Adaptively Secure Constrained Pseudorandom Functions

Dennis Hofheinz
dennis.hofheinz@kit.edu

Akshay Kamath
University of Texas at Austin
kamath@cs.utexas.edu

Venkata Koppula
University of Texas at Austin
kvenkata@cs.utexas.edu

Brent Waters
University of Texas at Austin
bwaters@cs.utexas.edu

Abstract

A constrained pseudo random function (PRF) behaves like a standard PRF, but with the added feature that the (master) secret key holder, having secret key K, can produce a constrained key, K_f, that allows for the evaluation of the PRF on a subset of the domain as determined by a predicate function f within some family F. While previous constructions gave constrained PRFs for poly-sized circuits, all reductions for such functionality were based in the selective model of security where an attacker declares which point he is attacking before seeing any constrained keys.

In this paper we give new constrained PRF constructions for circuits that have polynomial reductions to indistinguishability obfuscation in the random oracle model. Our solution is constructed from two recently emerged primitives: an adaptively secure Attribute-Based Encryption (ABE) for circuits and a Universal Parameters as introduced by Hofheinz et al. Both primitives are constructible from indistinguishability obfuscation (iO) (and injective pseudorandom generators) with only polynomial loss.

Supported by NSF CNS-0952692, CNS-1228599 and CNS-1414082. DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

1 Introduction

Recently, the concept of constrained pseudo random functions (PRFs) was proposed independently by Boneh and Waters [BW13], Boyle, Goldwasser and Ivan [BGI13] and Kiayias et al [KPTZ13]. A constrained PRF behaves like a standard PRF [GGM84], but with the added feature that the (master) secret key holder, having secret key K, can produce a constrained key, K_f, that allows for the evaluation of the PRF on a subset of the domain as determined by a predicate function f within some family F. The security definition of a constrained PRF system allows for a poly-time attacker to query adaptively on several functions $f_1, \ldots, f_Q$ and receive constrained keys $K_{f_1}, \ldots, K_{f_Q}$. Later the attacker chooses a challenge point x^* such that $f_i(x^*) = 0\ \forall i$. The attacker should not be able to distinguish between the output of the PRF $F(k, x^*)$ and a randomly chosen value with better than negligible probability. Constrained PRFs have been utilized for applications such as broadcast encryption [BW13], multiparty key exchange [BZ14] and the development of "punctured programming" techniques using obfuscation [SW14].

Ideally, we would would like to be able to construct constrained PRF systems for as expressive families as possible. In their initial work Boneh and Waters [BW13] gave a construction for building constrained PRFs for polynomial sized circuits (with an priori fixed depth) based on multilinear encodings [GGH13a, CLT13]. Furthermore, they demonstrated the power of constrained PRFs with several motivating applications.

One application (detailed in [BW13]) is a (secret encryption key) broadcast key encapsulation mechanism with "optimal size ciphertexts", where the ciphertext consists solely of a header describing the recipient list S. The main idea is that to the key assigned to a set S is simply the PRF evaluated on S as $F(k, S)$. A user i in the system is assigned a key for a function $f_i(\cdot)$, where $f_i(S) = 1$ if and only if $i \in S$. Other natural applications given include identity-based key exchange and a form of non-interactive policy-based key distribution. Later Sahai and Waters [SW14] showed the utility of (a limited form of) constrained PRFs in building cryptography from indistinguishability obfuscation and Boneh and Zhandry [BZ14] used them (along with obfuscation) in constructing recipient private broadcast encryption.

Adaptive Security While the functionality of the Boneh-Waters construction was expressive, their proof reduction was limited to selective security where the challenge point x^* is declared by the attacker before it makes any queries. For many applications of constrained PRFs achieving the "right" notion of adaptive security requires an underlying adaptively secure constrained PRF. In particular, this applies to the optimal size broadcast, policy-based encryption, non-interactive key exchange and recipient-private broadcast constructions mentioned above.

In this work we are interested in exploring adaptive security in constrained PRFs with polynomial time reductions (i.e. avoid complexity leveraging). To this point constructions that achieve adaptive security have relatively limited functionality. Hohenberger, Koppula, and Waters [HKW14] show how to build adaptive security from indistinguishability obfuscation for a special type of constrained PRFs called puncturable PRF. In a puncturable PRF system the attacker is allowed to make several point queries adaptively, before choosing a challenge point x^* and receiving a key that allows for evaluation at all points $x \neq x^*$. While their work presents progress in this area, there is a large functionality gap between the family of all poly-sized circuits and puncturing-type functions. Fuchsbauer et al [FKPR14] give a subexponential reduction to obfuscation for a larger class of "prefix-type" circuits, however, their reduction is still super polynomial. In addition, they give evidence that the problem of achieving full security with polynomial reductions might be difficult. They adapt the proof of [LW14] to show a black box impossibility result for a certain class of "fingerprinting" constructions that include the original Boneh-Waters [BW13] scheme.

Our Contributions In this paper we give new constrained PRF constructions for circuit classes that have polynomial reductions to indistinguishability obfuscation in the random oracle model [1].

Our solution is constructed from two recently emerged primitives: an adaptively secure Attribute-Based Encryption (ABE) [SW05] for circuits and Universal Parameters as introduced by Hofheinz et al. [HJK$^+$14]. Both primitives are constructible from indistinguishability obfuscation (iO) (and injective pseudorandom

[1] This paper supersedes an earlier eprint posting of Hofheinz [Hof14].

generators) with only polynomial loss. Waters [Wat14] recently gave an adaptively secure construction of ABE[2] based on indistinguishability obfuscation and Hofheinz et al. [HJK+14] showed how to build Universal Parameter from iO in the random oracle model — emphasizing that the random oracle heuristic is applied outside the obfuscated program.

Before we describe our construction we briefly overview the two underlying primitives. An ABE scheme (for circuits) has four algorithms. A setup algorithm ABE.setup(1^λ) that outputs public parameters $\mathsf{pk}_{\mathsf{ABE}}$, and a master secret key $\mathsf{msk}_{\mathsf{ABE}}$. The encryption algorithm ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, t, x$) takes in the public parameters, message t, an "attribute" string x and outputs a ciphertext ct. A key generation algorithm ABE.keygen($\mathsf{msk}_{\mathsf{ABE}}, C$) outputs a secret key given a boolean circuit C. Finally, the decryption algorithm ABE.dec(SK, ct) will decrypt an ABE ciphertext encrypted under attribute x iff $C(x) = 1$, where C is the circuit associated with the secret key.

The second primitive is a universal parameters scheme. Intuitively a universal parameters scheme behaves somewhat like a random oracle except it can sample from arbitrary distributions as opposed to just uniformly random strings. More concretely, a universal parameters scheme consists of two algorithms, UniversalGen and InduceGen. In a set-up phase, $U \leftarrow$ UniversalGen(1^λ) will take as input a security parameter and output "universal parameters" U. We can use these parameters to "obliviously" sample from a distribution specified by a circuit d, in the following sense. If we call InduceGen(U, d) the scheme will output $d(z)$ for hidden random coins z that are pseudorandomly derived from U and d.

Security requires that in the random oracle model, UniversalGen outputs images that look like independently and honestly generated d-samples, in the following sense. Namely, we require that an efficient simulator can simulate U and the random oracle such that the output of InduceGen on arbitrarily many adversarially chosen inputs d_i coincides with independently and honestly chosen images $d_i(z_i)$ (for truly random z_i that are hidden even from the simulator). Of course, the simulated U and the programmed random oracle must be computationally indistinguishable from the real setting.

Our Solution in a Nutshell We now describe our construction that shows how to build constrained PRFs from adaptively secure ABE and universal parameters. One remarkable feature is the simplicity of our construction once the underlying building blocks are in place.

The constrained PRF key is setup by first running $U \leftarrow$ UniversalGen(1^λ) and ($\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}$) $\leftarrow$ ABE.setup(1^λ). The master PRF key K is (U, ($\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}$)). To define the PRF evaluation on input x we let $d_{\mathsf{pk}_{\mathsf{ABE}};x}(z = (t, r))$ be a circuit in some canonical form that takes as input random $z = (t, r)$ and computes ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, t, x; r$). Here we view $\mathsf{pk}_{\mathsf{ABE}}, x$ as constants hardwired into the circuit d and t, r as the inputs, where we make the random coins of the encryption algorithm explicit. To evaluate the PRF $F(K, x)$ we first compute ct_x = InduceGen($U, d_{\mathsf{pk}_{\mathsf{ABE}};x}$). Then we compute and output ABE.dec($\mathsf{msk}_{\mathsf{ABE}}, \mathsf{ct}_x$)[3]. Essentially, the evaluation function on input x first uses the universal parameters to encrypt an ABE ciphertext under attribute x for a randomly chosen message t. Then it uses the master secret key to decrypt the ciphertext which gives t as the output.

To generate a constrained key for circuit C, the master key holder simply runs the ABE key generation to compute sk_C = ABE.keygen($\mathsf{msk}_{\mathsf{ABE}}, C$) and sets the constrained key to be $K\{C\} = (U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C))$. Evaluation can be done using $K\{C\}$ on input x where $C(x) = 1$. Simply compute ct_x from the universal parameters U as above, but then use sk_C to decrypt. The output will be consistent with the master key evaluation.

The security argument is organized as follows. We first introduce a hybrid game where the calls to the universal parameters scheme are answered by a parameters oracle that generates a fresh sample every time it is called. The security definition of universal parameters schemes argues (in the random oracle model) that the attacker's advantage in this game must be negligibly close to the original advantage. Furthermore, any polynomial time attacker will cause this parameters oracle to be called at most some polynomial Q number of times. One of these calls must correspond to the eventual challenge input x^*.

[2]The construction is actually for Functional Encryption which implies ABE.

[3]We use the convention that the master secret key can decrypt all honestly generated ABE ciphertexts. Alternatively, one could just generate a secret key for a circuit that always outputs 1 and use this to decrypt.

We can now reduce to the security of the underlying ABE scheme. First the reduction guesses with $1/Q$ success probability which parameter oracle call will correspond to x^* and embed and ABE challenge ciphertext here. An attacker on the constrained PRF scheme now maps straightforwardly to an ABE attacker.

Future Directions A clear future direction is to attempt to achieve greater functionality in the standard model. There is a significant gap between our random oracle model results of constrained PRFs for all circuits and the standard model results of Hohenberger, Koppula, and Waters for puncturable PRFs [HKW14]. It would be interesting to understand if there are fundamental limitations to achieving such results. Fuchsbauer et al [FKPR14] et al. give some initial steps to negative results, however, it is unclear if they generalize to larger classes of constructions.

Relationship to [Hof14] We note to the reader that this work supersedes/replaces an earlier eprint article of Hofheinz [Hof14]. We intend this paper to be viewed as the definitive source for ideas appearing both here and some related to [Hof14].

Other Related Work Attribute-Based Encryption for circuits was first achieved independently by Garg, Gentry, Halevi, Sahai and Waters [GGH+13b] from multilinear maps and by Gorbunov, Vaikuntanathan and Wee [GVW13] from the learning with errors [Reg05] assumption. Both works were proven selectively secure; requiring complexity leveraging for adaptive security. In two recent works, Waters [Wat14] and Garg, Gentry, Halevi and Zhandry [GGHZ14] achieve adaptively secure ABE for circuits under different cryptographic assumptions. We also note that Boneh and Zhandry [BZ14] show how to use indistinguishability obfuscation for circuits and punctured PRFs to create constrained PRFs for circuit. This construction is limited though to either selective security or utilizing complexity leveraging.

2 Preliminaries

2.1 Notations

Let $x \leftarrow \mathcal{X}$ denote a uniformly random element drawn from the set $\mathcal{X}$. Given integers $\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}}$, let $\mathcal{C}[\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}}]$ denote the set of circuits that can be represented using ℓ_{ckt} bits, take ℓ_{inp} bits input and output ℓ_{out} bits.

2.2 Constrained Pseudorandom Functions

The notion of constrained pseudorandom functions was introduced in the concurrent works of [BW13, BGI13, KPTZ13]. Let $\mathcal{K}$ denote the key space, $\mathcal{X}$ the input domain and $\mathcal{Y}$ the range space. A PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is said to be constrained with respect to a boolean circuit family $\mathcal{F}$ if there is an additional key space $\mathcal{K}_c$, and three algorithms F.setup, F.constrain and F.eval as follows:

- F.setup(1^λ) is a PPT algorithm that takes the security parameter λ as input and outputs a key $K \in \mathcal{K}$.

- F.constrain(K, C) is a PPT algorithm that takes as input a PRF key $K \in \mathcal{K}$ and a circuit $C \in \mathcal{F}$ and outputs a constrained key $K\{C\} \in \mathcal{K}_c$.

- F.eval$(K\{C\}, x)$ is a deterministic polynomial time algorithm that takes as input a constrained key $K\{C\} \in \mathcal{K}_c$ and $x \in \mathcal{X}$ and outputs an element $y \in \mathcal{Y}$. Let $K\{C\}$ be the output of F.constrain(K, C). For correctness, we require the following:

$$F.\mathrm{eval}(K\{C\}, x) = F(K, x) \text{ if } C(x) = 1.$$

2.2.1 Security of Constrained Pseudorandom Functions

Intuitively, we require that even after obtaining several constrained keys, no polynomial time adversary can distinguish a truly random string from the PRF evaluation at a point not accepted by the queried circuits. This intuition can be formalized by the following security game between a challenger and an adversary Att.

Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a constrained PRF with respect to a circuit family $\mathcal{F}$. The security game consists of three phases.

Setup Phase The challenger chooses a random key $K \leftarrow \mathcal{K}$ and a random bit $b \leftarrow \{0, 1\}$.

Query Phase In this phase, Att is allowed to ask for the following queries:

- **Evaluation Query** Att sends $x \in \mathcal{X}$, and receives $F(K, x)$.
- **Key Query** Att sends a circuit $C \in \mathcal{F}$, and receives $F.\mathsf{constrain}(K, C)$.
- **Challenge Query** Att sends $x \in \mathcal{X}$ as a challenge query. If $b = 0$, the challenger outputs $F(K, x)$. Else, the challenger outputs a random element $y \leftarrow \mathcal{Y}$.

Guess A outputs a guess b' of b.

Let $E \subset \mathcal{X}$ be the set of evaluation queries, $L \subset \mathcal{F}$ be the set of constrained key queries and $Z \subset \mathcal{X}$ the set of challenge queries. A wins if $b = b'$ and $E \cap Z = \phi$ and for all $C \in L, z \in Z, C(z) = 0$. The advantage of Att is defined to be $\mathsf{Adv}^{\mathcal{F}}_{\mathsf{Att}}(\lambda) = \left| \Pr[\text{Att wins}] - 1/2 \right|$.

Definition 2.1. The PRF F is a secure constrained PRF with respect to $\mathcal{F}$ if for all PPT adversaries A $\mathsf{Adv}^{\mathcal{F}}_{\mathsf{Att}}(\lambda)$ is negligible in λ.

In the above definition the challenge query oracle may be queried multiple times on different points, and either all the challenge responses are correct PRF evaluations or they are all random points. As argued in [BW13], such a definition is equivalent (via a hybrid argument) to a definition where the adversary may only submit one challenge query. For our proofs, we will use the single challenge point security definition.

Another simplification that we will use in our proofs is with respect to the evaluation queries. Note that since we are considering constrained PRFs for circuits, without loss of generality, we can assume that the attacker queries for only constrained key queries. This is because any query for evaluation at input x can be replaced by a constrained key query for a circuit C_x that accepts only x.

2.3 Universal Parameters

In a recent work, Hofheinz et al. [HJK+14] introduced the notion of universal parameters. Intuitively, a universal parameters scheme provides a concise way to sample pseudorandomly from arbitrary distributions. More formally, a universal parameters scheme $\mathcal{U}$, parameterized by polynomials $\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}$ and ℓ_{out}, consists of algorithms UniversalGen and InduceGen defined below.

- $\mathsf{UniversalGen}(1^\lambda)$ takes as input the security parameter λ and outputs the universal parameters U.
- $\mathsf{InduceGen}(U, d)$ is a deterministic algorithm that takes as input the universal parameters U and a circuit d of size at most ℓ_{ckt} bits. The circuit d takes as input ℓ_{inp} bits and outputs ℓ_{out} bits. The output of InduceGen also consists of ℓ_{out} bits.

Intuitively, InduceGen is supposed to sample from d, in the sense that it outputs a value $d(z)$ for pseudorandom and hidden random coins z. However, it is nontrivial to define what it means that the random coins z are hidden, and that even multiple outputs (for adversarially and possibly even adaptively chosen circuits d) look pseudorandom.

Hofheinz et al. [HJK+14] formalize security by mandating that InduceGen is programmable in the random oracle model. In particular, there should be an efficient way to simulate U and the random oracle, such that

InduceGen outputs an externally given value that is honestly sampled from d. This programming should work even for arbitrarily many InduceGen outputs for adversarially chosen inputs d simultaneously, and it should be indistinguishable from a real execution of UniversalGen and InduceGen.

In this work, we will be using a universal parameter scheme that is even adaptively secure. In order to formally define adaptive security for universal parameters, let us first define the notion of an admissible adversary A.

An admissible adversary A is defined to be an efficient interactive Turing Machine that outputs one bit, with the following input/output behavior:

- A takes as input security parameter λ and a universal parameter U.
- A can send a random oracle query (RO, x), and receives the output of the random oracle on input x.
- A can send a message of the form (params, d) where $d \in \mathsf{C}[\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}}]$. Upon sending this message, A is required to honestly compute $p_{\mathsf{d}} = \mathsf{InduceGen}(U, d)$, making use of any additional random oracle queries, and A appends (d, p_{d}) to an auxiliary tape.

Let SimUGen and SimRO be PPT algorithms. Consider the following two experiments:

$\mathsf{Real}^{\mathsf{A}}(1^\lambda)$:

1. The random oracle RO is implemented by assigning random outputs to each unique query made to RO.
2. $U \leftarrow \mathsf{UniversalGen}^{\mathsf{RO}}(1^\lambda)$.
3. $\mathsf{A}(1^\lambda, U)$ is executed, where every message of the form (RO, x) receives the response $\mathsf{RO}(x)$.
4. Upon termination of A, the output of the experiment is the final output of the execution of A.

$\mathsf{Ideal}^{\mathsf{A}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda)$:

1. A truly random function F that maps ℓ_{ckt} bits to ℓ_{inp} bits is implemented by assigning random ℓ_{inp}-bit outputs to each unique query made to F. Throughout this experiment, a Parameters Oracle O is implemented as follows: On input d, where $d \in \mathsf{C}[\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}}]$, O outputs $d(F(d))$.
2. $(U, \tau) \leftarrow \mathsf{SimUGen}(1^\lambda)$. Here, SimUGen can make arbitrary queries to the Parameters Oracle O.
3. $\mathsf{A}(1^\lambda, U)$ and $\mathsf{SimRO}(\tau)$ begin simultaneous execution.
   - Whenever A sends a message of the form (RO, x), this is forwarded to SimRO, which produces a response to be sent back to A.
   - SimRO can make any number of queries to the Parameter Oracle O.
   - Finally, after A sends any message of the form (params, d), the auxiliary tape of A is examined until an entry of the form (d, p_{d}) is added to it. At this point, if p_{d} is not equal to $d(F(d))$, then experiment aborts, resulting in an Honest Parameter Violation.
4. Upon termination of A, the output of the experiment is the final output of the execution of A.

Definition 2.2. A universal parameters scheme U = (UniversalGen, InduceGen), parameterized by polynomials $\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}$ and ℓ_{out}, is said to be adaptively secure in the random oracle model if there exist PPT algorithms SimUGen and SimRO such that for all PPT adversaries A, the following hold:

$$\Pr[\mathsf{Ideal}^{\mathsf{A}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda) \text{ aborts }] = 0^{4}$$

and

$$\left|\Pr[\mathsf{Real}^{\mathsf{A}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}^{\mathsf{A}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

Hofheinz et al. [HJK+14] construct a universal parameters scheme that is adaptively secure in the random oracle model, assuming a secure indistinguishability obfuscator, a selectively secure puncturable PRF and an injective pseudorandom generator.

[4] The definition in [HJK+14] only requires this probability to be negligible in λ. However, the construction actually achieves zero probability of Honest Parameter Violation. Hence, for the simplicity of our proof, we will use this definition

2.4 Attribute Based Encryption

An attribute based encryption scheme ABE for a circuit family $\mathcal{F}$ with message space $\mathcal{M}$ and attribute space $\mathcal{X}$ consists of algorithms ABE.setup, ABE.keygen, ABE.enc and ABE.dec defined below.

- ABE.setup(1^λ) is a PPT algorithm that takes as input the security parameter and outputs the public key $\mathsf{pk}_{\mathsf{ABE}}$ and the master secret key $\mathsf{msk}_{\mathsf{ABE}}$.
- ABE.keygen($\mathsf{msk}_{\mathsf{ABE}}, C$) is a PPT algorithm that takes as input the master secret key $\mathsf{msk}_{\mathsf{ABE}}$, a circuit $C \in \mathcal{F}$ and outputs a secret key sk_C for circuit C.
- ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, m, x$) takes as input a public key $\mathsf{pk}_{\mathsf{ABE}}$, message $m \in \mathcal{M}$, an attribute $x \in \mathcal{X}$ and outputs a ciphertext ct. We will assume the encryption algorithm takes ℓ_{rnd} bits of randomness [5]. The notation ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, m, x; r$) is used to represent the randomness r used by ABE.enc.
- ABE.dec(sk_C, ct) takes as input secret key sk_C, ciphertext ct and outputs $y \in \mathcal{M} \cup \{\perp\}$.

Correctness For any circuit $C \in \mathcal{F}$, $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow$ ABE.setup(1^λ), message $m \in \mathcal{M}$, attribute $x \in \mathcal{X}$ such that $C(x) = 1$, we require the following:

$$\mathsf{ABE.dec}(\mathsf{ABE.keygen}(\mathsf{msk}_{\mathsf{ABE}}, C), \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, m, x)) = m.$$

For simplicity of notation, we will assume ABE.dec($\mathsf{msk}_{\mathsf{ABE}}$, ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, m, x$)) $= m$ for all messages m, attributes x [6].

2.4.1 Security

Security for an ABE scheme is defined via the following adaptive security game between a challenger and adversary Att.

1. **Setup Phase** The challenger chooses $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow$ ABE.setup(1^λ) and sends $\mathsf{pk}_{\mathsf{ABE}}$ to Att.
2. **Pre-Challenge Phase** The challenger receives multiple secret key queries. For each $C \in \mathcal{F}$ queried, it computes $\mathsf{sk}_C \leftarrow$ ABE.keygen($\mathsf{msk}_{\mathsf{ABE}}, C$) and sends sk_C to Att.
3. **Challenge** Att sends messages $m_0, m_1 \in \mathcal{M}$ and attribute $x \in \mathcal{X}$ such that $C(x) = 0$ for all circuits queried during the Pre-Challenge phase. The challenger chooses $b \leftarrow \{0,1\}$, computes ct $\leftarrow$ ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, m_b, x$) and sends ct to Att.
4. **Post-Challenge Phase** Att sends multiple secret key queries $C \in \mathcal{F}$ as in the Pre-Challenge phase, but with the added restriction that $C(x) = 0$. It receives $\mathsf{sk}_C \leftarrow$ ABE.keygen($\mathsf{msk}_{\mathsf{ABE}}, C$).
5. **Guess** Finally, Att outputs its guess b'.

Att wins the ABE security game for scheme ABE if $b = b'$. Let $\mathsf{Adv}^{\mathsf{ABE}}_{\mathsf{Att}} = \left| \Pr[\mathsf{Att\ wins}] - 1/2 \right|$.

Definition 2.3. An ABE scheme ABE = (ABE.setup, ABE.keygen, ABE.enc, ABE.dec) is said to be adaptively secure if for all PPT adversaries Att, $\mathsf{Adv}^{\mathsf{ABE}}_{\mathsf{Att}} \leq \mathrm{negl}(\lambda)$.

In a recent work, Waters [Wat14] showed a construction for an adaptively secure functional encryption scheme, using indistinguishability obfuscation. An adaptively secure functional encryption scheme implies an adaptively secure attribute based encryption scheme. Garg, Gentry, Halevi and Zhandry [GGHZ14] showed a direct construction based on multilinear encodings.

[5] This assumption can be justified by the use of an appropriate pseudorandom generator that maps ℓ_{rnd} bits to the required length.

[6] We can assume this holds true, since given $\mathsf{msk}_{\mathsf{ABE}}$, one can compute a secret key sk for circuit C_{all} that accepts all inputs, and then use sk to decrypt ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, m, x$).

3 Adaptively Secure Constrained PRF

In this section, we will describe our constrained pseudorandom function scheme for circuit class $\mathcal{F}$. Let $n = n(\lambda), \ell_{\mathrm{rnd}} = \ell_{\mathrm{rnd}}(\lambda)$ be polynomials in λ, and let ℓ_{ckt} be a polynomial (to be defined in the construction below). We will use an adaptively secure ABE scheme (ABE.setup, ABE.keygen, ABE.enc, ABE.dec) for a circuit family $\mathcal{F}$ with message and attribute space $\{0,1\}^n$. Let us assume the encryption algorithm ABE.enc uses ℓ_{rnd} bits of randomness to compute the ciphertext. We will also use an $(\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}} = n + \ell_{\mathrm{rnd}}, \ell_{\mathrm{out}} = n)$ universal parameters scheme $\mathcal{U}$ = (UniversalGen, InduceGen).

The PRF $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, along with algorithms F.setup, F.constrain and F.eval are described as follows.

F.setup(1^λ): The setup algorithm computes the universal parameters $U \leftarrow$ UniversalGen(1^λ) and the key pair ($\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}$) $\leftarrow$ ABE.setup(1^λ). In order to define F, we will first define a program Prog$\{\mathsf{pk}_{\mathsf{ABE}}, x\}$.

> Prog$\{\mathsf{pk}_{\mathsf{ABE}}, x\}$:
>
> Input : $t \in \{0,1\}^n, r \in \{0,1\}^{\ell_{\mathrm{rnd}}}$.
>
> Constants : $\mathsf{pk}_{\mathsf{ABE}}$, $x \in \{0,1\}^n$.
>
> Output ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, t, x; r$).

Let C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}, x\}$ be an $\ell_{\mathrm{ckt}} = \ell_{\mathrm{ckt}}(\lambda)$[7] bit canonical description of Prog$\{\mathsf{pk}_{\mathsf{ABE}}, x\}$, where the last n bits of the representation are x, and let C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}$ be C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}, x\}$ without the last n bits; that is, for any $x \in \{0,1\}^n$, C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}||x$ = C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}, x\}$.

The PRF key K is set to be $(U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}),$ C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\})$. To compute $F(K, x)$, first compute ct = InduceGen(U, C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}||x$) and output ABE.dec($\mathsf{msk}_{\mathsf{ABE}}$, ct).

F.constrain($K = (U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}),$ C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}), C$): The constrain algorithm first computes an ABE secret key corresponding to circuit C. It computes sk_C = ABE.keygen($\mathsf{msk}_{\mathsf{ABE}}, C$) and sets the constrained key to be $K\{C\} = (U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C),$ C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\})$.

F.eval($K\{C\} = (U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C),$ C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}), x$): The evaluation algorithm first computes the canonical circuit C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}, x\}$ = C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}||x$. Next, it computes ct = InduceGen(U, C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}, x\}$) and outputs ABE.dec($\mathsf{sk}_C$, ct).

Correctness Consider any PRF key $K = (U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}),$ C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\})$ output by F.setup(1^λ). Let $C \in \mathcal{F}$ be any circuit, and let $\mathsf{sk}_C \leftarrow$ ABE.keygen($\mathsf{msk}_{\mathsf{ABE}}, C$), $K\{C\} = (U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}),$ C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\})$. Let x be any input such that $C(x) = 1$. We require that F.eval($K\{C\}, x$) = $F(K, x)$.

$$\begin{aligned} F.\mathsf{eval}(K\{C\}, x) &= \mathsf{ABE.dec}(\mathsf{sk}_C, \mathsf{InduceGen}(U, \mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}, x\})) \\ &= \mathsf{ABE.dec}(\mathsf{msk}_{\mathsf{ABE}}, \mathsf{InduceGen}(U, \mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}, x\}))^{8} \\ &= F(K, x) \end{aligned}$$

[7] Note that the value ℓ_{ckt} required by the universal parameters scheme is determined by the ABE scheme. It depends on the size of the encryption circuit ABE.enc and the length of $\mathsf{pk}_{\mathsf{ABE}}$.

[8] Recall ABE.dec($\mathsf{msk}_{\mathsf{ABE}}$, ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, m, x$)) $= m =$ ABE.dec(sk_C, ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, m, x$)) if $C(x) = 1$.

4 Proof of Security

In this section, we will prove adaptive security for our constrained PRF in the random oracle model. We assume the random oracle outputs ℓ_{RO} bit strings as output. We will first define a sequence of hybrid experiments, and then show that if any PPT adversary Att has non-negligible advantage in one experiment, then it has non-negligible advantage in the next experiment. Game 0 is the constrained PRF adaptive security game in the random oracle model. In Game 1, the challenger simulates the universal parameters and the random oracle queries. It also implements a Parameters Oracle O which is used for this simulation. Let $q_{\rm par}$ denote the number of queries to O during the Setup, Pre-Challenge and Challenge phases. In the next game, the challenger guesses the parameters oracle query which corresponds to the challenge input. Finally, in the last game, it modifies the output of the parameter oracle on challenge input.

4.1 Sequence of Games

Game 0**:** In this experiment, the challenger chooses PRF key K. It receives random oracle queries and constrained key queries from the adversary Att. On receiving the challenge input x^*, it outputs either $F(K, x^*)$ or a truly random string. The adversary then sends post-challenge random oracle/constrained key queries, and finally outputs a bit b'.

1. **Setup Phase** Choose $U \leftarrow$ UniversalGen(1^λ), $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow$ ABE.setup(1^λ). Let C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}$ be the canonical circuit as defined in the construction.
2. **Pre Challenge Phase**

   Constrained Key Queries: For every constrained key query C, compute $\mathsf{sk}_C \leftarrow$ ABE.keygen$(\mathsf{msk}_{\mathsf{ABE}}, C)$. Send $(U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C), $ C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\})$ to Att.
   Random Oracle Queries: For each random oracle query y_i, check if y_i has already been queried. If yes, let (y_i, α_i) be the tuple corresponding to y_i. Send α_i to Att.
   If not, choose $\alpha_i \leftarrow \{0,1\}^{\ell_{\mathsf{RO}}}$, send α_i to Att and add (y_i, α_i) to table.
3. **Challenge Phase** On receiving challenge input x^*, set $d^* =$ C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$. Compute ct = InduceGen$(U, d^*)$, $t_0 =$ ABE.dec$(\mathsf{msk}_{\mathsf{ABE}}, \mathsf{ct})$. Choose $b \leftarrow \{0,1\}$. If $b = 0$, send t_0 to Att. Else send $t_1 \leftarrow \{0,1\}^n$.
4. **Post Challenge Phase** Respond to constrained key/random oracle queries as in pre-challenge phase.
5. **Guess** Att outputs a bit b'.

Game 1**:** This game is similar to the previous one, except that the universal parameters U and responses to random oracle queries are simulated. The challenger implements a Parameter Oracle O, and O is used for simulating U and the random oracle. Also, instead of using InduceGen to compute $F(K, x^*)$, the challenger uses the parameters oracle O. Please note that even though O is defined during the Setup Phase, it is used in all the remaining phases.

1. **Setup Phase** Choose $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow$ ABE.setup(1^λ). Let C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}$ be the canonical circuit as defined in the construction. Implement the Parameters Oracle O as follows:

   Implement a table T. Initially T is empty.
   For each query $d \in \mathcal{C}[\ell_{\rm ckt}, \ell_{\rm inp}, \ell_{\rm out}]$(recall $\mathcal{C}[\ell_{\rm ckt}, \ell_{\rm inp}, \ell_{\rm out}]$ is the family of circuits whose bit representation is of length $\ell_{\rm ckt}$, takes input of length $\ell_{\rm inp}$ and provides output of length $\ell_{\rm out}$),
   - If there exists an entry of the form (d, α, β), output α.
   - Else if d is of the form C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}||x$ for some x, choose $t \leftarrow \{0,1\}^n$, $r \leftarrow \{0,1\}^{\ell_{\rm rnd}}$. Output ct = ABE.enc$(\mathsf{pk}_{\mathsf{ABE}}, t, x; r)$. Add (d, ct, t) to T.
   - Else, choose $t \leftarrow \{0,1\}^{\ell_{\rm inp}}$, compute $\alpha = d(t)$. Add $(d, \alpha, ?)$ to T and output α.

   Choose $U \leftarrow$ SimUGen(1^λ).

2. **Pre Challenge Phase**

   Constrained Key Queries: For every constrained key query C, compute $\mathsf{sk}_C \leftarrow \mathsf{ABE.keygen}(\mathsf{msk}_{\mathsf{ABE}}, C)$. Send $(U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C), \mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}\})$ to Att.

   Random Oracle Queries: For each random oracle query y_i, output $\mathsf{SimRO}(y_i)$ [9].

3. **Challenge Phase** On receiving challenge input x^*, set $d^* = \mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$.
   If T does not contain an entry of the form (d^*, α, β), query the Parameters Oracle O with input d^*.
   Let (d^*, α, β) be the entry in T corresponding to d^*. Set $t_0 = \mathsf{ABE.dec}(\mathsf{msk}_{\mathsf{ABE}}, O(d^*)) = \beta$ [10].
   Choose $b \leftarrow \{0,1\}$. If $b = 0$, send t_0 to Att. Else send $t_1 \leftarrow \{0,1\}^n$.
4. **Post Challenge Phase** Respond to constrained key/random oracle queries as in pre-challenge phase.
5. **Guess** Att outputs a bit b'.

Game 2: In this game, the challenger 'guesses' the parameters oracle query which will correspond to the challenge input. The attacker wins if this guess is correct, or if the challenge input has not been queried before. Recall q_{par} denotes the number of calls to the Parameters Oracle O during the Setup, Pre-Challenge and Challenge phases.

1. **Setup Phase** Choose $i^* \leftarrow [q_{\mathrm{par}}]$.
   Choose $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.setup}(1^\lambda)$. Let $\mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}$ be the canonical circuit as defined in the construction. Implement the Parameters Oracle O as follows:

   Implement a table T. Initially T is empty.
   For each query $d \in \mathcal{C}[\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}}]$,
   - If there exists an entry of the form (d, α, β), output α.
   - Else if d is of the form $\mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x$ for some x, choose $t \leftarrow \{0,1\}^n$, $r \leftarrow \{0,1\}^l$.
     Output $\mathsf{ct} = \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, t, x; r)$.
     Add (d, ct, t) to T.
   - Else, choose $t \leftarrow \{0,1\}^{\ell_{\mathrm{inp}}}$, compute $\alpha = d(t)$. Add $(d, \alpha, ?)$ to T and output α.

   Choose $U \leftarrow \mathsf{SimUGen}(1^\lambda)$.
2. **Pre Challenge Phase**

   Constrained Key Queries: For every constrained key query C, compute $\mathsf{sk}_C \leftarrow \mathsf{ABE.keygen}(\mathsf{msk}_{\mathsf{ABE}}, C)$. Send $(U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C), \mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}\})$ to Att.

   Random Oracle Queries: For each random oracle query y_i, output $\mathsf{SimRO}(y_i)$.

3. **Challenge Phase** On receiving challenge input x^*, set $d^* = \mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$.
   If T does not contain an entry of the form (d^*, α, β), query the Parameters Oracle O with input d^*.
   If d^* was not the $(i^*)^{\mathrm{th}}$ unique query to O, abort. Choose $\gamma \leftarrow \{0,1\}$. Att wins if $\gamma = 1$.
   Else if d^* was the $(i^*)^{\mathrm{th}}$ unique query to O, let (d^*, α^*, β^*) be the corresponding entry in T. Set $t_0 = \beta$.
   Choose $b \leftarrow \{0,1\}$. If $b = 0$, send t_0 to Att. Else send $t_1 \leftarrow \{0,1\}^n$.
4. **Post Challenge Phase** Respond to constrained key/random oracle queries as in pre-challenge phase.
5. **Guess** Att outputs a bit b'.

Game 3: The only difference between this game and the previous one is in the behavior of the Parameter Oracle on the $(i^*)^{\mathrm{th}}$ query. Suppose the $(i^*)^{\mathrm{th}}$ input is of the form $d^* = \mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$. In the previous game, the entry in table T corresponding to d^* is of the form (d^*, α^*, β^*) where α^* is an encryption of β^* for attribute x^* using public key $\mathsf{pk}_{\mathsf{ABE}}$. In this game, the entry corresponding to d^* is (d^*, α^*, β^*), where α^* is the encryption of a random message for attribute x^* using $\mathsf{pk}_{\mathsf{ABE}}$.

1. **Setup Phase** Choose $i^* \leftarrow [q_{\mathrm{par}}]$.
   Choose $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.setup}(1^\lambda)$. Let $\mathsf{C\text{-}Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}$ be the canonical circuit as defined in the construction. Implement the Parameters Oracle O as follows:

[9] Recall SimRO can make polynomially many calls to Parameter Oracle O.

[10] Recall $O(d^*) = \alpha$, and $\mathsf{ABE{:}dec}(\mathsf{msk}_{\mathsf{ABE}}; \alpha) = \beta$.

Implement a table T. Initially T is empty.
For each query $d \in \mathcal{C}[\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}}]$,

- If there exists an entry of the form (d, α, β), output α.
- Else if d is of the form C-Prog$\{\mathsf{pk}_{\mathsf{ABE}}\}||x$ for some x, choose $t, \tilde{t} \leftarrow \{0,1\}^n$, $r \leftarrow \{0,1\}^{\ell_{\mathrm{rnd}}}$.
  If d is not the $(i^*)^{\mathrm{th}}$ unique query, output ct $\leftarrow$ ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, t, x; r$) and add (d, ct, t) to T.
  Else compute ct $\leftarrow$ ABE.enc($\mathsf{pk}_{\mathsf{ABE}}, \tilde{t}, x; r$) and add (d, ct, t).
- Else, choose $t \leftarrow \{0,1\}^{\ell_{\mathrm{inp}}}$, compute $\alpha = d(t)$. Add $(d, \alpha, ?)$ to T and output α.

Choose $U \leftarrow$ SimUGen(1^λ).

2. **Pre Challenge Phase**

   Constrained Key Queries: For every constrained key query C, compute $\mathsf{sk}_C \leftarrow$ ABE.keygen($\mathsf{msk}_{\mathsf{ABE}}, C$). Send $(U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C), \text{C-Prog}\{\mathsf{pk}_{\mathsf{ABE}}\})$ to Att.
   Random Oracle Queries: For each random oracle query y_i, output SimRO(y_i).

3. **Challenge Phase** On receiving challenge input x^*, set $d^* = \text{C-Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$.
   If T does not contain an entry of the form (d^*, α, β), query the Parameters Oracle O with input d^*.
   If d^* was not the $(i^*)^{\mathrm{th}}$ unique query to O, abort. Choose $\gamma \leftarrow \{0,1\}$. Att wins if $\gamma = 1$.
   Else if d^* was the $(i^*)^{\mathrm{th}}$ unique query to O, let (d^*, α^*, β^*) be the corresponding entry in T. Set $t_0 = \beta$.
   Choose $b \leftarrow \{0,1\}$. If $b = 0$, send t_0 to Att. Else send $t_1 \leftarrow \{0,1\}^n$.
4. **Post Challenge Phase** Respond to constrained key/random oracle queries as in pre-challenge phase.
5. **Guess** Att outputs a bit b'.

4.2 Analysis

For any PPT adversary Att, let $\mathsf{Adv}^i_{\mathsf{Att}}$ denote the advantage of Att in Game i.

Claim 4.1. Assuming U = (UniversalGen, InduceGen) is a secure $(\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}})$ universal parameters scheme, for any PPT adversary Att,

$$\left|\mathsf{Adv}^0_{\mathsf{Att}} - \mathsf{Adv}^1_{\mathsf{Att}}\right| \leq \mathsf{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary Att such that $\left|\mathsf{Adv}^0_{\mathsf{Att}} - \mathsf{Adv}^1_{\mathsf{Att}}\right| = \epsilon$. We will construct a PPT algorithm B such that $\left|\Pr[\mathsf{Real}^{\mathsf{B}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}^{\mathsf{B}}_{\mathsf{SimUGen};\mathsf{SimRO}}(1^\lambda) = 1]\right| = \epsilon$.

B interacts with Att and participates in either the Real or Ideal game. It receives the universal parameters U. It chooses $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow$ ABE.setup(1^λ).

During the pre-challenge phase, B receives either secret key queries or random oracle queries. On receiving secret key query for circuit C, it computes $\mathsf{sk}_C \leftarrow$ ABE.keygen($\mathsf{msk}_{\mathsf{ABE}}, C$) and sends $K\{C\} = (U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C), \text{C-Prog}\{\mathsf{pk}_{\mathsf{ABE}}\})$ to Att. On receiving random oracle query y, it forwards it to the universal parameters challenger. It receives response α, which it forwards to Att.

On receiving the challenge message x^*, it sets $d^* = \text{C-Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$, computes ct = InduceGen(U, d^*), t_0 = ABE.dec($\mathsf{msk}_{\mathsf{ABE}}$, ct). It chooses $b \leftarrow \{0,1\}$. If $b = 0$, it sends t_0, else it sends $t_1 \leftarrow \{0,1\}$.

The post challenge queries are handled similar to the pre challenge queries. Finally, Att outputs b'. If $b = b'$, B send 0 to the universal parameters challenger, indicating Real experiment. Else it sends 1.

Note that due to the honest parameter violation probability being 0, Att participates in either Game 0 or Game 1. This concludes our proof. ∎

Observation 4.1. For any adversary Att, $\mathsf{Adv}^2_{\mathsf{Att}} \geq \mathsf{Adv}^1_{\mathsf{Att}}/q_{\mathrm{par}}$.

Proof. Since the challenger's choice i^* is independent of Att, if $d = \text{C-Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$ was queried before the challenge phase, then the challenger's guess is correct with probability $1/q_{\mathrm{par}}$. ∎

Claim 4.2. Assuming ABE = (ABE.setup, ABE.keygen, ABE.enc, ABE.dec) is an adaptively secure attribute based encryption scheme, for any PPT adversary Att,

$$\left|\mathsf{Adv}^2_{\mathsf{Att}} \quad \mathsf{Adv}^3_{\mathsf{Att}}\right| \quad \mathsf{negl}(\lambda).$$

Proof. Note that the only di erence between Game 2 and Game 3 is in the implementation of Parameters Oracle O. Suppose there exists a PPT adversary Att such that $\left|\mathsf{Adv}^2_{\mathsf{Att}} \quad \mathsf{Adv}^3_{\mathsf{Att}}\right| = \epsilon$. We will construct a PPT algorithm B that interacts with Att and breaks the adaptive security of ABE scheme with advantage ϵ.

B receives $\mathsf{pk}_{\mathsf{ABE}}$ from the ABE challenger. It chooses $i \quad [q_{\mathrm{par}}]$ and computes $U \quad \mathsf{SimUGen}(1 \ )$.

Implementing the Parameters Oracle O : B must implement the Parameters Oracle. It maintains a table T which is initially empty. On receiving a query d for O, if there exists an entry of the form (d, α, β) in T, it outputs α. Else, if d is a new query, and is not of the form C-Progfpk$_{\mathsf{ABE}}$gjj$x$ for some x, it chooses $t \quad$ f0, 1g$^{\ell_{\mathrm{inp}}}$, outputs $d(t)$ and stores $(d, d(t), ?)$. Else, if d = C-Progfpk$_{\mathsf{ABE}}$gjj$x$, and d is not the $(i \ )^{\mathrm{th}}$ query, it chooses t 2 f0, 1g^n, computes ct = ABE.enc(pk$_{\mathsf{ABE}}$, t, x) and stores (d, ct, t) in T. Else, if $d \ $ = C-Progfpk$_{\mathsf{ABE}}$gjj$x \ $ is the $(i \ )^{\mathrm{th}}$ query, B chooses $t, \tilde{t} \quad$ f0, 1g^n, sends $t, \tilde{t}$ as the challenge messages and $x \ $ as the challenge attribute to the ABE challenger. It receives ct in response. B stores $(d \ , \mathsf{ct}, t)$ in T and outputs ct.

The remaining parts are identical in both Game 2 and Game 3. During the pre-challenge query phase, B receives either constrained key queries or random oracle queries. On receiving constrained key query for circuit C, it sends C to the ABE challenger as a secret key query, and receives sk$_{\mathsf{C}}$. It sends $(U, (\mathsf{pk}, \mathsf{sk}_{\mathsf{C}}),$ C-Progfpk$_{\mathsf{ABE}}$g) to Att. On receiving a random oracle query y, it computes SimRO(y), where SimRO is allowed to query the Parameters Oracle O. If B receives any constrained key query C such that $C(x \ ) = 1$ (where $d \ $ = C-Progfpk$_{\mathsf{ABE}}$gjj$x \ $ was the $(i \ )^{\mathrm{th}}$ unique query to O), then B aborts.

In the challenge phase, B receives input $x \ $. If $d \ $ = C-Progfpk$_{\mathsf{ABE}}$gjj$x \ $ was not the $(i \ )^{\mathrm{th}}$ query to O, B aborts. Else, let $(d \ , \alpha \ , \beta \ )$ be the corresponding entry in T. It chooses $b \quad$ f0, 1g. If $b = 0$, it outputs $t_0 = \beta \ $, else it outputs $t_1 \quad$ f0, 1g^n.

The post challenge phase is handled similar to the pre-challenge phase. Finally, Att outputs b^0. If $b = b^0$, Att outputs 0, indicating ct is an encryption of t. Else it outputs 1.

We will now analyse B's winning probability. Let $x \ $ was the challenge input sent by Att. Note that if B aborts, then the $(i \ )^{\mathrm{th}}$ unique query to O was not $d \ $ = C-Progfpk$_{\mathsf{ABE}}$gjj$x \ $, in which case, Att wins with probability exactly 1/2.

If $d \ $ was the $(i \ )^{\mathrm{th}}$ query and ct is an encryption of t, then this corresponds to Game 2. Else, it corresponds to Game 3. Note that Pr[B outputs 0jct $\quad$ ABE.enc(pk$_{\mathsf{ABE}}$, $t, x \ $)] = Pr[Att wins in Game 2] and Pr[B outputs 0jct $\quad$ ABE.enc(pk$_{\mathsf{ABE}}$, $\tilde{t}, x \ $)] = Pr[Att wins in Game 3]. Therefore, $\mathsf{Adv}^{\mathsf{ABE}}_{\mathsf{B}} = \epsilon$.

■

Observation 4.2. For any adversary Att, $\mathsf{Adv}^3_{\mathsf{Att}} = 0$.

Proof. Note that Att receives no information about t_0 in the pre-challenge and post challenge phases. As a result, t_0 and t_1 look identical to Att. ■

References

[BGI13] Elette Boyle, Sha Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. IACR Cryptology ePrint Archive, 2013:401, 2013.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In ASIACRYPT, pages 280{300, 2013.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, e cient traitor tracing, and more from indistinguishability obfuscation. In Proceedings of CRYPTO 2014, 2014.

[CLT13] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. Cryptology ePrint Archive, Report 2013/183, 2013.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained prfs. IACR Cryptology ePrint Archive, 2014:416, 2014.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In EUROCRYPT, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2013/128, 2013. `http://eprint.iacr.org/`.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure attribute based encryption from multilinear maps. Cryptology ePrint Archive, Report 2014/622, 2014. `http://eprint.iacr.org/`.

[GGM84] Oded Goldreich, Sha Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In FOCS, pages 464{479, 1984.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In STOC, 2013.

[HJK+14] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal parameters. Cryptology ePrint Archive, Report 2014/507, 2014. `http://eprint.iacr.org/`.

[HKW14] Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. Cryptology ePrint Archive, Report 2014/521, 2014. `http://eprint.iacr.org/`.

[Hof14] Dennis Hofheinz. Fully secure constrained pseudorandom functions using random oracles. IACR Cryptology ePrint Archive, 2014:372, 2014.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In ACM Conference on Computer and Communications Security, pages 669{684, 2013.

[LW14] Allison B. Lewko and Brent Waters. Why proving HIBE systems secure is di cult. In Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings, pages 58{76, 2014.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005, pages 84{93, 2005.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In EUROCRYPT, pages 457{473, 2005.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In STOC, pages 475{484, 2014.

[Wat14] Brent Waters. A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, Report 2014/588, 2014. `http://eprint.iacr.org/`.