# 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 $\mathcal{F}$. 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 arbitrary circuits in the random oracle model based on indistinguishability obfuscation. Our solution is constructed from two recently emerged primitives: an adaptively secure Attribute-Based Encryption (ABE) for circuits and a Universal Sampler Scheme as introduced by Hofheinz et al. Both primitives are constructible from indistinguishability obfuscation ($i\mathcal{O}$) (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 [BGI14] 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 $\mathcal{F}$. 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 [BZ14a] 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 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 [BZ14a] 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 [HKW15] 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 quasipolynomial reduction to PRGs for a larger class of "prefix-type" circuits, however, their reduction is still not 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.

In order to describe the technical problem that arises with adaptively secure constrained PRFs, say that we want to construct a bit-fixing constrained PRF $F$, i.e., one that allows for constrained keys $K_{f_{\overline{x}}}$ for "bit-matching" predicates of the form $f_{\overline{x}}(x) = 1 \Leftrightarrow \forall i : x_i = \overline{x}_i \vee \overline{x}_i = \perp$ with $\overline{x} = (\overline{x}_i)_{i=1}^n \in (\{0,1\} \cup \{\perp\})^n$. An adversary $A$ on $F$ may first ask for polynomially many constrained keys $K_{f_{\overline{x}}}$, and then gets challenged on a preimage $x^*$. The goal of a successful simulation is to be able to prepare all $K_{f_{\overline{x}}}$, but *not* to be able to compute $F(K, x^*)$.

Now if $x^* = (x_i^*)_{i=1}^n$ is known in advance, then the simulation can set up the function $F(K, \cdot)$ in an "all-

but-one" way, such that all images except $F(K, x^*)$ can be computed. For instance, the selective-security simulation from [BW13] sets up

$$F(K, x) = e(g, \ldots, g)^{\prod_{i=1}^{n} \alpha_{i,x_i}} \qquad (\text{for } K = (\alpha_{i,b})_{i,b}), \tag{1}$$

where $e$ is an $(n-1)$-linear map, and the simulation knows all $\alpha_{i,1-x_i^*}$ (while the $\alpha_{i,x_i^*}$ are only known "in the exponent," as $g^{\alpha_{i,x_i^*}}$). This setup not only allows to compute $F(K, x)$ as soon as there is an $i$ with $x_i \neq x_i^*$ (such that the corresponding $\alpha_{i,x_i} = \alpha_{i,1-x_i^*}$ is known); also, assuming a graded multilinear map, evaluation can be *delegated*. (For instance, a constrained key that allows to evaluate all inputs with $x_1 = 1$ would contain $\alpha_{1,1}$ and $g^{\alpha_{i,b}}$ for all other $i, b$.)

However, observe now what happens when $A$ chooses the challenge preimage $x^*$ only *after* asking for constrained keys. Then, the simulation may be forced to commit to the full function $F(K, \cdot)$ (information-theoretically) before even knowing where "not to be able to evaluate." For instance, for the constrained PRF from [BW13] sketched above, already a few suitably chosen constrained keys (for predicates $f_i$) fully determine $F(K, \cdot)$, while the corresponding predicates $f_i$ leave exponentially many potential challenge preimages $x^*$ uncovered. If we assume that the simulation either can or cannot evaluate $F(K, x)$ on a given preimage $x$ (at least once $F(K, \cdot)$ is fully determined), we have the following dilemma. Let $\mathcal{C}$ be set of preimages that the simulation *cannot* evaluate. If $\mathcal{C}$ is too small, then $x^* \in \mathcal{C}$ will not happen sufficiently often, so that the simulation cannot learn anything from $A$. But if $\mathcal{C}$ is too large, then the simulation will not be able to construct "sufficiently general" constrained keys for $A$ (because the corresponding predicates $f$ would evaluate to 1 on some elements of $\mathcal{C}$).[1]

This argument eliminates not only guessing $x^*$ (at least when aiming at a polynomial reduction), but also the popular class of "partitioning arguments". (Namely, while guessing $x^*$ corresponds to $|\mathcal{C}| = 1$ above, partitioning arguments consider larger sets $\mathcal{C}$. However, the argument above excludes sets $\mathcal{C}$ of *any* size for relevant classes of constraining predicates and superpolynomial preimage space.) In particular, since the selectively-secure constrained PRFs from [KPTZ13, BW13, BGI14] fulfill the assumptions of the argument, it seems hopeless to prove them fully secure, at least for standard preimage sizes.

Hence, to obtain adaptively secure constrained PRFs, we feel that leaving the standard model of computation is unavoidable, and so we resort to a solution based on random oracles.

**Our Contributions** In this paper we give new constrained PRF constructions for circuit classes in the random oracle model, based on indistinguishability obfuscation. While our construction does use heavy tools such as indistinguishability obfuscation, and our proof involves the random oracle heuristic, we wish to emphasize that our solution is currently the only known one for this problem. Moreover, recent results [AKW16] have shown that for certain problems, it is impossible to get the desired security guarantees, even assuming the existence of indistinguishability obfuscation and the random oracle heuristic.

Our solution is constructed from two recently emerged primitives: an adaptively secure Attribute-Based Encryption (ABE) [SW05] for circuits and Universal Samplers as introduced by Hofheinz et al. [HJK+16]. Both primitives are constructible from indistinguishability obfuscation ($i\mathcal{O}$) (and injective pseudorandom generators) with only polynomial loss. Waters [Wat15] recently gave an adaptively secure construction of ABE[2] based on indistinguishability obfuscation and Hofheinz et al. [HJK+16] showed how to build Universal Samplers from $i\mathcal{O}$ in the random oracle model — emphasizing that the random oracle heuristic is applied outside the obfuscated program.[3]

Before we describe our construction we briefly overview the two underlying primitives. An ABE scheme (for circuits) has four algorithms. A setup algorithm $\mathsf{ABE.setup}(1^\lambda)$ that outputs public parameters $\mathsf{pk}_{\mathsf{ABE}}$,

[1] In fact, for many classes of allowed constraining predicates, $A$ can easily ask for constrained keys that, taken together, allow to evaluate $F(K, \cdot)$ *everywhere* except on $x^*$. For instance, in our case, $A$ could ask for all keys $K_{f_i}$ with $f_i(x) = 1 \Leftrightarrow x_i = 1 - x_i^*$. Hence, in this case, the simulation *must* fail already whenever $|\mathcal{C}| \geq 2$.

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

[3] The original construction of universal samplers from [HJK+16] was only proven using complexity leveraging, and thus suffers from an exponential security loss in the reduction. However, a recent result of Garg et al. [GPSZ16] gives a construction whose proof does *not* require complexity leveraging, and thus leads to a reduction with polynomial loss.

and a master secret key $\mathsf{msk}_{\mathsf{ABE}}$. The encryption algorithm $\mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, t, x)$ takes in the public parameters, message $t$, an "attribute" string $x$ and outputs a ciphertext ct. A key generation algorithm $\mathsf{ABE.keygen}(\mathsf{msk}_{\mathsf{ABE}}, C)$ outputs a secret key given a boolean circuit $C$. Finally, the decryption algorithm $\mathsf{ABE.dec}(\mathrm{SK}, \mathrm{ct})$ 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 sampler scheme. Intuitively, a universal sampler 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 sampler scheme consists of two algorithms, $\mathsf{US.setup}$ and $\mathsf{US.sample}$. In a set-up phase, $U \leftarrow \mathsf{US.setup}(1^\lambda)$ will take as input a security parameter and output "sampler 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 $\mathsf{US.sample}(U, d)$ 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, $\mathsf{US.setup}$ 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 $\mathsf{US.sample}$ 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 samplers. 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 $U \leftarrow \mathsf{US.setup}(1^\lambda)$ and $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.setup}(1^\lambda)$. 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 $\mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, t, x; r)$. 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 $\mathrm{ct}_x = \mathsf{US.sample}(U, d_{\mathsf{pk}_{\mathsf{ABE}}, x})$. Then we compute and output $\mathsf{ABE.dec}(\mathsf{msk}_{\mathsf{ABE}}, \mathrm{ct}_x)$[4]. Essentially, the evaluation function on input $x$ first uses the universal sampler 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 $\mathsf{sk}_C = \mathsf{ABE.keygen}(\mathsf{msk}_{\mathsf{ABE}}, C)$ 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 $\mathrm{ct}_x$ from the sampler parameters $U$ as above, but then use $\mathsf{sk}_C$ 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 sampler scheme are answered by a samples oracle that generates a fresh sample every time it is called. The security definition of universal samplers 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 samples oracle to be called at most some polynomial $Q$ number of times. One of these calls must correspond to the eventual challenge input $x^*$.

We can now reduce to the security of the underlying ABE scheme. First the reduction guesses with $1/Q$ success probability which samples 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 [HKW15]. It

[4]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.

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.

**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 [Wat15] and Garg, Gentry, Halevi and Zhandry [GGHZ14] achieve adaptively secure ABE for circuits under different cryptographic assumptions. We also note that Boneh and Zhandry [BZ14a] 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.

In a recent work, Brakerski and Vaikunthanathan [BV15] showed a constrained PRF construction that is secure against single query attackers based on the LWE assumption. However, our construction and motivating applications are concerned with the case of multiple queries or collusions.

We finally mention that one application of our adaptively secure constrained PRF, adaptively secure non-interactive group key exchange [BZ14b], has already been constructed from multlinear maps [Rao14]. However, the construction from [Rao14] relies on an interactive assumption over multilinear maps (which allows an adversary to adaptively request secret keys). Our approach, although being formulated in the random oracle model, provides a conceptually different solution that avoids non-interactive complexity assumptions.

# 2 Preliminaries

## 2.1 Notations

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

## 2.2 Constrained Pseudorandom Functions

The notion of constrained pseudorandom functions was introduced in the concurrent works of [BW13, BGI14, 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 $F.\mathsf{setup}$, $F.\mathsf{constrain}$ and $F.\mathsf{eval}$ as follows:

- $F.\mathsf{setup}(1^\lambda)$ is a PPT algorithm that takes the security parameter $\lambda$ as input and outputs a key $K \in \mathcal{K}$.
- $F.\mathsf{constrain}(K, C)$ 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 $K\{C\} \in \mathcal{K}_c$.
- $F.\mathsf{eval}(K\{C\}, x)$ 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 $F.\mathsf{constrain}(K, C)$. For correctness, we require the following:

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

### 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 $\mathsf{Att}$.

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, $\mathsf{Att}$ is allowed to ask for the following queries:

- **Evaluation Query** $\mathsf{Att}$ sends $x \in \mathcal{X}$, and receives $F(K,x)$.
- **Key Query** $\mathsf{Att}$ sends a circuit $C \in \mathcal{F}$, and receives $F.\mathsf{constrain}(K,C)$.
- **Challenge Query** $\mathsf{Att}$ 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 $\mathsf{Att}$ is defined to be $\mathsf{Adv}^F_{\mathsf{Att}}(\lambda) = \left| \Pr[\mathsf{Att} \text{ wins}] - 1/2 \right|$.

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

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 Samplers

In a recent work, Hofheinz et al. [HJK+16] introduced the notion of universal samplers. Intuitively, a universal sampler scheme provides a concise way to sample pseudorandomly from arbitrary distributions. More formally, a universal sampler scheme $\mathcal{U}$, parameterized by polynomials $\ell_{\text{ckt}}, \ell_{\text{inp}}$ and $\ell_{\text{out}}$, consists of algorithms $\mathsf{US.setup}$ and $\mathsf{US.sample}$ defined below.

- $\mathsf{US.setup}(1^\lambda)$ takes as input the security parameter $\lambda$ and outputs the sampler parameters $U$.
- $\mathsf{US.sample}(U,d)$ is a deterministic algorithm that takes as input the sampler parameters $U$ and a circuit $d$ of size at most $\ell_{\text{ckt}}$ bits. The circuit $d$ takes as input $\ell_{\text{inp}}$ bits and outputs $\ell_{\text{out}}$ bits. The output of $\mathsf{US.sample}$ also consists of $\ell_{\text{out}}$ bits.

Intuitively, $\mathsf{US.sample}$ 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+16] formalize security by mandating that $\mathsf{US.sample}$ is programmable in the random oracle model. In particular, there should be an efficient way to simulate $U$ and the random oracle, such that $\mathsf{US.sample}$ outputs an externally given value that is honestly sampled from $d$. This programming should work even for arbitrarily many $\mathsf{US.sample}$ outputs for adversarially chosen inputs $d$ simultaneously, and it should be indistinguishable from a real execution of $\mathsf{US.setup}$ and $\mathsf{US.sample}$.

In this work, we will be using a universal sampler scheme that is even adaptively secure. In order to formally define adaptive security for universal samplers, let us first define the notion of an admissible adversary $\mathcal{A}$.

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

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

Let $\mathsf{SimUGen}$ and $\mathsf{SimRO}$ be PPT algorithms. Consider the following two experiments:

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

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

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

1. A truly random function $F$ that maps $\ell_{\text{ckt}}$ bits to $\ell_{\text{inp}}$ bits is implemented by assigning random $\ell_{\text{inp}}$-bit outputs to each unique query made to $F$. Throughout this experiment, a Samples Oracle $O$ is implemented as follows: On input $d$, where $d \in \mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$, $O$ outputs $d(F(d))$.
2. $(U, \tau) \leftarrow \mathsf{SimUGen}(1^\lambda)$. Here, $\mathsf{SimUGen}$ can make arbitrary queries to the Samples Oracle $O$.
3. $\mathcal{A}(1^\lambda, U)$ and $\mathsf{SimRO}(\tau)$ begin simultaneous execution.
   - Whenever $\mathcal{A}$ sends a message of the form $(\mathsf{RO}, x)$, this is forwarded to $\mathsf{SimRO}$, which produces a response to be sent back to $\mathcal{A}$.
   - $\mathsf{SimRO}$ can make any number of queries to the Samples Oracle $O$.
   - Finally, after $\mathcal{A}$ sends any message of the form $(\mathsf{params}, d)$, the auxiliary tape of $\mathcal{A}$ 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 Sample Violation*.
4. Upon termination of $\mathcal{A}$, the output of the experiment is the final output of the execution of $\mathcal{A}$.

**Definition 2.2.** A universal sampler scheme $\mathcal{U} = (\mathsf{US.setup}, \mathsf{US.sample})$, parameterized by polynomials $\ell_{\text{ckt}}, \ell_{\text{inp}}$ and $\ell_{\text{out}}$, is said to be adaptively secure in the random oracle model if there exist PPT algorithms $\mathsf{SimUGen}$ and $\mathsf{SimRO}$ such that for all PPT adversaries $\mathcal{A}$, the following hold:

$$\Pr[\mathsf{Ideal}^{\mathcal{A}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda) \text{ aborts }] = 0,^{5}$$

and

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

Hofheinz et al. [HJK+16] construct a universal sampler 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.

## 2.4 Attribute Based Encryption

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

- $\mathsf{ABE.setup}(1^\lambda)$ 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 $\mathsf{msk}_{\mathsf{ABE}}$.

[5] The definition in [HJK+16] only requires this probability to be negligible in $\lambda$. However, the construction actually achieves zero probability of Honest Sample Violation. Hence, for the simplicity of our proof, we will use this definition

- $\mathsf{ABE.keygen}(\mathsf{msk}_{\mathsf{ABE}}, C)$ 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 $\mathsf{sk}_C$ for circuit $C$.
- $\mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, m, x)$ 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 $\ell_{\text{rnd}}$ bits of randomness [6]. The notation $\mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, m, x; r)$ is used to represent the randomness $r$ used by $\mathsf{ABE.enc}$.
- $\mathsf{ABE.dec}(\mathsf{sk}_C, \text{ct})$ takes as input secret key $\mathsf{sk}_C$, ciphertext ct and outputs $y \in \mathcal{M} \cup \{\perp\}$.

**Correctness** For any circuit $C \in \mathcal{F}$, $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.setup}(1^\lambda)$, 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 $\mathsf{ABE.dec}(\mathsf{msk}_{\mathsf{ABE}}, \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, m, x)) = m$ for all messages $m$, attributes $x$. [7]

### 2.4.1 Security

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

1. **Setup Phase** The challenger chooses $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.setup}(1^\lambda)$ and sends $\mathsf{pk}_{\mathsf{ABE}}$ to $\mathsf{Att}$.
2. **Pre-Challenge Phase** The challenger receives multiple secret key queries. For each $C \in \mathcal{F}$ queried, it computes $\mathsf{sk}_C \leftarrow \mathsf{ABE.keygen}(\mathsf{msk}_{\mathsf{ABE}}, C)$ and sends $\mathsf{sk}_C$ to $\mathsf{Att}$.
3. **Challenge** $\mathsf{Att}$ 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 $\text{ct} \leftarrow \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, m_b, x)$ and sends ct to $\mathsf{Att}$.
4. **Post-Challenge Phase** $\mathsf{Att}$ 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 $\mathsf{sk}_C \leftarrow \mathsf{ABE.keygen}(\mathsf{msk}_{\mathsf{ABE}}, C)$.
5. **Guess** Finally, $\mathsf{Att}$ outputs its guess $b'$.

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

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

In a recent work, Waters [Wat15] 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.

[6] This assumption can be justified by the use of an appropriate pseudorandom generator that maps $\ell_{\text{rnd}}$ bits to the required length.

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

# 3 Adaptively Secure Constrained PRF

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

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

$F.\mathsf{setup}(1^\lambda)$ The setup algorithm computes the sampler parameters $U \leftarrow \mathsf{US.setup}(1^\lambda)$ and $(\mathsf{pk_{ABE}}, \mathsf{msk_{ABE}}) \leftarrow \mathsf{ABE.setup}(1^\lambda)$. In order to define $F$, we will first define a program $\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\}$.

$\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\}$:

Input : $t \in \{0,1\}^n, r \in \{0,1\}^{\ell_{\text{rnd}}}$.

Constants : $\mathsf{pk_{ABE}}$, $x \in \{0,1\}^n$.

Output $\mathsf{ABE.enc}(\mathsf{pk_{ABE}}, t, x; r)$.

Let $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\}$ be an $\ell_{\text{ckt}} = \ell_{\text{ckt}}(\lambda)$ bit canonical description of $\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\}$, [8] where the last $n$ bits of the representation are $x$, and let $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}\}$ be $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\}$ without the last $n$ bits; that is, $\forall x \in \{0,1\}^n$, $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}\}||x = \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\}$.

The PRF key $K$ is set to be $(U, (\mathsf{pk_{ABE}}, \mathsf{msk_{ABE}}), \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}\})$. To compute $F(K, x)$, the setup algorithm first 'samples' a ciphertext $\text{ct} = \mathsf{US.sample}(U, \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}\}||x)$ and output $\mathsf{ABE.dec}(\mathsf{msk_{ABE}}, \text{ct})$.

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

$F.\mathsf{eval}(K\{C\} = (U, (\mathsf{pk_{ABE}}, \mathsf{sk}_C), \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}\}), x)$**:** The evaluation algorithm first computes the canonical circuit $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\} = \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}\}||x$. Next, it computes $\text{ct} = \mathsf{US.sample}(U, \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\})$. Finally, it outputs $\mathsf{ABE.dec}(\mathsf{sk}_C, \text{ct})$.

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

$$
\begin{aligned}
&F.\mathsf{eval}(K\{C\}, x) \\
&= \mathsf{ABE.dec}(\mathsf{sk}_C, \mathsf{US.sample}(U, \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\})) \\
&= \mathsf{ABE.dec}(\mathsf{msk_{ABE}}, \mathsf{US.sample}(U, \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk_{ABE}}, x\}))^{9} \\
&= F(K, x)
\end{aligned}
$$

[8] Note that the value $\ell_{\text{ckt}}$ required by the universal sampler scheme is determined by the ABE scheme. It depends on the size of the encryption circuit $\mathsf{ABE.enc}$ and the length of $\mathsf{pk_{ABE}}$.

[9] Recall $\mathsf{ABE.dec}(\mathsf{msk_{ABE}}, \mathsf{ABE.enc}(\mathsf{pk_{ABE}}, m, x))$ outputs $m$, and so does $\mathsf{ABE.dec}(\mathsf{sk}_C, \mathsf{ABE.enc}(\mathsf{pk_{ABE}}, m, x))$ 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 $\ell_{\mathsf{RO}}$ bit strings as output. We will first define a sequence of hybrid experiments, and then show that if any PPT adversary $\mathsf{Att}$ has non-negligible advantage in one experiment, then it has non-negligible advantage in the next experiment. $\mathsf{Game}$ 0 is the constrained PRF adaptive security game in the random oracle model. In $\mathsf{Game}$ 1, the challenger simulates the sampler parameters and the random oracle queries. It also implements a Samples Oracle $O$ which is used for this simulation. Let $q_{\text{par}}$ denote the number of queries to $O$ during the Setup, Pre-Challenge and Challenge phases. In the next game, the challenger guesses which samples oracle query corresponds to the challenge input. Finally, in the last game, it modifies the output of the samples oracle on challenge input.

## 4.1 Sequence of Games

$\mathsf{Game}$ 0 In this experiment, the challenger chooses PRF key $K$. It receives random oracle queries and constrained key queries from the adversary $\mathsf{Att}$. 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 $U \leftarrow \mathsf{US.setup}(1^\lambda)$, $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.setup}(1^\lambda)$.
   Let $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}$ be the canonical circuit as defined in the construction.
2. **Pre Challenge Phase**
   - **Constrained Key Queries:** For every constrained key query $C$, compute $\mathsf{sk}_C \leftarrow \mathsf{ABE.keygen}(\mathsf{msk}_{\mathsf{ABE}}, C)$.
     Send $(U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C), \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\})$ to $\mathsf{Att}$.
   - **Random Oracle Queries:** For each random oracle query $y_i$, check if $y_i$ has already been queried.
     If yes, let $(y_i, \alpha_i)$ be the tuple corresponding to $y_i$. Send $\alpha_i$ to $\mathsf{Att}$.
     If not, choose $\alpha_i \leftarrow \{0,1\}^{\ell_{\mathsf{RO}}}$, send $\alpha_i$ to $\mathsf{Att}$ and add $(y_i, \alpha_i)$ to table.
3. **Challenge Phase** On receiving challenge input $x^*$, set $d^* = \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$.
   Compute $\text{ct} = \mathsf{US.sample}(U, d^*)$, $t_0 = \mathsf{ABE.dec}(\mathsf{msk}_{\mathsf{ABE}}, \text{ct})$.
   Choose $b \leftarrow \{0,1\}$. If $b = 0$, send $t_0$ to $\mathsf{Att}$. Else send $t_1 \leftarrow \{0,1\}^n$.
4. **Post Challenge Phase** Respond to constrained key and random oracle queries as in pre-challenge phase.
5. **Guess** $\mathsf{Att}$ outputs a bit $b'$.

$\mathsf{Game}$ 1 This game is similar to the previous one, except that the sampler parameters $U$ and responses to random oracle queries are simulated. The challenger implements a Samples Oracle $O$, and $O$ is used for simulating $U$ and the random oracle. Also, instead of using $\mathsf{US.sample}$ to compute $F(K, x^*)$, the challenger uses the samples 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 $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.setup}(1^\lambda)$. Let $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}$ be the canonical circuit as defined in the construction.
   Implement the Samples Oracle $O$ as follows:
   - Implement a table $T$. Initially $T$ is empty.
   - For each query $d \in \mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$(recall $\mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$ is the family of circuits whose bit representation is of length $\ell_{\text{ckt}}$, takes input of length $\ell_{\text{inp}}$ and provides output of length $\ell_{\text{out}}$) ,
     - If $\exists$ an entry of the form $(d, \alpha, \beta)$, output $\alpha$.
     - Else if $d$ is of the form $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x$, choose $t \leftarrow \{0,1\}^n$, $r \leftarrow \{0,1\}^{\ell_{\text{rnd}}}$.
       Output $\text{ct} = \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, t, x; r)$. Add $(d, \text{ct}, t)$ to $T$.
     - Else, choose $t \leftarrow \{0,1\}^{\ell_{\text{inp}}}$, compute $\alpha = d(t)$. Add $(d, \alpha, \perp)$ to $T$ and output $\alpha$.

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 $(U, (\mathsf{pk}_{\mathsf{ABE}}, \mathsf{sk}_C), \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\})$ to $\mathsf{Att}$.
   - **Random Oracle Queries:** For each random oracle query $y_i$, output $\mathsf{SimRO}(y_i)$ (recall $\mathsf{SimRO}$ can make polynomially many calls to Samples Oracle $O$).
3. **Challenge Phase** On receiving challenge input $x^*$, set $d^* = \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$.
   If $T$ does not contain an entry of the form $(d^*, \alpha, \beta)$, query the Samples Oracle $O$ with input $d^*$.
   Let $(d^*, \alpha, \beta)$ 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 $\mathsf{Att}$. Else send $t_1 \leftarrow \{0,1\}^n$.
4. **Post Challenge Phase** Respond to constrained key and random oracle queries as in pre-challenge phase.
5. **Guess** $\mathsf{Att}$ outputs a bit $b'$.

Game 2 In this game, the challenger 'guesses' the samples 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 $q_{\text{par}}$ denotes the number of calls to the Samples Oracle $O$ during the Setup, Pre-Challenge and Challenge phases.

1. **Setup Phase** Choose $i^* \leftarrow [q_{\text{par}}]$.
   Choose $(\mathsf{pk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.setup}(1^\lambda)$. Let $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}$ be the canonical circuit as defined in the construction. Implement the Samples Oracle $O$ as follows:
   - Implement a table $T$. Initially $T$ is empty.
   - For each query $d \in \mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$,
     - If there exists an entry of the form $(d, \alpha, \beta)$, output $\alpha$.
     - Else if $d$ is of the form $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x$ for some $x$, choose $t \leftarrow \{0,1\}^n$, $r \leftarrow \{0,1\}^l$.
       Output $\mathrm{ct} = \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, t, x; r)$.
       Add $(d, \mathrm{ct}, t)$ to $T$.
     - Else, choose $t \leftarrow \{0,1\}^{\ell_{\text{inp}}}$, compute $\alpha = d(t)$. Add $(d, \alpha, \perp)$ to $T$ and output $\alpha$.

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

Game 3 The only difference between this game and the previous one is in the behavior of the Sample Oracle on the $(i^*)^{\text{th}}$ query. Suppose the $(i^*)^{\text{th}}$ input is of the form $d^* = \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$. In the previous game, the entry in table $T$ corresponding to $d^*$ is of the form $(d^*, \alpha^*, \beta^*)$ where $\alpha^*$ is an encryption of $\beta^*$ for

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

attribute $x^*$ using public key $\mathsf{pk}_{\mathsf{ABE}}$. In this game, the entry corresponding to $d^*$ is $(d^*, \alpha^*, \beta^*)$, where $\alpha^*$ 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 $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}$ be the canonical circuit as defined in the construction. Implement the Samples 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 $(d, \alpha, \beta)$, output $\alpha$.
     - Else if $d$ is of the form $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x$ 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 $\mathrm{ct} \leftarrow \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, t, x; r)$, add $(d, \mathrm{ct}, t)$ to $T$. Else set $\mathrm{ct} \leftarrow \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, \tilde{t}, x; r)$, add $(d, \mathrm{ct}, t)$.
     - Else, choose $t \leftarrow \{0,1\}^{\ell_{\mathrm{inp}}}$, compute $\alpha = d(t)$. Add $(d, \alpha, \perp)$ to $T$ and output $\alpha$.

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

## 4.2 Analysis

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

**Claim 4.1.** Assuming $\mathcal{U} = (\mathsf{US.setup}, \mathsf{US.sample})$ is a secure $(\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}})$ universal sampler scheme, for any PPT adversary $\mathsf{Att}$,

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

*Proof.* Suppose there exists a PPT adversary $\mathsf{Att}$ such that $\left|\mathsf{Adv}^0_{\mathsf{Att}} - \mathsf{Adv}^1_{\mathsf{Att}}\right| = \epsilon$. For any $\mathsf{SimUGen}, SimRO$, we will construct a PPT algorithm $\mathcal{B}$ such that

$$\left|\Pr[\mathsf{Real}^{\mathcal{B}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}^{\mathcal{B}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda) = 1]\right| = \epsilon.$$

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

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

On receiving the challenge message $x^*$, it sets $d^*$ to be the circuit $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$, computes $\mathrm{ct} = \mathsf{US.sample}(U, d^*)$, $t_0 = \mathsf{ABE.dec}(\mathsf{msk}_{\mathsf{ABE}}, \mathrm{ct})$. 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, $\mathsf{Att}$ outputs $b'$. If $b = b'$, $\mathcal{B}$ send 0 to the universal sampler challenger, indicating Real experiment. Else it sends 1.

Note that due to the honest sample violation probability being 0, $\mathsf{Att}$ participates in either $\mathsf{Game}$ 0 or $\mathsf{Game}$ 1. This concludes our proof.

■

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

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

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

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

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

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

Implementing the Samples Oracle $O$ : $\mathcal{B}$ must implement the Samples 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 $(d, \alpha, \beta)$ in $T$, it outputs $\alpha$. Else, if $d$ is a new query, and is not of the form $\mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x$ for some $x$, it chooses $t \leftarrow \{0,1\}^{\ell_{\mathrm{inp}}}$, outputs $d(t)$ and stores $(d, d(t), \perp)$. Else, if $d = \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x$, and $d$ is not the $(i^*)^{\mathrm{th}}$ query, it chooses $t \in \{0,1\}^n$, computes $\mathrm{ct} = \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, t, x)$ and stores $(d, \mathrm{ct}, t)$ in $T$. Else, if $d^* = \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$ is the $(i^*)^{\mathrm{th}}$ query, $\mathcal{B}$ chooses $t, \tilde{t} \leftarrow \{0,1\}^n$, sends $t, \tilde{t}$ as the challenge messages and $x^*$ as the challenge attribute to the ABE challenger. It receives ct in response. $\mathcal{B}$ stores $(d^*, \mathrm{ct}, t)$ in $T$ and outputs ct.

The remaining parts are identical in both $\mathsf{Game}$ 2 and $\mathsf{Game}$ 3. During the pre-challenge query phase, $\mathcal{B}$ 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 $\mathsf{sk}_C$. It sends $(U, (\mathsf{pk}, \mathsf{sk}_C), \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\})$ to $\mathsf{Att}$. On receiving a random oracle query $y$, it computes $\mathsf{SimRO}(y)$, where $\mathsf{SimRO}$ is allowed to query the Samples Oracle $O$. If $\mathcal{B}$ receives any constrained key query $C$ such that $C(x^*) = 1$ (where $d^* = \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$ was the $(i^*)^{\mathrm{th}}$ unique query to $O$), then $B$ aborts.

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

The post challenge phase is handled similar to the pre-challenge phase. Finally, $\mathsf{Att}$ outputs $b'$. If $b = b'$, $\mathsf{Att}$ outputs 0, indicating ct is an encryption of $t$. Else it outputs 1.

We will now analyse $\mathcal{B}$'s winning probability. Let $x^*$ be the challenge input sent by $\mathsf{Att}$. Note that if $\mathcal{B}$ aborts, then the $(i^*)^{\mathrm{th}}$ unique query to $O$ was not $d^* = \mathcal{C}\text{-}\mathsf{Prog}\{\mathsf{pk}_{\mathsf{ABE}}\}||x^*$, in which case, $\mathsf{Att}$ 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 $\mathsf{Game}$ 2. Else, it corresponds to $\mathsf{Game}$ 3. Note that $\Pr[\mathcal{B}$ outputs $0 \text{ — } \mathrm{ct} \leftarrow \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, t, x^*)] = \Pr[\mathsf{Att}$ wins in $\mathsf{Game}$ 2] and similarly, $\Pr[\mathcal{B}$ outputs $0 \text{ — } \mathrm{ct} \leftarrow \mathsf{ABE.enc}(\mathsf{pk}_{\mathsf{ABE}}, \tilde{t}, x^*)] = \Pr[\mathsf{Att}$ wins in $\mathsf{Game}$ 3]. Therefore, $\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{B}} = \epsilon$.

■

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

*Proof.* Note that $\mathsf{Att}$ 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 $\mathsf{Att}$. ■

# References


[AKW16] Shashank Agrawal, Venkata Koppula, and Brent Waters. Impossibility of simulation secure functional encryption even with random oracles. Cryptology ePrint Archive, Report 2016/959, 2016.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 501–519, 2014.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 1–30, 2015.

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

[BZ14a] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Proceedings of CRYPTO 2014*, 2014.

[BZ14b] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *CRYPTO*, 2014.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained prfs. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 82–101, 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. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 479–499, 2013.

[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, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *FOCS*, pages 464–479, 1984.

[GPSZ16] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the subexponential barrier in obfustopia. Cryptology ePrint Archive, Report 2016/102, 2016. `http://eprint.iacr.org/2016/102`.

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

[HJK+16] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal parameters. In *ASIACRYPT*, 2016.

[HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, pages 79–102, 2015.

[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 difficult. 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.

[Rao14] Vanishree Rao. Adaptive multiparty non-interactive key exchange without setup in the standard model. Cryptology ePrint Archive, Report 2014/910, 2014. `http://eprint.iacr.org/2014/910`.

[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.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 678–697, 2015.