Adaptive Security via Deletion in Attribute-Based Encryption: Solutions from Search Assumptions in Bilinear Groups

Rishab Goyal* Jiahui Liu† Brent Waters‡

Abstract

One of the primary research challenges in Attribute-Based Encryption (ABE) is constructing and proving cryptosystems that are adaptively secure. To date the main paradigm for achieving adaptive security in ABE is dual system encryption. However, almost all such solutions in bilinear groups rely on (variants of) either the subgroup decision problem over composite order groups or the decision linear assumption. Both of these assumptions are decisional rather than search assumptions and the target of the assumption is a source or bilinear group element. This is in contrast to earlier selectively secure ABE systems which can be proven secure from either the decisional or search Bilinear Diffie-Hellman assumption. In this work we make progress on closing this gap by giving a new ABE construction for the subset functionality and prove security under the Search Bilinear Diffie-Hellman assumption.

We first provide a framework for proving adaptive security in Attribute-Based Encryption systems. We introduce a concept of ABE with deletable attributes where any party can take a ciphertext encrypted under the attribute string $x \in \{0,1\}^n$ and modify it into a ciphertext encrypted under any string $x' \in \{0,1,\perp\}^n$ where x' is derived by replacing any bits of x with $\perp$ symbols (i.e. "deleting" attributes of x). The semantics of the system are that any private key for a circuit C can be used to decrypt a ciphertext associated with x' if none of the input bits read by circuit C are $\perp$ symbols and $C(x') = 1$.

We show a pathway for combining ABE with deletable attributes with constrained pseudorandom functions to obtain adaptively secure ABE building upon the recent work of Tsabary [Tsa19]. Our new ABE system will be adaptively secure and be a ciphertext-policy ABE that supports the same functionality as the underlying constrained PRF as long as the PRF is "deletion conforming". Here we also provide a simple constrained PRF construction that gives subset functionality.

Our approach enables us to access a broader array of Attribute-Based Encryption schemes support deletion of attributes. For example, we show that both the Goyal et al. (GPSW) [GPSW06] and Boyen [Boy13] ABE schemes can trivially handle a deletion operation. And, by using a hardcore bit variant of GPSW scheme we obtain an adaptively secure ABE scheme under the *Search* Bilinear Diffie-Hellman assumption in addition to pseudo random functions in NC1. This gives the first adaptively secure ABE from a *search* assumption as all prior work relied on decision assumptions over source group elements.

1 Introduction

Attribute-Based Encryption (ABE), since its introduction by Sahai and Waters [SW05], has significantly propelled the concept of secure communication. The traditional notion of Public Key Encryption (PKE) [DH76, RSA78, GM84] was meant to enable a one-to-one private communication channel with a specific targeted

*MIT. Email: `goyal@utexas.edu`. Research supported in part by NSF CNS Award #1718161, an IBM-MIT grant, and by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR00112020023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA. Work done in part while at UT Austin supported by IBM PhD Fellowship, and at the Simons Institute for the Theory of Computing supported by Simons-Berkeley research fellowship.

†University of Texas at Austin. Email: `jiahui@cs.utexas.edu`.

‡University of Texas at Austin and NTT Research. Email: `bwaters@cs.utexas.edu`. Supported by NSF CNS-1908611, CNS-1414082, DARPA SafeWare, Packard Foundation Fellowship, and Simons Investigator Award.

user over an insecure network. ABE, on the other hand, provides a more fine-grained access control over plaintext delivery where it allows the encryptor to specify a policy f which is attached to the ciphertext. In such systems, each user decryption key is associated with an attribute string x such that it can recover the encrypted message only when $f(x) = 1$, that is when the policy f accepts the attribute x.[1]

Since its inception in 2005, the notion of Attribute-Based Encryption has received tremendous amount of attention. Initial developments in the context of provably secure ABE constructions as well as new proof techniques were driven by bilinear map-based realizations. The earliest such constructions (e.g. [SW05, GPSW06]) were proven secure under only a relaxed notion of security called *selective* security where an attacker is required to declare the descriptor f^* that will be associated with the challenge ciphertext at the beginning of the game, i.e. even *before* seeing the public parameters. This relaxation enabled the use of a so-called "partitioning" strategy for proving security. Intuitively, availability of the challenge descriptor f^* to the reduction algorithm, before it needs to sample the system public-secret parameters, enables the reduction algorithm to shape its view of the system parameters into a *partition.* Such a partitioned view of the parameters allows the reduction algorithm to generate a secret key sk_x for every attribute x as long as $f^*(x) = 0$ (that is, whenever f^* rejects the attribute x), while simultaneously being able to translate a distinguishing attack on a challenge ciphertext associated with f^* into breaking a number theoretic assumption. Unfortunately, such a partitioning strategy does not naturally translate [LW14] to the case of full or adaptive security where an attacker gets to choose the challenge function f^* after it sees the public parameters as well as makes a polynomial number of secret key queries. In this scenario the best known partitioning-style reductions will simply have to guess the function f^* to be chosen by the attacker and abort the reduction if the guess does not align with the actual choice of the attacker. This guessing approach incurs a security loss in the reduction proportional to the number of functions to choose from, and thus necessitates the use of a subexponentially secure variant of the underlying number theoretic assumption.

The shortcomings of the partitioning paradigm suggested the need for a new set of proof techniques for attaining adaptive security. The most well-known proof technique in that direction is Waters' dual system methodology [Wat09] which led to the first adaptively secure ABE scheme whose security was proven under a static assumption by Lewko et al. [LOS$^+$10]. Their approach allowed for adaptive security by moving beyond partitioning proofs.[2] Subsequently, several other works achieved adaptive security in ABE systems with various desiderata [OT10, LW11, LW12, Wee14, Att14]. One prominent trait of all these dual system solutions is that they almost exclusively rely on (variants of) the decision subgroup decision assumption or the decision linear assumption. Briefly, the decision linear assumption over a prime order bilinear group $\mathbb{G}$ states that given $g, v, w, v^a, w^b \in \mathbb{G}$ it is hard to distinguish between g^{a+b} and a random group element in $\mathbb{G}$. This is a potentially stronger assumption due to the facts that (1) it is decisional and (2) the target of the assumption g^{a+b} is in the bilinear group.[3] In contrast earlier selectively secure schemes (such as [SW05, GPSW06]) can be proven secure under the Search Bilinear Diffie-Hellman assumption which states that given g, g^a, g^b, g^c it is difficult to compute $e(g, g)^{abc}$. In our work we work toward closing this gap by constructing new ABE systems provably secure from search assumptions.

We start by building upon a recent breakthrough due to Tsabary [Tsa19] for proving adaptively secure ABE systems from the Learning with Errors (LWE) assumption [Reg05]. Until this work all prior ABE systems (that go beyond Identity-Based Encryption) from the LWE assumption (e.g. [GVW13, Boy13, BGG$^+$14, GVW15]) relied on a partitioning argument and were thus selectively secure. Tsabary's ABE construction is for the family of subset predicates where both private keys and ciphertexts are associated with subsets over $[N]$ and a secret key for subset S can decrypt a ciphertext for subset T iff $S \subseteq T$.[4] While the subset predicate class is rather limited in comparison to the functionalities mentioned earlier, the work

[1]For readers familiar with the notions of "ciphertext-policy" ABE and "key-policy" ABE, we will be using the ciphertext-policy vernacular in the sequel.

[2]Notably, earlier works of Gentry [Gen06] and Gentry-Halevi [GH09] moved beyond partitioning for IBE and Hierarchical IBE.

[3]If $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map, then we refer to elements in $\mathbb{G}$ as being in the source group or bilinear group.

[4]Tsabary actually presents their construction as realizing t-CNF for any constant t. However, this can be viewed as a special application of ABE for subsets. For this reason we will interpret their construction in terms of subset semantics for the purposes of this introduction.

is remarkable given the lack of progress towards realizing adaptive security from LWE for so many years. (It was known [BV16, GKW16] how to prove security in a slightly weaker model where the attacker sees the public parameters, but is not allowed any private key queries before committing to f^*; however, these works do not appear to give any further insight into achieving full/adaptive security.)

Tsabary's idea is to start with a *selectively* secure Attribute-Based Encryption scheme with certain special partial evaluation properties, and combine it with an *adaptively* secure Constrained Pseudorandom Function (CPRF) [BGI14, BW13, KPTZ13] that satisfies complementary "conforming" properties. Intuitively, the central idea in the work can be interpreted as a mechanism to leverage adaptive security of the CPRF for proving adaptive security of the resulting ABE system, while relying on the underlying selectively secure ABE scheme mostly for the encryption-decryption capability. Tsabary cleverly executed the above idea, and showed that combining these primitives in the right manner the resulting ABE system is adaptively secure, and the policy class it supports matches the constraint class of the underlying CPRF. For instantiating the entire framework, Tsabary derived a simple construction for constrained PRFs for subset constraints with requisite conforming properties from CPRF construction by [DKNY18], thereby giving an adaptively secure ABE scheme for subset predicates.

The framework requires the starting selectively secure ABE system to support partial ciphertext evaluation. Such a partial computation feature is not supported in many existing ABE systems, with the Boneh et al. [BGG+14] construction being the only known construction providing requisite capability. In particular, none of the bilinear map schemes such as [GPSW06], or the simpler (albeit less powerful) LWE-based ABE scheme of Boyen [Boy13] support partial evaluation.

This work. In this work, we provide a framework to both broaden and simplify the adaptively secure ABE transformation. At the core of our work is the observation that while [Tsa19] relies on the partial ciphertext evaluation framework of Boneh et al. [BGG+14], there is hardly any computation performed on the ciphertext. Concretely, the transformation the partial evaluation performed on the ciphertext exactly corresponds to the CPRF constrain operation. Now in a CPRF scheme for subset constraints over a universe of elements $[N]$, the CPRF master key msk consists of $N+1$ regular PRF keys $k_0, k_1, \ldots, k_N$. And, to evaluate the CPRF on a set $S \subseteq [N]$, the evaluator computes the following:

$$\mathsf{CPRF}(\mathsf{msk}, S) = F(k_0, S) \bigoplus_{i \in S} F(k_i, S).$$

For constraining the master key to a constraint set $T \subseteq [N]$ such that evaluation works on all input sets $S \subseteq T$, all we need to do is "delete" all the regular PRF keys k_j for which $j \notin T$ — thus no elaborate computation is required in constraining the key.

Our work builds around this key insight wherein we introduce the complementary notions of ABE with *deletable attributes* and *deletion conforming* CPRFs. At a high level, a *key-policy* ABE with deletable attributes allows encryption to a non-binary attribute string $x \in \{0, 1, \perp\}^n$, where $\perp$ represents a "deleted" attribute. The semantics of such an ABE scheme are that a user decryption key for a policy circuit C can decrypt the ciphertext associated with attribute x as long as the circuit C does not *touch* any of the deleted input wires and $C(x) = 1$.[5] Moreover, any user given just the public parameters can take a ciphertext ct for attribute string x and produce another ciphertext ct' encrypting the same message but for an attribute string x', where x' is the same as x except it can have some further attribute bits deleted (i.e., changed to $\perp$s). Armed with these abstractions we are able to compile these into an adaptively secure ciphertext-policy ABE scheme using a transformation that follows [Tsa19] in spirit.

The potential benefits of our approach are twofold. First, we show that the framework of ABE with deletable attributes encompasses a much broader range of ABE systems. Notably, this includes the early bilinear map based GPSW construction [GPSW06] as well as the LWE-based scheme of Boyen [Boy13].[6]

[5] Here by not touching an input wire, we mean that the circuit must not read/depend upon that particular input wire.

[6] We recently learned of the existence of an attack [ABN+20] on Boyen's ABE scheme. We still include the proof that it is deletable to demonstrate wider applicability of our framework, but do not claim extension of Boyen's scheme as an instantiation from LWE. To instantiate our framework under LWE, we believe that one could show the [BGG+14] scheme to be deletable.

As it turns out, showing that these schemes support attribute deletion is extremely simple — e.g., in GPSW one just has to literally "delete" ciphertext components associated with the corresponding attributes. Furthermore, following this paradigm leads to the first fully secure ABE scheme from a search problem in bilinear map setting. This is done by applying a very minor tweak to original GPSW which is to hide the message under a hardcore bit. With this tweak, we can show that the resulting scheme is adaptively secure under the *Search* Bilinear Diffie-Hellman (BDH) assumption [BF01] in addition to assuming pseudorandom functions in NC1 which is a minicrypt assumption. We also note that pseudorandom functions in NC1 are implied by the Bilinear decisional Diffie-Hellman assumption; thus we can alternatively base our security entirely on that assumption. We emphasize that all prior work on adaptively secure ABE from bilinear maps relied on *decision* assumptions over the source group.

A second (and perhaps more nuanced) benefit of trading off partial evaluation for deletion is in simplicity. Given that deletion is a more restricted operation arguing security is inherently simpler when we only perform deletion on input wires, compared to arbitrary partial circuit evaluation. We remark that there can be a tradeoff in the direction of functionality. While our construction using deletion matches the subset functionality given in [Tsa19], it is entirely possible that in the future we may find a larger class of functionalities that are supported by a partial computation framework and not by deletion. Doing so is an intriguing open question.

1.1 Technical Overview

Following the framework developed in [Tsa19], our work provides a mechanism to leverage adaptive security of a constrained PRF for upgrading the security of an ABE system from selective to adaptive. Concretely, we show that starting with a selectively secure *key-policy* ABE (KP-ABE) system that permits attribute deletion, we could pair it with an adaptively secure CPRF scheme to build an adaptively secure *ciphertext-policy* ABE (CP-ABE) system. Such a pairing mandates the CPRF scheme to satisfy certain special properties that we refer to as deletion conforming. The transformation flips the semantics of the underlying ABE system from key-policy to ciphertext-policy, and the constraint class associated with the CPRF maps directly to the predicate class for the resulting ciphertext-policy ABE system.

We now provide an overview of our framework and techniques. The overview is broken into four parts — first, we introduce the concept of attribute deletion for key-policy ABE systems; second, we define the complementary notion of deletion conforming CPRFs, and describe a simple construction for the family of subset constraints; third, we show how these aformentioned KP-ABE and CPRFs systems (for the right functionalities) be combined to construct an adaptively secure CP-ABE scheme; and lastly, we provide two concrete instantiations for KP-ABE with deletable attributes from standard assumptions.

A Key-Policy ABE with Deletable Attributes. We begin by informally introducing the concept of attribute deletion with formal definitions provided in Section 3. Recall that in the key-policy setting, the semantics of an ABE scheme are that every ciphertext ct_x is associated with an attribute string $x \in \{0,1\}^n$, while every secret decryption key sk_C is associated with a policy circuit $C : \{0,1\}^n \to \{0,1\}$. Here the functionality provided by the scheme is that decryption recovers the encrypted messages whenever the policy circuit accepts the attribute (i.e., $C(x) = 1$). An ABE system with deletable attributes provides two additional capabilities — (1) the encryption algorithm can now compute ciphertexts for non-binary attribute strings $x \in \{0,1,\perp\}^n$ as well, where the '$\perp$' symbol is interpreted as an '*unset*' attribute bit, (2) given any ciphertext ct_x, one can publicly reduce it to another ciphertext $\mathsf{ct}_{x'}$ encrypting the same message with the associated attribute string x' so long as x' can be obtained by having some attribute bits of x deleted (i.e., changed from *set* to *unset*).

Formally, such schemes have a special Delete algorithm that take as input the public parameters pp, a ciphertext ct_x, and an index set $\mathcal{I} \subseteq [n]$ and it outputs a modified ciphertext ct'. Here the set $\mathcal{I}$ denotes the indices of attribute bits that the user wants to delete, and let $\mathsf{Restrict}(x, \mathcal{I})$ denote the string x' that is obtained by deleting attribute bits of x that lie in set $\mathcal{I}$. The correctness requirement in presence of attribute deletion is expanded as follows: a secret key sk_C can decrypt a ciphertext ct_x if the circuit C does not *read* any of the *unset* input wires in attribute x, and evaluating C on x outputs 1. (For example, consider the following circuit: $C(x) = x_2 \oplus x_3$, where x_i denotes the i-th bit of x. For such a circuit C, we have that a corresponding secret key sk_C can not be used to decrypt a ciphertext ct_x whenever either $x_2/x_3 = \perp$, or $x_2 \oplus x_3 \neq 1$. That is, if $x_2 = x_3 \neq \perp$, then decryption succeeds irrespective of how other attribute bits are set.)

For security, such schemes must satisfy a special deletion indistinguishability property (in addition to the regular IND-CPA security). Briefly, deletion indistinguishability states that the distributions of ciphertexts generated by either running the encryption algorithm directly, or the encryption algorithm followed by the deletion algorithm should be computationally indistinguishable as long as they encrypt the same message and w.r.t. the same attribute string. That is, we have the following:

$$\{\mathsf{Delete}(\mathsf{pp}, \mathsf{Enc}(\mathsf{pp}, m, x), \mathcal{I})\} \approx_c \{\mathsf{Enc}(\mathsf{pp}, m, x')\}, \qquad \text{where } x' = \mathsf{Restrict}(x, \mathcal{I}).$$

Here the distributions must remain indistinguishable even if the distinguisher gets the ABE master key.

Intuitively, the goal of such a deletable key-policy ABE system is to enable arbitrary attribute deletion on ciphertexts while extending the usual policy circuit evaluation functionality over to partial/incomplete input strings. Typically, evaluating circuits on incomplete inputs is regarded as an invalid operation, but here our abstraction relies on the fact that as long as all the input wires actually used by the circuit are set (i.e., are 0/1), then we could still legally evaluate the circuit and define its output for partial inputs. As we describe later on, such a attribute deletion framework is already powerful enough for realizing adaptive security in ABE systems for subset predicates.

Deletion Conforming CPRFs. A regular constrained PRF (CPRF) [BGI14, BW13, KPTZ13] consists of a pseudorandom function (PRF) $\mathsf{CPRF}(\cdot, \cdot)$ with a key msk. The constrained property states that given master key msk, there is a way to generate a constrained key ck_f for any constraint function f such that $\mathsf{CPRF}(\mathsf{msk}, x) = \mathsf{CPRF}(\mathsf{ck}_f, x)$ whenever $f(x) = 1$. Also, the standard constrained pseudorandomness property states that an attacker cannot distinguish PRF evaluations $\mathsf{CPRF}(\mathsf{msk}, x_i)$ from uniformly random values on all inputs x_i for which $f(x_i) = 0$, even after it gets to see the constrained key ck_f. The CPRF scheme is said to be adaptively secure if the adversary can choose the challenge constraint function f after making polynomially many PRF evaluation queries. In this work, similar to [Tsa19], we instead require the CPRF to achieve adaptive key simulation security. Key simulation property states that there exists an efficient key simulation algorithm KeySim such that an attacker cannot distinguish a simulated key $\widetilde{\mathsf{ck}}_f \leftarrow \mathsf{KeySim}(f)$ from a honestly constrained key ck_f for any adaptively chosen challenge constraint f as long as all its PRF evaluation queries x_i are not satisfied by the constraint f, i.e. $f(x_i)$ for all evaluation queries x_i. Tsabary provided a CPRF construction for subset constraints which satisfies both adaptive pseudorandomness and key simulation security properties.[7] As a side contribution, in the main body we show that the standard constrained pseudorandomness already implies key simulation security.

Inspired by our deletable attribute framework for ABE systems, we define the notion of deletion conforming CPRFs, or DCCPRF in short. Intuitively, it states a CPRF system is deletion conforming if any constrained key ck_f in such a scheme can be *deterministically* computed by simply "deleting" specific bits of the master key msk (i.e., replacing some bits of the master key with the special $\perp$ symbol). Additionally, it must be the case that the PRF evaluation algorithm for any given input x be simplified into a circuit C_x such that evaluating C_x on a master key msk and a constrained key ck_f matches on all valid inputs (i.e., all x such that $f(x) = 1$). Here evaluating the circuit on a constrained key is defined similar to that for partial inputs as in the deletable KP-ABE setting, since a constrained key could have partially *unset* key bits (i.e., contain $\perp$ symbols). All these notions are formally defined later in Section 4.

As mentioned previously, here we construct a deletion conforming CPRF for subset constraints. A subset constraint family is defined over a universe of elements $[N] := \{1, \ldots, N\}$, where input to the PRF is a set $S \subseteq [N]$ (which could be represented as an N-bit binary string), and each constraint function is associated with another set $T \subseteq [N]$ such that an input set S satisfies the constraint iff $S \subseteq T$. A CPRF scheme for

[7] As we pointed out before, Tsabary gives a construction for t-CNF (for constant t) constraint functions, but this can be viewed as a special case of subset constraints.

such a constraint family can be built using a combinatorial strategy as introduced in [DKNY18], where the CPRF master key msk consists of $N+1$ regular PRF keys $k_0, k_1, \ldots, k_N$, and the CPRF output on a set S is computed by first selecting all PRF keys k_i such that the associated index $i \in S$, which is then followed by independent PRF evaluation under all selected keys and finally XORing all the evaluations together.[8] Concretely, the evaluator proceeds as follows:

$$\mathsf{CPRF}(\mathsf{msk}, S) = F(k_0, S) \bigoplus_{i \in S} F(k_i, S).$$

Note that a constrained key for a subset T can be simply set as the corresponding subset of underlying PRF keys, that is $\mathsf{ck}_T = \{k_0\} \cup \{k_i\}_{i \in T}$. Observe that for every input set S satisfying the constraint set T (i.e., $S \subseteq T$), the constrained key ck_T already contains the necessary PRF keys for performing the PRF evaluation, thus correctness of evaluation for constrained keys follows immediately. Next, the proof of adaptive constrained pseudorandomness security follows from a simple observation that a reduction algorithm can simply guess an index $i \in \{0, 1, \ldots, N\}$ which is meant to denote the index of the regular PRF key that is not required for answering the constrained key query, but is needed for evaluating the CPRF on the challenge input. Since N is a polynomial, thus such a reduction strategy gives a proof of adaptive security with just polynomial security loss.

Finally, to complete our overview of CPRFs, we just need to argue that our CPRF construction satisfies the desired deletion conforming properties. This mostly follows by inspection of our aformentioned construction thereby aligning with our goal of simplicity and precision. Concretely, note that a constrained key ck_T can simply be deterministically obtained by "deleting" all the regular PRF keys k_i for which $i \notin T$. Also, for any input set S, the corresponding CPRF evaluation circuit can be described as: first, it reads the input wires (encoding the appropriate PRF key) corresponding to set S, and then evaluates the circuit $F(\cdot, S)$ on each block of input wires, which is finally followed up by XORing them together. Observe that since this circuit does not even read/touch the input wires corresponding to PRF keys k_i for which $i \notin S$, thus evaluating the circuit on a master key msk and constrained key ck_T is well-defined and gives the same output whenever $S \subseteq T$. Thus, this completes the proof sketch for the above CPRF to be deletion conforming. More details on our construction are provided in Section 7.

Building adaptively secure Ciphertext-Policy ABE. Moving on to our main transformation, our approach is to decouple the adaptivity and functionality (delivering the message to users) requirements of a CP-ABE scheme, and deal with them separately. Following Tsabary's paradigm, we rely on our deletion conforming CPRFs for enabling the reduction algorithm to be able to answer the adaptive key queries, while still using the selectively secure deletable KP-ABE system for guaranteeing that the message is hidden. At a very high level, the idea is to handle the adaptivity problem outside of the underlying KP-ABE system, while using its attribute deletion capabilities to compute the CP-ABE challenge ciphertext from a KP-ABE challenge ciphertext that was selectively obtained. Below we sketch our transformation.

The public parameters of the CP-ABE system contains the deletable (KP-)ABE parameters $\mathtt{del}.\mathsf{pp}$, while the master secret key consists of a DCCPRF master key $\mathtt{prf}.\mathsf{msk}$ as well as the deletable ABE master key $\mathtt{del}.\mathsf{msk}$. Recall that in a CP-ABE system, each secret key is associated with an attribute string $x \in \{0,1\}^N$. To sample a secret key for attribute x, the key generator first computes a tag value t as the CPRF evaluation with input x, i.e. $t = \mathsf{CPRF}(\mathtt{prf}.\mathsf{msk}, x)$. Let C_x denote the simplified explicit circuit that performs the CPRF evaluation on input x, i.e. $C_x(\mathsf{key}) = \mathsf{CPRF}(\mathsf{key}, x)$. The key generator then creates a policy circuit $f_{x,t}$, given the tag value t and circuit description C_x, as:

$$f_{x,t}(z) = \begin{cases} 1 & \text{if } C_x(z) \neq t, \\ 0 & \text{otherwise.} \end{cases}$$

The (CP-ABE) secret key sk_x for attribute x now corresponds to a (KP-ABE) secret key for the above policy circuit, i.e. $\mathsf{sk}_x = \mathtt{del}.\mathsf{sk}_{f_{x,t}}$. To encrypt a message m under a policy circuit g, the encryptor first samples

[8]In the construction the master key consists of $N+1$ PRF keys instead of N keys just so that pseudorandomness holds for empty set as well.

a *simulated* constrained key $\widetilde{\texttt{prf.sk}_g}$ with g being used as the constraint function, and then it computes the ciphertext as an KP-ABE encryption of message m with attribute string set as sk_g. The resulting decryption algorithm is exactly the decryption algorithm of the underlying KP-ABE scheme.

First, note that, by the deletion conforming properties, evaluating C_x is well-defined and accurately matches the corresponding CPRF output on every accepting constrained key. Thus with this observation we get that correctness of the above construction follows from the fact that whenever $g(x) = 1$, then $f_{x,t}(\widetilde{\texttt{prf.sk}_g}) = 1$ with all but negligible probability, since $C_x(\widetilde{\texttt{prf.sk}_g}) = t = C_x(\texttt{prf.msk})$ happens only with negligible probability by pseudorandomness of the underlying CPRF.

Next we describe the intuition behind the proof of adaptive security. Note that initially the challenge ciphertext for policy g^* with message m is computed as KP-ABE encryption of message m with a simulated CPRF constrained key $\widetilde{\texttt{prf.sk}_{g^*}}$ as the attribute string. As a first step, we instead switch this to be a honestly constrained key $\texttt{prf.sk}_{g^*} = \mathsf{Constrain}(\texttt{prf.msk}, g^*)$. Since the CPRF satisfies the adaptive key simulation property, thus this change will be indistinguishable. Note that it is important that the CPRF is adaptively secure for this reduction to work since to answer the pre-challenge key queries, the reduction algorithm needs to query for the respective CPRF evaluations. Next, by the deletion conforming property of the constrained PRF scheme, we have that the constrained key $\texttt{prf.sk}_{g^*}$ can be computed by simply deleting certain specific key bits of the master key $\texttt{prf.msk}$. Let $\mathcal{I}_{g^*}$ denote such a set of indices, i.e. $\texttt{prf.sk}_{g^*} = \mathsf{Restrict}(\texttt{prf.msk}, \mathcal{I}_{g^*})$. By relying on the deletion indistinguishability property of the KP-ABE scheme, we get that the challenge ciphertext can instead be computed as first encrypting the message m under attribute string $\texttt{prf.msk}$, and then deleting the attribute bits as specified by set $\mathcal{I}_{g^*}$ by running the KP-ABE deletion algorithm. Finally, since the attribute string $\texttt{prf.msk}$ is sampled at the beginning of the security game, thus $\texttt{prf.msk}$ can be selectively specified to the KP-ABE challenger thereby allowing us to argue that the message is also hidden. Our construction and its proof is formally provided in Section 6.

Perfect correctness? Although at first glance it may seem that imperfect correctness is an inherent and unavoidable feature of the above framework, we later show in Appendix B that this is not the case where we provide an alternate construction which is perfectly correct. Very briefly, our idea is to have two deletable ABE sub-systems working in parallel, instead of just one, where both the ciphertexts and secret keys contain two copies (one under each ABE sub-system). The only difference is that while sampling a secret key under both the systems independently, the key generator uses two distinct tag values, where one of the tag values is computed as is now, whereas the other tag value will be its complement. Such a trick gets around the imperfect correctness problem since it can never happen that $C_x(\widetilde{\texttt{prf.sk}_g}) = t_0$ as well as $C_x(\widetilde{\texttt{prf.sk}_g}) = t_1$ where t_0, t_1 are the complementary tag pairs. It turns out that the proof of adaptive security now is more involved, as we need to first use the existing proof structure to erase the information about the challenge message from the first deletable ABE sub-ciphertext, then we would have to undo correlations created between parts of the challenge ciphertext and secret keys, and finally use a similar proof structure to erase the information about the challenge message from the second deletable ABE sub-ciphertext as well.

Another interpretation. Abstractly, the deletion paradigm described above can be interpreted as a mechanism to selectively activate the trapdoors embedded inside the secret keys such that whenever trapdoor is activated then the challenger can simulate the secret keys for all possible attributes. The property such simulated keys satisfy is that they are indistinguishable from honestly sampled secret keys as long as the challenge policy does not accept the corresponding key attribute. On a more intuitive level, one could also observe some similarities between the above framework and the Dual System methodology [Wat09], where switching from a simulated CPRF key to an honestly constrained CPRF key could be comparable to moving from a *normal* to a *semi-functional* ciphertext, and the secret keys are already sampled in the semi-functional mode.

Deleting attributes in [GPSW06, Boy13]. Finally, we show that existing ABE schemes by Goyal et al. (GPSW) [GPSW06] from bilinear maps, and by Boyen [Boy13] from LWE[9] already lie in the class of

[9]We want to remind the reader the existence of an attack [ABN+20] on Boyen's ABE scheme. Deletions in Boyen's scheme are merely provided for illustrative purposes in Appendix D.

ABE schemes with deletable attributes, thereby displaying the generality of our framework. Below we give an overview of our deletion algorithms. More details are provided later in Section 8, where we also show that a KP-ABE scheme with deletable attributes for monotonic access structures can be generically upgraded to non-monotonic log-depth circuits (i.e., $\mathbf{NC}^1$).

Deletions in [GPSW06]. First, we look at the bilinear map based ABE construction by GPSW. They proposed a KP-ABE scheme for monotone access structures and proved its security under the Decisional Bilinear Diffie-Hellman (DBDH) assumption that can also be readily adapted to a scheme provably secure under the Search Bilinear Diffie-Hellman assumption. The public parameters in the GPSW scheme contain n group elements in the base group $\{T_i\}_{i\in[n]}$ and one group element in the target group K, where n denotes the length of the attributes. A ciphertext encrypting a message m under an attribute $x \in \{0,1\}^n$ is of the following form:

$$\mathsf{ct} = (m \cdot K^s, \{T_i^s\}_{i\in[n]:x_i=1}),$$

where s is a random exponent. Basically the term T_i^s encodes the i-th bit of the attribute, and during decryption the algorithm pairs the ciphertext component T_i^s with a corresponding key component (iff the policy circuit reads the i-th input wire) and performs a polynomial interpolation in the exponent to reconstruct the masking term K^s.

Our observation is that to delete an attribute bit, say j, one could simply drop the term T_j^s from ciphertext (if it exists). Multiple attribute bits could be deleted analogously. As long as the policy circuit does not read the deleted input wire, the correctness for deleted ciphertexts follows immediately from the correctness of GPSW scheme itself. Similarly, to encrypt a message m under a non-binary attribute string $x \in \{0,1,\perp\}^n$, we simply treat each $\perp$ symbol as a 0 bit, and therefore do not encode it in the ciphertext. Clearly, the distributions of freshly computed ciphertexts and deleted ciphertexts (encrypting the same message m and attribute x) are identical, thus deletion indistinguishability for GPSW is merely a statistical property. Combining this with the fact that GPSW provides selective IND-CPA security, we obtain that GPSW augmented with the deletion procedure is KP-ABE scheme with deletable attributes. Later in Appendix C we also describe a hardcore bit variant of the above scheme whose security relies on the Computational Bilinear Diffie-Hellman (CBDH) assumption.

Deletions in [Boy13]. Next, we look at the LWE-based ABE construction by Boyen. Boyen's scheme is also for monotone access structures and its security relies on the LWE assumption. The public parameters in Boyen's scheme consist $\ell + 1$ matrices of appropriate dimensions $(\mathbf{A}_0, \{\mathbf{A}_i\}_{i\in[\ell]})$ and a vector $\mathbf{u}$, where ℓ denotes the length of the attributes. Now a ciphertext ct encrypting a message bit msg under an attribute $x \in \{0,1\}^\ell$ is of the following form $\mathsf{ct} = (c_0, \mathbf{c}_{1,0}, \mathbf{c}_{1,1}, \ldots, \mathbf{c}_{1,\ell})$, where

$$c_0 = \mathbf{s}^\top \cdot \mathbf{u} + \nu_0 + \lfloor \frac{q}{2} \rfloor \cdot \mathsf{msg},$$

$$\forall i \in [0,\ell], \quad \mathbf{c}_{1,i} = \begin{cases} \mathbf{s}^\top \cdot \mathbf{A}_i + \boldsymbol{\nu}_{1,i} & \text{if } i = 0 \text{ or } x_i = 1, \\ \boldsymbol{\nu}_{1,i} & \text{otherwise.} \end{cases}$$

and $\mathbf{s}$ is a random secret vector, and $\nu_0, \{\boldsymbol{\nu}_{1,i}\}_{i\in[\ell]}$ are sampled i.i.d. according to the LWE noise distribution. Here the vector $\mathbf{c}_{1,i}$ encodes the i-th bit of the attribute, and during decryption the algorithm combines the ciphertext component $\mathbf{c}_{1,i}$ with a corresponding key component (iff the policy circuit reads the i-th input wire).

For deleting attributes in Boyen's scheme, instead of dropping the respective ciphertext component, we replace with freshly sampled noise. Concretely, to delete an attribute bit, say j, we replace the vector $\mathbf{c}_{1,j}$ in the ciphertext with a freshly sampled noise vector $\boldsymbol{\nu}'_{1,i}$. Multiple attribute bits could be deleted analogously.[10] And as for our augmented GPSW scheme, during encryption we treat each $\perp$ symbol as a 0 bit, and the correctness and deletion indistinguishability of the resultant follows either immediately from Boyen's scheme or by inspection.

[10]We could also drop the deleted ciphertext components instead of replacing them with LWE noise, however for ensuring consistency with Boyen's scheme we keep it this way.

Recent Independent Work Recently, Katsumata, Nishimaki, Yamada, and Yamakawa (KNYY) [KNYY20] gave an exciting construction showing how to expand the framework of [Tsa19] to encompass an inner product encryption and Fuzzy IBE functionality within the LWE setting. An important insight was showing that a specific cryptosystem could relax the earlier conforming property to just functional equivalence and thus leverage a particular constrained PRF of [DKN+20] to achieve greater functionality.

In contrast, our work shows how to relax the conforming property to deletion so that it is realizable in a broader setting that includes bilinear maps. But we show that is still sufficient to maintain the t-CNF functionality. KNYY show that in the LWE setting one can strengthen the framework to handle a broader class of LWE specific constrained PRFs. The works were performed independently.

Comparing techniques with [Tsa19]. We conclude by giving some further technical comparisons between our framework and the earlier work of Tsabary [Tsa19] that we build upon. Our work follows a similar pathway which is to leverage adaptive security of constrained PRFs (with special properties) inside a key-policy ABE scheme (with special properties) to achieve an adaptively secure ciphertext-policy ABE scheme, but differences lie in the flavour of these special properties required from the underlying constrained PRF and key-policy ABE systems. Tsabary started with the LWE-based ABE construction of Boneh et al. [BGG+14], and using the homomorphic properties of the underlying ABE scheme, Tsabary developed a framework for partial ciphertext evaluation and a circuit splitting/composition abstraction, wherein the ABE scheme allows a user to encrypt messages under *partially evaluated* attributes such that they are indistinguishable from *partially evaluated* ciphertexts encrypting same message under the original (unevaluated) attribute. Concretely, [Tsa19] relies on the fact that for any attribute x and circuit C, one could compute ciphertexts of the form: $\mathsf{ct}_0 = \mathsf{Enc}(\mathsf{pp}, m, x)$, $\mathsf{ct}_1 = \mathsf{Enc}(\mathsf{pp}, m, C(x))$ such that given a secret key $\mathsf{sk}_{\widetilde{C}}$ for some circuit $\widetilde{C}$ s.t. $\widetilde{C}(x) = 1$, a user can not only decrypt ciphertexts of the form ct_0, but it can also decrypt ciphertexts of the form ct_1 as long as there exists another circuit C' with the semantics that $\widetilde{C}(\cdot) = C'(C(\cdot))$ that the decryptor knows. Here the equality between the circuit $\widetilde{C}$ and the composition of C, C' mandates the resultant 'gate-by-gate' circuit descriptions must be *identical.* With such an ABE scheme with these special properties as the centerpiece, [Tsa19] built a constrained PRF that conforms with the necessary circuit splitting/composition semantics. Very briefly, [Tsa19] required that the PRF evaluation circuit with the input hardwired can be split into two sub-circuits such that one of those sub-circuits can be used during generating the CP-ABE ciphertext. Combining all these things in an extremely careful manner gives the desired result of an adaptively secure CP-ABE scheme for subset policies.

Our approach, on the other hand, is to skip the entire partial evaluation and circuit splitting/composition framework, and instead go with a simpler abstraction of input deletion while also demanding (as part of our definitional framework) an explicit descriptions for all the circuits used throughout the analysis.

2 Preliminaries

Notation. Let PPT denote probabilistic polynomial-time. We denote the set of all positive integers upto n as $[n] := \{1, \ldots, n\}$. Also, we use $[0, n]$ to denote the set of all non-negative integers upto n, i.e. $[0, n] := \{0\} \cup [n]$. Throughout this paper, unless specified, all polynomials we consider are positive polynomials. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. The distribution $\mathcal{D}^n$ is used to represent a distribution over vectors of n components, where each component is drawn independently from the distribution $\mathcal{D}$.

For any $n \in \mathbb{N}$, string $x \in \{0, 1, \perp\}^n$ and index set $\mathcal{I} \subseteq [n]$, let $\mathsf{Restrict}(x, \mathcal{I})$ denote the string $\widetilde{x} \in \{0, 1, \perp\}^n$ such that

$$\forall i \in [n], \quad \widetilde{x}_i = \begin{cases} x_i & \text{if } j \notin \mathcal{I}, \\ \perp & \text{otherwise.} \end{cases}$$

where x_i and $\widetilde{x}_i$ denote the ith elements of strings x and $\widetilde{x}$, respectively. For any string $x \in \{0,1,\perp\}^n$, let $\mathsf{BotSet}(x)$ denote the subset of indices in $[n]$ such that for every $i \in \mathsf{BotSet}(x)$, $x_i = \perp$ and for every $i \notin \mathsf{BotSet}(x)$, $x_i \in \{0,1\}$. Formally, $\mathsf{BotSet}(x) := \{i \in [n] : x_i = \perp\}$.

Circuit notation. Also, throughout the paper we use the circuit model of computation. Consider any circuit $C : \{0,1\}^n \to \{0,1\}$ that takes n-bits of input and outputs a single bit. For any circuit C, we define $\mathsf{Unsupported}(C) \subseteq [n]$ to be set of indices $i \in [n]$ such that the circuit C does not use on the ith input wire (i.e., C does not read the ith input bit).[11]

Lastly, we use CEval to denote an "expanded" notion of circuit evaluation. The algorithm CEval takes as input a circuit $C : \{0,1\}^n \to \{0,1\}^m$, and a string $x \in \{0,1,\perp\}^n$, and it first checks that $\mathsf{BotSet}(x) \subseteq \mathsf{Unsupported}(C)$. If the check fails, it outputs the all-zeros string 0^m; otherwise it evaluates the circuit C on string x, and outputs the same result as the circuit which is $C(x)$. Note that evaluating the circuit C on string x (that could possibly contain non-binary input bits) is well-defined in the last step, since the evaluator CEval only runs the circuit C after its checks that $\mathsf{BotSet}(x) \subseteq \mathsf{Unsupported}(C)$, and thus we know that if the check succeeds then all the input wires/bits read by circuit C are defined and not set as $\perp$. Formally, CEval can be defined as:

$$\mathsf{CEval}(C, x) = \begin{cases} C(x) & \text{if } \mathsf{BotSet}(x) \subseteq \mathsf{Unsupported}(C), \\ 0^m & \text{otherwise.} \end{cases}$$

2.1 Pseudorandom Functions

A pseudorandom function (PRF) consists a pair of algorithms Setup and Eval with the following syntax:

$\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{sk}$. The setup algorithm takes as input the security parameter λ and input length parameter n, and outputs a secret key sk.

$\mathsf{Eval}(\mathsf{sk}, x) \to y$. The evaluation algorithm, on input the secret key sk and string $x \in \{0,1\}^n$, outputs a bit string $y \in \{0,1\}^m$. Here $m = m(\lambda)$ denotes the output length of the PRF.

Definition 2.1 (Pseudorandomness)**.** A PRF scheme $\mathsf{PRF} = (\mathsf{Setup}, \mathsf{Eval})$ is said to be secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, n \in \mathbb{N}$, the following holds:

$$\Pr\left[\mathcal{A}^{\mathsf{Eval}(\mathsf{sk},\cdot)}(r_b) = b : \begin{array}{c} \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^n),\ b \leftarrow \{0,1\} \\ x^* \leftarrow \mathcal{A}^{\mathsf{Eval}(\mathsf{sk},\cdot)}(1^\lambda, 1^n) \\ r_0 \leftarrow \{0,1\}^m,\ r_1 = \mathsf{Eval}(\mathsf{sk}, x^*) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where $\mathcal{A}$ must not query the challenge input x^* to the evaluation oracle $\mathsf{Eval}(\mathsf{sk}, \cdot)$.

3 Key Policy Attribute-Based Encryption with Deletable Attributes

In this section, we introduce the notion of Key Policy Attribute-Based Encryption (KP-ABE) with deletable attributes. First, we provide the syntax, and later describe our definitions for KP-ABE with deletable attributes.

[11]Note that our definition of the *unsupported indices* for a circuit C is very restrictive. Concretely, we say that an index $i \in \mathsf{Unsupported}(C)$ *iff* as per the circuit description of C the ith input wire is unused/untouched. For instance, consider two circuits $C, \widetilde{C}$ which takes length 2-bit strings as inputs: $C(x) = (x_1 \vee \neg x_1) \wedge x_2$ and $\widetilde{C}(x) = x_2$. Here $\mathsf{Unsupported}(C) = \emptyset$ and $\mathsf{Unsupported}(\widetilde{C}) = \{1\}$, i.e. circuits $C, \widetilde{C}$ have different unsupported indices even though they are functionally identical. This is because as per the circuit description of C, it does use both input wires/bits; whereas $\widetilde{C}$ ignores the first input wire/bit.

Syntax. A key-policy attribute based encryption (KP-ABE) scheme with deletable attributes for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ and message space $\mathcal{M}$ consists of the following PPT algorithms:

$\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{pp}, \mathsf{msk})$. On input the security parameter λ and attribute length n, the setup algorithm outputs a set of public parameters pp, and master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. On input the master secret key msk and a circuit $f \in \mathcal{C}_n$, the key generation algorithm outputs a predicate key sk_f.

$\mathsf{Enc}(\mathsf{pp}, x, m) \to \mathsf{ct}$. On input the public parameters pp, an attribute string $x \in \{0, 1, \perp\}^n$, and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct. Note that here the attribute string x is possibly a non-binary string as it could contain $\perp$ symbols.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to m/\mathtt{fail}$. On input a secret key sk_f and a ciphertext ct, the decryption algorithm either outputs a message m or a special string $\mathtt{fail}$ (to denote decryption failure).

$\mathsf{Delete}(\mathsf{pp}, \mathsf{ct}, \mathcal{I}) \to \mathsf{ct}'$. On input of the public parameters pp, a ciphertext ct and a set of indices $\mathcal{I} \subseteq [n]$, the deletion algorithm outputs a modified ciphertext ct'.

We require such an ABE scheme to satisfy the following properties.

Correctness. Intutively, it says that the above scheme is correct if decrypting a ciphertext, which was either directly computed using the encryption algorithm or generated by the ciphertext deletion algorithm, outputs the correct message as long as the policy circuit accepts the attribute associated with the ciphertext.

Formally, an KP-ABE scheme with deletable attributes is said to be correct if for all $\lambda, n \in \mathbb{N}$, $f \in \mathcal{C}_n$, $m \in \mathcal{M}$, $x_0 \in \{0, 1, \perp\}^n$ and a sequence of indices sets $\mathcal{I}_1, \mathcal{I}_2 \cdots, \mathcal{I}_k \subseteq [n]$, for any $k \geq 0$, the following holds:

$$\mathsf{CEval}(f, x_k) = 1 \implies \Pr\left[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_k) = m : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pp}, x_0, m) \\ (\forall i \in [k])\ \mathsf{ct}_i \leftarrow \mathsf{Delete}(\mathsf{pp}, \mathsf{ct}_{i-1}, \mathcal{I}_i) \end{array}\right] = 1,$$

where x_k is defined by the following sequence of operations: $x_i \leftarrow \mathsf{Restrict}(x_{i-1}, \mathcal{I}_i)$ for all $i \in [k]$.

Security. For security, we have two requirements. First, we require the scheme to provide standard semantic security as for standard ABE schemes. Here we consider both selective and adaptive IND-CPA security definitions. Second, we introduce a notion of indistinguishability for ciphertexts with deleted attributes, in which the adversary cannot distinguish between a ciphertext modified by the Delete algorithm and a ciphertext directly encrypted from the same message with respect to the same attribute string after deletion. Formally, they are defined as below.

Definition 3.1 (Adaptive IND-CPA Security)**.** A KP-ABE scheme is adaptively secure if for every stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, n \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ ((m_0, m_1), x^* \in \{0,1\}^n) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, 1^n, \mathsf{pp}) \\ b \leftarrow \{0,1\}; \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x^*, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the adversary $\mathcal{A}$ is admissible as long as every secret key query f made by $\mathcal{A}$ to the oracle $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ satisfies the condition that $f(x^*) = 0$. Here x^* is the challenge attribute chosen by $\mathcal{A}$. Note that the adversary must choose x^* as a binary string, that is it must not contain any $\perp$ symbols.[12]

[12]Note that since x^* does not contain $\perp$ symbols, thus $f(x^*)$ is always well-defined and we do not need define the admissibility constraint as $\mathsf{CEval}(f, x^*) = 0$ instead.

Definition 3.2 (Selective IND-CPA Security)**.** A KP-ABE scheme is said to be *selectively* secure if in the above security game (see Definition 3.1), the adversary must instead declare the challenge attribute $x^* \in \{0,1\}^n$ at the beginning of the game, that is even before it receives the public paramters pp from the challenger.

Definition 3.3 (Deletion Indistinguishability)**.** A KP-ABE scheme with deletable atrributes satisfies deletion indistinguishability property if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda, n \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}(\mathsf{ct}_b) = b : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n),\ b \leftarrow \{0,1\} \\ (m, x \in \{0,1,\bot\}^n, \mathcal{I} \subseteq [n]) \leftarrow \mathcal{A}(1^\lambda, 1^n, \mathsf{pp}, \mathsf{msk}) \\ \widetilde{\mathsf{ct}} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m),\ \mathsf{ct}_0 \leftarrow \mathsf{Delete}(\mathsf{pp}, \widetilde{\mathsf{ct}}, \mathcal{I}) \\ \widetilde{x} \leftarrow \mathsf{Restrict}(x, \mathcal{I}),\ \mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pp}, \widetilde{x}, m) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Note that the attribute vector x chosen by the adversary $\mathcal{A}$ can contain $\bot$ symbols.

4 Constrained PRFs: Defining Deletion Conformity

In this section, we first recall the notion of constrained PRFs (CPRFs) [BGI14, BW13, KPTZ13], and later introduce our notion of *deletion conforming* CPRFs.

Syntax. A constrained PRF (CPRF) for constraint class $\mathcal{F} = \{\mathcal{F}_N\}_{N \in \mathbb{N}}$ consists of three PPT algorithms ($\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval}$) with the following syntax:

$\mathsf{Setup}(1^\lambda, 1^N) \to \mathsf{msk}$. On input the security parameter λ and input length N, the setup algorithm outputs a master secret key $\mathsf{msk} \in \{0,1\}^k$. Let $k = k(\lambda, N)$ denote the length of secret key, where $k(\cdot,\cdot)$ is an a-priori fixed polynomial.

$\mathsf{Constrain}(\mathsf{msk}, f) \to \mathsf{sk}_f$. On input a constraint function $f \in \mathcal{F}_N$ and master secret key msk, the constrain algorithm outputs a constrained key sk_f.

$\mathsf{Eval}(\mathsf{sk}, x) \to y$. The evaluation algorithm takes as input a (possibly constrained) secret key sk and a string $x \in \{0,1\}^N$, and outputs a string y. Let $m = m(\lambda, N)$ denote the length of the output string y for some polynomial $m(\cdot,\cdot)$.[13]

Correctness of CPRF evaluation. A CPRF scheme is said to be correct if for all $\lambda, N \in \mathbb{N}$, $f \in \mathcal{F}_N$, and $x \in \{0,1\}^N$, the following holds:

$$f(x) = 1 \implies \Pr\left[\mathsf{Eval}(\mathsf{msk}, x) = \mathsf{Eval}(\mathsf{sk}_f, x) : \begin{array}{c} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^N) \\ \mathsf{sk}_f \leftarrow \mathsf{Constrain}(\mathsf{msk}, f) \end{array}\right] = 1$$

Security. Next, we recall the notion of single-key adaptive pseudorandomness security for constrained PRFs. Later on we also define the notion of key simulation security as defined in [Tsa19].

Definition 4.1 (Adaptive single-key constrained pseudorandomness)**.** We say that a $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval})$ satisfies adaptive single-key constrained pseudorandomness security if for any stateful admissible PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda, N \in \mathbb{N}$, the following holds:

$$\Pr\left[\mathcal{A}^{\mathsf{Eval}(\mathsf{msk},\cdot),\mathsf{Constrain}(\mathsf{msk},\cdot)}(r_b) = b : \begin{array}{c} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^N),\ b \leftarrow \{0,1\} \\ x^* \leftarrow \mathcal{A}^{\mathsf{Eval}(\mathsf{msk},\cdot),\mathsf{Constrain}(\mathsf{msk},\cdot)}(1^\lambda, 1^N) \\ r_0 \leftarrow \{0,1\}^m,\ r_1 = \mathsf{Eval}(\mathsf{msk}, x^*) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

[13]Here we consider a single PRF evaluation algorithm that could take as input a master key as well as a constrained key. Thus, both the master and constrained keys are of same length k. Note that one could instead split it into two seperate evaluation algorithms, however for ease of exposition we avoid it.

Here the adversary $\mathcal{A}$ is said to be admissible as long as it satisfies the following conditions — (1) it makes at most one query to the constrain oracle $\mathsf{Constrain}(\mathsf{msk}, \cdot)$, and its queried function f must be such that $f(x^*) = 0$, (2) it must not send x^* as one of its evaluation queries to $\mathsf{Eval}(\mathsf{msk}, \cdot)$.

The above pseudorandomness security could be extended to collusion-resistant notions where the adversary could make polynomially many constrain queries, however in this work we only require single-key security. Next, we define key simulation security for CPRFs.

Definition 4.2 (Adaptive key simulation)**.** We say that a $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval})$ satisfies adaptive key simulation security if there exists a PPT algorithm KeySim such that for any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda, N \in \mathbb{N}$, the following holds:

$$\Pr\left[\mathcal{A}^{\mathsf{Eval}(\mathsf{msk},\cdot)}(sk_b) = b : \begin{array}{c} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^N),\ b \leftarrow \{0,1\} \\ f^* \leftarrow \mathcal{A}^{\mathsf{Eval}(\mathsf{msk},\cdot)}(1^\lambda, 1^N) \\ \mathsf{sk}_0 \leftarrow \mathsf{KeySim}(1^\lambda, 1^N, f^*),\ \mathsf{sk}_1 \leftarrow \mathsf{Constrain}(\mathsf{msk}, f^*) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Here the adversary $\mathcal{A}$ is said to be admissible if all its evaluation queries $x \in \{0,1\}^N$ satisfy the condition that $f^*(x) = 0$. That is, none of the queried inputs are satisfied by the constraint f^*.

Non-colliding property. A constrained PRF CPRF that satisfies key simulation security (Definition 4.2) is said to be *non-colliding* if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, N \in \mathbb{N}$, every input $x \in \{0,1\}^N$, constraint function $f \in \mathcal{F}_N$, the following holds:

$$\Pr\left[\mathsf{Eval}(\mathsf{msk}, x) = \mathsf{Eval}(\mathsf{sk}'_f, x) : \begin{array}{c} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^N) \\ \mathsf{sk}'_f \leftarrow \mathsf{KeySim}(1^\lambda, 1^N, f) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Later on in Appendix A, we show that if the CPRF satisfies (0-key) pseudorandomness security, then it also satisfies the non-colliding property as long as the output length of the PRF is large enough. Additionally, we also show that the adaptive single-key constrained pseudorandomness security in fact implies adaptive key simulation security.

4.1 Deletion Conforming CPRFs

Now we define the deletion conforming property for CPRFs. Intuitively, it states that a constrained key in such a CPRF scheme must be deterministically computable by simply deleting specific bits of the master key (i.e., replacing some bits of the master key with a special $\perp$ symbol). Formally we define it below.

Definition 4.3 (Deletion Conforming CPRF)**.** We say that a constrained PRF scheme $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval})$ for a function class $\mathcal{F} = \{\mathcal{F}_N\}_{N\in\mathbb{N}}$ is a *deletion conforming* CPRF if the constrain algorithm $\mathsf{Constrain}$ is deterministic, and there exists two polynomial time algorithms $(\mathsf{CircuitGen}, \mathsf{DeleteFunc})$ with the following syntax and properties:

$\mathsf{CircuitGen}(1^\lambda, 1^N, x) \rightarrow C_x$. The circuit generation algorithm is a *deterministic* algorithm that takes as input the security parameter λ, length parameter N, and input string $x \in \{0,1\}^N$. It outputs the description of a circuit C_x.

$\mathsf{DeleteFunc}(1^\lambda, 1^N, f) \rightarrow \mathcal{I}_f$. The key deletion algorithm is a *deterministic* algorithm that takes as input the security parameter λ, length parameter N, and a constraint function $f \in \mathcal{F}_N$. It outputs a set of indices $\mathcal{I}_f \subseteq [k]$, where k denotes the length of the master secret key.

We say that $\mathsf{DCCPRF} = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{CircuitGen}, \mathsf{DeleteFunc})$ is a deletion conforming CPRF if for all $\lambda, N \in \mathbb{N}$, every function $f \in \mathcal{F}_N$, input $x \in \{0,1\}^N$, and master key $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$, the following properties are satisfied.

1. **Function deletion property:** $\mathsf{Constrain}(\mathsf{msk}, f) = \mathsf{Restrict}(\mathsf{msk}, \mathcal{I}_f)$, where index set $\mathcal{I}_f$ is computed as $\mathcal{I}_f = \mathsf{DeleteFunc}(1^\lambda, 1^N, f)$.

2. **Circuit evaluation property:** Let $C_x = \mathsf{CircuitGen}(1^\lambda, 1^N, x)$. It states that $\mathsf{Eval}(\mathsf{msk}, x) = C_x(\mathsf{msk})$ irrespective of whether $f(x) = 0/1$, and $\mathsf{Eval}(\mathsf{sk}_f, x) = \mathsf{CEval}(C_x, \mathsf{sk}_f)$ whenever $f(x) = 1$ where $\mathsf{sk}_f = \mathsf{Constrain}(\mathsf{msk}, f)$ or $\mathsf{sk}_f \leftarrow \mathsf{KeySim}(1^\lambda, 1^N, f)$.

Here recall that the Restrict and CEval operations are as defined in Section 2 — $\mathsf{Restrict}(s, \mathcal{I})$ denotes a string after replacing the bits in s with indices corresponding to indices in set $\mathcal{I}$ with $\perp$; and $\mathsf{CEval}(C, x)$ denotes evaluating the circuit C on input x, but setting the circuit output to be the all zeros string 0^m if the circuit C depends on the input wires whose indices have $\perp$ symbol in x.

5 Ciphertext Policy Attribute-Based Encryption

In this section, we recall the notion of Ciphertext Policy Attribute-Based Encryption (CP-ABE). First, we provide the syntax and definitions, and later define the predicate class we study in this work.

Syntax. A ciphertext-policy attribute based encryption (CP-ABE) scheme for a class of predicates $\mathcal{F} = \{\mathcal{F}_N\}_{N \in \mathbb{N}}$ and message space $\mathcal{M}$ consists of the following PPT algorithms:

$\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pp}, \mathsf{msk})$. On input the security parameter λ and attribute length N, the setup algorithm outputs a set of public parameters pp, and master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, x) \to \mathsf{sk}_x$. On input the master secret key msk and a key attribute $x \in \{0,1\}^N$, the key generation algorithm outputs a predicate key sk_x.

$\mathsf{Enc}(\mathsf{pp}, f, m) \to \mathsf{ct}$. On input the public parameters pp, a predicate $f \in \mathcal{F}_N$, and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_x, \mathsf{ct}) \to m/\mathtt{fail}$. On input a secret key sk_x and a ciphertext ct, the decryption algorithm either outputs a message m or a special string $\mathtt{fail}$ (to denote decryption failure).

Correctness. A CP-ABE scheme is said to be correct if for all $\lambda, N \in \mathbb{N}$, $f \in \mathcal{F}_N$, $m \in \mathcal{M}$, $x \in \{0,1\}^N$, the following holds:

$$f(x) = 1 \Longrightarrow \Pr\left[\mathsf{Dec}(\mathsf{sk}_x, \mathsf{ct}) = m : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N) \\ \mathsf{sk}_x \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x),\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m) \end{array}\right] = 1.$$

Security. For security, we require the scheme to achieve adaptive security (see Definition 3.1). Note that the admissibility condition for the adversary $\mathcal{A}$ in the security game is modified as follows. The adversary $\mathcal{A}$ is admissible as long as every secret key query $x \in \{0,1\}^N$ made by $\mathcal{A}$ to the oracle $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ satisfies the condition that $f^*(x) = 0$, where f^* is the challenge predicate chosen by $\mathcal{A}$.

6 Building Adaptively Secure CP-ABE

In this section, we build an adaptively secure CP-ABE scheme from a selectively secure KP-ABE scheme with deletable attributes DelABE and a single-key adaptively secure deletion conforming CPRF scheme DCCPRF.

6.1 Construction

Let DelABE = (DelABE.Setup, DelABE.KeyGen, DelABE.Enc, DelABE.Dec, DelABE.Delete) be a KP-ABE scheme with deletable attributes for predicate class $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$, and DCCPRF = (PRF.Setup, PRF.Constrain, PRF.Eval, PRF.CircuitGen, PRF.DeleteFunc, PRF.KeySim) be a deletion conforming CPRF for constraint class $\mathcal{F} = \{\mathcal{F}_N\}_{N \in \mathbb{N}}$. We require the predicate class $\mathcal{C}$ to be sufficiently expressive such that it contains circuits which perform comparison on top of a circuit generated by the $\mathsf{PRF.CircuitGen}$ algorithm. The requirement will become evident after the construction.

Below we describe our CP-ABE scheme $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for predicate class $\mathcal{F} = \{\mathcal{F}_N\}_{N\in\mathbb{N}}$.

$\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pp}, \mathsf{msk})$. The setup algorithm first runs DCCPRF setup to generate the corresponding master secret key: $\mathtt{prf}.\mathsf{msk} \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^N)$. Let $k = k(\lambda, N)$ denote the length of the master secret key $\mathtt{prf}.\mathsf{msk}$. Next, it runs the deletable ABE setup algorithm $\mathsf{DelABE.Setup}$ to get deletable ABE public parameters and master secret key as: $(\mathtt{del}.\mathsf{msk}, \mathtt{del}.\mathsf{pp}) \leftarrow \mathsf{DelABE.Setup}(1^\lambda, 1^k)$.

It sets public parameters and master key as $\mathsf{pp} = \mathtt{del}.\mathsf{pp}$, $\mathsf{msk} = (\mathtt{prf}.\mathsf{msk}, \mathtt{del}.\mathsf{msk})$.

$\mathsf{KeyGen}(\mathsf{msk}, x) \to \mathsf{sk}$. Let $\mathsf{msk} = (\mathtt{prf}.\mathsf{msk}, \mathtt{del}.\mathsf{msk})$. The key generation algorithm first computes $t = \mathsf{PRF.Eval}(\mathtt{prf}.\mathsf{msk}, x)$ and generates a circuit $C_x : \{0,1\}^k \to \{0,1\}^m$ as $C_x = \mathsf{PRF.CircuitGen}(x)$. Next, it creates the following circuit $(f_{x,t} : \{0,1\}^k \to \{0,1\})$

$$f_{x,t}(z) = \begin{cases} 1 & \text{if } C_x(z) \neq t, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Finally, the algorithm runs the deletable ABE key generation to sample the secret key sk as $\mathsf{sk} \leftarrow \mathsf{DelABE.KeyGen}(\mathtt{del}.\mathsf{msk}, f_{x,t})$.

$\mathsf{Enc}(\mathsf{pp}, f, m) \to \mathsf{ct}$. The encryption algorithm runs the CPRF key simulation to generate a simulated key as $\mathsf{sk}'_f \leftarrow \mathsf{PRF.KeySim}(1^\lambda, 1^N, f)$. Next, it runs the deletable ABE encryption algorithm with attribute sk'_f as $\mathsf{ct} \leftarrow \mathsf{DelABE.Enc}(\mathsf{pp}, m, \mathsf{sk}'_f)$, and outputs ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m/\mathtt{fail}$. The decryption algorithm runs the deletable ABE decryption as $z = \mathsf{DelABE.Dec}(\mathsf{sk}, \mathsf{ct})$, and outputs z as decryption output.

6.2 Correctness and Efficiency

We start by proving that our construction satisfies the CP-ABE correctness condition, and also discuss the efficiency of the resulting scheme. First, we prove correctness.

Lemma 6.1 (Correctness)**.** If the deletable KP-ABE scheme DelABE satisfies correctness, and the deletion conforming CPRF scheme DCCPRF satisfies non-colliding and circuit evaluation properties, then the CP-ABE scheme ABE described above is correct.

Proof. We show that the scheme decrypts correctly with all but negligible probability. In Appendix B, we will discuss how to boost the imperfect correctness to perfect correctness.

Fix any security parameter λ and attribute length N. For every predicate $f \in \mathcal{F}_N$, message $m \in \mathcal{M}$, and attribute $x \in \{0,1\}^N$, we have that the decryption algorithm Dec, on inputs ciphertext ct and secret key sk, simply outputs $z = \mathsf{DelABE.Dec}(\mathsf{sk}, \mathsf{ct})$. Consider $(\mathtt{del}.\mathsf{msk}, \mathtt{del}.\mathsf{pp})$ and $\mathtt{prf}.\mathsf{msk}$ to be the deletable KP-ABE and CPRF parameters sampled during setup. Note that the ciphertext ct is computed as $\mathsf{ct} \leftarrow \mathsf{DelABE.Enc}(\mathtt{del}.\mathsf{pp}, m, \mathsf{sk}'_f)$, where $\mathsf{sk}'_f \leftarrow \mathsf{PRF.KeySim}(1^\lambda, 1^N, f)$. Also, the secret key sk is sampled as $\mathsf{sk} \leftarrow \mathsf{DelABE.KeyGen}(\mathtt{del}.\mathsf{msk}, f_{x,t})$, where $t = \mathsf{PRF.Eval}(\mathtt{prf}.\mathsf{msk}, x)$ and $f_{x,t}$ is as defined in the construction. First, observe that by correctness of the deletable KP-ABE scheme, if $\mathsf{CEval}(f_{x,t}, \mathsf{sk}'_f) = 1$, then the decryption algorithm outputs message m correctly, i.e. $z = m$. Thus, to complete the completeness argument, we just need to show that whenever $f(x) = 1$, then $\mathsf{CEval}(f_{x,t}, \mathsf{sk}'_f) = 1$ as well with all but negligible probability (over the choice of random coins used during setup and encryption).

Recall that circuit $f_{x,t}(\mathsf{sk}'_f) = 1$ if and only if $C_x(\mathsf{sk}'_f) \neq t$, where $C_x = \mathsf{PRF.CircuitGen}(x)$. Now if $f(x) = 1$, by the circuit evaluation property of deletion conforming CPRF, we get that $C_x(\mathsf{sk}'_f) = \mathsf{PRF.Eval}(\mathsf{sk}'_f, x)$. Since $t = \mathsf{PRF.Eval}(\mathtt{prf}.\mathsf{msk}, x)$, thus by the non-colliding property, we know that the event $C_x(\mathsf{sk}'_f) = t$ happens with only negligible probability. Therefore, whenever $f(x) = 1$, the decryption algorithm outputs message m with all but negligible probability. This completeness the correctness argument. □

Next, we state the depth of the circuit $f_{x,t}$ for which we run the KP-ABE key generation algorithm.

Lemma 6.2 (Circuit depth)**.** For every $\lambda, N \in \mathbb{N}$, predicate $f \in \mathcal{F}_N$ and attribute $x \in \{0,1\}^N$, we have that $\mathsf{depth}(f_{x,t}) = \mathsf{depth}(C_x) + O(\log \lambda)$.

Proof. This follows immediately from our construction. Note that the circuit depth of $f_{x,t}$ is depth of C_x plus the depth of a circuit to check equality on two strings in $\{0,1\}^m$. Since m is a polynomial in the security parameter λ, and equality check on two strings in $\{0,1\}^m$ can be efficiently performed in depth $O(\log m) = O(\log \lambda)$ using XOR gates and OR gates, thus the lemma follows. □

6.3 Security

Next, we prove that the CP-ABE scheme constructed above is adaptively secure. Formally, we prove the following.

Theorem 6.3. If the deletion KP-ABE scheme DelABE satisfies selective IND-CPA security and deletion indistinguishability (Definitions 3.2 and 3.3), and the deletion conforming CPRF scheme DCCPRF satisfies adaptive key simulation security, and circuit evaluation and function deletion properties (Definitions 4.2 and 4.3), then the CP-ABE scheme ABE satisfies adaptive IND-CPA security as per Definition 3.1.

Proof. We prove the security via a sequence of hybrid games. We will first define the sequence of hybrid games, and then show that they are indistinguishable for any PPT adversary.

Game 0. This corresponds to the original adaptive IND-CPA security game.

- **Setup Phase.** The challenger runs $\mathtt{prf}.\mathsf{msk} \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^N)$ and $(\mathtt{del}.\mathsf{msk}, \mathtt{del}.\mathsf{pp}) \leftarrow \mathsf{DelABE.Setup}(1^\lambda, 1^k)$. Next, it sets $\mathsf{pp} = \mathtt{del}.\mathsf{pp}$ and $\mathsf{msk} = (\mathtt{prf}.\mathsf{msk}, \mathtt{del}.\mathsf{msk})$ and sends pp to the adversary $\mathcal{A}$.
- **Pre-Challenge Query Phase.** The adversary $\mathcal{A}$ makes polynomially many key queries on attributes it chooses. For each key query on attribute $x \in \{0,1\}^N$, the challenger proceeds as follows:
  1. It computes $t = \mathsf{PRF.Eval}(\mathtt{prf}.\mathsf{msk}, x)$, and generates a circuit $C_x : \{0,1\}^k \to \{0,1\}^m$ as $C_x = \mathsf{PRF.CircuitGen}(1^\lambda, 1^N, x)$. Next, it creates a circuit $f_{x,t}$ as described in Eq. (1).
  2. Then it computes a secret key as $\mathsf{sk} \leftarrow \mathsf{DelABE.KeyGen}(\mathtt{del}.\mathsf{msk}, f_{x,t})$, and sends sk to $\mathcal{A}$.
- **Challenge Phase.** $\mathcal{A}$ sends two messages (m_0, m_1) and a predicate function $f^* \in \mathcal{F}_N$ as its challenge to the challenger. The challenger responds with ciphertext ct^* to $\mathcal{A}$, where ct^* is computed as follows:
  1. The challenger generates a simulated key as $\mathsf{sk}_{f^*} \leftarrow \mathsf{PRF.KeySim}(1^\lambda, 1^N, f^*)$.
  2. Next, it chooses a random bit $b \leftarrow \{0,1\}$, and computes the challenge ciphertext as $\mathsf{ct}^* \leftarrow \mathsf{DelABE.Enc}(\mathtt{del}.\mathsf{pp}, \mathsf{sk}'_{f^*}, m_b)$.
- **Post-Challenge Query Phase.** This is identical to the pre-challenge query phase.
- **Guess.** The adversary $\mathcal{A}$ finally sends the guess b', and wins if $b = b'$.

Game 1. This game is identical to **Game 0** except that in the **Challenge Phase** step **1**, the challenger encrypts the challenge ciphertext to a *real constrained PRF key* with respect to challenge function f^* instead of the simulated key.

- **Challenge Phase.** $\mathcal{A}$ sends two messages (m_0, m_1) and a predicate function $f^* \in \mathcal{F}_N$ as its challenge to the challenger. The challenger responds with ciphertext ct^* to $\mathcal{A}$, where ct^* is computed as follows:
  1. The challenger generates a constrained key as $\mathsf{sk}_{f^*} \leftarrow \mathsf{PRF.Constrain}(\mathtt{prf}.\mathsf{msk}, f^*)$.

Game 2. This game is identical to **Game 1** except that in the **Challenge Phase** step **1**, the challenger generates the real constrained PRF key sk_{f^*} with respect to f^* directly using the PRF.DeleteFunc and Restrict algorithms on the PRF master secret key $\mathtt{prf}.\mathsf{msk}$.

- **Challenge Phase.** $\mathcal{A}$ sends two messages (m_0, m_1) and a predicate function $f^* \in \mathcal{F}_N$ as its challenge to the challenger. The challenger responds with ciphertext ct^* to $\mathcal{A}$, where ct^* is computed as follows:
    1. The challenger first computes a set of indices $\mathcal{I}_{f^*} := \mathsf{PRF.DeleteFunc}(1^\lambda, 1^N, f^*)$, and then it computes the constrained key as $\mathsf{sk}_{f^*} = \mathsf{Restrict}(\mathtt{prf}.\mathsf{msk}, \mathcal{I}_{f^*})$.

Game 3. This game is identical to **Game 2** except that in the **Challenge Phase** step **2**, the challenger encrypts the message to the *PRF master secret key* $\mathtt{prf}.\mathsf{msk}$ and then uses DelABE.Delete to modify the ciphertext according the indices set $\mathcal{I}_{f^*}$.

- **Challenge Phase.** $\mathcal{A}$ sends two messages (m_0, m_1) and a predicate function $f^* \in \mathcal{F}_N$ as its challenge to the challenger. The challenger responds with ciphertext ct^* to $\mathcal{A}$, where ct^* is computed as follows:
    1. The challenger first computes a set of indices $\mathcal{I}_{f^*} := \mathsf{PRF.DeleteFunc}(1^\lambda, 1^N, f^*)$.
    2. Next, it chooses a random bit $b \leftarrow \{0, 1\}$, and computes a KP-ABE ciphertext as $\mathsf{ct}' \leftarrow \mathsf{DelABE.Enc}(\mathtt{del}.\mathsf{pp}, \mathtt{prf}.\mathsf{msk}, m_b)$. Then it computes challenge ciphertext as $\mathsf{ct}^* \leftarrow \mathsf{DelABE.Delete}(\mathtt{del}.\mathsf{pp}, \mathsf{ct}', \mathcal{I}_{f^*})$.

Analysis. Next, we show by a sequence of lemmas that no PPT adversary can distinguish between any two adjacent games with non-negligible advantage. In the last game, we show that the advantage of any PPT adversary is negligible. This completes the proof of adaptive security of our CP-ABE scheme ABE.

Let $\mathcal{A}$ denote the PPT attacker playing the adaptive IND-CPA security game with the ABE challenger. In the sequel, we denote the advantage of adversary $\mathcal{A}$ in **Game** i as $\mathsf{Adv}^i_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ wins in Game } i] - \frac{1}{2}$, where recall that $\mathcal{A}$ wins in **Game** i if it guesses the challenger's bit b correctly.

Lemma 6.4. Assuming the key simulation security of the deletion conforming CPRF DCCPRF holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}_1(\cdot)$, such that for all $\lambda, N \in \mathbb{N}$, we have that $\mathsf{Adv}^0_{\mathcal{A}}(\lambda) - \mathsf{Adv}^1_{\mathcal{A}}(\lambda) \leq \mathsf{negl}_1(\lambda)$.

Proof. Suppose there exists an adversary $\mathcal{A}$ and a non-negligible function $\epsilon(\cdot)$ such that $\mathsf{Adv}^0_{\mathcal{A}}(\lambda) - \mathsf{Adv}^1_{\mathcal{A}}(\lambda) \geq \epsilon(\lambda)$, then we construct a reduction algorithm $\mathcal{B}$ such that $\mathcal{B}$ has non-negligible advantage in the *key simulation game* of the deletion conforming CPRF. Below we describe our reduction algorithm $\mathcal{B}$.

- In the setup phase, the key simulation challenger $\mathcal{K}$ runs PRF.Setup, and $\mathcal{B}$ runs DelABE.Setup to sample a key pair as $(\mathtt{del}.\mathsf{pp}, \mathtt{del}.\mathsf{msk}) \leftarrow \mathsf{DelABE.Setup}(1^\lambda, 1^k)$. $\mathcal{B}$ then sends $\mathtt{del}.\mathsf{pp}$ to $\mathcal{A}$ as the public parameters.

- In the pre-challenge query phase, when $\mathcal{A}$ sends a key query on attribute x to $\mathcal{B}$, $\mathcal{B}$ sends x to the key simulation challenger $\mathcal{K}$ as its PRF evaluation query. $\mathcal{K}$ answers $\mathcal{B}$ with t, where $t = \mathsf{PRF.Eval}(\mathtt{prf}.\mathsf{msk}, x)$. $\mathcal{B}$ uses t and x to generate circuit C_x and circuit $f_{x,t}$ as in Game 0; then it computes the secret key $\mathsf{sk} \leftarrow \mathsf{DelABE.KeyGen}(\mathtt{del}.\mathsf{msk}, f_{x,t})$, and sends sk to $\mathcal{A}$ as the secret key for attribute x.

- In the challenge phase, $\mathcal{A}$ sends the predicate function f^* and messages m_0, m_1 to the reduction algorithm $\mathcal{B}$. $\mathcal{B}$ then forwards f^* to $\mathcal{K}$ as its challenge constraint function. Let sk_{f^*} denote $\mathcal{K}$'s response. $\mathcal{B}$ chooses as random bit $b \leftarrow \{0, 1\}$, and computes the challenge ciphertext as $\mathsf{ct}^* \leftarrow \mathsf{DelABE.Enc}(\mathtt{del}.\mathsf{pp}, \mathsf{sk}_{f^*}, m_b)$, and sends ct^* to $\mathcal{A}$.

- The post-challenge phase is identical to the pre-challenge query phase. Finally, $\mathcal{A}$ outputs its guess b', and if $b = b'$ then $\mathcal{B}$ outputs 0 as its guess (to denote that sk_{f^*} was a simulated key). Otherwise, $\mathcal{B}$ outputs 1 as its guess.

First, note that $\mathcal{A}$ must be an admissible adversary in the CP-ABE security game, thus it must hold that $f^*(x) = 0$ for all attributes x queried by $\mathcal{A}$. Therefore, $\mathcal{B}$ is also an admissible adversary in the key simulation game since it also satisfies condition that $f^*(x) = 0$ for all inputs x queried by $\mathcal{B}$. Next, observe that if the challenger $\mathcal{K}$ samples sk_{f^*} as a simulated key, then $\mathcal{B}$ perfectly simulates Game 0 for $\mathcal{A}$, otherwise it simulates Game 1. Thus, $\mathcal{B}$'s advantage in the key simulation game is at least $\epsilon(\lambda)$, which is non-negligible and contradicts the key simulation security. $\square$

Lemma 6.5. Assuming the function deletion property of the deletion conforming CPRF DCCPRF holds, then for any adversary $\mathcal{A}$, parameters $\lambda, N \in \mathbb{N}$, we have that $\mathsf{Adv}^1_{\mathcal{A}}(\lambda) = \mathsf{Adv}^2_{\mathcal{A}}(\lambda)$.

Proof. This follows immediately from the function deletion property. Recall that function deletion property states that for all $\lambda, N \in \mathbb{N}$, every constraint function $f^* \in \mathcal{F}_N$, and master key $\mathtt{prf.msk} \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^N)$, we have that:

$$\Pr\left[\mathsf{sk}^{(1)}_{f^*} = \mathsf{sk}^{(2)}_{f^*} : \begin{array}{c} \mathcal{I}_{f^*} = \mathsf{PRF.DeleteFunc}(1^\lambda, 1^N, f^*) \\ \mathsf{sk}^{(1)}_{f^*} = \mathsf{PRF.Constrain}(\mathtt{prf.msk}, f^*) \\ \mathsf{sk}^{(2)}_{f^*} = \mathsf{Restrict}(\mathtt{prf.msk}, \mathcal{I}_{f^*}) \end{array}\right] = 1.$$

Note that $\mathsf{sk}^{(1)}_{f^*}$ and $\mathsf{sk}^{(2)}_{f^*}$ exactly correspond to the CPRF keys as generated in **Game 1** and **Game 2**, respectively. Since they are identical, thus the adversary's advantage is also identical in these two games. $\square$

Lemma 6.6. Assuming the deletion indistinguishability security of the deletable KP-ABE DelABE holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}_2(\cdot)$, such that for all $\lambda, N \in \mathbb{N}$, we have that $\mathsf{Adv}^2_{\mathcal{A}}(\lambda) - \mathsf{Adv}^3_{\mathcal{A}}(\lambda) \leq \mathsf{negl}_2(\lambda)$.

Proof. Suppose there exists an adversary $\mathcal{A}$ and a non-negligible function $\epsilon(\cdot)$ such that $\mathsf{Adv}^2_{\mathcal{A}}(\lambda) - \mathsf{Adv}^3_{\mathcal{A}}(\lambda) \geq \epsilon(\lambda)$, then we construct a reduction algorithm $\mathcal{B}$ such that $\mathcal{B}$ has non-negligible advantage in the *deletion indistinguishability game* of the deletable KP-ABE. Below we describe our reduction algorithm $\mathcal{B}$.

- In the setup phase, the deletion indistinguishability challenger $\mathcal{D}$ runs $\mathsf{DelABE.Setup}$ and sends the deletable ABE parameters $(\mathtt{del.pp}, \mathtt{del.msk})$ to $\mathcal{B}$. $\mathcal{B}$ then samples a CPRF master key as $\mathtt{prf.msk} \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^N)$, and sends $\mathtt{del.pp}$ to $\mathcal{A}$ as the CP-ABE public parameters.
- In the pre-challenge query phase, $\mathcal{A}$ sends a key query on attribute x to $\mathcal{B}$. $\mathcal{B}$ first evaluates the CPRF as $t = \mathsf{PRF.Eval}(\mathtt{prf.msk}, x)$, and uses t and x to generate circuits C_x and $f_{x,t}$ as in Game 2. It then computes the secret key $\mathsf{sk} \leftarrow \mathsf{DelABE.KeyGen}(\mathtt{del.msk}, f_{x,t})$, and sends sk to $\mathcal{A}$ as the secret key for attribute x.
- In the challenge phase, $\mathcal{A}$ sends the predicate function f^* and messages m_0, m_1 to $\mathcal{B}$. The reduction algorithm $\mathcal{B}$ samples a random bit $b \leftarrow \{0,1\}$, and computes a set of indices $\mathcal{I}_{f^*} = \mathsf{PRF.DeleteFunc}(1^\lambda, 1^N, f^*)$, and sends $(m_b, \mathtt{prf.msk}, \mathcal{I}_{f^*})$ to the deletion challenger $\mathcal{D}$. Let ct^* denote the challenger's response. $\mathcal{B}$ forwards ct^* to $\mathcal{A}$ as its challenge ciphertext.
- The post-challenge phase is identical to the pre-challenge query phase. Finally, $\mathcal{A}$ outputs its guess b', and if $b = b'$ then $\mathcal{B}$ outputs 0 as its guess (to denote that ct^* was a freshly encrypted ciphertext). Otherwise, $\mathcal{B}$ outputs 1 as its guess.

Note that if the challenger $\mathcal{D}$ computes ct^* by first restricting the attribute to the constrained key and then encrypting it directly using the KP-ABE encryption algorithm, then $\mathcal{B}$ perfectly simulates Game 2 for $\mathcal{A}$, otherwise it simulates Game 3. Thus, $\mathcal{B}$'s advantage in the deletion indistinguishability game is at least $\epsilon(\lambda)$, which is non-negligible and contradicts the deletion indistinguishability security. $\square$

Lemma 6.7. Assuming the selective IND-CPA security of the deletable KP-ABE DelABE holds and the deletion conforming CPRF DCCPRF satisfies circuit evaluation property, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}_3(\cdot)$, such that for all $\lambda, N \in \mathbb{N}$, we have that $\mathsf{Adv}^3_{\mathcal{A}}(\lambda) \leq \mathsf{negl}_3(\lambda)$.

Proof. Suppose there exists an adversary $\mathcal{A}$ and a non-negligible function $\epsilon(\cdot)$ such that $\mathsf{Adv}^3_{\mathcal{A}}(\lambda) \geq \epsilon(\lambda)$, then we construct a reduction algorithm $\mathcal{B}$ such that $\mathcal{B}$ has non-negligible advantage in the *selective IND-CPA game* of the deletable KP-ABE. Below we describe our reduction algorithm $\mathcal{B}$.

- In the setup phase, $\mathcal{B}$ first samples a CPRF master key as $\texttt{prf}.\mathsf{msk} \leftarrow \mathsf{PRF}.\mathsf{Setup}(1^\lambda, 1^N)$, and sends $\texttt{prf}.\mathsf{msk}$ as its challenge attribute to the selective IND-CPA challenger $\mathcal{D}$. The challenger runs $\mathsf{DelABE}.\mathsf{Setup}$ and sends the deletable public parameters $\texttt{del}.\mathsf{pp}$ to $\mathcal{B}$. $\mathcal{B}$ simply forwards $\texttt{del}.\mathsf{pp}$ to $\mathcal{A}$ as the CP-ABE public parameters.
- In the pre-challenge query phase, when $\mathcal{A}$ sends a key query on attribute x to $\mathcal{B}$, $\mathcal{B}$ computes $t_x = \mathsf{PRF}.\mathsf{Eval}(\texttt{prf}.\mathsf{msk}, x)$ and generates the circuit f_{x,t_x} using x and t_x. Next, $\mathcal{B}$ sends secret key query on predicate f_{x,t_x} to the challenger $\mathcal{D}$. $\mathcal{D}$ replies $\mathcal{B}$'s query with sk and $\mathcal{B}$ forwards sk to $\mathcal{A}$ as the secret key for attribute x.
- In the challenge phase, $\mathcal{A}$ sends the predicate function f^* and messages m_0, m_1 to $\mathcal{B}$. $\mathcal{B}$ sends (m_0, m_1) to $\mathcal{D}$. Let ct' denote the KP-ABE challenge ciphertext sent by $\mathcal{D}$. $\mathcal{B}$ first computes the index set $\mathcal{I}_{f^*} = \mathsf{PRF}.\mathsf{DeleteFunc}(1^\lambda, 1^N, f^*)$, and then computes challenge ciphertext as $\mathsf{ct}^* \leftarrow \mathsf{DelABE}.\mathsf{Delete}(\texttt{del}.\mathsf{pp}, \mathsf{ct}', \mathcal{I}_{f^*})$. $\mathcal{B}$ sends ct^* to $\mathcal{A}$ as its challenge ciphertext.
- The post-challenge phase is identical to the pre-challenge query phase. Finally, $\mathcal{A}$ outputs its guess b', and $\mathcal{B}$ outputs the same bit b' as its guess.

First, note that for each key query on attribute x made by $\mathcal{A}$, we have that $C_x(\texttt{prf}.\mathsf{msk}) = \mathsf{PRF}.\mathsf{Eval}(\texttt{prf}.\mathsf{msk}, x)$. This follows from the circuit evaluation property of the deletion conforming CPRF. Since $t_x = \mathsf{PRF}.\mathsf{Eval}(\texttt{prf}.\mathsf{msk}, x)$, thus by definition of the circuit f_{x,t_x} (see Eq. (1)), we have that $f_{x,t_x}(\texttt{prf}.\mathsf{msk}) = 0$ for every attribute x. Thus, the reduction algorithm $\mathcal{B}$ is an admissible adversary in the selective IND-CPA game. Next, observe that $\mathcal{B}$ perfectly simulates Game 3 for $\mathcal{A}$, therefore $\mathcal{B}$'s advantage in the selective IND-CPA game is at least $\epsilon(\lambda)$, which is non-negligible and contradicts the selective IND-CPA security of the deletable KP-ABE system. □

Combining Lemmas 6.4 to 6.7, the Theorem 6.3 follows.

□

7 Deletion Conforming CPRFs for Subset Constraints

In this section, we build an adaptively secure deletion conforming constrained PRF scheme for subset constraints from a regular PRF scheme PRF. We start by describing the constraint class $\{\mathcal{F}_N\}_{N \in \mathbb{N}}$ that we study in this work. As stated in the introduction the construction below follows the lines of Davidson et al. [DKNY18] and [Tsa19].

Subset constraints. Here we focus on the class of subset constraints. At a high level, in this setting a binary input string $x \in \{0,1\}^N$ could alternatively be interpreted as a set $\mathsf{Set}(x) \subseteq [N]$ instead, where an index $i \in [N]$ lies in the set $\mathsf{Set}(x)$ *iff* $x_i = 1$. Formally, we define the Set function as:

$$\forall x \in \{0,1\}^N, \quad \mathsf{Set}(x) := \{i \in [N] \ : \ x_i = 1\}.$$

Now every constraint in the class $\mathcal{F}_N$ is associated with a length N binary string y, and the corresponding constraint function Subset_y is defined as follows:

$$\forall x \in \{0,1\}^N, \quad \mathsf{Subset}_y(x) = 1 \iff \mathsf{Set}(x) \subseteq \mathsf{Set}(y).$$

Concretely, we have that $\mathcal{F}_N = \{\mathsf{Subset}_y\}_{y \in \{0,1\}^N}$. For ease of exposition, throughout this paper we simply denote each constraint with only the associated string y instead of the entire function Subset_y.

7.1 Construction

Let $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$ be a standard PRF with key space $\left\{\{0,1\}^{k(\lambda)}\right\}_{\lambda \in \mathbb{N}}$ and output length $m(\lambda)$, for polynomials $k(\cdot), m(\cdot)$. Below we describe our constrained PRF scheme CPRF for subset constraints $\left\{\{\mathsf{Subset}_y\}_{y \in \{0,1\}^N}\right\}_{N \in \mathbb{N}}$.

$\mathsf{Setup}(1^\lambda, 1^N) \to \mathsf{msk}$. The CPRF setup algorithm samples $N+1$ regular PRF keys as $\mathsf{msk}_i \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^N)$ for $i \in [0, N]$. The algorithm sets the CPRF master key msk as $\mathsf{msk} = (\mathsf{msk}_0, \mathsf{msk}_1, \cdots, \mathsf{msk}_N)$.

Throughout, we interpret the key msk as length $(N+1) \cdot k$ bit string which is divided into $N+1$ blocks of k bits each, where $k = k(\lambda)$. Also, let $m = m(\lambda)$

$\mathsf{Constrain}(\mathsf{msk}, y) \to \mathsf{sk}_y$. Let $\mathsf{msk} = (\mathsf{msk}_0, \mathsf{msk}_1, \cdots, \mathsf{msk}_N)$. The constrain algorithm sets the constrained key sk_y as the master key msk with blocks of regular PRF keys omitted as per constraint $y \in \{0,1\}^N$. Concretely, the algorithm outputs constrained key as $\mathsf{sk}_y = (\mathsf{sk}_{y,0}, \mathsf{sk}_{y,1}, \cdots, \mathsf{sk}_{y,N})$ where

$$\forall i \in [0, N], \quad \mathsf{sk}_{y,i} = \begin{cases} \mathsf{msk}_i & \text{if } i = 0 \text{ or } y_i = 1, \\ \perp^k & \text{otherwise.} \end{cases} \tag{2}$$

$\mathsf{Eval}(\mathsf{sk}, x) \to t$. The evaluation algorithm intep館rets the input key sk as $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1, \cdots, \mathsf{sk}_N)$ where each sub-key $\mathsf{sk}_i \in \{0, 1, \perp\}^k$ for $i \in [0, N]$. It first checks that for every $i \in \mathsf{Set}(x)$, $\mathsf{sk}_i \in \{0,1\}^k$ (that is, sk_i is a binary string). If the check fails, it outputs an all zeros string 0^m. Otherwise, it computes output string t as

$$t = \bigoplus_{\substack{i \in [0,N] \text{ s.t.} \\ i=0 \ \vee \ x_i = 1}} \mathsf{PRF.Eval}(\mathsf{sk}_i, x).$$

Below we argue that the above construction satisfies correctness as well as achieves single-key adaptive pseudorandomness security.

Correctness of CPRF evaluation. Fix parameters $\lambda, N \in \mathbb{N}$, constraint $y \in \{0,1\}^N$, and input $x \in \{0,1\}^N$. We know that whenever $\mathsf{Subset}_y(x) = 1$ (i.e., the constraint is satisfied), then for all $i \in [N]$ we have that $(x_i = 1) \to (y_i = 1)$. In words, if the input x satisfies the constraint y, then for every position i where $x_i = 1$, the corresponding bit position of y is also 1.

Consider any master secret key $\mathsf{msk} = (\mathsf{msk}_0, \mathsf{msk}_1, \cdots, \mathsf{msk}_N)$ where each msk_i is sampled using the PRF setup algorithm. Let $\mathsf{sk}_y = (\mathsf{sk}_{y,0}, \mathsf{sk}_{y,1}, \cdots, \mathsf{sk}_{y,N}) = \mathsf{Constrain}(\mathsf{msk}, y)$. By construction, we have that for all $i \in [0, N]$, $\mathsf{sk}_{y,i} = \mathsf{msk}_i$ if $i = 0$ or $x_i = 1$, otherwise $\mathsf{sk}_{y,i} = \perp^k$. Since the CPRF only evaluates on $\mathsf{sk}_{y,i}$ where $i = 0$ or $x_i = 1$, thus the CPRF output is unchanged irrespective of whether the evaluator uses master key msk or constrained key sk_y, i.e. $\mathsf{Eval}(\mathsf{sk}_y, x) = \mathsf{Eval}(\mathsf{msk}, x)$ for all x such that $\mathsf{Subset}_y(x) = 1$. This completes the correctness argument.

Adaptive single-key pseudorandomness security. Here we prove the following.

Theorem 7.1. If the PRF scheme PRF satisfies pseudorandomness security (Definition 2.1), then the CPRF scheme CPRF satisfies adaptive single-key constrained pseudorandomness security as per Definition 4.1.

Proof. Suppose there exists an adversary $\mathcal{A}$ that has non-negligible advantage in the adaptive single-key constrained pseudorandomness game with the CPRF challenger, then we construct a reduction algorithm $\mathcal{B}$ such that $\mathcal{B}$ has non-negligible advantage in the standard PRF security game. Below we describe our reduction algorithm $\mathcal{B}$.

- In the setup phase, $\mathcal{B}$ first randomly picks an index $j \leftarrow [0, N]$, and samples N PRF keys as $\mathsf{msk}_i \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^N)$ for all $i \in [0, N] \setminus \{j\}$.

- In the pre-challenge query phase, for answering an evaluation query x sent by $\mathcal{A}$, $\mathcal{B}$ proceeds as follows. If $x_j = 1$, then $\mathcal{B}$ forwards x to the standard PRF challenger $\mathcal{D}$ as its evaluation query. Let v denote the challenger's response. If $x_j = 0$, then $\mathcal{B}$ sets $v = \mathbf{0}$. $\mathcal{B}$ computes the CPRF output t as follows and sends t to the adversary $\mathcal{A}$.

$$t = v \oplus \left(\bigoplus_{\substack{i \in [0,N] \setminus \{j\} \\ \text{s.t. } i=0 \vee x_i = 1}} \mathsf{PRF.Eval}(\mathsf{msk}_i, x) \right). \tag{3}$$

  For answering a constrained key query y by $\mathcal{A}$, the reduction algorithm $\mathcal{B}$ proceeds as follows. If $j \in \{0\} \cup \mathsf{Set}(y)$, then $\mathcal{B}$ aborts and guesses a random bit. Otherwise, $\mathcal{B}$ computes the constrained key $\mathsf{sk}_y = (\mathsf{sk}_{y,0}, \mathsf{sk}_{y,1}, \cdots, \mathsf{sk}_{y,N})$ as described in Eq. (2), and sends to $\mathcal{A}$. Note that if $\mathcal{B}$ does not abort, then it has all the required PRF keys msk_i for answering the constrained key query since it will be the case that $y_j = 0$.

- In the challenge phase, $\mathcal{A}$ sends challenge input x^* to $\mathcal{B}$. If $x^*_j \neq 1$ or $j \neq 0$, then $\mathcal{B}$ aborts and guesses randomly. Otherwise, $\mathcal{B}$ sends x^* to the PRF challenger as its challenge query, and let v^* denote the challenger's response. The reduction algorithm $\mathcal{B}$ computes the CPRF output t^* using v^* and PRF keys $\{\mathsf{msk}_i\}_{i \neq j}$ as in Eq. (3). It sends t^* to $\mathcal{A}$ as its challenge.

- The post-challenge phase queries are answered identical to that in the pre-challenge query phase. Finally, $\mathcal{A}$ outputs its guess b', and $\mathcal{B}$ forwards b' to the challenger $\mathcal{D}$ as its guess.

Note that for any admissible adversary $\mathcal{A}$, there must exist an index $i \in [0, N]$ such that either — (1) $x^*_i = 1$ and $y_i = 0$, where x^* is the challenge input and y is the constrained key query, or (2) $\mathcal{A}$ makes no constrained key queries. In case (1), with probability at least $\frac{1}{N+1}$, $\mathcal{B}$'s random choice of index j matches the special index i. While, in case (2), with probability $\frac{1}{N+1}$ $\mathcal{B}$ chooses $j = 0$. Thus, we have that with probability at least $\frac{1}{N+1}$ $\mathcal{B}$ does not abort. Next, observe that whenever $\mathcal{B}$ does not abort, then $\mathcal{B}$ perfectly simulates the single-key CPRF security game for $\mathcal{A}$. Thus, if $\mathcal{A}$'s advantage is at least $\epsilon(\lambda)$, then $\mathcal{B}$'s advantage is also at least $\frac{\epsilon(\lambda)}{N+1}$. Hence, the lemma follows. □

7.2 Deletion Conforming Properties

In this section, we show that the constrained PRF construction CPRF described above is a deletion conforming CPRF scheme. First, we describe the $\mathsf{CircuitGen}$ and $\mathsf{DeleteFunc}$ algorithms. Later on, we show that these algorithms satisfy the deletion conforming properties as discussed in Section 4.

$\mathsf{CircuitGen}(1^\lambda, 1^N, x) \to C_x$. Let $\Delta_x : \{0,1\}^k \to \{0,1\}^m$ denote the PRF evaluation circuit that takes as input a key $\mathsf{key} \in \{0,1\}^k$ and evaluates the PRF on input x using key. That is, $\Delta_x := \mathsf{PRF.Eval}(\cdot, x)$. The circuit generation algorithm outputs a circuit $C_x : \{0,1\}^{(N+1)\cdot k} \to \{0,1\}^m$ where the circuit C_x is described as follows:

- Let $\mathsf{sk} \in \{0,1\}^{(N+1)\cdot k}$ represent the input to circuit C_x. Parse the input sk as $N+1$ blocks of PRF keys — $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1, \cdots, \mathsf{sk}_N)$, where $\mathsf{sk}_i \in \{0,1\}^k$ for all $i \in [0, N]$. (Note that this simply means grouping the input wires to circuits into $N+1$ disjoint blocks.)
- Circuit C_x *ignores* the input wires corresponding to all sk_i where $x_i = 0$. And, for all remaining input wires, it applies the circuit Δ_x on every size k wire block. More concretely, it computes in parallel $t_i = \Delta_x(\mathsf{sk}_i)$ for all $i \in \{0\} \cup \mathsf{Set}(x)$.

  (Note that when we say it ignores the input wires, then we mean that the circuit does not depend on those input wires. That is, all these input wires belong to the set $\mathsf{Unsupported}(C_x)$.)

- Next, C_x XORs all the computed m-bit strings $\{t_i\}$ together to output an m-bit string t. Concretely, C_x applies an XOR gate (in parallel) to every two neighboring t_i values obtained above, then it again applies an XOR gate to the previously computed values, and continues until it computes t. In words, it XORs t_i values in a binary tree-like manner.

$\mathsf{DeleteFunc}(1^\lambda, 1^N, y) \to \mathcal{I}_y$. The deletion function outputs the following set of indices $\mathcal{I}_y \subseteq [(N+1)\cdot k]$:

$$\mathcal{I}_y = \left\{ j \in [(N+1)\cdot k] \ : \ y_i = 0 \text{ where } i = \left\lceil \frac{j}{k} \right\rceil - 1 \right\}.$$

Next, we argue that $\mathsf{DCCPRF} = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{CircuitGen}, \mathsf{DeleteFunc})$ is a deletion conforming CPRF as per Definition 4.3. Additionally we also discuss the depth of the circuits generated by the $\mathsf{CircuitGen}$ algorithm below. Fix $\lambda, N \in \mathbb{N}$, a constraint $y \in \{0,1\}^N$, and input $x \in \{0,1\}^N$. Consider any master secret key $\mathsf{msk} = (\mathsf{msk}_0, \cdots, \mathsf{msk}_N)$ as sampled by the setup algorithm.

Circuit depth. Suppose the circuit depth of the evaluation circuit of the standard PRF $\mathsf{PRF.Eval}$ is $d = d(\lambda, N)$. Then we have that the depth of circuit C_x is $d + O(\log N)$. This follows from the circuit description C_x as described in $\mathsf{CircuitGen}$. Note that C_x consists of two layers of sub-circuits, where initially in the lower layer of C_x we simply have the PRF evaluation circuit Δ_x applied on the accepting parts of the input key sk; and in the upper layer of C_x, we apply the XOR gates pair-by-pair and level-by-level which forms a binary tree of XOR gates. Since there are at most $N+1$ strings $\{t_i\}_i$ in the upper layer that C_x needs to XOR together, thus it has depth $O(\log N)$. Therefore the depth of the C_x circuit is $d + O(\log N)$.

Function deletion property. First, note that the output of the constrain algorithm on inputs msk and y is a secret key $\mathsf{sk}_y = (\mathsf{sk}_{y,0}, \cdots, \mathsf{sk}_{y,N})$, where if $y_i = 0$ then $\mathsf{sk}_{y,i} = \perp^k$, otherwise $\mathsf{sk}_{y,i} = \mathsf{msk}_i$.

Now we know that an index $j \in \mathcal{I}_y$ iff $y_{\lceil \frac{j}{k} \rceil - 1} = 0$, where $\mathcal{I}_y = \mathsf{DeleteFunc}(1^\lambda, 1^N, y)$. Thus, by definition of the $\mathsf{Restrict}$ operation, we get that $\mathsf{Restrict}(\mathsf{msk}, \mathcal{I}_y) = \mathsf{sk}_y = \mathsf{Constrain}(\mathsf{msk}, y)$ since $\mathsf{Restrict}$ simply replaces the master key components to the bot string $\perp^k$ wherever $y_i = 0$.

Circuit evaluation property. For the case of circuit evaluation w.r.t. the master key msk, it follows directly from our construction that $\mathsf{Eval}(\mathsf{msk}, x) = C_x(\mathsf{msk})$. This is because circuits $\mathsf{Eval}(\cdot, x)$ and C_x do the exact same computation.

For a constraint y, we know that if $\mathsf{Subset}_y(x) = 1$ then for every $i \in [N]$ wherever $x_i = 1$ we also have $y_i = 1$. Thus C_x will not touch any input wires in a constrained key sk_y which are set as $\perp$, i.e. we have $\mathcal{I}_y \subseteq \mathsf{Unsupported}(C_x)$. Therefore, $\mathsf{Eval}(\mathsf{sk}_y, x) = \mathsf{CEval}(C_x, \mathsf{sk}_y)$ whenever $\mathsf{Subset}_y(x) = 1$ as the associated circuits perform identical computation.

8 Deletable ABE from standard assumptions

In this section we show that [GPSW06] is already a KP-ABE scheme with deletable attributes. First, we show that the KP-ABE schemes for monotone access structures in [GPSW06] have efficient deletion algorithms such that the resulting scheme satisfies both the semantic security as well deletion indistinguishability properties. Later on, we briefly elaborate the well-known approach for building a KP-ABE scheme for $\mathbf{NC}^1$ (i.e., log-depth circuits) from any KP-ABE scheme for monotone access structures, and describe that it preserves the deletion property of the underlying system.

8.1 Deletable ABE from Bilinear Maps via [GPSW06]

Goyal et al. (GPSW) [GPSW06] proposed a KP-ABE scheme for monotone access structures and proved its security under the Decisional Bilinear Diffie-Hellman (DBDH) assumption [BF01]. Here we show that the GPSW scheme, described in [GPSW06, Section 4], is also a deletable KP-ABE scheme for the same predicate class. Let $\mathsf{GPSW} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ represent the KP-ABE construction provided in [GPSW06, Section 4]. Formally, they proved the following.

Theorem 8.1 ([GPSW06, Theorem 1, Paraphrased])**.** If the Decisional Bilinear Diffie-Hellman (DBDH) assumption holds, then the scheme GPSW is a selective IND-CPA secure scheme as per Definition 3.2.

Now we describe a simple deletion algorithm for the GPSW scheme, and argue that the augmented GPSW scheme satisfies all the required properties described in Section 3. We start by briefly discussing some notational changes that we make to the GPSW syntax.

Notation. For consistency with our ABE definitions, we interpret the attribute string as a bit string $x \in \{0,1\}^n$, where as is the GPSW construction [GPSW06, Section 4] the attribute was parsed as a set of subset of the attribute universe $\mathcal{U} = \{1, 2, \ldots, n\}$. Here n denotes the length of the attributes selected during system setup. Note that this is mostly a syntactic change, and does not affect the GPSW scheme in any significant way.

Below we recall the Setup and Enc algorithms as provided in [GPSW06, Section 4], and also describe our Delete algorithm.

$\mathsf{Setup}(1^\lambda, 1^n) \rightarrow (\mathsf{pp}, \mathsf{msk})$. The setup algorithm chooses a bilinear group $\mathbb{G}_1$ of prime order p. Let g denote the generator of the group $\mathbb{G}_1$, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be associated the bilinear map. It chooses a random key exponent $\alpha \in \mathbb{Z}_p$, and also chooses a random exponent per bit position of the attribute, that is $t_i \leftarrow \mathbb{Z}_p$ for $i \in [n]$.

It outputs the public parameters and master secret key as $\mathsf{pp} = (g, e(g,g)^\alpha, \{g^{t_i}\}_{i\in[n]})$ and $\mathsf{msk} = (\alpha, \{t_i\}_{i\in[n]})$.[14]

$\mathsf{Enc}(\mathsf{pp}, x, m) \rightarrow \mathsf{ct}$. The encryption algorithm parses the public parameters as $\mathsf{pp} = (g, K, \{T_i\}_{i\in[n]})$, and an attribute $x \in \{0,1\}^n$. It chooses a random exponent $s \in \mathbb{Z}_p$, and publishes the ciphertext as

$$\mathsf{ct} = (x, m \cdot K^s, \{T_i^s\}_{i\in[n]:x_i=1}).$$

Encrypting to attributes with $\perp$ symbols. First, note that in the GPSW encryption algorithm the input attribute string x is a binary string, that is $x \in \{0,1\}^n$. However, in our deletable ABE framework, we allow the encryptor to choose attributes with $\perp$ symbols, thus the attribute string x now lies in $\{0,1,\perp\}^n$ instead of $\{0,1\}^n$. Now our augmented encryption algorithm is identical to the above encryption algorithm, that is the ciphertext is computed as

$$\mathsf{ct} = (x, m \cdot K^s, \{T_i^s\}_{i\in[n]:x_i=1}).$$

Note that previously the algorithm does not compute T_i^s for all i wherever $x_i = 0$. Now the augmented encryption algorithm also does not compute T_i^s for all i wherever $x_i = \perp$. That is, it treats $\perp$ symbols as a 0 bit during encryption. Therefore, the deletion algorithm can simply *delete* the T_i^s terms from the ciphertext wherever $i \in \mathcal{I}$ to compute a corresponding deleted ciphertext. Formally, we describe it below.

$\mathsf{Delete}(\mathsf{pp}, \mathsf{ct}, \mathcal{I}) \rightarrow \mathsf{ct}'$. The algorithm parses the ciphertext as $\mathsf{ct} = (x, E', \{E_i\}_{i\in[n]:x_i=1})$. It sets the output ciphertext ct' as

$$\mathsf{ct}' = (\mathsf{Restrict}(x, \mathcal{I}), E', \{E_i\}_{i\in[n]\setminus\mathcal{I}:\ x_i=1}).$$

[14]The parameters also contain the bilinear map parameters, but here we don't explicitly write it for simplicity.

Deletion Indistinguishability. First, we show that the augmented GPSW scheme $\mathsf{AugGPSW} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delete})$ satisfies the deletion indistinguishability property. Below we prove a much stronger statement which in turn implies deletion indistinguishability. Intuitively, we argue that, for every choice of system parameters, the distribution of a freshly encrypted ciphertext and a (corresponding) deleted ciphertext are *identical*.

Lemma 8.2. For every $\lambda, n \in \mathbb{N}$, parameters $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, attribute $x \in \{0,1\}^n$, message $m \in \mathcal{M}$, and index set $\mathcal{I} \in [n]$, the following two distributions are identical:

$$\mathcal{D}_1 = \left\{ \mathsf{ct} : \begin{array}{c} x' = \mathsf{Restrict}(x, \mathcal{I}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x', m) \end{array} \right\}, \qquad \mathcal{D}_2 = \left\{ \mathsf{ct}' : \begin{array}{c} \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m) \\ \mathsf{ct}' \leftarrow \mathsf{Delete}(\mathsf{pp}, \mathsf{ct}, \mathcal{I}) \end{array} \right\}.$$

That is, $\mathcal{D}_1 \equiv \mathcal{D}_2$.

Proof. The proof of this lemma immediately follows by inspection of the encryption and deletion algorithms described above. Consider any λ, n, key pair $(\mathsf{pp}, \mathsf{msk})$, attribute x, message m and index set $\mathcal{I}$. First, note that the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ can be expanded as follows:

$$\mathcal{D}_1 = \left\{ (x', m \cdot K^s, \{T_i^s\}_{i \in S_1}) : x' = \mathsf{Restrict}(x, \mathcal{I}), s \leftarrow \mathbb{Z}_p, S_1 = \{i \in [n] : x'_i = 1\} \right\},$$
$$\mathcal{D}_2 = \left\{ (x', m \cdot K^s, \{T_i^s\}_{i \in S_2}) : x' = \mathsf{Restrict}(x, \mathcal{I}), s \leftarrow \mathbb{Z}_p, S_2 = \{i \in [n] \setminus \mathcal{I} : x_i = 1\} \right\}.$$

Recall by definition of $\mathsf{Restrict}$, we have that $x'_i = 1$ if and only if $x_i = 1$ and $i \notin \mathcal{I}$. Therefore, it follows that $\mathcal{D}_1 \equiv \mathcal{D}_2$. □

Correctness. Note that since a deleted ciphertext is *identically* distributed to a freshly encrypted ciphertext, and also GPSW is a perfectly correct ABE scheme, thus correctness of our $\mathsf{AugGPSW}$ scheme follows.

Selective IND-CPA Security. Note that even though in our scheme, attribute vectors could contain $\perp$ symbols (i.e., lie in $\{0,1,\perp\}^n$), the IND-CPA attacker is only allowed to specify a binary string as a challenge attribute (i.e., it must lie in $\{0,1\}^n$) in the selective security game (Definition 3.2). Therefore, the selective IND-CPA security proof of $\mathsf{AugGPSW}$ follows from selective IND-CPA security proof of GPSW.

Hence, combining above facts, Lemma 8.2 and Theorem 8.1, we obtain the following:

Theorem 8.3. If the Decisional Bilinear Diffie-Hellman (DBDH) assumption holds, then the scheme $\mathsf{AugGPSW}$ is a KP-ABE scheme with deletable attributes that satisfies selective IND-CPA security as well as deletion indistinguishability (Definitions 3.2 and 3.3).

Also, later on in Appendix C we describe how to get deletable ABE from Computational Bilinear Diffie-Hellman (CBDH) assumption. It follows from a straightforward use of hardcore predicate on top of the GPSW scheme.

8.2 Deletable ABE: Monotonic Access Structures to $\mathbf{NC}^1$

Suppose we start with a KP-ABE scheme for arbitrary polynomial-sized monotone boolean formulas, then there is a well-known folklore transformation that gives us a KP-ABE scheme for log-depth circuits ($\mathbf{NC}^1$) generically from the underlying scheme. The idea can be described as follows. First, the key generation algorithm, on input a log-depth (non-monotone) circuit C, generates a polynomial-sized (non-monotone) boolean formula f_C that evaluates the same circuit. (Note that size of the formula f_C grows exponentially with the depth of circuit C, thus the same transformation does not work for larger depth circuits.) Now the formula f_C is a possibly *non-monotone* boolean formula, thus it could apply negation ($\neg$) gates on non-atomic formulae. Next, one using De Morgan's identities can translate the non-monotone boolean formula f_C into another formula $\widetilde{f}_C$ such that in the description of formula $\widetilde{f}_C$, negation gates are only applied on input wires. In other words, formula $\widetilde{f}_C$ can alternatively be interpreted as a monotone boolean formula being applied on the literals. (Recall that a literal is an atomic formula or its negation, i.e. either an input wire or its negation). With this observation, one could use KP-ABE scheme for monotone boolean formulas to obtain a KP-ABE scheme for $\mathbf{NC}^1$.

Concretely, to initialize the KP-ABE scheme for $\mathbf{NC}^1$ circuits with n-bit inputs, the setup algorithm runs the setup for the underlying KP-ABE scheme (for monotone boolean formulas) with $2n$-bit inputs. Intuitively, an attribute $x \in \{0,1\}^n$ will be encoded as string $y \in \{0,1\}^{2n}$ such that $y_{2i-1}, y_{2i} = 1, 0$ if $x_i = 0$, otherwise $y_{2i-1}, y_{2i} = 0, 1$ (for all $i \in [n]$). During encryption to attribute x, the encryptor computes the ciphertext with attribute string y as described above. And, the key generator first computes the reduced boolean formula $\widetilde{f}_C : \{0,1\}^n \to \{0,1\}$ for the predicate circuit C as described previously. Next, it computes a predicate key for a monotone boolean formula $g_C : \{0,1\}^{2n} \to \{0,1\}$ where the formula g_C is exactly the formula f_C except whenever f_C uses i-th input wire directly (i.e., x_i) then g_C uses $2i$-th input wire instead (i.e., y_{2i}), whereas wherever f_C uses the complete of i-th input wire (i.e., $\neg x_i$) then g_C uses $(2i-1)$-th input wire instead (i.e., y_{2i-1}). Note that correctness and IND-CPA security follows directly from the respective properties of the underlying KP-ABE system.

It turns out that the above transformation preserves the deletion properties as well. Thus, if we start with a deletable KP-ABE scheme for monotone boolean formulas, then the above folklore transformation gives a deletable KP-ABE scheme for $\mathbf{NC}^1$ circuits. This follows from the fact that now the deletion algorithm for the $\mathbf{NC}^1$ KP-ABE scheme deletes two underlying attributes instead of one. Concretely, on input an index set $\mathcal{I} \subseteq [n]$, the deletion algorithm simply runs the monotone boolean formula KP-ABE deletion algorithm with index set $\widetilde{\mathcal{I}} \subseteq [2n]$, where $\widetilde{\mathcal{I}} = \{2i-1, 2i\}_{i \in \mathcal{I}}$. In words, the deletion algorithm deletes encodings of both literals from the underlying ciphertext. Now the deletion indistinguishability of the new scheme follows from deletion indistinguishability of the underlying KP-ABE scheme.

Thus, combining this with Theorem 8.3, we get deletable KP-ABE schemes for $\mathbf{NC}^1$ circuits.

9 The Tsabary Framework

Our deletion framework builds upon the work of Tsabary [Tsa19]. In examining the work of [Tsa19] we noticed two points of ambiguity in the framework which we detail below. We believe that both points can be resolved by applying the correct interpretation or additional detail, however, we include the observations below for the sake of completeness. We communicated these points to the author.

Potential mismatch between conforming PRF definion and construction. While defining the concept of a conforming PRF in [Tsa19, Definition 3.1], the author use the circuit $U_{f \to x} : \{0,1\}^{\ell_f} \to \{0,1\}^k$ to denote the constrained PRF evaluation circuit that takes as input a constrained key sk_f and outputs a k-bit PRF output value on input x (which is hardwired). Later in [Tsa19, Section 3.1], they described their PRF construction wherein the evaluation algorithm states that — "If $f(x) = 0$ then abort, o.w. note that $S_x \subseteq S_f$ and compute r_x as in Eq. (4)".

At this point there is a mismatch between the definition and construction. The definition only allows the circuit to output k bits for the PRF output. However, the construction needs to be able to handle both a normal output as well as indicating an abort. There is no room to do both with just k bits of output. It is unclear what a circuit $U_{f \to x}$ should actually do when the abort condition ($f(x) = 0$) is triggered. Here are a few possibilities:

1. Have the circuit $U_{f \to x}$ output some arbitrary value whenever the abort condition is triggered. This will actually result in an insecure system. Suppose that whenever the abort is triggered then conforming PRF outputs a fixed value c. With high probability this will be different from the value 'r' from KeyGen on [Tsa19, page 14]. Thereby allowing one to decrypt when the abort condition is triggered.

2. Add an extra wire to indicate abort. So now the circuit will output $k+1$ bits; k normal output bits and an extra one to indicate whether abort was triggered. The decryption routine should then check both that $r \neq r'$ AND the abort flag was not triggered. We believe that this path should be able to make the framework work, however, it was not actually used.

3. On abort, one could set the output of the PRF to something like say the all zeros string 0^k. Then the logic of Section 4 could check that $r \neq r'$ AND the output is not 0^k. This is basically the same idea as above, but avoids the extra bit for aborting by designating the 0^k string as special in some way.

The main point is that the framework seems like it can be made to work with either the second or third strategy above (or maybe some other tweak). However, as it is there are ambiguities on what would actually be done and the writeup as it is does not address this.

Potential ambiguities in proving gradual evaluation. Similarly in [Tsa19, Definition 3.1], when the authors define the gradual evaluation property, they denote circuits $U_{\sigma\to x}$, $U_{\sigma\to f}$, $U_{f\to x}$ as performing the appropriate PRF constraining/computation, and it is crucial that the circuit obtained by composing the second and third circuits to be *identical* to the first circuit. Now in [Tsa19, Section 3.1], they provide their PRF construction and claim that the corresponding circuits satisfy this property by virtue of the fact that the evaluation circuit can be split into two layers.

At this point, there are some missing details about what the authors define the circuits to be, and how precisely these sub-circuits are set. In a little more detail, they state that in the circuit $U_{\sigma\to x}$, the first layer chooses secret keys depending upon the input x, while the second layer computes the PRF using them as well as XORing them in the end. And they claim that the circuit $U_{\sigma\to f}$ is the first layer, whereas $U_{f\to x}$ is the second layer. Now there are some subtle issues here:

1. The circuit $U_{\sigma\to f}$ which performs the constraining operation does not exactly correspond to the first layer, but instead some portion of the first layer. Recall that in their construction, $U_{\sigma\to f}$ selects some of the base PRF keys depending upon f. Whereas $U_{f\to x}$ selects the remaining of the base PRF keys depending upon input x, followed by layer 2 computation. Thus, layer 1 is split across these two circuits.

2. Thus, the proof of gradual evaluation needs to be more fine-grained where the authors must argue why composition of these two circuits matches the gate-by-gate description of $U_{\sigma\to x}$ circuit. Intuitively, the part that needs extra care is the one responsible for selecting a PRF key.

3. More broadly, the subtle issue with the above style of argument is that it is not clear what the underlying circuits are, and what does partial evaluation of a circuit look like (or, in other words, how is partial hardwiring in a circuit performed). For instance, If you had a circuit say (a OR b) AND (c OR d) and you hardwire say $a = 1$, then is the new circuit (c OR d) or is it (1 OR b) AND (c OR d). In the current description, the authors seem to implicitly assume a mechanism for constructing such circuits that provide the requisite structural guarantees, but they require additional interpretation from the reader.

Acknowledgements. We thank Benedikt Wagner for pointing out a useful subtlety in the circuit evaluation property in the definition of deletion conforming CPRFs.

References

[ABN+20] Shweta Agrawal, Rajarshi Biswas, Ryo Nishimaki, Keita Xagawa, Xiang Xie, and Shota Yamada. Attacks on boyen's attribute-based encryption scheme in tcc 2013. Personal communication, 2020.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. 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 557–577, 2014.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil Pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, 2001.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. 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 533–556, 2014.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *Public-Key Cryptography PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519. Springer Berlin Heidelberg, 2014.

[Boy13] Xavier Boyen. Attribute-based functional encryption on lattices. In Amit Sahai, editor, *Theory of Cryptography*, pages 122–142, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-abe from lwe: unbounded attributes and semi-adaptive security. In *Annual International Cryptology Conference*, pages 363–384. Springer, 2016.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300. Springer Berlin Heidelberg, 2013.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography, 1976.

[DKN$^+$20] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure constrained pseudorandom functions in the standard model. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 559–589. Springer, 2020.

[DKNY18] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, and Shota Yamada. Constrained prfs for bit-fixing (and more) from owfs with adaptive security and constant collusion resistance. Cryptology ePrint Archive, Report 2018/982, 2018.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT '06*, volume 4004 of LNCS, pages 445–464, 2006.

[GH09] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 437–456, 2009.

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, 2016.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *STOC*, 2018.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, 2006.

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

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Annual Cryptology Conference*, 2015.

[KNYY20] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure inner product encryption from LWE. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 375–404. Springer, 2020.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684. ACM, 2013.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588, 2011.

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 180–198, 2012.

[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.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[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.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

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

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-cnf from LWE. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, pages 62–85, 2019.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 616–637, 2014.

A Constrained Pseudorandomness to Key Simulation and More

First, we show that if the CPRF satisfies (0-key) pseudorandomness security, then it also satisfies the non-colliding property as long as the output length of the PRF is large enough.

Lemma A.1. If $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval})$ satisfies constrained pseudorandomness security (Definition 4.1) and $m = \omega(\log \lambda)$, where m is the output length and λ is the security parameter, then it also satisfies the non-colliding property.

Proof. Suppose the CPRF does not satisfy the non-colliding property, then we can construct a PPT adversary $\mathcal{A}$ which has non-negligible advantage in the pseudorandomness security game. For the sake of contradiction, consider that CPRF does not satisfy non-colliding property, and let input x^* and function f^* be such that

$$\Pr\left[\mathsf{Eval}(\mathsf{msk}, x^*) = \mathsf{Eval}(\mathsf{sk}'_f, x^*) : \begin{array}{c} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^N) \\ \mathsf{sk}'_f \leftarrow \mathsf{KeySim}(1^\lambda, 1^N, f^*) \end{array}\right] \geq \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\cdot)$.

Now consider the following attacker $\mathcal{A}$. $\mathcal{A}$ first samples a simulated key as $\mathsf{sk}^* \leftarrow \mathsf{KeySim}(1^\lambda, 1^N, f^*)$, and computes $z = \mathsf{Eval}(\mathsf{sk}^*, x^*)$. Next, it sends x^* as its challenge input to the CPRF challenger. Let y^* denote the challenger's response. The attacker $\mathcal{A}$'s checks if $y^* = z$, and outputs 1 as its guess (to denote y^* is honest evaluation) if the check passes. Otherwise, it outputs a random bit as its guess.

Let us next analyze $\mathcal{A}$'s advantage. First, note that if the challenger samples y^* uniformly at random, then $\Pr[\mathcal{A} \text{ outputs } 1] \leq \frac{1}{2} + \frac{1}{2^m}$. This follows from the fact that if $y^* \neq z$, then $\mathcal{A}$ outputs a random bit, and the event $y^* = z$ occurs with probability $\frac{1}{2^m}$. Second, note that if y^* is the PRF output, then we have that with at least ϵ probability it will be the case that $y^* = z$ (since CPRF does not satisfy non-colliding) and $\mathcal{A}$ outputs 1. And, with at most $1 - \epsilon$ probability, it outputs a random bit. Thus, we get that in this case, $\Pr[\mathcal{A} \text{ outputs } 1] \geq \frac{1}{2} + \frac{\epsilon}{2}$. Lastly, since we have that $m = \omega(\log \lambda)$, thus $\mathcal{A}$ has has non-negligible advantage $(\epsilon/2 - \mathsf{negl}(\lambda))$ in the pseudorandomness security game. Hence, the lemma follows. □

Key simulation from constrained pseudorandomness. Here we show that the adaptive single-key constrained pseudorandomness security in fact implies adaptive key simulation security.

Lemma A.2. If $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval})$ satisfies adaptive single-key constrained pseudorandomness security (Definition 4.1), then it also satisfies the adaptive key simulation security (Definition 4.2).

Proof. We first define the key simulation algorithm $\mathsf{KeySim}(1^\lambda, 1^N, f)$ to do the following:

— Generate a fresh master secret key msk' as $\mathsf{msk}' \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$.

— Output the simulated key as $\mathsf{sk}'_f \leftarrow \mathsf{Constrain}(\mathsf{msk}', f)$.

We define a sequence of games $\mathsf{Game}_0, \mathsf{Game}_1, \cdots, \mathsf{Game}_q$ where q is the total number of queries made by the adversary $\mathcal{A}$. If $\mathcal{A}$ can have a non-negligible advantage difference between Game_i and Game_{i+1}, then we can have an adversary $\mathcal{B}$ which runs $\mathcal{A}$ and has non-negligible advantage in the single-key adaptive security game for CPRF. We denote the advantage of adversary $\mathcal{A}$ in Game_i as $\mathsf{Adv}^i_{\mathcal{A}}$, which is $\Pr[\mathcal{A} \text{ wins } \mathsf{Game}_i] - \frac{1}{2}$.

Game_0 : This corresponds to the original key simulation security game, in which the adversary makes polynomially many (pre/post-challenge) evaluation queries $\{x_j\}_j$ such that $f^*(x_j) = 0$ for all j, where f^* is the challenge constraint adaptively chosen by $\mathcal{A}$.

Game_i : This is identical to Game_0, except the challenger now answers the first i queries with uniform random values $t \leftarrow \{0,1\}^m$, and the remaining queries are answered as before.

Claim A.3. If CPRF satisfies adaptive single-key constrained pseudorandomness security, then for any PPT adversary $\mathcal{A}$ there exists negligible function $\mathsf{negl}(\cdot)$, such that for every $\lambda, N \in \mathbb{N}, i \in [q]$, we have $\mathsf{Adv}_{\mathcal{A}}^{i-1} - \mathsf{Adv}_{\mathcal{A}}^{i} \leq \mathsf{negl}(\lambda)$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that can have a non-negligible advantage difference between Game_i and Game_{i+1}, then we can build a reduction algorithm $\mathcal{B}$ which has non-negligible advantage in the single-key adaptive security game for CPRF. (For ease of exposition, we assume throughout the proof that all the evaluation queries made by $\mathcal{A}$ are distinct. Note that this can be easily handled by the reduction algorithm by storing all the query-response pairs and responding to any new query by first checking if it was asked previously.)

The reduction algorithm $\mathcal{B}$ proceeds as follows. $\mathcal{B}$ answers the first $i-1$ evaluation queries made by $\mathcal{A}$ with uniform random values. For answering the i-th query x_i, $\mathcal{B}$ forwards it to the CPRF challenger as its challenge input. Let y^* be the challenger's response. $\mathcal{B}$ then sends y^* to $\mathcal{A}$ as the corresponding output. Next, to answer remaining evaluation queries ($i+1$ to q), $\mathcal{B}$ forwards $\mathcal{A}$'s evaluation query as its own evaluation query to the challenger and sends back the challenger's response to $\mathcal{A}$. For answering the key query f^* made by $\mathcal{A}$, $\mathcal{B}$ first samples a random bit $b \leftarrow \{0,1\}$. If $b = 0$, it forwards f^* to the CPRF challenger as its key query, and sends back the challenger's response to $\mathcal{A}$. Otherwise, it samples a simulated key as $\mathsf{sk}_{f^*} \leftarrow \mathsf{KeySim}(1^\lambda, 1^N, f^*)$, and sends sk_{f^*} to $\mathcal{A}$. (Note that the key query and evaluation queries could be arbitrarily interleaved.) Finally, $\mathcal{A}$ outputs its guess b'. If $b = b'$, then $\mathcal{B}$ outputs 0 as its guess (to signify that y^* was the PRF output). Otherwise, $\mathcal{B}$ outputs 1.

First, note that whenever $\mathcal{A}$ is an admissible adversary as per the key simulation game, then $\mathcal{B}$ is also an admissible adversary as per the constrained pseudorandomness game. This follows from the fact that $f^*(x_i) = 0$ for all inputs x_i queried by $\mathcal{A}$. Next, observe that if the challenger samples y^* as a uniform random value, then $\mathcal{B}$ perfectly simulates Game_i for $\mathcal{A}$, otherwise it simulates Game_{i-1}. Thus, if $\mathsf{Adv}_{\mathcal{A}}^{i-1} - \mathsf{Adv}_{\mathcal{A}}^{i}$ is non-negligible, then $\mathcal{B}$ also has non-negligible advantage. The claim follows. □

Claim A.4. For any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda, N \in \mathbb{N}$, we have that $\mathsf{Adv}_{\mathcal{A}}^{q}(\lambda) = 0$.

Proof. This follows from inspection and definition of our key simulation algorithm. Note that in Game_q, all queries are answered with uniform random values. Thus, all the query responses are independent of the master key. Since the simulated key and constrained key are sampled identically (with different keys), thus the corresponding distributions are identical. Hence, the adversary has no advantage in distinguishing. □

From Claim A.3 and Claim A.4, the Lemma A.2 follows. □

B KP-ABE and DCCPRFs to CP-ABE: Preserving Perfect Correctness

In this section, we give an alternate construction for constructing CP-ABE such that it satisfies perfect correctness. Note that the construction described in Section 6 does not achieve perfect correctness since with negligible probability it could happen that the simulated key used in the ciphertext evaluates to the tag value encoded in the secret key. Below we sketch a construction that avoids this problem. Our construction draws inspiration from the perfectly correct PLBE scheme of Goyal et al. [GKW18, Appendix B].

B.1 Construction

Let $\mathsf{DelABE} = (\mathsf{DelABE.Setup}, \mathsf{DelABE.KeyGen}, \mathsf{DelABE.Enc}, \mathsf{DelABE.Dec}, \mathsf{DelABE.Delete})$ be a KP-ABE scheme with deletable attributes for predicate class $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$, and $\mathsf{DCCPRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Constrain}, \mathsf{PRF.Eval}, \mathsf{PRF.CircuitGen}, \mathsf{PRF.DeleteFunc}, \mathsf{PRF.KeySim})$ be a deletion conforming CPRF for constraint class $\mathcal{F} = \{\mathcal{F}_N\}_{N\in\mathbb{N}}$. We require the predicate class $\mathcal{C}$ to be sufficiently expressive such that it contains circuits which perform comparison on top of a circuit generated by the $\mathsf{PRF.CircuitGen}$ algorithm. The requirement will become evident after the construction.

Below we describe our CP-ABE scheme $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for predicate class $\mathcal{F} = \{\mathcal{F}_N\}_{N\in\mathbb{N}}$.

$\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pp}, \mathsf{msk})$. The setup algorithm first runs DCCPRF setup to generate the corresponding master secret key: $\mathtt{prf}.\mathsf{msk} \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^N)$. Let $k = k(\lambda, N)$ denote the length of the master secret key $\mathtt{prf}.\mathsf{msk}$. Next, it runs the deletable ABE setup algorithm twice independently to obtain deletable ABE parameters as: $(\mathtt{del}.\mathsf{msk}_b, \mathtt{del}.\mathsf{pp}_b) \leftarrow \mathsf{DelABE.Setup}(1^\lambda, 1^k)$ for $b \in \{0,1\}$.

It sets public parameters and master key as $\mathsf{pp} = (\mathtt{del}.\mathsf{pp}_0, \mathtt{del}.\mathsf{pp}_1)$, $\mathsf{msk} = (\mathtt{prf}.\mathsf{msk}, \mathtt{del}.\mathsf{msk}_0, \mathtt{del}.\mathsf{msk}_1)$.

$\mathsf{KeyGen}(\mathsf{msk}, x) \to \mathsf{sk}$. Let $\mathsf{msk} = (\mathtt{prf}.\mathsf{msk}, \mathtt{del}.\mathsf{msk}_0, \mathtt{del}.\mathsf{msk}_1)$. The key generation algorithm first computes $t_0 = \mathsf{PRF.Eval}(\mathtt{prf}.\mathsf{msk}, x)$, sets $t_1 = \overline{t_0}$ (i.e., t_1 is bit-wise complement of t_0), and generates a circuit C_x as $C_x = \mathsf{PRF.CircuitGen}(x)$. Next, it samples two secret keys $\mathsf{sk}_0, \mathsf{sk}_1$ as $\mathsf{sk}_b \leftarrow \mathsf{DelABE.KeyGen}(\mathtt{del}.\mathsf{msk}_b, f_{x,t_b})$ for $b \in \{0,1\}$ (where f_{x,t_b} is as defined in Eq. (1)). Finally, it outputs $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$.

$\mathsf{Enc}(\mathsf{pp}, f, m) \to \mathsf{ct}$. Let $\mathsf{pp} = (\mathtt{del}.\mathsf{pp}_0, \mathtt{del}.\mathsf{pp}_1)$. The encryption algorithm runs the CPRF key simulation to generate a simulated key as $\mathsf{sk}'_f \leftarrow \mathsf{PRF.KeySim}(1^\lambda, 1^N, f)$. Next, it runs the deletable ABE encryption algorithm with attribute sk'_f as $\mathsf{ct}_b \leftarrow \mathsf{DelABE.Enc}(\mathtt{del}.\mathsf{pp}_b, m, \mathsf{sk}'_f)$ for $b \in \{0,1\}$, and outputs ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m$. Let $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$ and $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$. The decryption algorithm runs the deletable ABE decryption as $z_b = \mathsf{DelABE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b)$ for $b \in \{0,1\}$. It outputs z_0 as decryption output if $z_0 \neq \mathtt{fail}$, otherwise it outputs z_1.

Correctness. We start by proving that our construction satisfies the CP-ABE correctness condition.

Lemma B.1 (Correctness)**.** If the deletable KP-ABE scheme DelABE satisfies correctness, and the deletion conforming CPRF scheme DCCPRF satisfies non-colliding and circuit evaluation properties, then the CP-ABE scheme ABE described above is perfectly correct.

Proof. Fix any security parameter λ and attribute length N. For every predicate $f \in \mathcal{F}_N$, message $m \in \mathcal{M}$, and attribute $x \in \{0,1\}^N$, we have that the decryption algorithm Dec, on inputs ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ and secret key $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$, computes $z_b = \mathsf{DelABE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b)$ for both $b \in \{0,1\}$. Consider $(\mathtt{del}.\mathsf{msk}_b, \mathtt{del}.\mathsf{pp}_b)$ for $b \in \{0,1\}$ and $\mathtt{prf}.\mathsf{msk}$ to be the deletable KP-ABE and CPRF parameters sampled during setup. Note that the ciphertext ct_b is computed as $\mathsf{ct}_b \leftarrow \mathsf{DelABE.Enc}(\mathtt{del}.\mathsf{pp}_b, m, \mathsf{sk}'_f)$, where $\mathsf{sk}'_f \leftarrow \mathsf{PRF.KeySim}(1^\lambda, 1^N, f)$. Also, the secret key sk_b is sampled as $\mathsf{sk}_b \leftarrow \mathsf{DelABE.KeyGen}(\mathtt{del}.\mathsf{msk}, f_{x,t_b})$, where $t_0 = \mathsf{PRF.Eval}(\mathtt{prf}.\mathsf{msk}, x)$ and $t_1 = \overline{t_0}$. First, observe that by correctness of the deletable KP-ABE scheme, if $\mathsf{CEval}(f_{x,t_b}, \mathsf{sk}'_f) = 1$, then the decryption algorithm outputs message m correctly, i.e. $z_b = m$. Thus, to complete the completeness argument, we just need to show that whenever $f(x) = 1$, then $\mathsf{CEval}(f_{x,t_b}, \mathsf{sk}'_f) = 1$ for either $b = 0$ or $b = 1$.

Recall that circuit $f_{x,t_b}(\mathsf{sk}'_f) = 1$ if and only if $C_x(\mathsf{sk}'_f) \neq t_b$, where $C_x = \mathsf{PRF.CircuitGen}(x)$. Now if $f(x) = 1$, by the circuit evaluation property of deletion conforming CPRF, we get that $C_x(\mathsf{sk}'_f) = \mathsf{PRF.Eval}(\mathsf{sk}'_f, x)$. Since $t_0 = \mathsf{PRF.Eval}(\mathtt{prf}.\mathsf{msk}, x)$ and $t_1 = \overline{t_0}$, thus we have that $t_0 \neq t_1$, therefore we have that it can never happen that $C_x(\mathsf{sk}'_f) = t_0$ as well as $C_x(\mathsf{sk}'_f) = t_1$ simultaneously. Therefore, whenever $f(x) = 1$, the decryption algorithm always outputs message m. Hence perfect correctness follows. $\square$

Security proof sketch. The proof of security builds on the proof of Theorem 6.3 provided in Section 6.3. Here we sketch the high level idea. The proof again follows via a sequence of hybrid games. The first game corresponds to the original adaptive IND-CPA security game. As in the proof of Theorem 6.3, we first switch the constrained key sk_{f^*} (used in the challenge ciphertext) to be an honestly computed constrained key instead of a simulated key. This follows from adaptive key simulation security of the CPRFs. Next, we switch the way challenge ciphertext $\mathsf{ct}^* = (\mathsf{ct}_0^*, \mathsf{ct}_1^*)$ is computed. We start by computing ct_0^* as an encryption of all zeros string, instead of encrypting the challenge message honestly. However, ct_1^* still encrypts the challenge message. Such a hybrid jump is carried in a couple of steps where we start by relying on the deletion indistinguishability property of the deletable KP-ABE scheme to change the way ct_0^* is encrypted (i.e., we use ciphertext deletion algorithm), and then by using selective IND-CPA security of the KP-ABE scheme, we can switch ct^* from encrypting the challenge message to all zeros string.

Next, we switch the constrained key sk_{f^*} back to being a simulated key. Then we switch the way tags t_0 and t_1 are computed while answering any secret key query. Now we compute t_1 as $t_1 = \mathsf{PRF.Eval}(\mathtt{prf}.\mathsf{msk}, x)$ while setting $t_0 = \overline{t_1}$. By relying on pseudorandomness security of the CPRF scheme, we can argue that this is indistinguishable. Next, we use identical strategy (as used for ct_0^*) to turn ct_1^* into an encryption of an all zeros string. As before, this is performed by identical hybrid jumps where we first switch sk_{f^*} to be a honestly constrained PRF key, and then by relying on deletion indistinguishability and selective IND-CPA of KP-ABE, we can make ct_1^* to encrypt the all zeros string as well. Thus, in the final hybrid game, the challenge ciphertext $\mathsf{ct}^* = (\mathsf{ct}_0^*, \mathsf{ct}_1^*)$ is independent of the challenge messages. Therefore, adaptive IND-CPA security follows.

C Deletable ABE from CBDH

In this section, we describe a natural variant of the GPSW KP-ABE system [GPSW06] that has deletable attributes while also provably secure under the Computational Bilinear Diffie-Hellman (CBDH) assumption [BF01]. The idea is to simply use hardcore predicates on top of the GPSW scheme. Below we sketch the GPSW-based Key Encapsulation Mechanism whose security is based on the CBDH assumption.
The GPSW Setup and KeyGen algorithms are unchanged. Below we describe the key encapsulation algorithm.

$\mathsf{Enc}(\mathsf{pp}, x) \to (\mathsf{ct}, \mathsf{key})$. The algorithm parses the public parameters as $\mathsf{pp} = (g, K, \{T_i\}_{i\in[n]})$, and an attribute $x \in \{0, 1, \perp\}^n$. It chooses a random exponent $s \in \mathbb{Z}_p$, randomness r for the hardcore predicate, and publishes the ciphertext and KEM key as

$$\mathsf{ct} = (r, x, \{T_i^s\}_{i\in[n]:x_i=1}), \qquad \mathsf{key} = \mathsf{HardCore}(r, K^s),$$

where $\mathsf{HardCore}$ computes the hardcore bit.

Next, the deletion algorithm is defined analogous to as in Section 8.1.

$\mathsf{Delete}(\mathsf{pp}, \mathsf{ct}, \mathcal{I}) \to \mathsf{ct}'$. The algorithm parses the ciphertext as $\mathsf{ct} = (r, x, \{E_i\}_{i\in[n]:x_i=1})$. It sets the output ciphertext ct' as $\mathsf{ct}' = (r, \mathsf{Restrict}(x, \mathcal{I}), \{E_i\}_{i\in[n]\setminus\mathcal{I}:\ x_i=1})$.

Finally, the decryption is almost identical to GPSW decryption algorithm [GPSW06, Section 4.2], except the algorithm now outputs the extracted KEM key instead of a message.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to \mathsf{key}/\mathtt{fail}$. The algorithm parses a secret key as $\mathsf{sk}_f = \{D_i\}_i$ and a ciphertext $\mathsf{ct} = (r, x, \{E_i\}_{i\in[n]:x_i=1})$. Here the secret key sk_f contains a group element per each leaf node as per the description of its associated boolean formula f.

It runs the recursive $\mathsf{DecryptNode}$ procedure as described in the original GPSW construction using the ciphertext and secret key components $\{E_i\}_i$, $\{D_i\}_i$. Let F denote the final output of the $\mathsf{DecryptNode}$ procedure. The algorithm outputs $\mathtt{fail}$ if the $\mathsf{DecryptNode}$ procedure fails, otherwise it outputs the KEM key as $\mathsf{key} = \mathsf{HardCore}(F)$.

Correctness. The proof of correctness immediately follows from the correctness proof provided in [GPSW06, Section 4]. Note that the GPSW decryption algorithm first runs the decryption algorithm exactly as described above, and follows that by a single group operation in the target group. For our correctness proof, we can simply ignore the last group operation performed in GPSW decryption as the remaining is identical.

Security proof sketch. The proof of deletion indistinguishability is identical to that provided for Lemma 8.2. Now for proving selective IND-CPA security under CBDH assumption, we first show that any successful IND-CPA attacker on the above scheme can be reduced to obtain a successful attacker that can distinguish the hardcore bit of a BDH challenge from a random bit. Then such a distinguisher can be used to guess the BDH challenge with non-negligible probability by the standard hardcore bit extraction techniques.

D Deletable ABE from [Boy13]

In this section we show that [Boy13] is also a KP-ABE schemes with deletable attributes. As mentioned in the introduction, we want to remind the reader the existence of an attack [ABN+20] on Boyen's ABE scheme. Deletions in Boyen's scheme are provided for illustrative purposes, and to demonstrate wider applicability of our framework. To instantiate our framework under LWE, we believe that one could show the [BGG+14] scheme to be deletable.

D.1 Deletable ABE from LWE via [Boy13]

Boyen [Boy13] proposed a KP-ABE scheme for monotone access structures and proved its security under the Learning with Errors (LWE) assumption [Reg05]. Here we show that Boyen's scheme, described in [Boy13, Section 4], is also a deletable KP-ABE scheme. Let $\mathsf{Boyen} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ represent the KP-ABE construction provided in [Boy13, Section 4].

Now we describe our deletion algorithm for Boyen's scheme. As in Section 8.1, we switch the set notation for attributes to bit strings in the Boyen's original scheme. Also, throughout this section, we use ℓ to denote the length of attributes, and n instead denotes the dimension of the underlying lattice. Below we recall the Setup and Enc algorithms as provided in [Boy13, Section 4], and then describe our Delete algorithm.

$\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{pp}, \mathsf{msk})$. The setup algorithm first chooses LWE parameters (q, n, m, χ) such that they satisfy the required constraints.[15] It then proceeds as follows:

1. Run algorithm $\mathsf{TrapGen}$ to sample ℓ matrices $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ for $i \in [\ell]$ with trapdoor information as $(\mathbf{A}, T_i) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ for $i \in [\ell]$.
2. Select a uniformly random matrix as $\mathbf{A}_0 \leftarrow \mathbb{Z}_q^{n\times m}$.
3. Select a uniformly random vector as $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
4. Output the parameters as $\mathsf{pp} = (\{\mathbf{A}_i\}_{i\in[\ell]}, \mathbf{A}_0, \mathbf{u})$, and $\mathsf{msk} = (\{T_i\}_{i\in[\ell]})$.[16]

$\mathsf{Enc}(\mathsf{pp}, x, \mathsf{msg}) \to \mathsf{ct}$. The encryption algorithm parses the public parameters as $\mathsf{pp} = (\{\mathbf{A}_i\}_{i\in[\ell]}, \mathbf{A}_0, \mathbf{u})$, and an attribute $x \in \{0,1\}^\ell$. It proceeds as follows:

1. Assemble an "encryption matrix" $\mathbf{F} \in \mathbb{Z}_q^{n\times(\ell+1)m}$, obtained as the concatenation of, for each $i \in [\ell]$, either $\mathbf{A}_i$ if $x_i = 1$, or $\mathbf{0}$ if $x_i = 0$, and matrix $\mathbf{A}_0$, as follows,

$$\mathbf{F} = [\mathbf{F}_1 \mid \cdots \mid \mathbf{F}_\ell \mid \mathbf{F}_0], \qquad \text{where } \forall i \in [\ell],\ \mathbf{F}_i = \begin{cases} \mathbf{A}_i & \text{if } x_i = 1, \\ \mathbf{0} & \text{if } x_i = 0. \end{cases} \tag{4}$$

2. Select a uniformly random vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$.

[15] We suggest the reader to look at [Boy13] for discussion on how to sample the LWE parameters.

[16] The parameters also contain the LWE parameters, but here we don't explicitly write it for simplicity.

3. Select a low-norm Gaussian noise scalar $\nu_0 \leftarrow \chi$, and then compute:
$$c_0 = (\mathbf{s}^\top \cdot \mathbf{u} + \nu_0 + \lfloor \frac{q}{2} \rfloor \cdot \mathsf{msg}) \bmod q.$$
4. Select a low-norm Gaussian noise vector $\boldsymbol{\nu}_1 \leftarrow \chi^{(\ell+1)m}$, where each component is sampled i.i.d. according to the noise distribution χ, and computes:
$$\mathbf{c}_1 = (\mathbf{s}^\top \cdot \mathbf{F} + \boldsymbol{\nu}_1) \bmod q.$$
5. Output the ciphertext $\mathsf{ct} = (c_0, \mathbf{c}_1)$.

Encrypting to attributes with $\perp$ symbols. As in the case of our AugGPSW scheme, our augmented encryption algorithm for Boyen's scheme treats $\perp$ symbols as a 0 bit during encryption. Concretely, it assembles matrix $\mathbf{F}$ as described in Eq. (4), except it sets the sub-matrices $\mathbf{F}_i$ as:
$$\forall i \in [\ell], \; \mathbf{F}_i = \begin{cases} \mathbf{A}_i & \text{if } x_i = 1, \\ \mathbf{0} & \text{if } x_i \in \{0, \perp\}. \end{cases}$$
The remaining encryption procedure proceeds identically. However, here the deletion algorithm does not simply *delete* the ciphertext components in vector $\mathbf{c}_1$ wherever $i \in \mathcal{I}$, but instead replaces the corresponding vector components with freshly sampled noise terms. Formally, we describe it below.

$\mathsf{Delete}(\mathsf{pp}, \mathsf{ct}, \mathcal{I}) \to \mathsf{ct}'$. The algorithm parses the ciphertext as $\mathsf{ct} = (c_0, \mathbf{c}_1)$, where $\mathbf{c}_1 \in \mathbb{Z}_q^{(\ell+1)m}$. It proceeds as follows:

1. Parse the vector $\mathbf{c}_1$ as $[\mathbf{c}_{1,1} \mid \cdots \mid \mathbf{c}_{1,\ell} \mid \mathbf{c}_{1,0}]$ where each sub-vector $\mathbf{c}_{1,i} \in \mathbb{Z}_q^m$. (That is, view $\mathbf{c}_1$ as a concatenation of $(\ell+1)$ blocks of length m vectors.)
2. Compute a vector $\mathbf{c}'_1 \in \mathbb{Z}_q^{(\ell+1)m}$ as follows:
$$\mathbf{c}'_1 = \left[\mathbf{c}'_{1,1} \mid \cdots \mid \mathbf{c}'_{1,\ell} \mid \mathbf{c}_{1,0}\right], \qquad \text{where} \quad \begin{array}{ll} \forall i \in \mathcal{I}, & \mathbf{c}'_{1,i} \leftarrow \chi^m, \\ \forall i \in [\ell] \setminus \mathcal{I}, & \mathbf{c}'_{1,i} = \mathbf{c}_{1,i}. \end{array}$$
3. Output the deleted ciphertext as $\mathsf{ct}' = (c_0, \mathbf{c}'_1)$.

Deletion Indistinguishability. First, we show that the augmented GPSW scheme AugBoyen = (Setup, KeyGen, Enc, Dec, Delete) satisfies the deletion indistinguishability property. As before, we prove a much stronger statement which in turn implies deletion indistinguishability. Intuitively, we argue that, for every choice of system parameters, the distribution of a freshly encrypted ciphertext and a (corresponding) deleted ciphertext are *identical*.

Lemma D.1. For every $\lambda, \ell \in \mathbb{N}$, parameters $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, attribute $x \in \{0,1\}^\ell$, message $\mathsf{msg} \in \{0,1\}$, and index set $\mathcal{I} \in [\ell]$, the following two distributions are identical:
$$\mathcal{D}_1 = \left\{ \mathsf{ct} : \begin{array}{c} x' = \mathsf{Restrict}(x, \mathcal{I}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x', m) \end{array} \right\}, \qquad \mathcal{D}_2 = \left\{ \mathsf{ct}' : \begin{array}{c} \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m) \\ \mathsf{ct}' \leftarrow \mathsf{Delete}(\mathsf{pp}, \mathsf{ct}, \mathcal{I}) \end{array} \right\}.$$
That is, $\mathcal{D}_1 \equiv \mathcal{D}_2$.

Proof. The proof of this lemma immediately follows by inspection of the encryption and deletion algorithms described above. Consider any λ, ℓ, key pair $(\mathsf{pp}, \mathsf{msk})$, attribute x, message msg and index set $\mathcal{I}$. First, we expand out the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ as follows:

$$\mathcal{D}_1 = \left\{ (c_0, [\mathbf{c}_{1,1} \mid \cdots \mid \mathbf{c}_{1,\ell} \mid \mathbf{c}_{1,0}]) : \begin{array}{l} \mathbf{s} \leftarrow \mathbb{Z}_q^n,\ \nu_0 \leftarrow \chi,\ \boldsymbol{\nu}_{1,i} \leftarrow \chi^m (\forall i \in [0,\ell]), \\ x' = \mathsf{Restrict}(x, \mathcal{I}), S_1 = \{i \in [\ell] : x'_i = 1\}, \\ c_0 = \mathbf{s}^\top \cdot \mathbf{u} + \nu_0 + \lfloor \frac{q}{2} \rfloor \cdot \mathsf{msg}, \\ \forall i \in \{0\} \cup S_1,\ \mathbf{c}_{1,i} = \mathbf{s}^\top \cdot \mathbf{A}_i + \boldsymbol{\nu}_{1,i}, \\ \forall i \in [\ell] \setminus S_1,\ \mathbf{c}_{1,i} = \boldsymbol{\nu}_{1,i} \end{array} \right\},$$

$$\mathcal{D}_2 = \left\{ \left(c_0, \left[\mathbf{c}'_{1,1} \mid \cdots \mid \mathbf{c}'_{1,\ell} \mid \mathbf{c}_{1,0}\right]\right) : \begin{array}{l} \mathbf{s} \leftarrow \mathbb{Z}_q^n,\ \nu_0 \leftarrow \chi,\ \boldsymbol{\nu}_{1,i} \leftarrow \chi^m (\forall i \in [0,\ell]), \\ S_2 = \{i \in [\ell] : x_i = 1\}, \\ c_0 = \mathbf{s}^\top \cdot \mathbf{u} + \nu_0 + \lfloor \frac{q}{2} \rfloor \cdot \mathsf{msg}, \\ \forall i \in \{0\} \cup S_2,\ \mathbf{c}_{1,i} = \mathbf{s}^\top \cdot \mathbf{A}_i + \boldsymbol{\nu}_{1,i}, \\ \forall i \in [\ell] \setminus S_2,\ \mathbf{c}_{1,i} = \boldsymbol{\nu}_{1,i}, \\ \forall i \in \mathcal{I},\ \mathbf{c}'_{1,i} \leftarrow \chi^m, \\ \forall i \in [\ell] \setminus \mathcal{I},\ \mathbf{c}'_{1,i} = \mathbf{c}_{1,i}. \end{array} \right\}$$

Recall by definition of Restrict, we have that $x'_i = 1$ if and only if $x_i = 1$ and $i \notin \mathcal{I}$. Therefore, it follows that $\mathcal{D}_1 \equiv \mathcal{D}_2$ since the vectors $\mathbf{c}_{1,i}$ for $i \in \{0\} \cup S_1$ are identically computed as $\mathbf{s}^\top \cdot \mathbf{A}_i + \boldsymbol{\nu}_{1,i}$, and the vectors $\mathbf{c}_{1,i}$ for $i \in [\ell] \setminus S_1$ are sampled i.i.d. according to the noise distribution χ in both $\mathcal{D}_1$ and $\mathcal{D}_2$. Here the last statement follows from the fact that since each original noise value is sampled i.i.d, thus they can be replaced with newly sampled noise values. This completes the proof. □

Correctness. Note that since a deleted ciphertext is *identically* distributed to a freshly encrypted ciphertext, and also Boyen is a statistically correct ABE scheme, thus correctness of our AugBoyen scheme follows.

Selective IND-CPA Security. As in the case of AugGPSW, selective IND-CPA security of our AugBoyen scheme follows from selective IND-CPA security proof of Boyen. Thus, if [Boy13] is secure, then so is the above ABE scheme with deletable attributes.