided into two forms: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In a KP-ABE system, the secret key of a user is associated with an access policy function (denoted by f) defined over a set of attributes while the ciphertext is associated with a set of attributes (denoted by x). A ciphertext can be decrypted by a user only if the attribute set x with the ciphertext satisfies the policy f which associates to his secret key, that is $f(x) = 1$. A CP-ABE is a complementary form of the KP-ABE, wherein the secret key is associated with an attribute set x, while a ciphertext is associated with a policy function f. A ciphertext can be decrypted by a user who has sk_x only if $f(x) = 1$.

Naturally, we want to know that whether we can realize NIKE in the attribute-based setting, that is attribute-based non-interactive key exchange (ABNIKE). Learning from ABE, we divide the notion of ABNIKE into two forms: key-policy ABNIKE (KP-ABNIKE) and shared-key-policy ABNIKE (SP-ABNIKE). Intuitively, in a KP-ABNIKE scheme, a user who is associated with a policy function f has a secret key sk_f from the KGC. The shared key K_x will be established according to an attribute set x. A user can non-interactively generate a shared key K_x if and only if the attribute set x with the shared key satisfies the policy f associated the user's secret key, i.e., $f(x) = 1$. On the contrary, in an SP-ABNIKE scheme, a user's secret key is associated with an attribute set x, while a shared key is associated with a policy function f. A shared key K_f can be generated by a user who has sk_x only if $f(x) = 1$.

To consider the applications of the ABNIKE, we can see the following scenario which has been mentioned in some previous works, e.g., [15]. In an internet forum where the members are organized into user groups based on their skills or privileges. It is a natural requirement that the members of a group should be able to establish secure communication channel with the others members belonging to some particular groups. Here, we may notice that ABE actually can be applied to such scenario. However, the application of ABE in this scenario has a drawback that each message to shared should to be encrypted separately. Someone may say that encrypting a shared key directly by using ABE. Such implementation also has a problem that the sender who is not a member of the particular groups knows the encrypted shared key. Therefore, ABNIKE is a preferable choice in such application scenario.

There are many NIKE and IBNIKE schemes have been proposed in literatures. The classic Diffie-Hellman NIKE scheme solves the two-party case $n = 2$. Joux [19] made use of bilinear maps to construct the first three-party NIKE scheme. By using multilinear maps [5, 16], Boneh and Silverberg [5] gave an NIKE scheme for general n. Recently, Boneh and Zhandry [8] gave a general NIKE scheme by using indistinguishability obfuscation [17] (iO, see [17] for details). IBNIKE is an extension of NIKE. In IBNIKE schemes, each user has a secret key sk_{id} for an unique identity id. Then, in order

* Corresponding author (email: tangfei127@163.com, r-zhang@iie.ac.cn, lihongda@iie.ac.cn)

to build a common shared key $K_{\mathcal{I}}$ on behalf of n users $\mathcal{I} = (id_1, \ldots, id_n)$, each party $id_i \in \mathcal{I}$ only needs to take as input his/her secret key sk_{id_i} and the set of identities $\mathcal{I}$ to compute it. The SOK IBNIKE [9, 21] scheme solves the case $n = 2$. Recently, Freire et al. [13] and Boneh et al. [8] devised an IBNIKE scheme for general n by using multilinear maps and iO, respectively.

In addition, in the past several years, a handful of attribute-based (interactive) key exchange schemes were proposed. However, as far as we know, most of them were considered only for two- or three-party setting, for example, schemes in [2, 6, 14, 24–28] consider two-party ABKE,[1)] scheme in [3] considers three-party ABKE. Such two- or three-party ABKE schemes do not capture the intuitive (or natural) concept described above. In other words, as a natural extension of the standard KE and identity-based KE, the notion of ABKE should capture the intuition that the shared key is on behalf of some attribute set x (resp. policy f) rather than two or three users. Actually, the intuitive concept of ABNIKE implies the notion of two- or three-party ABNIKE. The only exceptions are [15] and [7]. In [15], Gorantla, Boyd, and Nieto designed an attribute-based authenticated key exchange scheme based on encapsulation policy attribute-based key encapsulation mechanism. However, as said by [7], the schemes in [15] are interactive, analyzed in the generic group model, and only apply to policies represented as polynomial size formulas. In [7], Boneh and Waters introduced the notion of policy-based key distribution which is a particular case of the SP-ABNIKE. Furthermore, they gave a concrete realization based on constrained pseudorandom function for circuit predicates.

In this work, we study the notion of ABNIKE. Our main works are as follows:

1. First of all, we present formal definitions for ABNIKE based on the notions of NIKE and IBNIKE. Our definitions capture the intuitive concept described above. In addition, we define security model for ABNIKE in the dishonest key registration (DKR) setting which is a rather strong model, that is, m-CKS-heavy for ABNIKE. In the m-CKS-heavy model, challenger chooses a random bit b and answers `register honest user`, `register corrupt user`, `extract`, `reveal`, and `test` queries for adversary until it outputs a bit b'. The adversary succeeds if $b' = b$. Our security model for ABNIKE is extended from the model of m-CKS-heavy for NIKE [12] and IBNIKE [8].
2. Next, by using a powerful tool, differing-input obfuscation (diO) [1, 4], we give a concrete realization of ABNIKE. The main technique for our construction is the utilization of punctured program technique [23].
3. Finally, we show that our ABNIKE notion implies IBNIKE and two- or more-party ABNIKE which have been realized by some previous works.

Appendix B Security models

Cash, Kiltz, and Shoup [10] introduced a security notion for NIKE in the public key setting, namely CKS model. The CKS model allows an adversary to obtain honestly generated public keys, and also register dishonestly generated public keys for which the adversary need not know the corresponding secret keys. This dishonest key registration (DKR) setting describes a real-world situation that the certificate authority is not assumed to check that a public key has not been previously registered to another user.

However, the CKS model missed some possible attacks, such as the adversary to "corrupt" honestly generated public keys to learn the corresponding secret keys and obtain the shared key between honest parties in the system. To fix up this problem, Freire, Hofheinz, Kiltz, and Paterson [12] augmented the original CKS model with the "missing" queries, introducing the m-CKS-heavy model in the public key setting. In the m-CKS-heavy model, the challenger chooses a random bit b and answers `register honest user`, `register corrupt user`, `extract`, `reveal`, and `test` queries for the adversary until it outputs a bit b'. The adversary is said that wins the game if $b' = b$.

The m-CKS-heavy is defined for two-party NIKE schemes. Then Boneh and Zhandry [8] extended the m-CKS-heavy model into the multi-party NIKE and identity-based multi-party NIKE settings. In addition, Boneh et al. [8] defined two weaker notions called *semi-static* and *static* security. In the semi-static security model, the adversary is required that to commit to a set $\hat{S}$ of users at the beginning of the game. The adversary then must only makes test queries on a subsets $S^* \subseteq \hat{S}$, and can only make register corrupt and extract queries on users $i \notin \hat{S}$. The static security model is a more weaker notion in which the adversary only can make a single test query on a set S^*, and it must commit to S^* before seeing the public parameters.

Based on the security models, i.e., m-CKS-heavy, for NIKE [12] and IBNIKE [8], we now define security models for ABNIKE. For consistency, we define m-CKS-heavy model for KP-ABNIKE for circuits which is played by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In the beginning, $\mathcal{C}$ takes as input a security parameter λ to run the $\mathsf{AB.Setup}$ algorithm and gives $\mathcal{A}$ the public parameter pp. The challenger $\mathcal{C}$ takes a random bit b and answers oracle queries for $\mathcal{A}$ until $\mathcal{A}$ outputs a bit b'. The challenger answers the following types of queries made by $\mathcal{A}$:

• `Register honest user`: Adversary $\mathcal{A}$ submits a function $f \in \mathbb{F}$. Challenger $\mathcal{C}$ runs $\mathsf{AB.KeyGen}$ algorithm to generate a secret key sk_f and records the tuple $(honest, f, sk_f)$.

• `Register corrupt user`: Adversary $\mathcal{A}$ submits a function $f \in \mathbb{F}$. Challenger $\mathcal{C}$ records the tuple $(corrupt, f, \perp)$. $\mathcal{A}$ can make multiple "Register corrupt user" queries for a same f during the experiment, in which case $\mathcal{C}$ only uses the most recent record.

• `Extract queries`: Adversary $\mathcal{A}$ submits a function $f \in \mathbb{F}$ that was registered as an honest user. Challenger $\mathcal{C}$ searches for a tuple $(honest, f, sk_f)$ containing f and returns sk_f to $\mathcal{A}$.

1) Scheme in [2] is called fuzzy secret handshake protocol which supports only a single threshold gate. Schemes in [6, 14] are called predicate-based key exchange protocols which actually are a variant of KP-ABNIKE.

• `Reveal queries`: Adversary $\mathcal{A}$ submits an attribute set $x \subseteq \mathbb{A}$. Challenger $\mathcal{C}$ runs $\mathsf{AB.SharedKey}$ algorithm and returns the result K_x to $\mathcal{A}$.

• `Test queries`: Adversary $\mathcal{A}$ submits an attribute set $x \subseteq \mathbb{A}$. If $b = 0$, $\mathcal{C}$ runs $\mathsf{AB.SharedKey}$ algorithm and returns the result K_x to $\mathcal{A}$. If $b = 1$, $\mathcal{C}$ chooses a random key from $\mathbb{SHK}$, records it for later, and returns it to $\mathcal{A}$. To keep consistent, $\mathcal{C}$ returns the same random key for the same x every time $\mathcal{A}$ queries for its shared key.

The adversary's queries may be made adaptively and arbitrary polynomial in number. To avoid trivial attacks, no query to the `Reveal` oracle is allowed on any attribute set x chosen for `Test` queries, and no `Extract` query is allowed on any f such that $f(x) = 1$ where x is involved in `Test` queries. In addition, we require that no policy function registered as corrupt can later be subject of a `Register honest user` query, and vice versa.

The advantage of a PPT adversary $\mathcal{A}$ (taken over the random coins of the challenger and adversary) in the above game is defined as

$$\mathbf{Adv}_{\mathcal{A},ABNIKE}^{m-CKS-heavy} = |\Pr[b' = b] - 1/2|.$$

Definition 1. We say that a KP-ABNIKE scheme is secure if all PPT adversaries have at most negligible advantage in the above m-CKS-heavy game.

We adapt Boneh and Zhandry's [8] semi-static security notion to the KP-ABNIKE setting.

Definition 2. We say that a KP-ABNIKE scheme is semi-statically secure if for any PPT adversary $\mathcal{A}$ satisfying the following properties, $\mathbf{Adv}_{\mathcal{A},ABNIKE}^{m-CKS-heavy}$ is negligible:

- $\mathcal{A}$ commits to a set $X^* \subseteq \mathbb{A}$ of attributes before seeing the public parameters.
- Each query f to `Extract` oracle must be satisfy the property $f(x) = 0$ for all $x \subseteq X^*$.
- Each query x to `Reveal` oracle must have $x \nsubseteq X^*$.
- Each query x^* to `Test` oracle must be on a subset $x^* \subseteq X^*$.

We also adapt Boneh and Zhandry's [8] static security notion to the KP-ABNIKE setting.

Definition 3. We say that a KP-ABNIKE scheme is statically secure if for any PPT adversary $\mathcal{A}$ satisfying the following properties, $\mathbf{Adv}_{\mathcal{A},ABNIKE}^{m-CKS-heavy}$ is negligible:

- $\mathcal{A}$ commits to an attribute set $x^* \subseteq \mathbb{A}$ before seeing the public parameters.
- Each query f to `Extract` oracle must be satisfy $f(x^*) = 0$.
- The attribute set x^* cannot be taken as input to the `Reveal` oracle.
- $\mathcal{A}$ makes only a single query to `Test` oracle on the attribute set x^*.

Appendix C Security of the KP-ABNIKE scheme

We now prove the static security of our KP-ABNIKE scheme (in the main work).

Theorem 1. If diO is a differing-input obfuscator, $\mathfrak{F}$ is a punctured PRF, and $\mathfrak{S}$ is a secure signature scheme, then our construction is a statically secure KP-ABNIKE scheme.

Proof. We describe the proof as a sequence of the following hybrid games.

• $\mathcal{G}_0$: This game corresponds to the honest execution of the static m-CKS-heavy game where the adversary initially submits a challenging attribute set $x^* \subseteq \mathbb{A}$. The challenger first runs $\mathsf{Sig.Key}(1^\lambda)$ and $\mathsf{PRF.Key}(1^\lambda)$ algorithms to generate (sk, vk) and K. Then it builds an obfuscated program $diO(\mathcal{P})$ and sets the public parameters pp and master key $msk = sk$. The adversary is given pp and then makes the following queries:

• `Register honest user`: $\mathcal{A}$ submits a function $f \in \mathbb{F}$. Challenger $\mathcal{C}$ runs $\mathsf{Sig.Sign}(sk, f) \to sk_f$ and records the tuple $(honest, f, sk_f)$.

• `Register corrupt user`: $\mathcal{A}$ submits a function $f \in \mathbb{F}$ where f was not and will not be registered as honest. Challenger $\mathcal{C}$ records the tuple $(corrupt, f, \perp)$.

• `Extract queries`: $\mathcal{A}$ submits a function $f \in \mathbb{F}$, where $f(x^*) = 0$, that was registered as an honest user. Challenger $\mathcal{C}$ searches for a tuple $(honest, f, sk_f)$ containing f and returns sk_f to $\mathcal{A}$.

• `Reveal queries`: $\mathcal{A}$ submits an attribute set $x \subseteq \mathbb{A}$ and $x \neq x^*$. Challenger $\mathcal{C}$ computes $K_x = F(K, x)$ and returns it to $\mathcal{A}$.

• `Test queries`: If $b = 0$, $\mathcal{C}$ computes $K_{x^*} = F(K, x^*)$ and returns it to $\mathcal{A}$. If $b = 1$, $\mathcal{C}$ chooses a random key from $\mathbb{SHK}$ and returns it to $\mathcal{A}$.

• $\mathcal{G}_1$: The adversary $\mathcal{A}$ initially gives a challenging attribute set $x^* \subseteq \mathbb{A}$ to the challenger $\mathcal{C}$. In addition, $\mathcal{C}$ receives a verification key vk with respect to the signature scheme $\mathfrak{S}$ from the signature scheme challenger. $\mathcal{C}$ first runs $\mathsf{PRF.Key}(1^\lambda)$ algorithm to generate K. Then it builds an obfuscated program $diO(\mathcal{P}^*)$, where $\mathcal{P}^*$ is defined below and has the same size as $\mathcal{P}$ by appropriate padding.

Shared key generation program $\mathcal{P}^*$

Constants: verification key vk, punctured PRF key K_{x^*}, set x^*
Inputs: attribute set x, function f, and secret key sk_f

1. Check that $f(x) \stackrel{?}{=} 1$ and $\mathsf{Sig.Vrfy}(vk, f, sk_f) \stackrel{?}{=} 1$.
2. If any checks fail, output $\perp$; else,
   (a) if $x = x^*$: it outputs $\perp$;

(b) if $x \neq x^*$: it outputs $K_x \leftarrow F(K_{x^*}, x)$.

Finally, $\mathcal{C}$ sets the public parameters pp and answers the following queries:

- `Register honest user`: $\mathcal{A}$ submits a function $f \in \mathbb{F}$. If $f(x^*) = 0$, $\mathcal{C}$ makes a signing query on f to the signature scheme challenger and it will receive a signature sk_f. It then records the tuple $(honest, f, sk_f)$. If $f(x^*) = 1$, then $\mathcal{C}$ directly records the tuple $(honest, f, \emptyset)$. Note that in the static security model, the adversary $\mathcal{A}$ cannot make query to the `Extract` oracle for functions f which satisfy $f(x^*) = 1$, and hence the tuple $(honest, f, \emptyset)$ will not affect $\mathcal{C}$'s capability to answer $\mathcal{A}$'s queries.
- `Register corrupt user`: It is same as that in $\mathcal{G}_0$.
- `Extract queries`: It is same as that in $\mathcal{G}_0$.
- `Reveal queries`: It is same as that in $\mathcal{G}_0$.
- `Test queries`: It is same as that in $\mathcal{G}_0$.
- $\mathcal{G}_2$: This game is identical to the game $\mathcal{G}_1$, except that instead of setting the challenge shared key as $K_{x^*} \leftarrow F(K, x^*)$, it is chosen uniformly at random from $\mathbb{SHK}$, which is independent of F.

We now need to argue that each of these hybrid games are computationally indistinguishable.

Lemma 1. If diO is a secure differing-input obfuscation and $\mathfrak{S}$ is a secure signature scheme, then the games $\mathcal{G}_0$ and $\mathcal{G}_1$ are computational indistinguishable.

Proof. First of all, we argue that $\mathcal{P}$ and $\mathcal{P}^*$ form a differing-inputs circuit family. We note that the only difference between circuits $\mathcal{P}$ and $\mathcal{P}^*$ is on the challenge point $x^* \subseteq \mathbb{A}$, hence the input, that can cause $\mathcal{P}$ and $\mathcal{P}^*$ output two different values, can only be the form (x^*, f, sk_f), where $f(x^*) = 1$ and $\mathsf{Sig.Vrfy}(vk, f, sk_f) = 1$. Given a such input circuit $\mathcal{P}$ outputs $F(K, x^*)$ but $\mathcal{P}^*$ outputs $\perp$. Given any other inputs $\mathcal{P}$ and $\mathcal{P}^*$ will output the same values. If $\mathcal{P}$ and $\mathcal{P}^*$ cannot form a differing-inputs circuit family which means that there exists a sample algorithm can sample $(C_0 = \mathcal{P}, C_1 = \mathcal{P}^*, aux = x^*)$ and an adversary can find out the differing-input (x^*, f, sk_f), such that $f(x^*) = 1$ and $\mathsf{Sig.Vrfy}(vk, f, sk_f) = 1$. The validation of $\mathsf{Sig.Vrfy}(vk, f, sk_f) = 1$ means that sk_f is a valid signature for f, where $f(x^*) = 1$. Here f was not queried to the `Extract` oracle and thus f was not taken as input to the signing oracle by $\mathcal{C}$. Based on this analysis, $\mathcal{C}$ can make use of (f, sk_f) as the forgery with respect to the signature scheme $\mathfrak{S}$. Therefore, the security of the signature scheme shows that $\mathcal{P}$ and $\mathcal{P}^*$ form a differing-inputs function family. Hence, according to the security of diO, the obfuscations $diO(\mathcal{P})$ and $diO(\mathcal{P}^*)$ are indistinguishable. This in turn shows that $\mathcal{G}_0$ is indistinguishable from $\mathcal{G}_1$. □

Lemma 2. If $\mathfrak{F}$ is a secure punctured PRF, then the advantages of any PPT adversary in games $\mathcal{G}_1$ and $\mathcal{G}_2$ must be negligibly close.

Proof. We show that if there is a PPT adversary $\mathcal{A}$ with different advantages in games $\mathcal{G}_1$ and $\mathcal{G}_2$, then we can construct a pair of attackers $(\mathcal{A}_1, \mathcal{A}_2)$ to break the pseudorandomness property of the punctured PRF $\mathfrak{F}$. $\mathcal{A}_1(1^\lambda)$ simply invokes the adversary to obtain the challenge x^*. It then sets and gives $\tau = (x^*, O_{sk}^{Sign}, vk, K_{x^*})$ to $\mathcal{A}_2$, where O_{sk}^{Sign} denotes the signing oracle with respect to the signature scheme $\mathfrak{S}$. $\mathcal{A}_2$ obtains τ from $\mathcal{A}_1$ and a value z^* from the PRF challenger, where $z^* = F(K, x^*)$ or a random t. Note that (1) given τ, algorithm $\mathcal{A}_2$ can answer the adversary $\mathcal{A}$'s `Extract` and `Reveal` queries because O_{sk}^{Sign} and K_{x^*} in τ; and (2) this yields either the value z^* computed in $\mathcal{G}_1$ (when $z^* = F(K, x^*)$) or $\mathcal{G}_2$ (when $z^* = t$). Finally, $\mathcal{A}_2$ outputs 1 if the adversary succeeds. In conclusion, any adversary with different advantages in games $\mathcal{G}_1$ and $\mathcal{G}_2$ leads an attacker on the pseudorandomness of the punctured PRF $\mathfrak{F}$. □

Finally, in the last game $\mathcal{G}_2$, any PPT adversary obviously cannot win the game with non-negligible advantage because the real shared key is replaced by a random value. This completes the proof of Theorem 1. □

Appendix D Semi-statically secure KP-ABNIKE

Before giving the semi-statically secure KP-ABNIKE scheme, we first show that why the KP-ABNIKE with punctured PRF is hard to be semi-statically secure. In the model of semi-static security described in Definition 3, the adversary $\mathcal{A}$ commits to an attribute set $X^* \subseteq \mathbb{A}$ before seeing the public parameters. Then $\mathcal{A}$ can make `Test` queries for arbitrary polynomial times on inputs $x^* \subseteq X^*$. From the proof of Lemma 1, we know that we need to puncture all possible subsets $x^* \subseteq X^*$ that the adversary may challenge on. Therefore, if the set X^* is oversize, then we may not be able to efficiently puncture its all subsets x^*.

To circumvent the above problem, we make use of the notion of constrained PRF for circuit predicates [7,8]. Informally, constrained PRFs for circuits support constraining to sets S accepted by a polynomial size circuit C. That is to say, given a constrained key K_C, if $C(x) = 1$ then it can compute $F(K_C, x) = F(K, x)$; Else, it remains pseudorandomness for x which satisfies $C(x) = 0$. The constrained PRF for circuits can be realized from indistinguishability obfuscation [8] and multilinear maps [7].

The formal definition of constrained PRF for circuits is defined below.

Definition 4. A family of constrained PRFs for circuits consist of a triple of algorithms $\mathfrak{F}^C = (\mathsf{PRF.Key}, \mathsf{PRF.Pun}, F)$, and a pair of computable functions $n(\cdot)$ and $m(\cdot)$, satisfying the following conditions:

- Functionality preserved under puncturing: For any PPT adversary $\mathcal{A}$ such that it outputs a circuit C, then for all x which satisfies $C(x) = 1$, have:

$$\Pr[F(K, x) = F(K_C, x) : K \leftarrow \mathsf{PRF.Key}(1^\lambda), K_C \leftarrow \mathsf{PRF.Pun}(K, C)] = 1.$$

- Pseudorandom at punctured points: For any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1(1^\lambda)$ outputs a circuit C and state τ, consider an experiment where $K \leftarrow \mathsf{PRF.Key}(1^\lambda)$ and $K_C \leftarrow \mathsf{PRF.Pun}(K, C)$. Then we have:

$$\Pr[\mathcal{A}_2(\tau, K_C, C, F(K,x)|_{C(x)=0}) = 1] - \Pr[\mathcal{A}_2(\tau, K_C, C, U_{m(\lambda)}) = 1] = negl(\lambda),$$

where U_ℓ denotes the uniform distribution over ℓ bits.

We now prove that, if the PRF in the KP-ABNIKE construction is a constrained PRF for circuits, then it is semi-statically secure.

Theorem 2. If diO is a differing-input obfuscator, $\mathfrak{F}^C$ is a constrained PRF for circuits, and $\mathfrak{S}$ is a secure signature scheme, then the KP-ABNIKE construction is semi-statically secure.

Proof. We describe the proof as a sequence of the following hybrid games.

• $\mathcal{G}_0$: This game corresponds to the honest execution of the semi-static m-CKS-heavy game where the adversary initially submits a set $X^* \subseteq \mathbb{A}$. The challenger first runs $\mathsf{Sig.Key}(1^\lambda)$ and $\mathsf{PRF.Key}(1^\lambda)$ algorithms to generate (sk, vk) and K. Then it builds an obfuscated program $diO(\mathcal{P})$ and sets the public parameters pp and master key $msk = sk$. The adversary is given pp and then makes the following queries:

• `Register honest user`: $\mathcal{A}$ submits a function $f \in \mathbb{F}$. Challenger $\mathcal{C}$ runs $\mathsf{Sig.Sign}(sk, f) \to sk_f$ and records the tuple $(honest, f, sk_f)$.

• `Register corrupt user`: $\mathcal{A}$ submits a function $f \in \mathbb{F}$ where f was not and will not be registered as honest. Challenger $\mathcal{C}$ records the tuple $(corrupt, f, \perp)$.

• `Extract queries`: $\mathcal{A}$ submits a function $f \in \mathbb{F}$ that was registered as an honest user. Challenger $\mathcal{C}$ searches for a tuple $(honest, f, sk_f)$ containing f and returns sk_f to $\mathcal{A}$.

• `Reveal queries`: $\mathcal{A}$ submits an attribute set $x \subseteq \mathbb{A}$ but $x \nsubseteq X^*$. Challenger $\mathcal{C}$ computes $K_x = F(K, x)$ and returns it to $\mathcal{A}$.

• `Test queries`: $\mathcal{A}$ submits an attribute set $x^* \subseteq X^*$. If $b = 0$, $\mathcal{C}$ computes $K_{x^*} = F(K, x^*)$ and returns it to $\mathcal{A}$. If $b = 1$, $\mathcal{C}$ chooses a random key from $\mathbb{SHK}$, records it for later, and returns it to $\mathcal{A}$.

• $\mathcal{G}_1$: The adversary $\mathcal{A}$ initially commits an attribute set $x^* \subseteq \mathbb{A}$. $\mathcal{C}$ receives a verification key vk with respect to the signature scheme $\mathfrak{S}$ from the signature scheme challenger. $\mathcal{C}$ then runs $\mathsf{PRF.Key}(1^\lambda)$ algorithm to generate K. Next, $\mathcal{C}$ needs to constrain the PRF so that it can only be evaluated at points $\hat{x}$ where $\hat{x}$ is not contained in X^*. Formally, it constructs a circuit C that takes as input $\hat{x}$ and accepts only if there is some attribute $a \in \hat{x}$ that is not contained in X^*. $\mathcal{C}$ then constructs the constrained PRF for circuits C, i.e., $\mathfrak{F}^C$. Then it builds an obfuscated program $diO(\mathcal{P}^*)$, where $\mathcal{P}^*$ is defined below and has the same size as $\mathcal{P}$ by appropriate padding.

Shared key generation program $\mathcal{P}^*$

Constants: verification key vk, punctured PRF key K_C, circuit C
Inputs: attribute set x, function f, and secret key sk_f

1. Check that $f(x) \stackrel{?}{=} 1$ and $\mathsf{Sig.Vrfy}(vk, f, sk_f) \stackrel{?}{=} 1$.
2. If any checks fail, output $\perp$; else,
   (a) if $C(x) = 0$: it outputs $\perp$;
   (b) if $C(x) = 1$: it outputs $K_x \leftarrow F(K_C, x)$.

Finally, $\mathcal{C}$ sets the public parameters pp and answers the following queries:

• `Register honest user`: $\mathcal{A}$ submits a function $f \in \mathbb{F}$. $\mathcal{C}$ makes a signing query on f to the signature scheme challenger and it will receive a signature sk_f. It then records the tuple $(honest, f, sk_f)$.

• `Register corrupt user`: It is same as that in $\mathcal{G}_0$.

• `Extract queries`: It is same as that in $\mathcal{G}_0$.

• `Reveal queries`: It is same as that in $\mathcal{G}_0$.

• `Test queries`: It is same as that in $\mathcal{G}_0$.

In the end of the game, we change the conditions that the adversary $\mathcal{A}$ succeeds as follows: (1)$b' = b$; (2) all f queried to `Register honest user` and `Extract` oracles satisfy $f(x^*) = 0$ for all x^* queried to `Test` oracle.

• $\mathcal{G}_2$: This game is identical to the game $\mathcal{G}_1$, except that instead of setting all challenge shared keys as $K_{x^*} \leftarrow F(K, x^*)$, they are chosen uniformly at random from $\mathbb{SHK}$, which are independent of F.

We need to argue that each of these hybrid games are computationally indistinguishable.

Lemma 3. If diO is a secure differing-input obfuscation and $\mathfrak{S}$ is a secure signature scheme, then the games $\mathcal{G}_0$ and $\mathcal{G}_1$ are computational indistinguishable.

Proof. First of all, we argue that $\mathcal{P}$ and $\mathcal{P}^*$ form a differing-inputs circuit family. We note that the only difference between circuits $\mathcal{P}$ and $\mathcal{P}^*$ is on the points x where $C(x) = 0$, hence the input, that can cause $\mathcal{P}$ and $\mathcal{P}^*$ output two different values, can only be the form (x, f, sk_f), where $C(x) = 0, f(x) = 1$, and $\mathsf{Sig.Vrfy}(vk, f, sk_f) = 1$. Given a such input circuit $\mathcal{P}$ outputs $F(K, x)$ but $\mathcal{P}^*$ outputs $\perp$. Given any other inputs $\mathcal{P}$ and $\mathcal{P}^*$ will output the same values. If $\mathcal{P}$ and $\mathcal{P}^*$ cannot form a differing-inputs circuit family which means that there exists a sample algorithm can sample $(C_0 = \mathcal{P}, C_1 = \mathcal{P}^*, aux = C)$ and an adversary can find out the differing-input (x, f, sk_f) such that $C(x) = 0, f(x) = 1$, and $\mathsf{Sig.Vrfy}(vk, f, sk_f) = 1$. The validation of $\mathsf{Sig.Vrfy}(vk, f, sk_f) = 1$ means that sk_f is a valid signature for f, where $f(x) = 1$ and f was not queried to the `Register honest user` and `Extract` oracles.[2)] We can use such a differing input to break the security of the signature

2) This only can be checked after the game and thus increase $\mathcal{C}$'s workload. However, this will not affect $\mathcal{A}$'s advantage to win game $\mathcal{G}_1$.

scheme. Therefore, the security of the signature scheme shows that $\mathcal{P}$ and $\mathcal{P}^*$ form a differing-inputs function family. Hence, according to the security of diO, the obfuscations $diO(\mathcal{P})$ and $diO(\mathcal{P}^*)$ are indistinguishable. This in turn shows that $\mathcal{G}_0$ is indistinguishable from $\mathcal{G}_1$. □

Lemma 4. If $\mathfrak{F}^C$ is a secure constrained PRF for circuits, then the advantages of any PPT adversary in games $\mathcal{G}_1$ and $\mathcal{G}_2$ must be negligibly close.

Proof. We show that if there is a PPT adversary $\mathcal{A}$ with different advantages in games $\mathcal{G}_1$ and $\mathcal{G}_2$, then we can construct a pair of attackers $(\mathcal{A}_1, \mathcal{A}_2)$ to break the pseudorandomness property of the PRF $\mathfrak{F}^C$. $\mathcal{A}_1(1^\lambda)$ simply invokes the adversary to obtain the challenge X^*. It then sets C and $\mathfrak{F}^C$ as describe in $\mathcal{G}_1$. Next, $\mathcal{A}_1$ gives $\tau = (O_{sk}^{Sign}, vk, C, K_C)$ to $\mathcal{A}_2$, where O_{sk}^{Sign} is the signing oracle with respect to the signature scheme $\mathfrak{S}$. $\mathcal{A}_2$ obtains τ from $\mathcal{A}_1$ and a value z^* from the PRF challenger, where $z^* = F(K, x^*)$ and or a random t. Note that (1) given τ, algorithm $\mathcal{A}_2$ can answer the adversary $\mathcal{A}$'s `Register honest user`, `Extract`, and `Reveal` queries because O_{sk}^{Sign} and K_C in τ; and (2) this yields either the value z^* computed in $\mathcal{G}_1$ or $\mathcal{G}_2$. Finally, $\mathcal{A}_2$ outputs 1 if the adversary succeeds. In conclusion, any adversary with different advantages in games $\mathcal{G}_1$ and $\mathcal{G}_2$ leads an attacker on the pseudorandomness of the punctured PRF $\mathfrak{F}^C$. □

Finally, in the last game $\mathcal{G}_2$, any PPT adversary obviously cannot win the game because the real shared key is replaced by a random value. This completes the proof of Theorem 2. □

Appendix E Connection to IBNIKE and two- or more-party ABNIKE

In this section, we show that how to construct identity-based non-interactive key exchange (IBNIKE) schemes and two- or more-party shared-key-policy attribute-based non-interactive key exchange (SP-ToMABNIKE) schemes from shared-key-policy attribute-based non-interactive key exchange (SP-ABNIKE) schemes.

An IBNIKE scheme consists of three PPT algorithms: $\mathsf{IB.Setup}$ algorithm takes as input a security parameter λ to generate public parameter pp' and master key msk'; $\mathsf{IB.KeyGen}$ algorithm takes as inputs the master key msk' and user's identity id to generate a secret key sk_{id}; $\mathsf{IB.SharedKey}$ takes as inputs the public parameters pp', a group of users $\mathcal{I} = (id_1, \ldots, id_n)$, and a secret key sk_{id_s} to generate a session key $K_{\mathcal{I}}$.

The idea of the transformation is the following: In SP-ABNIKE scheme, we assume that user's attribute set is defined by an unique identity, i.e., $x := id$. Then, set the policy for the session key as $f = id_1 \bigvee \cdots \bigvee id_n$ which means that $f(id) = 1$ if and only if $id = id_i$ for $i \in [1, n]$. Concretely, the IBNIKE construction which is based on SP-ABNIKE is as following:

- $\mathsf{IB.Setup}(1^\lambda)$: Run $(pp, msk) \leftarrow \mathsf{AB.Setup}(1^\lambda)$. Set $pp' := pp, msk' := msk$.
- $\mathsf{IB.KeyGen}(msk', id)$: Run $sk_{id} \leftarrow \mathsf{AB.KeyGen}(msk, id)$.
- $\mathsf{IB.SharedKey}(pp, \mathcal{I} = (id_1, \ldots, id_n), sk_{id_s}, s)$: Set the policy function as $f_{\mathcal{I}} = id_1 \bigvee \cdots \bigvee id_n$, where $\mathcal{I} = (id_1, \ldots, id_n)$ is sorted by lexicographic order. Then run $K_{f_{\mathcal{I}}} \leftarrow \mathsf{AB.SharedKey}(pp, f_{\mathcal{I}}, sk_{id_s}, s)$.

In SP-ABNIKE schemes, we assume that an attribute set x uniquely determines a user. An SP-ToMABNIKE scheme consists of three PPT algorithms: $\mathsf{ToMAB.Setup}$ algorithm takes as input a security parameter λ to generate public parameter pp' and master key msk'; $\mathsf{ToMAB.KeyGen}$ algorithm takes as inputs the master key msk' and user's attribute set x_i, to generate a secret key sk_{x_i}; $\mathsf{ToMAB.SharedKey}$ takes as inputs the public parameters pp', attribute sets of the participants $x_1, \ldots, x_n, n \geqslant 2$, and a secret key $sk_{x_i}, i \in [1, n]$ to generate a session key $K_{1,\ldots,n}$. The difference betweens SP-ABNIKE and SP-ToMABNIKE is the following: in SP-ABNIKE schemes, the session key is established on behalf of some policy function; However, in SP-ToMABNIKE schemes, the session key is established on behalf of some specific group of users.

The transformation for SP-ToMABNIKE which is similar to the IBNIKE case. The detail is as following:

- $\mathsf{ToMAB.Setup}(1^\lambda)$: Run $(pp, msk) \leftarrow \mathsf{AB.Setup}(1^\lambda)$. Set $pp' := pp, msk' := msk$.
- $\mathsf{ToMAB.KeyGen}(msk', x)$: Run $sk_x \leftarrow \mathsf{AB.KeyGen}(msk, x)$.
- $\mathsf{ToMAB.SharedKey}(pp', (x_1, \ldots, x_n), sk_{x_i}, i)$: Set the policy function as $f_n = x_1 \bigvee \cdots \bigvee x_n$, where $x_1, \ldots, x_n$ is sorted by lexicographic order. Then run $K_{f_n} \leftarrow \mathsf{AB.SharedKey}(pp, f_n, sk_{x_i}, i)$.

References

1 Ananth P, Boneh D, Garg S, Sahai A, Zhandry M. Deffering-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013

2 Ateniese G, Kirsch J, Blanton M. Secret handshakes with dynamic and fuzzy mathcing. NDSS Symposium 2007, San Diego, CA, USA, 2007. 1–19

3 Bayat M, Aref M R. An attribute-based tripartite key agreement protocol. International Journal of Communication Systems, 2015, 28(8): 1419–1431

4 Boyle E, Chung K M, Pass R. On extractability obfuscation. TCC 2014, San Diego, CA, USA, 2014. 52–73

5 Boneh D, Silverberg A. Applications of multilinear forms to cryptography. Contemporary Mathematics, 2003, 324(1): 71–90

6 Birkett J, Stebila D. Predicate-based key exchange. ACISP 2010, Sydney, Australia, 2010. 282–299

7 Boneh D, Waters B. Constrained pseudorandom functions and their applications. ASIACRYPT 2013, Bengaluru, India, 2013. 280–300

8 Boneh D, Zhandry M. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. CRYPTO 2014, Santa Barbara, CA, USA, 2014. 480–499

9 Chen Y, Huang Q, Zhang Z. Sakai-ohgishi-kasahara identity-based non-interactive key exchange scheme, revisited. ACISP 2014, Wollongong, Australia, 2014. 274–289

10 Cash D, Kiltz E, Shoup V. The twin Diffie-Hellman problem and applications. CRYPTO 2008, Santa Barbara, CA, USA, 2008. 127–145
11 Diffie W, Hellman M E. New directions in cryptography. IEEE Transactions on Information Theory, 1976, 22(6): 644–654
12 Freire E S V, Hofheinz D, Kiltz E, Paterson K G. Non-interactive key exchange. PKC 2013, Nara, Japan, 2013. 254–271
13 Freire E S V, Hofheinz D, Paterson K G, Striecks C. Programmable hash functions in the multilinear setting. CRYPTO 2013, Santa Barbara, CA, USA, 2013. 513–530
14 Fujioka A, Suzuki K, Yoneyama K. Predicate-based authenticated key exchange resilient to ephemeral key leakage. WISA 2010, Jeju Island, Korea, 2011. 15–30
15 Gorantla M C, Boyd C, Nieto J M G. Attribute-based authenticated key exchange. ACISP 2010, Sydney, Australia, 2010. 300–317
16 Garg S, Gentry C, Halevi S. Candidate multilinear maps from ideal lattices. EUROCRYPT 2013, Athens, Greece, 2013. 1–17
17 Garg S, Gentry C, Halevi S, Raykova M, Sahai A, Waters B. Candidate indistinguishability obfuscation and functional encryption for all circuits. FOCS 2013, Berkeley, CA, USA, 2013. 40–49
18 Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. CCS 2006, Alexandria, VA, USA, 2006. 89–98
19 Joux A. A one round protocol for tripartite Diffie-Hellman. Journal of Cryptology, 17(4), 2004: 263–276
20 Shamir A. Identity-based cryptosystems and signature schemes. CRYPTO 1984, Santa Barbara, CA, USA, 1985. 47–53
21 Sakai R, Ohgishi K, Kasahara M. Cryptosystems based on pairing. The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan, 2000. 135–148
22 Sahai S, Waters B. Fuzzy identity-based encryption. EUROCRYPT 2005, Aarhus, Denmark, 2005. 457–473
23 Sahai A, Waters B. How to use indistinguishability obfuscation: deniable encryption, and more. STOC 2014, New York, NY, USA, 2014. 475–484
24 Wang H, Xu Q, Ban T. A provably secure two-party attribute-based key agreement protocol. Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2009), Kyoto, Japan, 2009. 1042–1045
25 Wang H, Xu Q, Fu X. Revocable attribute-based key agreement protocol without random oracles. Journal of Networks, 4(8): 787–794
26 Wang H, Xu Q, Fu X. Two-party attribute-based key agreement protocol in the standard model. International Symposium on Information Processing (ISIP 2009), Huangshan, China, 2009. 325–328
27 Yoneyama K. Strongly secure two-pass attribute-based authenticated key exchange. Pairing 2010, Yamanaka Hot Spring, Japan, 2010. 147–166
28 Yoneyama K. Generic construction of two-party round-optimal attribute-based authenticated key exchange without random oracles. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2013, 96(6): 1112–1123