Realizing Chosen Ciphertext Security Generically in Attribute-Based Encryption and Predicate Encryption

Venkata Koppula
Weizmann Institute of Science[*]

Brent Waters
UT Austin[†]

February 25, 2019

Abstract

We provide generic and black box transformations from any chosen plaintext secure Attribute-Based Encryption (ABE) or One-sided Predicate Encryption system into a chosen ciphertext secure system. Our transformation requires only the IND-CPA security of the original ABE scheme coupled with a pseudorandom generator (PRG) with a special security property.

In particular, we consider a PRG with an n bit input $s \in \{0,1\}^n$ and $n \cdot \ell$ bit output $y_1, \ldots, y_n$ where each y_i is an ℓ bit string. Then for a randomly chosen s the following two distributions should be computationally indistinguishable. In the first distribution $r_{i,s_i} = y_i$ and $r_{i,\bar{s}_i}$ is chosen randomly for $i \in [n]$. In the second distribution all $r_{i,b}$ are chosen randomly for $i \in [n], b \in \{0,1\}$.

1 Introduction

In Attribute-Based Encryption [SW05] (ABE) every ciphertext CT that encrypts a message m is associated with an attribute string x, while each secret, as issued by an authority, will be associated with a predicate function C. A user with a secret key sk that is associated with function C will be able to decrypt a ciphertext associated with x and recover the message if and only if $C(x) = 1$. Additionally, security of ABE systems guarantees that an attacker with access to several keys cannot learn the contents of an encrypted message so long as none of them are so authorized.

Since the introduction of Attribute-Based Encryption and early constructions [GPSW06] over a decade ago, there have been many advances in the field ranging from supporting expressive functionality [GVW13, BGG$^+$14], to techniques for adaptive security[Wat09, LOS$^+$10, OT10, LW12, Att14, Wee14, CGW15], short sized ciphertexts[ALdP11], multi-authority [Cha07, CC09, LW11] and partially hiding attributes [GVW15, GKW17, WZ17] to name just a few. In almost all of these cases and in most other papers, the treatment of ABE focused on the chosen plaintext (IND-CPA) definition of ABE. This is despite the fact that chosen ciphertext security [NY90, RS91, DDN91] — where the attacker can make oracle decryption queries to keys it does not have — is arguably the right definition of security for the same reasons it is the right definition for standard public key cryptography[Sho98]. Likely, most of these works target IND-CPA security since the authors already have their hands full with putting forth new concepts and techniques in manuscripts that often run for many pages. In these circumstances it seems reasonable for such works to initially target chosen plaintext definitions and then for later works to circle back and build toward chosen ciphertext security.

Unfortunately, closing the loop to chosen ciphertext security can be tricky in practice. First, there are a rather large and growing number of ABE constructions. Writing papers to address moving each of these to chosen ciphertext security seems burdensome to authors and program committees alike. One line of work [GPSW06, YAHK11, NP15, BL16] to mediate this problem is to identify features in ABE constructions,

[*]E-mail:`venkata.koppula@weizmann.ac.il`

[†]E-mail:`bwaters@cs.utexas.edu`. Research supported by NSF CNS-1414082, DARPA SafeWare, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

which if present mean that CPA security implies chosen ciphertext security. Yamada et. al [YAHK11] showed that certain delegability or verifiability properties in ABE systems imply chosen ciphertext security by the Canetti-Halevi-Katz[CHK04] transformation.

Their generality, however, is limited by the need to manually inspect and prove that each construction has such a property. In fact, many schemes might not have these properties. Recent trends for both functionality and proofs techniques might actually work against these properties. For example, an ABE scheme has the verification property roughly if it is possible to inspect a ciphertext and determine if it is well formed and what keys can decrypt it. This property emerged naturally in many of the pairing-based schemes prominent at the time, but is less obvious to prove in LWE-based constructions and actually can runs contrary to the predicate encryption goal of hiding an attribute string x from users that cannot decrypt. See for example the one-sided predicate encryption constructions of [GVW15, GKW17, WZ17].

If we desire a truly generic transformation to chosen ciphertext security, then there are essentially two pathways available. The first option is to apply some variant of the Fujisaki-Okamoto [FO99] transformation (first given for transforming from IND-CPA to IND-CCA security in public key encryption). Roughly, the encryption algorithm will encrypt as its message the true message m appended with a random bitstring r using the random coins $H(r)$ where H is a hash function modeled as a random oracle. The CCA-secure decryption algorithm will apply the original decryption algorithm to a ciphertext CT and recover $m'|r'$. Next, it re-encrypts the ciphertext under $H(r')$ to get a ciphertext CT' and outputs the message if $\mathrm{CT} = \mathrm{CT}'$; otherwise it rejects. The upside of this approach is that the added overhead is fairly low as it just adds one additional call to encryption as part of the decryption routine. On the downside the security analysis of this technique appears intrinsically tied to the random oracle model [BR93].

The second option is to augment encryption by appending a non-interactive zero knowledge proof [BFM88] that a ciphertext was well formed. This approach has been well studied and explored in the context of standard public key encryption [NY90] and should translate to the ABE context. Additionally, there are standard model NIZK proof assumptions under factoring and pairing-based and lattice based [PS19] assumptions.[1] A drawback of this approach is that applying any generic gate by gate NIZK to an encryption system will be quite expensive in terms of computational overhead— this will be needed for any generic conversion.

1.1 Our Contribution

In this work we provide a black box transformation for chosen ciphertext security of any ABE or one-sided predicate encryption system.[2]

Our transformation requires only the existence of a IND-CPA secure ABE system as well as a pseudorandom generator (PRG) with a special security property which we call the *hinting property.* This special security property can either be assumed for an "ordinary" (e.g., AES-based) PRG or provably obtained from either the Computational Diffie-Hellman assumption or the Learning with Errors assumption. Our transformation increases ciphertext size by roughly a factor of the security parameter — it requires $2 \cdot n$ sub-ciphertexts for a parameter n. Additionally, it requires about $2n$ additional encryptions of the original system for both the new encryption and decryption routines. While this overhead is an increase over the original CPA system and will likely incur more overhead than hand-tailored CCA systems, it is a significant performance improvement over NIZKs that operate gate by gate over the original encryption circuit.

We also wish to emphasize that our transformation applies to ordinary public key encryption as well as ABE. While chosen ciphertext security for PKE has been known for sometime from the CDH and LWE assumptions, we believe that our work provides new insights into the problem and might lead to furthering the understanding of whether IND-CPA security ultimately implies chosen ciphertext security.

[1]The realization of NIZKs from the Learning with Errors assumption is a very recent and exciting development[PS19] and occured after the initial posting of this work.

[2]The original definition of predicate encryption [BW07, KSW08] required hiding whether an attribute string of a challenge ciphertext was x_0 or x_1 from an attacker that had a key C where $C(x_0) = C(x_1)$. A weaker form of predicate encryption is where this guarantee is given only if $C(x_0) = C(x_1) = 0$, but not when $C(x_0) = C(x_1) = 1$. This weaker form has been called predicate encryption with one-sided security and anonymous Attribute-Based Encryption. For this paper we will use the term one-sided predicate encryption.

Hinting Property for PRGs : Let PRG be a function that maps n bits to $(n+1)\cdot n$ bits (output to be parsed as $n+1$ strings, each of length n). Consider the following experiment between a challenger and an adversary. The challenger chooses an n bit string s, computes $\mathsf{PRG}(s) = z_0 z_1 z_2 \ldots z_n$ (each $z_i \in \{0,1\}^n$). Next, it chooses n uniformly random strings $v_1, v_2, \ldots, v_n$ each from $\{0,1\}^n$. It then constructs a $2\times n$ matrix M as follows: if the i^{th} bit of s is 0, then $M_{0,i} = z_i, M_{1,i} = v_i$, else $M_{0,i} = v_i, M_{1,i} = z_i$.[3] Finally, the challenger either outputs z_0 together with M, or it outputs $2n+1$ uniformly random strings. A pseudorandom generator is said to satisfy the hinting property if any polynomial time adversary has negligible advantage in this experiment. Note that the seed s is used at two places : first to compute the strings $z_0, z_1, z_2, \ldots, z_n$, and then to decide where to place each z_i in the matrix M. Hence, the second piece of information (i.e. the position of z_i strings serves as an extra *hint* on the PRG). One could simply assume this property of a particular pseudo random generator. Indeed, this seems rather plausible that ordinary types of PRGs would have it. Alternately, we show how to construct PRGs that provably have this property under either the Computational Diffie-Hellman assumption or the LWE assumption. Our constructions of these PRGs use techniques that closely follow previous works [DG17a, DG17b, BLSV18, DGHM18, GH18] for a related group of primitives going under a variety of names: Chameleon Encryption, One-Time Signature with Encryption, Batch Encryption, One Way Function with Encryption. We note that while the technical innards for the CDH and LWE realizations of our PRG are similar to the above works, (unlike the above examples) our definition itself does not attach any new functionality requirements to PRG; it simply demands a stronger security property.

Next, we present an overview of our CCA construction. As a warm-up, we will first show how to use any CPA-secure public key encryption (PKE) scheme, together with hinting PRGs to construct a CCA-1 secure PKE scheme. Recall, CCA-1 security is a weaker variant of the CCA security game where the adversary is allowed decryption queries only before sending the challenge messages. After sending the challenge messages, the adversary receives the challenge ciphertext, and must send its guess.

CCA-1 secure PKE from CPA-secure PKE and hinting PRGs The construction also uses a (standard) pseudorandom generator G with sufficiently long stretch. Let $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be any CPA secure scheme, and $H : \{0,1\}^n \to \{0,1\}^{(n+1)\cdot n}$ a hinting PRG. We require the CPA scheme to have two properties which can be obtained 'for free'. First, we require that the scheme should have perfect decryption correctness for most public/secret keys. This can be obtained via the transformation of [DNR04]. Next, we require that any ciphertext can be decrypted given the encryption randomness. This is also easily obtained by choosing a random string r during encryption, and appending a one-time pad of the message using r.

The setup of our CCA-1 scheme runs the PKE setup $2n$ times, obtaining $2n$ public key/secret key pairs $\left\{\mathsf{pk}_{b,i}, \mathsf{sk}_{b,i}\right\}_{i\in[n], b\in\{0,1\}}$. The new public key consists of the $2n$ public keys $\left\{\mathsf{pk}_{b,i}\right\}_{i\in[n],b\in\{0,1\}}$, while the new secret key includes only of n out of the $2n$ secret keys, namely $\{\mathsf{sk}_{0,i}\}_{i\in[n]}$ (this *secret hiding principle*[FS90] has been used in many CCA constructions, including the initial CCA systems [NY90, DDN91]). To encrypt a message m, the encryption algorithm first chooses a uniformly random tag $t = t_1 t_2 \ldots t_n$, where each t_i is a sufficiently long string. Next, it chooses a seed $s \leftarrow \{0,1\}^n$ and computes $H(s) = z_0 z_1 \ldots z_n$. It uses z_0 to mask the message m; that is, it computes $c = m \oplus z_0$. The remaining ciphertext will contain n 'signals' that help the decryption algorithm to recover s bit by bit, which in turn will allow it to compute z_0 and hence unmask c.

The i^{th} signal (for each $i \in [n]$) has three components $c_{0,i}, c_{1,i}, c_{2,i}$. If the i^{th} bit of s is 0, then $c_{0,i}$ is an encryption of a random string x_i using the public key $\mathsf{pk}_{0,i}$ and randomness z_i, $c_{1,i}$ is an encryption of 0^n using $\mathsf{pk}_{1,i}$ (encrypted using true randomness), and $c_{2,i} = G(x_i)$. If the i^{th} bit of s is 1, then $c_{0,i}$ is an encryption of 0^n using public key $\mathsf{pk}_{0,i}$ (encrypted using true randomness), $c_{0,i}$ is an encryption of a random string x_i using public key $\mathsf{pk}_{1,i}$ and randomness z_i, and $c_{2,i} = G(x_i) + t_i$ (recall t_i is the i^{th} component in the tag). So half the ciphertexts are encryptions of zero, while the remaining are encryptions of random strings (with blocks of the hinting PRG output being used as randomness), and the positioning of the zero/random encryptions reveals the seed s.

[3]More compactly, $M_{s_i,i} = z_i$ and $M_{\overline{s_i},i} = v_i$.

The final ciphertext includes the tag $t = (t_1 t_2 \dots t_n)$, the 'main' component c, and n signals $(c_{0,i}, c_{1,i}, c_{2,i})$. A noteworthy point about the ciphertext: first, the components $\{c_{2,i}\}_i$ serve as a perfectly-binding commitment to the seed s. However, this commitment also has certain 'non-malleability' feature; given a ciphertext/commitment with tag t, an adversary cannot obtain a ciphertext/commitment for some other tag t' - this feature is necessary for the scheme to be CCA secure.

To decrypt, the decryption algorithm first decrypts each $c_{0,i}$ (recall the secret key is $\{\mathsf{sk}_{0,i}\}_{i \in [n]}$) to obtain $y_1 y_2 \dots y_n$. It then checks if $G(y_i) = c_{2,i}$. If so, it guesses that $s_i = 0$, else it guesses that $s_i = 1$. With this estimate for s, the decryption algorithm can compute $H(s) = z_0 z_1 \dots z_n$ and then compute $c \oplus z_0$ to learn the message m. While this decryption procedure works correctly, we would like to prevent malicious decryption queries (made during the CCA/CCA-1 experiment), and hence the decryption algorithm needs to enforce additional checks. In particular, the decryption algorithm therefore needs to check that the guess for s is indeed correct. If the i^{th} bit of s is guessed to be 0, then the decryption algorithm checks that $c_{0,i}$ is a valid ciphertext - it simply re-encrypts y_i and checks if this equals $c_{0,i}$. If the i^{th} bit of s is guessed to be 1, then the decryption algorithm first recovers the messaage underlying ciphertext $c_{1,i}$. Note that $c_{1,i}$ should be encrypted using randomness z_i, hence using z_i, one can recover message $\tilde{y}_i$ from $c_{1,i}$ (using the randomness recovery property of the PKE scheme). It then re-encrypts $\tilde{y}_i$ and checks if it is equal to $c_{1,i}$, and also checks that $c_{2,i} = G(\tilde{y}_i) + t_i$. Finally, if all these checks pass, the decryption algorithm outputs $z_0 \oplus c$.

To summarize, at a very high level, we build a partial trapdoor where the decryption algorithm will recover some of the coins used for encryption. These are then used to partially re-encrypt the ciphertext and test for validity. Influenced by Garg and Hajiabadi[GH18], we will prove security not by removing the signals for each bit position, but by adding misinformation that drowns out the original signal. Note that to prove security, we need to remove the information of z_0 (and hence information of s) from two places - first, from the commitment $\{c_{2,i}\}_i$; second, from the positions where the z_i values are used for encrypting. For the first one, in the proof, we set the challenge tag t^* such that the signal is ambiguous at each index. More formally, in the challenge ciphertext $c^*_{0,i}$ is an encryption of y_i, $c^*_{1,i}$ is encryption of $\tilde{y}_i$, and $G(y_i) = G(\tilde{y}_i) + t^*_i$. Replacing the encryptions of zeroes with encryptions of these carefully chosen strings involves a delicate argument, which crucially relies on the perfect correctness of our scheme (see discussion in Section 4 for more details).

When this is done for all indices, all information about s will be lost from the message space and we are almost done; however, one loose end remains. Each ciphertext at position (s_i, i) will be encrypted under randomness r_i which came from running the pseudorandom generator on s; whereas each ciphertext at position $(\bar{s}_i, i)$ will be encrypted under fresh random coins. To complete the proof we need a computational assumption that will allow us to change all the encryption coins to being chosen freshly at random. Here, we use the security of hinting PRGs, and that completes our proof.

CCA Security To achieve CCA security, we need to make a few tweaks to the above scheme. The setup algorithm also chooses n pairwise independent hash functions $h_1, h_2, \dots, h_n$. The encryption algorithm chooses a signing/verification key for a (one-time) signature scheme. Next, instead of choosing a tag t uniformly at random, it sets $t_i = h_i(\mathsf{vk})$ (where vk is the verification key). Finally, the encryption algorithm computes a signature on all the ciphertext components, and the final ciphertext consists of all these components together with the signature and the verification key. To prove security, we first ensure that none of the decryption queries correspond to the challenge ciphertext's verification key (this follows from the security of the signature scheme). After this point, the proof follows along the lines of the CCA-1 scheme.

Moving to Attribute Based Encryption/Predicate Encryption - For ABE/PE, the scheme is altered as follows. First, the public key consists of n ABE/PE public keys and n PKE public keys. Let $\mathsf{pk}_{0,i}$ denote the i^{th} ABE/PE public key, and $\mathsf{pk}_{1,i}$ the i^{th} PKE public key. The master secret key only consists of the n ABE/PE master secret keys. The main difference in the encryption algorithm is that the ciphertexts $c_{0,i}$ are now ABE/PE ciphertexts. Suppose we want to encrypt message m for attribute x. Then m is masked using z_0 as before, and the $c_{0,i}$ component is an encryption of zero/random message for attribute x using public key $\mathsf{pk}_{0,i}$ and randomness being truly random/z_i, depending on the i^{th} bit of seed s.

We conclude by remarking that while this work focuses on Attribute-Based Encryption and One-sided Predicate Encryption, we believe our transformation could apply to other specialized forms of encryption. For example, we believe it should immediately translate to any secure broadcast encryption [FN94] system. As another example, we believe our technique should also apply to ABE systems that are IND-CPA secure under a bounded number of key generation queries. Our technique, however, does not appear to apply to standard predicate encryption as defined in [BW07, KSW08] (notions very similar to full blown functional encryption). The core issue is that to test the validity of a ciphertext our decryption algorithm needs to obtain the attribute string x to perform re-encryption. In one-sided predicate encryption, if a user has a secret key for C and $C(x) = 1$ we essentially give up on hiding x and allow this to be recovered; whereas for full hiding we might want to still hide information about x even if $C(x) = 1$.

Finally, we note that even if we cast the notions of ABE aside our work might provide another path to exploring the longstanding open problem of achieving chosen ciphertext security from chosen plaintext security. The primary barrier is in how to achieve a PRG with this hinting security.

1.2 Additional Comparisons

It is instructive to take a closer look at how our work relates to and builds upon the trapdoor function construction of Garg and Hajiabadi[GH18]. Briefly and in our terminology, Garg and Hajiabadi gave a framework where the evaluation algorithm chooses an input $s \in \{0,1\}^n$ and use this to first produces a value y that produces part of the output. Next, for each position (s_i, i) the evaluation algorithm produces a signal using s and the public parameters of the TDF using a primitive called "one way function with encryption". At the opposite position $(\bar{s}_i, i)$ the evaluation algorithm outputs a random string r_i of sufficient length. With very high probability the random string z_i will not correspond to the valid signal for y at position $(\bar{s}_i, i)$. The inversion algorithm will use knowledge of the TDF secret key plus y to go recover the input s bit by bit. At each position i if a signal is present at $(0, i)$ it records $s_i = 0$ and sets $s_i = 1$ if the signal is at $(1, i)$. If the signal is at both 0 and 1, then recovery fails. One can observe that for almost all choices of public parameters there exist some valid inputs that will cause failure on inversion. To prove security the reduction algorithm at each position change the string z_i from random to a signal under y. The security properties of the one way function with encryption make this undetectable. Once, this is done the only information about s will be contained in y. Since many choices of s will map to y, inverting to the chosen s at this point will be statistically infeasible.

Our work as described above follows a similar approach in that a seed s is signaled bit by bit. And that a step of proving security is to add misinformation in by adding a counter signal in at positions $(\bar{s}_i, i)$. An important distinction is that in the work of Garg and Hajiabadi the signaling and inversion process is very tightly coupled in the one way function with encryption primitive. One could imagine trying to build an Attribute-Based version of one way function with encryption and then try to yield a CCA encryption from the resulting trapdoor. This runs into two problems. First, it would require a tailored construction for each type of ABE scheme that we want and then we are back to hacking CCA into each type of ABE variant. Second, since the GH scheme allows for ambiguous inputs, it can be difficult for mapping into chosen ciphertext secure schemes. In particular, this issue caused GH to need an adaptive version of one way function with encryption to bridge from TDFs to CCA security and this adaptive version was not realizable from the CDH assumption.

In our work the signaling strategy is decoupled from the recovery of the signals. In particular, the form of the signals comes from our computation of the (non-hinting) PRG, while recovery is realized from simply invoking the ABE decryption algorithm. We also get perfect correctness since a non-signal will be an encryption of the all 0's string. Also, with high probability our setup algorithm will choose parameters for which it is (information theoretically) impossible to create ambiguous signals. So once the ABE parameters are setup by an honest party (and with overwhelming probability, land in a good spot), there will be no further opportunity to take advantage of conflicting signals by an attacker via a decryption query.

We also believe that it might be interesting to swing some of our techniques back to the trapdoor function regime. For example, consider the GH TDF, but where we added values $a_1, \ldots, a_n$ to the public parameters. We could modify the evaluation algorithm such that at position i, the algorithm gives the one-way function with encryption output e_i if $s_i = 0$ and gives $e_i \oplus a_i$ if $s_i = 1$. This modification would allow us to drop the additional z_i values from the GH construction and make the output of the TDF shorter. In addition, while there would still be a negligible correctness error, it could be possible to rest this error solely in the choice of public parameters and for a "good" choice of parameters there would be no further error from evaluation. This last claim would require making sure that the a_i values were sufficiently long relative to y. We believe the techniques from [RS10] can be used here to achieve CCA security.

We finally remark again that our realizations of our hinting PRG largely follow in line with recent works [DG17a, DG17b, BLSV18, DGHM18, GH18]. In particular, our CDH realization follows closely to [DG17a] and our LWE realization to [BLSV18, DGHM18]. It may have been possible to build our hinting PRG from one of the previous abstractions, but we chose to provide direct number theoretic realizations. We believe that one important distinction is that our hinting PRG is simply a PRG with stronger security properties; unlike the above abstractions our definition in of itself does not ask for expanded functionality. An intriguing open question is if this can be leveraged to obtain further instances with provable security.

Independent Work Independently, Garg, Gay and Hajiabadi [GGH18] recently built upon the work of [GH18] to build trapdoors from one way function with encryption that has improved correctness properties. In particular, the base construction of [GGH18] generates parameters that with high probability will allow inversion on all inputs, whereas any parameters generated from the [GH18] construction will always have inversion failure on some small fraction of inputs. They then build upon using erasure codes and a "smoothness" property to get CCA secure determinstic encryption with shorter ciphertexts. In addition, they show a modification to the Peikert-Waters [PW08] DDH-based Lossy TDF that gets a better ciphertext rate. The initial direction of getting better correctness in TDFs is similar to our "swinging techniques back" comment above, but otherwise the works pursue separate goals and techniques.

1.3 Toward Bridging Chosen Ciphertext Security in PKE

One classical open problem in cryptography is whether chosen plaintext security implies chosen ciphertext security in standard public key encryption. From a cursory glance one can see that it is easy to swap out the ABE system from our construction for a plain old public key encryption system and the same proof will go through — this time for obtaining chosen ciphertext secure public key encryption. Thus the "only" barrier for moving from IND-CPA to IND-CCA security is in the hinting PRG.

An interesting open question is just how strong this barrier is. From our viewpoint, the hinting security is something that most natural PRGs would likely have. In trying to understand whether it or something similar could be built from general assumptions (e.g. PKE or one way functions) it could be useful to first try to build a separation from our hinting PRG and a standard one. *Do there exist PRGs that do not meet the security definition of hinting PRG?*

As a first stab at the problem, one might consider PRGs where there is an initial trusted setup algorithm that produces a set of public parameters, which are then used for every subsequent evaluation. In this setting one could imagine a counterexample where the public parameters produced by the setup algorithm include an obfuscated program which will assist in breaking the hinting security, but not be helpful enough to break standard security. Using obfuscation in a similar manner has been useful for achieving other separation results. If we consider PRGs that do not allow for such setup, the task appears to be more challenging. One could try to embed such an obfuscated program in the first block of the PRG output, but this block would need to still look random for standard PRG security.

However, as it turns out there is a much simpler way to achieve a separation. Consider the case where $\ell = 1$ then the identity function on the seed will be a pseudorandom function for the trivial reason that it does not expand. However, this function will not be hinting secure. To get a separation with expansion one can build on this to consider a PRG where the seed is split into two parts; where in one part of the seed has each of its bits given out in the clear exactly once. And where the rest of the bits of the output are generated pseudorandomly from the rest of the seed.

Altogether we believe that our work opens up a new avenue for exploring the connection of chosen plaintext and ciphertext security.

2 One-sided Predicate Encryption

A predicate encryption (PE) scheme $\mathcal{PE}$, for set of attribute spaces $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, predicate classes $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ and message spaces $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$, consists of four polytime algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ with the following syntax.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and a description of attribute space $\mathcal{X}_\lambda$, predicate class $\mathcal{C}_\lambda$ and message space $\mathcal{M}_\lambda$, and outputs the public parameters pp and the master secret key msk.

$\mathsf{Enc}(\mathsf{pp}, m, x) \to \mathsf{ct}$. The encryption algorithm takes as input public parameters pp, a message $m \in \mathcal{M}_\lambda$ and an attribute $x \in \mathcal{X}_\lambda$. It outputs a ciphertext ct.

$\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm takes as input master secret key msk and a predicate $C \in \mathcal{C}_\lambda$. It outputs a secret key sk_C.

$\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) \to m$ or $\perp$. The decryption algorithm takes as input a secret key sk_C and a ciphertext ct. It outputs either a message $m \in \mathcal{M}_\lambda$ or a special symbol $\perp$.

Correctness. A key-policy predicate encryption scheme is said to be correct if for all $\lambda \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, for all $x \in \mathcal{X}_\lambda$, $C \in \mathcal{C}_\lambda$, $m \in \mathcal{M}_\lambda$, $\mathsf{sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C)$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, m, x)$, the following holds

$$\begin{aligned} \text{Correctness for decryptable ciphertexts} : C(x) = 1 &\Rightarrow \Pr\left[\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) = m\right] = 1, \\ \text{Correctness for non-decryptable ciphertexts} : C(x) = 0 &\Rightarrow \Pr\left[\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) = \perp\right] \geq 1 - \mathsf{negl}(\lambda), \end{aligned}$$

where $\mathsf{negl}(\cdot)$ are negligible functions, and the probabilities are taken over the random coins used during key generation and encryption procedures.

Recovery from Randomness Property. A key-policy predicate encryption scheme is said to have *recovery from randomness property* if there is an additional algorithm $\mathsf{Recover}$ that takes as input public parameters pp, ciphertext ct, string r and outputs $y \in (\mathcal{M}_\lambda \times \mathcal{X}_\lambda) \cup \{\perp\}$ and satisfies the following condition: for all $\lambda \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, for all $x \in \mathcal{X}_\lambda$, $m \in \mathcal{M}_\lambda$, $\mathsf{ct} = \mathsf{Enc}(\mathsf{pp}, m, x; r)$, $\mathsf{Recover}(\mathsf{pp}, \mathsf{ct}, r) = (m, x)$. If there is no (m, x, r) tuple such that $\mathsf{ct} = \mathsf{Enc}(\mathsf{pp}, m, x; r)$, then $\mathsf{Recover}(\mathsf{pp}, \mathsf{ct}, r) = \perp$.

Security. In this work, we will be considering predicate encryption systems with one-sided security. One can consider both security against *chosen plaintext attacks* and *chosen ciphertext attacks*. First, we will present one-sided security against chosen plaintext attacks.

Definition 2.1 (One-Sided Security against Chosen Plaintext Attacks)**.** A predicate encryption scheme $\mathcal{PE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be one-sided secure against chosen plaintext attacks if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that the following holds:

$$\left| \Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ ((m_0, x_0), (m_1, x_1)) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{pp}) \\ b \leftarrow \{0, 1\};\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, m_b, x_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

where every predicate query C, made by adversary $\mathcal{A}$ to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle, must satisfy the condition that $C(x_0) = C(x_1) = 0$.

The notion of one-sided security against chosen plaintext attacks could alternatively be captured by a simulation based definition [GVW15]. Goyal et al. [GKW17] showed that if a PE scheme satisfies Definition 2.1, then it also satisfies the simulation based definition of [GVW15].

Next, we present the definition for capturing chosen ciphertext attacks on predicate encryption schemes. Here, we will assume that the key generation algorithm is deterministic.

Definition 2.2 (One-Sided Security against Chosen Ciphertext Attacks)**.** A predicate encryption scheme $\mathcal{PE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ with deterministic key generation is said to be one-sided secure against chosen ciphertext attacks if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that the following quantity

$$\left| \Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathcal{O}_{\mathrm{Dec}}(\mathsf{msk},\cdot,\cdot)}(\mathsf{ct}^*) = b : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ ((m_0, x_0), (m_1, x_1)) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathcal{O}_{\mathrm{Dec}}(\mathsf{msk},\cdot,\cdot)}(\mathsf{pp}) \\ b \leftarrow \{0,1\};\ \mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, m_b, x_b) \end{array} \right] - \frac{1}{2} \right|$$

is at most $\mathsf{negl}(\lambda)$, where

- the oracle $\mathcal{O}_{\mathrm{Dec}}(\mathsf{msk}, \cdot, \cdot)$ takes as input a ciphertext ct and a circuit C. It computes $\mathsf{sk}_C = \mathsf{KeyGen}(\mathsf{msk}, C)$ and outputs $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct})$.
- every predicate query C, made by adversary $\mathcal{A}$ to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle, must satisfy the condition that $C(x_0) = C(x_1) = 0$.
- every post-challenge query (C, ct) made by the adversary $\mathcal{A}$ to $\mathcal{O}_{\mathrm{Dec}}$ must satisfy the condition that either $\mathsf{ct} \neq \mathsf{ct}^*$ or if $\mathsf{ct} = \mathsf{ct}^*$, then $C(x_0) = C(x_1) = 0$.

Remark 2.1. Note that the above definition addresses chosen ciphertext attacks against systems with deterministic key generation. An analogous definition for general schemes (that is, with randomized key generation) would involve maintaining key handles and allowing the adversary to choose the key to be used for the decryption queries. We choose the simpler definition since any scheme's key generation can be made deterministic by using a pseudorandom function. In particular, the setup algorithm chooses a PRF key K which is included as part of the master secret key. To derive a key for circuit C, the algorithm first computes $r = \mathrm{PRF}(K, C)$ and then uses r as randomness for the randomized key generation algorithm.

2.1 ABE schemes with 'recovery from randomness' property

Any ABE scheme satisfying one-sided CPA security can be transformed into another ABE scheme that is also one-sided CPA secure, and has the 'recovery from randomness' property. The encryption algorithm simply uses part of the randomness to compute a symmetric key encryption of the message and attribute, with part of the randomness as the encryption key.

More formally, let $E = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ be an ABE scheme that satisfies one-sided CPA security (see Definition 2.1), and let $(\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a symmetric key CPA secure encryption schceme. consider the following scheme $E' = (\mathsf{Setup}', \mathsf{Enc}', \mathsf{KeyGen}', \mathsf{Dec}', \mathsf{Recover})$, where $\mathsf{Setup}' = \mathsf{Setup}$ and $\mathsf{KeyGen}' = \mathsf{KeyGen}$.

$\mathsf{Enc}'(\mathsf{pk}, m, x)$: The encryption algorithm first samples three random strings r_1, r_2, r_3. It computes $\mathsf{ct}_1 = \mathsf{Enc}(\mathsf{pk}, m, x; r_1)$. Next, it computes $\mathsf{ske.sk} = \mathsf{SKE.Setup}(1^\lambda; r_2)$. Finally, it computes $\mathsf{ct}_2 = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (m, x); r_3)$ and outputs $(\mathsf{ct}_1, \mathsf{ct}_2)$.

$\mathsf{Dec}'(\mathsf{sk}, (\mathsf{ct}_1, \mathsf{ct}_2))$: The decryption algorithm simply decrypts ct_1 using sk, and ignores ct_2. It outputs $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_1)$.

$\mathsf{Recover}((\mathsf{ct}_1, \mathsf{ct}_2), r = (r_1, r_2, r_3))$: The recovery algorithm first computes $\mathsf{ske.sk} = \mathsf{SKE.Setup}(1^\lambda; r_2)$. It outputs $y \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ct}_2)$.

Assuming the symmetric key encryption scheme satisfies perfect correctness, this ABE scheme has perfect recovery from randomness property. To argue CPA security, we can first use the security of the SKE scheme to switch ct_2 to an encryption of $0^{|m|+|x|}$. Then, we can use the one-sided CPA security.

3 Hinting PRGs

A hinting PRG scheme is a PRG with a stronger security guarantee than standard PRGs. A hinting PRG takes n bits as input, and outputs $n \cdot \ell$ output bits. In this security game, the challenger outputs $2n$ strings, each of ℓ bits. In one scenario, all these $2n$ strings are uniformly random. In the other case, half the strings are obtained from the PRG evaluation, and the remaining half are uniformly random. Moreover, these $2n$ strings are output as a $2 \times n$ matrix, where in the i^{th} column, the top entry is pseudorandom if the i^{th} bit of the seed is 0, else the bottom entry is pseudorandom. As a result, these $2n$ strings give a 'hint' about the seed, and hence this property is stronger than regular PRGs. Note, if this hint is removed and the top entries in each column were pseudorandom (and the rest uniformly random), then this can be achieved using regular PRGs.

Below, we define this primitive formally. The informal description above had two simplifications. First, the definition below considers PRGs with setup (although one can analogously define such a primitive without setup). Second, we assume the PRG outputs $(n+1) \cdot \ell$ bits, where the first ℓ bits do not contain any extra hint about the seed. Finally, for our CCA application, we introduce some notation in order to represent the $n+1$ blocks of the PRG output. Instead of describing the PRG as a function that outputs $(n+1) \cdot \ell$ bits, we have an evaluation algorithm that takes as input an index $i \in \{0, 1, \ldots, n\}$, and outputs the i^{th} block of the PRG output.

Let $n(\cdot, \cdot)$ be a polynomial. A n-hinting PRG scheme consists of two PPT algorithms $\mathsf{Setup}, \mathsf{Eval}$ with the following syntax.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm takes as input the security parameter λ, and length parameter ℓ, and outputs public parameters pp and input length $n = n(\lambda, \ell)$.

$\mathsf{Eval}(\mathsf{pp}, s \in \{0,1\}^n, i \in [n] \cup \{0\})$: The evaluation algorithm takes as input the public parameters pp, an n bit string s, an index $i \in [n] \cup \{0\}$ and outputs an ℓ bit string y.

Definition 3.1. A hinting PRG scheme $(\mathsf{Setup}, \mathsf{Eval})$ is said to be secure if for any PPT adverasry $\mathcal{A}$, polynomial $\ell(\cdot)$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left| \Pr\left[1 \leftarrow A\left(\mathsf{pp}, \left(y_0^\beta, \left\{y_{i,b}^\beta\right\}_{i \in [n], b \in \{0,1\}}\right)\right) : \begin{array}{c} (\mathsf{pp}, n) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell(\lambda)}), s \leftarrow \{0,1\}^n, \\ \beta \leftarrow \{0,1\}, y_0^0 \leftarrow \{0,1\}^\ell, y_0^1 = \mathsf{Eval}(\mathsf{pp}, s, 0), \\ y_{i,b}^0 \leftarrow \{0,1\}^\ell \ \forall\, i \in [n], b \in \{0,1\}, \\ y_{i,s_i}^1 = \mathsf{Eval}(\mathsf{pp}, s, i), y_{i,\overline{s_i}}^1 \leftarrow \{0,1\}^\ell \ \forall\, i \in [n] \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\cdot)$$

4 CCA Secure Public Key Encryption Scheme

Let $\mathsf{PKE}_{\mathsf{CPA}} = (\mathsf{CPA.Setup}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec})$ be a $\mathsf{IND\text{-}CPA}$ secure public key encryption scheme with randomness-decryptable ciphertexts and perfect decryption correctness, $\mathsf{S} = (\mathsf{ss.Setup}, \mathsf{ss.Sign}, \mathsf{ss.Verify})$ a strongly unforgeable one time signature scheme and $\mathsf{HPRG} = (\mathsf{HPRG.Setup}, \mathsf{HPRG.Eval})$ a hinting PRG scheme. We will assume that our encryption scheme has message space $\{0,1\}^{\lambda+1}$. Let $\ell_{\mathsf{PKE}}(\cdot)$ be a polynomial repesenting the number of bits of randomness used by $\mathsf{CPA.Enc}$, and $\ell_{\mathsf{vk}}(\cdot)$ the size of verification keys output by $\mathsf{ss.Setup}$. For simplicity of notation, we will assume $\ell(\cdot) = \ell_{\mathsf{PKE}}(\cdot)$,[4] $\ell_{\text{out}}(\lambda) = \ell_{\mathsf{vk}}(\lambda) + 3\lambda$ and $\mathsf{PRG}_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell_{\text{out}}(\lambda)}$ a family of secure pseudorandom generators.

We will now describe our CCA secure public key encryption scheme $\mathsf{PKE}_{\mathsf{CCA}} = (\mathsf{CCA.Setup}, \mathsf{CCA.Enc}, \mathsf{CCA.Dec})$ with message space $\mathcal{M}_\lambda = \{0,1\}^{\ell(\lambda)}$. For simplicity of notation, we will skip the dependence of ℓ and ℓ_{out} on λ.

$\mathsf{CCA.Setup}(1^\lambda)$: The setup algorithm performs the following steps.

[4]Alternatively, we could set ℓ to be max of these two polynomials.

1. It chooses $(\mathsf{HPRG.pp}, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
2. It chooses $2n$ different $\mathsf{PKE}_{\mathsf{CPA}}$ keys. Let $(\mathsf{cpa.sk}_{b,i}, \mathsf{cpa.pk}_{b,i}) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$ for each $b \in \{0,1\}$, $i \in [n]$.
3. It then chooses $a_i \leftarrow \{0,1\}^{\ell_{\text{out}}}$ for each $i \in [n]$ and $B \leftarrow \{0,1\}^{\ell_{\text{out}}}$.
4. It sets $\mathsf{cca.pk} = \left(\mathsf{HPRG.pp}, B, \left(a_i, \mathsf{cpa.pk}_{b,i}\right)_{b\in\{0,1\}, i\in[n]}\right)$ and $\mathsf{cca.sk} = (\mathsf{cpa.sk}_{0,i})_{i\in[n]}$.

$\mathsf{CCA.Enc}(\mathsf{cca.pk}, m, x)$: Let $\mathsf{cca.pk} = \left(\mathsf{HPRG.pp}, B, \left(a_i, \mathsf{cpa.pk}_{b,i}\right)_{b\in\{0,1\}, i\in[n]}\right)$. The encryption algorithm does the following:

1. It first chooses $s \leftarrow \{0,1\}^n$. It sets $c = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m$.
2. Next, it chooses signature keys $(\mathsf{ss.sk}, \mathsf{ss.vk}) \leftarrow \mathsf{ss.Setup}(1^\lambda)$.
3. For each $i \in [n]$, it chooses $v_i \leftarrow \{0,1\}^\lambda$ and $r_i \leftarrow \{0,1\}^\ell$, sets $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i)$.
4. Next, for each $i \in [n]$, it does the following:
   - If $s_i = 0$, it sets $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{0,i}, 1|v_i; \widetilde{r_i})$, $c_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{1,i}, 0^{\lambda+1}; r_i)$ and $c_{2,i} = \mathsf{PRG}(v_i)$.
   - If $s_i = 1$, it sets $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{0,i}, 0^{\lambda+1}; r_i)$, $c_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{1,i}, 1|v_i; \widetilde{r_i})$ and $c_{2,i} = \mathsf{PRG}(v_i) + a_i + B \cdot \mathsf{ss.vk}$.[5].
5. Finally, it sets $M = \left(c, (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\right)$, computes $\sigma \leftarrow \mathsf{ss.Sign}(\mathsf{ss.sk}, M)$ and outputs $(\mathsf{ss.vk}, M, \sigma)$ as the ciphertext.

$\mathsf{CCA.Dec}(\mathsf{cca.sk}, \mathsf{cca.pk}, \mathsf{cca.ct})$: Let the ciphertext $\mathsf{cca.ct}$ be parsed as $\left(\mathsf{ss.vk}, \left(c, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\right), \sigma\right)$ and $\mathsf{cca.sk} = \left((\mathsf{cpa.sk}_{0,i})_{i\in[n]}\right)$. The decryption algorithm does the following:

1. It first verifies the signature σ. It checks if $\mathsf{ss.Verify}(\mathsf{ss.vk}, M, \sigma) = 1$, else it outputs $\perp$.
2. Next, the decryption algorithm computes $d = \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \mathsf{cca.ct})$ (where $\mathsf{PKE.Find}$ is defined in Figure 1), and outputs $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$ (where $\mathsf{PKE.Check}$ is defined in Figure 2).

```
                    PKE.Find(cca.pk, cca.sk, cca.ct)

Inputs: Public Key $\mathsf{cca.pk} = \left(\mathsf{HPRG.pp}, B, \left(a_i, \mathsf{cpa.pk}_{b,i}\right)_{b\in\{0,1\}, i\in[n]}\right)$
        Secret Key $\mathsf{cca.sk} = (\mathsf{cpa.sk}_{0,i})_{i\in[n]}$
        Ciphertext $\mathsf{cca.ct} = \left(\mathsf{ss.vk}, \left(c, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\right), \sigma\right)$
Output: $d \in \{0,1\}^n$

• For each $i \in [n]$, do the following:
    1. Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}_{0,i}, c_{0,i})$.
    2. If $m_i = 1|v_i$ and $\mathsf{PRG}(v_i) = c_{2,i}$, set $d_i = 0$. Else set $d_i = 1$.
• Output $d = d_1 d_2 \ldots d_n$.
```

Figure 1: Routine $\mathsf{PKE.Find}$

[5]Here, we assume the verification key is embedded in $\mathbb{F}_{2^{\ell_{\text{out}}}}$, and the addition and multiplication are performed in $\mathbb{F}_{2^{\ell_{\text{out}}}}$.

```
PKE.Check(cca.pk, cca.ct, $d$)

Inputs: Public Key $\mathsf{cca.pk} = \Big(\mathsf{HPRG.pp}, B, \big(a_i, \mathsf{cpa.pk}_{b,i}\big)_{b\in\{0,1\}, i\in[n]}\Big)$
        Ciphertext $\mathsf{cca.ct} = \Big(\mathsf{ss.vk}, \Big(c, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\Big), \sigma\Big)$
        $d \in \{0,1\}^n$
Output: $\mathsf{msg} \in \{0,1\}^\ell$

• Let $\mathsf{flag} = \mathsf{true}$. For $i = 1$ to $n$, do the following:
    1. Let $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, i)$.
    2. If $d_i = 0$, let $y \leftarrow \mathsf{CPA.Recover}(\mathsf{cpa.pk}_{0,i}, c_{0,i}, \widetilde{r_i})$. Perform the following checks. If any of the
       checks fail, set $\mathsf{flag} = \mathsf{false}$ and exit loop.
        – $m \neq \perp$.
        – $\mathsf{CPA.Enc}(\mathsf{cpa.pk}_{0,i}, y; \widetilde{r_i}) = c_{0,i}$.
        – $m = 1|v$ and $\mathsf{PRG}(v) = c_{2,i}$.
    3. If $d_i = 1$, let $m \leftarrow \mathsf{CPA.Recover}(\mathsf{cpa.pk}_{1,i}, c_{1,i}, \widetilde{r_i})$. Perform the following checks. If any of the
       checks fail, set $\mathsf{flag} = \mathsf{false}$ and exit loop.
        – $m \neq \perp$.
        – $\mathsf{CPA.Enc}(\mathsf{cpa.pk}_{1,i}, m; \widetilde{r_i}) = c_{1,i}$.
        – $m = 1|v$ and $c_{2,i} = \mathsf{PRG}(v) + a_i + B \cdot \mathsf{ss.vk}$.
• If $\mathsf{flag} = \mathsf{true}$, output $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, 0)$. Else output $\perp$.
```

Figure 2: Routine PKE.Check

4.1 Discussion

We will now make a few observations about our construction and then proceed to give a brief overview our proof that appears in the next subsection.

First, for each $i \in [n]$ if $s_i = 0$ the encryption algorithm will choose a random v_i and signal that this bit is a 0 by encrypting $1|v_i$ to the position $(0, i)$ and giving $c_{2,i} = \mathsf{PRG}(v_i)$ in the clear. In the opposite position of $(1, i)$ it will encrypt the all 0's string. Likewise, if $s_i = 1$ it will signal a 1 by encrypting $1|v_i$ to the position $(1, i)$ and giving $c_{2,i} = \mathsf{PRG}(v_i) + a_i + B \cdot \mathsf{ss.vk}$ in the clear. With all but negligible probability it is impossible to signal both 0 and 1 simultaneously for an index i. This follows from the fact that a_i is chosen randomly and that the space of verification keys is much smaller than $2^{\ell_{\text{out}}(\lambda)}$. We observe that this argument has some flavor of Naor's bit commitment scheme [Nao89].

Second, we observe that even though one is supposed to encrypt the all 0's string to position $(\bar{s}_i, i)$ the PKE.Find routine will not immediately quit if discovers something else. Instead it simply sets $d_i = 0$ if decryption outputs $1|v_i$ and $c_{2,i} = \mathsf{PRG}(v_i)$; otherwise it sets $d_i = 1$. Thus, the decryption routine may refrain from immediately aborting even though when it "knows" the ciphertext was not formed entirely correctly. This will be critical to a proof step.

Our proof of security will be organized as a sequence of security games which we show to be indistinguishable. In the first proof step we apply a standard argument using strongly unforgeable signatures to change the decryption oracle to reject all ciphertexts corresponding to $\mathsf{ss.vk}^*$ where $\mathsf{ss.vk}^*$ is the verification key used by the challenge ciphertext.

Next, for each i we choose the public parameter values a_i such that it is possible for one to signal both 0 and 1 at index i, but that this ambiguity is only possible for a ciphertext corresponding to $\mathsf{ss.vk}^*$. To do this it chooses uniformly random $w_i \leftarrow \{0,1\}^\lambda$, and sets $a_i = \mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i) - \mathsf{ss.vk}^* \cdot B$ if $s_i^* = 0$, else $a_i = \mathsf{PRG}(w_i) - \mathsf{PRG}(v_i^*) - \mathsf{ss.vk}^* \cdot B$. This change can be shown to be undetectable by a standard pseudorandom generator argument. The effect of this change is that it allows the possibility of ambiguous signaling at both 0 and 1 in the challenge ciphertext. However, for all possible decryption queries where $\mathsf{ss.vk} \neq \mathsf{ss.vk}^*$ this remains impossible.

Our next goal will be to use the IND-CPA security of the underlying PKE scheme to introduce signals on the opposite path $\overline{s^*}$. To do this, however, for all i where $s_i^* = 1$ we must first change the decryption routine to use $\mathsf{cpa.sk}_{1,i}$ to decrypt the sub-ciphertext at position $(1, i)$ instead of using $\mathsf{cpa.sk}_{0,i}$ (at position $(0, i)$). Consider a particular ciphertext query and let d_i be the bit reported by the original find algorithm on that ciphertext query and d'_i be the bit reported by a the new decryption procedure on that same ciphertext. We want to argue that if $d_i \neq d'_i$ then the $\mathsf{PKE.Check}$ procedure will abort and output $\perp$ on both encryptions. The first possibility is that $d_i = 0$ and $d'_i = 1$; however, that should be information theoretically impossible as it would entail signaling both a 0 and 1 for a query with $\mathsf{ss.vk} \neq \mathsf{ss.vk}^*$. The other possibility is that $d_i = 1$ and $d'_i = 0$; i.e. that there is not a signal present at either side. In this case the first decryption routine will have $d_i = 1$, but then when running $\mathsf{PKE.Check}$ it will fail to find a signal at position $(1, i)$ and abort. Likewise, the second decryption routine will have $d'_i = 0$, but then fail to find a signal at position $(0, i)$, so both routines will behave identically in this case as well.

Once the oracle decryption is set to follow the seed s^* we can straightforwardly use CPA security to introduce ambiguous signals in the messages for all positions $(\overline{s^*}_i, i)$. Once this change is made we can change the oracle decryption routine again. This time it will only decrypt at positions $(1, i)$ for all $i \in [n]$ and only use $\mathsf{cpa.sk}_{1,i}$. A similar argument to before can be applied to make this change.

All information about s is gone except to the lingering amount in the random coins used to encrypt. We can immediately apply the hinting PRG to change to a game where these values can be moved to be uniformly at random. At this point the message will be hidden.

4.2 Security Proof

We will now show that the above construction satisfies the CCA security definition. Our proof proceeds via a sequence of hybrids. First, we will describe all hybrids, and then show that the hybrids are computationally indistinguishable.

4.2.1 Hybrids

Hybrid H_0 : This corresponds to the original security game.

- **Setup Phase**
  1. The challenger first chooses $(\mathsf{HPRG.pp}, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
  2. Next it chooses $2n$ different $\mathsf{PKE_{CPA}}$ keys. Let $(\mathsf{cpa.sk}_{b,i}, \mathsf{cpa.pk}_{b,i}) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$ for each $i \in [n]$, $b \in \{0, 1\}$.
  3. The challenger chooses $s^* \leftarrow \{0, 1\}^n$, $v_i^* \leftarrow \{0, 1\}^\lambda$ for each $i \in [n]$, and $(\mathsf{ss.sk}^*, \mathsf{ss.vk}^*) \leftarrow \mathsf{ss.Setup}(1^\lambda)$. It sets $\widetilde{r_i}^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$. (These components will be used during the challenge phase.)
  4. It then chooses $a_i \leftarrow \{0, 1\}^{\ell_{\text{out}}}$ for each $i \in [n]$ and $B \leftarrow \{0, 1\}^{\ell_{\text{out}}}$.
  5. It sends $\mathsf{cca.pk} = \Big(\mathsf{HPRG.pp}, B, \big(a_i, \mathsf{cpa.pk}_{b,i}\big)_{b \in \{0,1\}, i \in [n]}\Big)$ to $\mathcal{A}$, and sets the secret key $\mathsf{cca.sk} = (\mathsf{cpa.sk}_{0,i})_{i \in [n]}$.

- **Pre-challenge Query Phase**
  - *Decryption Queries*
    1. For each query $\left(\mathsf{ct} = \left(\mathsf{ss.vk}, M = \left(c, (c_{0,i}, c_{1,i}, c_{2,i})_i\right), \sigma\right)\right)$, the challenger first checks the signature σ.
    2. Next, the challenger first computes $d = \mathsf{PKE.Find}\,(\mathsf{cca.pk}, \mathsf{cca.sk}, \mathsf{cca.ct})$.
    3. It outputs $\mathsf{PKE.Check}\,(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$.

- **Challenge Phase**
  1. The adversary sends two challenge messages (m_0^*, m_1^*).
  2. The challenger chooses a bit $\beta \in \{0,1\}$.
  3. It sets $c^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m_\beta^*$.
  4. It sets $(c_{0,i}^*, c_{1,i}^*, c_{2,i}^*)$ as follows.
     - If $s_i^* = 0$, it sets $c_{0,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{0,i}, 1|v_i^*; \widetilde{r_i}^*)$, $c_{1,i}^* \leftarrow \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{1,i}, 0^{\lambda+1})$ and $c_{2,i}^* = \mathsf{PRG}(v_i^*)$.
     - If $s_i^* = 1$, it sets $c_{0,i}^* \leftarrow \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{0,i}, 0^{\lambda+1})$, $c_{1,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{1,i}, 1|v_i^*; \widetilde{r_i}^*)$ and $c_{2,i}^* = \mathsf{PRG}(v_i^*) + a_i + B \cdot \mathsf{ss.vk}^*$.
  5. Finally, it computes a signature σ^* on $M^* = \left(c^*, \left(c_{0,i}^*, c_{1,i}^*, c_{2,i}^*\right)\right)$ using $\mathsf{ss.sk}^*$ and sends $(\mathsf{ss.vk}^*, M, \sigma^*)$ to $\mathcal{A}$.

- **Post-challenge Query Phase**
  - *Decryption Queries* These are handled as in the pre-challenge phase, with the restriction that all queries (ct, C) must satisfy that $\mathsf{ct} \neq \mathsf{ct}^*$.

- Finally, the adversary sends its guess b.

Hybrid H_1 : This hybrid is similar to the previous one, except that during the decryption queries, the challenger checks if $\mathsf{ss.vk} = \mathsf{ss.vk}^*$. If so, it rejects.

Hybrid H_2 : In this hybrid, the challenger changes Step 4 of the setup phase. It chooses uniformly random $w_i \leftarrow \{0,1\}^\lambda$, and sets $a_i = \mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i) - \mathsf{ss.vk}^* \cdot B$ if $s_i^* = 0$, else $a_i = \mathsf{PRG}(w_i) - \mathsf{PRG}(v_i^*) - \mathsf{ss.vk}^* \cdot B$.

Hybrid H_3 : This hybrid is similar to the previous one, except that the challenger modifies the way decryption queries are handled. Instead of using $\mathsf{PKE.Find}$, the challenger uses $\mathsf{PKE.Find\text{-}1}$ (defined in Figure 3). The $\mathsf{PKE.Find}$ routine decrypts only the $c_{0,i}$ values. If decryption works, it guesses $d_i = 0$, else it guesses $d_i = 1$. The $\mathsf{PKE.Find\text{-}1}$ routine decrypts either $c_{0,i}$ or $c_{1,i}$, depending on the i^{th} bit of s^*. Note that the $\mathsf{PKE.Check}$ routine is identical in both experiments.

Hybrid H_4 : In this step, the challenger modifies the challenge ciphertext. For all $i \in [n]$ such that $s_i^* = 0$, the challenger sets $c_{1,i}^* \leftarrow \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{1,i}, 1|w_i)$.

Hybrid H_5 : In this step, the challenger modifies the challenge ciphertext. For all $i \in [n]$ such that $s_i^* = 1$, the challenger sets $c_{0,i}^* \leftarrow \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{0,i}, 1|w_i)$. [6]

Hybrid H_6 : This step is similar to the previous one, except for the decryption queries in the pre-challenge/post-challenge phase. Instead of using $\mathsf{PKE.Find\text{-}1}$, the challenger uses $\mathsf{PKE.Find\text{-}2}$ (defined in Figure 4). Note that this routine does not require any of the predicate encryption secret keys.[7]

[6] Note that hybrids H_4 and H_5 could have been clubbed into a single hybrid. We chose this distinction so that the hybrids for the PKE CCA proof are similar to the hybrids for our Predicate Encryption security proof.

[7] We could have simplified this step by using $\mathsf{PKE.Find}$ instead of using $\mathsf{PKE.Find\text{-}2}$. However, looking ahead, our proof for ABE/PE systems will require an analogous $\mathsf{PKE.Find\text{-}2}$ routine. Hence, we chose to add this minor additional complication here as well.

```
                $\mathsf{PKE.Find\text{-}1}(\mathsf{cca.pk}, \left(\mathsf{cpa.sk}_{s_i^*,i}\right)_{b\in\{0,1\},i\in[n]}, \mathsf{cca.ct}, s^*)$

        Inputs: Public Key $\mathsf{cca.pk} = \left(\mathsf{HPRG.pp}, B, \left(a_i, \mathsf{cpa.pk}_{b,i}\right)_{b\in\{0,1\},i\in[n]}\right)$
                Secret Keys $\left(\mathsf{cpa.sk}_{s_i^*,i}\right)_{i\in[n]}$
                Ciphertext $\mathsf{cca.ct} = \left(\mathsf{ss.vk}, \left(c, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\right), \sigma\right)$
                String $s^* \in \{0,1\}^n$
        Output: $d \in \{0,1\}^n$

• For each $i \in [n]$, do the following:
    – If $s_i^* = 0$,
        1. Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}_{0,i}, c_{0,i})$.
        2. If $m_i = 1|v_i$ and $\mathsf{PRG}(v_i) = c_{2,i}$, set $d_i = 0$. Else set $d_i = 1$.
    – Else if $s_i^* = 1$,
        1. Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}_{1,i}, c_{1,i})$.
        2. If $m_i = 1|v_i$ and $\mathsf{PRG}(v_i) + a_i + B \cdot \mathsf{ss.vk}^* = c_{2,i}$, set $d_i = 1$. Else set $d_i = 0$.
• Output $d = d_1 d_2 \dots d_n$.
```

Figure 3: Routine $\mathsf{PKE.Find\text{-}1}$

```
                $\mathsf{PKE.Find\text{-}2}(\mathsf{cca.pk}, \left(\mathsf{cpa.sk}_{1,i}\right)_{i\in[n]}, \mathsf{cca.ct})$

        Inputs: Public Key $\mathsf{cca.pk} = \left(\mathsf{HPRG.pp}, B, \left(a_i, \mathsf{cpa.pk}_{b,i}\right)_{b\in\{0,1\},i\in[n]}\right)$
                Secret Keys $\left(\mathsf{cpa.sk}_{1,i}\right)_{i\in[n]}$
                Ciphertext $\mathsf{cca.ct} = \left(\mathsf{ss.vk}, \left(c, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\right), \sigma\right)$
        Output: $d \in \{0,1\}^n$

• For each $i \in [n]$, do the following:
    1. Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}_{1,i}, c_{1,i})$.
    2. If $m_i = 1|v_i$ and $\mathsf{PRG}(v_i) + a_i + B \cdot \mathsf{ss.vk}^* = c_{2,i}$, set $d_i = 1$. Else set $d_i = 0$.
• Output $d = d_1 d_2 \dots d_n$.
```

Figure 4: Routine $\mathsf{PKE.Find\text{-}2}$

Hybrid H_7 : This hybrid is identical to the previous one, and the only difference here is change of variable names. In particular, we will swap the variable names v_i^* and w_i if $s_i^* = 1$. This change affects the setup phase (where the a_i values are set), and the challenge phase (where we set $c_{0,i}^*$ and $c_{1,i}^*$). Also, we rename the $\widetilde{r_i}^*$ and r_i^* variables to $r_{i,0}^*$ and $r_{i,1}^*$, depending on s_i^*. For clarity, we will present the entire setup phase and challenge phase below. Note that with this change, the challenger only uses s^* to set $\widetilde{r_i}^*$.

- **Setup Phase**
    1. The challenger first chooses $(\mathsf{HPRG.pp}, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
    2. Next it chooses $2n$ different PKE keys. Let $(\mathsf{cpa.sk}_{b,i}, \mathsf{cpa.pk}_{b,i}) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$ for each $b \in \{0,1\}$, $i \in [n]$.

3. The challenger chooses $s^* \leftarrow \{0,1\}^n$, $v_i^* \leftarrow \{0,1\}^\lambda$, $w_i \leftarrow \{0,1\}^\lambda$ for each $i \in [n]$, and $(\mathsf{ss.sk}^*, \mathsf{ss.vk}^*) \leftarrow \mathsf{ss.Setup}(1^\lambda)$. It sets $r^*_{i,s^*_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$ and $r^*_{i,\overline{s^*_i}} \leftarrow \{0,1\}^\ell$. (These components will be used during the challenge phase.)
4. It then chooses $B \leftarrow \{0,1\}^{\ell_{\mathrm{out}}}$ and sets $a_i = \mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i) - B \cdot \mathsf{ss.vk}^*$ for each $i \in [n]$.
5. It sends $\mathsf{cca.pk} = \Big(\mathsf{HPRG.pp}, B, \big(a_i, \mathsf{cpa.pk}_{b,i}\big)_{b\in\{0,1\}, i\in[n]}\Big)$ to $\mathcal{A}$, and sets the secret key $\mathsf{cca.msk} = (\mathsf{cpa.sk}_{1,i})_{i\in[n]}$.

- **Challenge Phase**

1. The adversary sends two tuples (m_0^*, m_1^*).
2. The challenger chooses a bit $\beta \in \{0,1\}$.
3. It sets $c^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m^*_\beta$.
4. It sets $(c^*_{0,i}, c^*_{1,i}, c^*_{2,i})$ as follows.
   - It sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{0,i}, 1|v_i^*; r^*_{i,0})$, $c^*_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{1,i}, 1|w_i; r^*_{i,1})$ and $c^*_{2,i} = \mathsf{PRG}(v_i^*) = \mathsf{PRG}(w_i) + a_i + B \cdot \mathsf{ss.vk}^*$.
5. Finally, it computes a signature σ^* on $M^* = \big(c^*, \big(c^*_{0,i}, c^*_{1,i}, c^*_{2,i}\big)\big)$ using $\mathsf{ss.sk}^*$ and sends $(\mathsf{ss.vk}^*, M, \sigma^*)$ to $\mathcal{A}$.

Hybrid H_8 : In this hybrid, the challenger chooses both $r^*_{i,b}$ uniformly at random from $\{0,1\}^\ell$.

4.2.2 Analysis

Let $\mathsf{adv}^x_{\mathcal{A}}$ denote the advantage of an adversary $\mathcal{A}$ in Hybrid H_x.

Lemma 4.1. Assuming ss is a strongly unforgeable one-time signature scheme, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^0_{\mathcal{A}} - \mathsf{adv}^1_{\mathcal{A}}| \le \mathsf{negl}(\lambda)$.

Proof. This proof follows from the security of ss. The only difference between these two hybrids is that the challenger, on receiving a decryption query, rejects if it contains $\mathsf{ss.vk}^*$. Suppose there exists a PPT adversary $\mathcal{A}$ such that $|\mathsf{adv}^0_{\mathcal{A}} - \mathsf{adv}^1_{\mathcal{A}}|$ is non-negligible. We can use $\mathcal{A}$ to break the security of ss. The reduction algorithm $\mathcal{B}$ receives a verification key vk^* from the signature scheme's challenger. The reduction algorithm chooses all other components by itself. Next, during the pre-challenge decryption queries, if any decryption query has vk^* in it and the signature verification passes, then the reduction algorithm outputs this as a forgery.

During the challenge phase, the reduction algorithm receives (m_0^*, m_1^*). It chooses β, and computes $M^* = \big(c_0^*, \big(c^*_{0,i}, c^*_{1,i}, c^*_{2,i}\big)\big)$ as in H_0. Finally, it sends M^* to the challenger, and receives signature σ^*. It sends $(\mathsf{vk}^*, M^*, \sigma^*)$ to $\mathcal{A}$.

The adversary then makes polynomially many decryption/key generation queries. If there exists some decryption query with verification key vk^* that verifies, then the reduction algorithm outputs the corresponding message and signature as a forgery.

Clearly, $\mathcal{B}'s$ advantage is at least $\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^2_{\mathcal{A}}$. □

Lemma 4.2. Assuming PRG is a secure pseudorandom generator, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^2_{\mathcal{A}}| \le \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma follows from the security of PRG. The only difference between the two hybrids is the choice of a_i. In H_1, all a_i are chosen uniformly at random. In H_2, the challenger chooses $w_i \leftarrow \{0,1\}^\lambda$ for each i, and sets a_i as either $\mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i) - \mathsf{ss.vk}^* \cdot B$ or $\mathsf{PRG}(w_i) - \mathsf{PRG}(v_i^*) - \mathsf{ss.vk}^* \cdot B$, depending on s_i. Since w_i is not used anywhere else in both these hybrid experiments, we can use PRG security to argue that any PPT adversary has nearly identical advantage in H_1 and H_2. □

Lemma 4.3. Assuming correctness for decryptable ciphertexts for PKE scheme, for any adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^2_{\mathcal{A}} - \mathsf{adv}^3_{\mathcal{A}}| \le \mathsf{negl}(\lambda)$.

Proof. This is an information-theoretic step, and holds for all adversaries (not necessarily polynomial time). The only difference between these two hybrids is with respect to the decryption queries. In H_2, the challenger uses the routine $\mathsf{PKE.Find}$ to get a string d, and then checks if d is valid (using $\mathsf{PKE.Check}$). In H_3, the challenger uses $\mathsf{PKE.Find\text{-}1}$ to compute the string d. In fact, one can prove a more general statement: note that $\mathsf{PKE.Find}$ corresponds to $\mathsf{PKE.Find\text{-}1}$ with last input set to be 0^n. We can show that for any two strings s^* and s', decryption using $\mathsf{PKE.Find\text{-}1}(\cdot,\cdot,\cdot,s^*)$ is statistically indistinguishable from decryption using $\mathsf{PKE.Find\text{-}1}(\cdot,\cdot,\cdot,s')$. For simplicity, we will present indistinguishability of H_2 and H_3, where in H_2, the challenger uses $\mathsf{PKE.Find}$ for decryption queries.

We will argue that with overwhelming probability, for any decryption query ct, either $\mathsf{PKE.Find}$ and $\mathsf{PKE.Find\text{-}1}$ output the same d, or they output d and d' respectively but $\mathsf{PKE.Check}$ rejects both. In particular, it suffices to show that there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $s^* \in [n]$ and $\mathsf{ss.vk}^*$, the following event's probability (denoted by p, parameterized by s^* and $\mathsf{ss.vk}^*$) is at most $\mathsf{negl}(\lambda)$:

$$\left[\begin{array}{c|c} \exists \mathsf{ct} \text{ s.t.} & \mathsf{HPRG.pp} \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell), B \leftarrow \{0,1\}^{\ell_{\text{out}}} \\ \mathsf{ct} = (\mathsf{ss.vk}, (c_0, (c_{0,i}, c_{1,i}, c_{2,i})), \sigma), \mathsf{ss.vk} \neq \mathsf{ss.vk}^* & v_i^*, w_i \leftarrow \{0,1\}^\lambda, \\ \mathsf{PKE.Find}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) = d & a_i = (\mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*} - B \cdot \mathsf{ss.vk}^*, \\ \mathsf{PKE.Find\text{-}1}(\mathsf{pk}, \mathsf{sk}', \mathsf{ct}, s^*) = d' & (\mathsf{cpa.pk}_{b,i}, \mathsf{cpa.sk}_{b,i}) \leftarrow \mathsf{CPA.Enc}(1^\lambda) \\ \mathsf{PKE.Check}(\mathsf{pk}, \mathsf{ct}, d) \neq \mathsf{PKE.Check}(\mathsf{pk}, \mathsf{ct}, d') & \mathsf{sk} = (\mathsf{cpa.sk}_{0,i})_i, \mathsf{sk}' = (\mathsf{cpa.sk}_{s_i^*,i})_i \end{array}\right]$$

where the probability is over the random coins used in $\mathsf{PredE}_{\mathsf{CCA}}.\mathsf{Setup}$. Now, $p \leq p_0 + p_1$, where p_b is defined as the following event's probability:

$$\left[\begin{array}{c|c} \exists \mathsf{ct} \text{ s.t.} & \mathsf{HPRG.pp} \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell), B \leftarrow \{0,1\}^{\ell_{\text{out}}} \\ \mathsf{ct} = (\mathsf{ss.vk}, (c_0, (c_{0,i}, c_{1,i}, c_{2,i})), \sigma), \mathsf{ss.vk} \neq \mathsf{ss.vk}^* & v_i^*, w_i \leftarrow \{0,1\}^\lambda, \\ \mathsf{PKE.Find}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) = d & a_i = (\mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*} - B \cdot \mathsf{ss.vk}^*, \\ \mathsf{PKE.Find\text{-}1}(\mathsf{pk}, \mathsf{sk}', \mathsf{ct}, s^*) = d' & (\mathsf{cpa.pk}_{b,i}, \mathsf{cpa.sk}_{b,i}) \leftarrow \mathsf{CPA.Enc}(1^\lambda) \\ i: \text{ first index s.t. } s_i^* = 1, d_i = b, d_i' = \overline{b} & \mathsf{sk} = (\mathsf{cpa.sk}_{0,i})_i, \mathsf{sk}' = (\mathsf{cpa.sk}_{s_i^*,i})_i \\ \mathsf{PKE.Check}(\mathsf{pk}, \mathsf{ct}, d) \neq \mathsf{PKE.Check}(\mathsf{pk}, \mathsf{ct}, d') & \end{array}\right]$$

We will show that $p_b \leq \mathsf{negl}(\cdot)$ for both $b \in \{0,1\}$. To prove this, let us first consider the following event:

$$p_{\mathsf{PRG}} = \Pr\left[\exists\, \alpha_1, \alpha_2 \in \{0,1\}^\lambda, i \in [n], \mathsf{ss.vk} \text{ s.t. } \mathsf{PRG}(\alpha_1) = \mathsf{PRG}(\alpha_2) + a_i + B \cdot \mathsf{ss.vk}\right]$$

where the probability is over the choice of $B \leftarrow \{0,1\}^{\ell_{\text{out}}}$ and $v_i^*, w_i \leftarrow \{0,1\}^\lambda$. Then $p_b \leq p_{\mathsf{PRG}} + p_b'$, where p_b' is like p_b', except for an additional condition that $\forall \gamma, \delta, \mathsf{PRG}(\gamma) \neq \mathsf{PRG}(\delta) + a_i + B \cdot \mathsf{ss.vk}$. It is formally defined as the following event's probability:

$$\left[\begin{array}{c|c} \exists \mathsf{ct} \text{ s.t.} & \\ \mathsf{ct} = (\mathsf{ss.vk}, (c_0, (c_{0,i}, c_{1,i}, c_{2,i})), \sigma), \mathsf{ss.vk} \neq \mathsf{ss.vk}^* & \mathsf{HPRG.pp} \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell), B \leftarrow \{0,1\}^{\ell_{\text{out}}} \\ \mathsf{PKE.Find}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) = d & v_i^*, w_i \leftarrow \{0,1\}^\lambda, \\ \mathsf{PKE.Find\text{-}1}(\mathsf{pk}, \mathsf{sk}', \mathsf{ct}, s^*) = d' & a_i = (\mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*} - B \cdot \mathsf{ss.vk}^*, \\ i: \text{ first index s.t. } s_i^* = 1, d_i = b, d_i' = \overline{b} & (\mathsf{cpa.pk}_{b,i}, \mathsf{cpa.sk}_{b,i}) \leftarrow \mathsf{CPA.Enc}(1^\lambda) \\ \forall \gamma, \delta, \mathsf{PRG}(\gamma) \neq \mathsf{PRG}(\delta) + a_i + B \cdot \mathsf{ss.vk} & \mathsf{sk} = (\mathsf{cpa.sk}_{0,i})_i, \mathsf{sk}' = (\mathsf{cpa.sk}_{s_i^*,i})_i \\ \mathsf{PKE.Check}(\mathsf{pk}, \mathsf{ct}, d) \neq \mathsf{PKE.Check}(\mathsf{pk}, \mathsf{ct}, d') & \end{array}\right]$$

Hence, it suffices to show that $p_{\mathsf{PRG}} \leq \mathsf{negl}(\lambda)$, $p_0' \leq \mathsf{negl}(\lambda)$ and $p_1' \leq \mathsf{negl}(\lambda)$.

Claim 4.1. $p_{\mathsf{PRG}} \leq \mathsf{negl}(\lambda)$.

Proof. We will prove a stronger statement: for all $\mathsf{ss.vk}^*$,s^* and $\{v_i, w_i\}_{i \in [n]}$, the following probability is at most $n \cdot 2^{-\lambda}$:

$$\Pr\left[\begin{array}{l} \exists\, \gamma, \delta \in \{0,1\}^\lambda, i \in [n], \mathsf{ss.vk} \neq \mathsf{ss.vk}^* \text{ s.t.} \\ \mathsf{PRG}(\gamma) = \mathsf{PRG}(\delta) + (\mathsf{PRG}(v_i) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*} + B \cdot \mathsf{ss.vk} \end{array}\right]$$

where the probability is over the choice of B. Fix any integer $i \in [n]$. Consider the following sets.

$$\begin{aligned}
&S = \{\mathsf{PRG}(x) : x \in \{0,1\}^\lambda\} \\
&S^- = \{\mathsf{PRG}(x) - \mathsf{PRG}(y) - (\mathsf{PRG}(v_i) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*} : x, y \in \{0,1\}^\lambda\} \\
&S^-_{\mathsf{vk}} = \{(\mathsf{PRG}(x) - \mathsf{PRG}(y) - (\mathsf{PRG}(v_i) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*}) / (\mathsf{ss.vk} - \mathsf{ss.vk}^*) : \\
&x, y \in \{0,1\}^\lambda, \mathsf{ss.vk} \in \{0,1\}^{\ell_{\mathsf{vk}}}\}
\end{aligned}$$

The set S has size at most 2^λ. As a result, the set S^- has size at most $2^{2\lambda}$. Finally, the set S^-_{vk} has size at most $2^{2\lambda+\ell_{\mathsf{vk}}}$. If we choose a uniformly random element from $\{0,1\}^{\ell_{\mathrm{out}}} \equiv \{0,1\}^{3\lambda+\ell_{\mathsf{vk}}}$, then this element falls in S^-_{vk} with probability at most $2^{-\lambda}$. This concludes our proof. □

Claim 4.2. $p'_0 = 0$.

Proof. This follows from the definitions of $\mathsf{PKE.Find}$, $\mathsf{PKE.Find\text{-}1}$ and p'_0. Note that $\mathsf{PKE.Find}$ sets $d_i = 0$ only if the decrypted value $1|v_i$ satisfies $\mathsf{PRG}(v_i) = c_{2,i}$, and $\mathsf{PKE.Find\text{-}1}$ sets $d_i = 1$ only if the decrypted value $1|w_i$ satisfies $\mathsf{PRG}(w_i) + a_i + B \cdot \mathsf{ss.vk} = c_{2,i}$. This, together with the requirement in p'_0 that $\forall\ \gamma, \delta$, $\mathsf{PRG}(\gamma) \neq \mathsf{PRG}(\delta) + a_i + B \cdot \mathsf{ss.vk}$, implies that $p'_0 = 0$. □

Claim 4.3. Assuming correctness for decryptable ciphertexts , $p'_1 = 0$.

Proof Intuition. We will first present an overview of the proof, and discuss a subtle but important point in the construction/proof.

Let E'_1 denote the event corresponding to p'_1. For this event to happen, there exists an index i such that $s_i^* = 1$, and the i^{th} iteration of both $\mathsf{PKE.Find}$ and $\mathsf{PKE.Find\text{-}1}$ fail to find a signal (that is, either the decryption fails, or the PRG check fails). Let d be the string output by $\mathsf{PKE.Find}$, and d' the string output by $\mathsf{PKE.Find\text{-}1}$ (therefore $d_i = \overline{d'_i} = 1$). We need to show that $\mathsf{PKE.Check}$ outputs $\perp$ for both d and d'. Suppose $\mathsf{PKE.Check}$ does not output $\perp$ for d. Then, this means that there exists a v such that $c_{1,i}$ is a PKE encryption of $1|v$ and $\mathsf{PRG}(v) + a_i + B \cdot \mathsf{ss.vk} = c_{2,i}$. In this case, the i^{th} iteration of $\mathsf{PKE.Find\text{-}1}$ should set $d'_i = 1$, which is a contradiction.

The other case, where $\mathsf{PKE.Check}$ does not output $\perp$ for d', is similar. This means there exists v, x such that $c_{0,i}$ is an encryption of $1|v$ for attribute x, $C(x) = 1$ and $\mathsf{PRG}(v) = c_{2,i}$. Using perfect correctness of the PKE scheme, we can argue that $\mathsf{PKE.Find}$ should have set $d_i = 0$, which is a contradiction.

Proof. Suppose $s_i^* = 1$, $d_i = 1$, $d'_i = 0$, and $\mathsf{PKE.Check}$ outputs different value for both d and d'. Let $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, i)$, $\widetilde{r_i}' = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d', i)$, $m \leftarrow \mathsf{CPA.Recover}(\mathsf{cpa.pk}_{1,i}, c_{1,i}, \widetilde{r_i})$, $m' \leftarrow \mathsf{CPA.Recover}(\mathsf{cpa.pk}_{0,i}, c_{0,i}, \widetilde{r_i}')$. Since $\mathsf{PKE.Check}$ outputs different values for d and d', it does not output $\perp$ for at least one of them in the i^{th} iteration. We will consider two cases.

Case 1: $\mathsf{PKE.Check}$ does not output $\perp$ for d in the i^{th} iteration: As a result, $m = 1|v$, $c_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{1,i}, m; \widetilde{r_i})$ and $\mathsf{PRG}(v) + a_i + B \cdot \mathsf{ss.vk} = c_{2,i}$. This means that $\mathsf{CPA.Dec}(\mathsf{sk}_{1,i}, c_{1,i}) = 1|v$ (by perfect correctness of the PKE decryption algorithm). However, this means $d'_i = 1$ (by definition of $\mathsf{PKE.Find\text{-}1}$). Hence Case 1 cannot occur.

Case 2: $\mathsf{PKE.Check}$ does not output $\perp$ for d' in the i^{th} iteration: As a result, $m = 1|v$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{0,i}, m; \widetilde{r_i})$, and $\mathsf{PRG}(v) = c_{2,i}$. This means that $\mathsf{CPA.Dec}(\mathsf{cpa.sk}_{0,i}, c_{0,i}) = 1|v$ (since we have perfect correctness for PredE decryption).However, by definition of $\mathsf{PKE.Find}$, $d_i = 0$. Hence Case 2 cannot occur. □

□

Lemma 4.4. Assuming PKE is $\mathsf{IND\text{-}CPA}$ secure, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^3_{\mathcal{A}} - \mathsf{adv}^4_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. The only difference in the two hybrids is with respect to the challenge ciphertext. In H_3, the challenger sets $c^*_{1,i}$ to be encryption of $0^{\lambda+1}$ for all $i \in [n]$ such that $s_i^* = 0$. In H_4, the challenger sets $c^*_{1,i}$ to be encryption of $1|w_i$. Note that the decryption queries require $\mathsf{cpa.sk}_{1,i}$ only if $s_i^* = 1$. As a result, using the $\mathsf{IND\text{-}CPA}$ security of PKE, it follows that the two hybrids are computationally indistinguishable. □

Lemma 4.5. Assuming PKE is $\mathsf{IND\text{-}CPA}$ secure, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^4_{\mathcal{A}} - \mathsf{adv}^5_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma is similar to the proof of the previous lemma (Lemma 4.4). In H_4, the challenger sets $c^*_{0,i}$ to be encryption of $0^{\lambda+1}$ for all $i \in [n]$ such that $s^*_i = 1$. In H_5, the challenger sets $c^*_{0,i}$ to be encryption of $1|w_i$. Note that the decryption queries require $\mathsf{cpa.sk}_{0,i}$ only if $s^*_i = 0$. As a result, using the $\mathsf{IND\text{-}CPA}$ security of PKE, it follows that the two hybrids are computationally indistinguishable. □

Lemma 4.6. Assuming correctness for decryptable ciphertexts for PKE scheme, for any adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^5_{\mathcal{A}} - \mathsf{adv}^6_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. This proof is similar to the proof of Lemma 4.3. In particular, recall that the proof of Lemma 4.3 works for any s^*, s', and note that $\mathsf{PKE.Find\text{-}2}$ simply corresponds to $\mathsf{PKE.Find\text{-}1}(\cdot,\cdot,\cdot,1^n)$. □

Lemma 4.7. $\mathsf{adv}^6_{\mathcal{A}} = \mathsf{adv}^7_{\mathcal{A}}$.

Proof. This follows from the definition of the two hybrids. The only difference between H_6 and H_7 is that the variable names v^*_i and w_i are swapped if $s^*_i = 1$. As a result, any adversary has identical advantage in both hybrids. □

Lemma 4.8. Assuming HPRG satisfies Definition 3.1, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^7_{\mathcal{A}} - \mathsf{adv}^8_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $|\mathsf{adv}^7_{\mathcal{A}} - \mathsf{adv}^8_{\mathcal{A}}| = \epsilon$. We will use $\mathcal{A}$ to build a PPT reduction algorithm $\mathcal{B}$ that breaks the security of HPRG.

The reduction algorithm first receives $\mathsf{HPRG.pp}$ and $\left(r^*_0, \left(r^*_{b,i}\right)_{i\in[n],b\in\{0,1\}}\right)$ from the challenger. It chooses $\{v^*_i, w_i\}$, $(\mathsf{ss.sk}^*, \mathsf{ss.vk}^*)$, sets $\{a_i\}$, chooses $B \leftarrow \{0,1\}^{\ell_{\mathrm{out}}}$, $\left\{(\mathsf{cpa.pk}_{b,i}, \mathsf{cpa.sk}_{b,i}) \leftarrow \mathsf{PredE.Setup}(1^\lambda)\right\}$ and sends $\left(\mathsf{HPRG.pp}, B, \left(a_i, \mathsf{cpa.pk}_{b,i}\right)\right)$ to $\mathcal{A}$. Next, it receives decryption queries, which can be handled using $\{\mathsf{cpa.sk}_{1,i}\}$ (as in H_6/H_7). For the challenge ciphertext, it chooses $\beta \leftarrow \{0,1\}$, sets $c^*_0 = m^*_b \oplus r^*_0$, computes $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{0,i}, 1|v^*_i; r^*_{0,i})$, $c^*_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_{1,i}, 1|w_i; r^*_{1,i})$, $c^*_{2,i} = \mathsf{PRG}(v^*_i) = \mathsf{PRG}(w^*_i) + a_i + B \cdot \mathsf{ss.vk}^*$ and finally computes a signature on $\left(c^*, \left(c^*_{0,i}, c^*_{1,i}, c^*_{2,i}\right)\right)$. It sends the ciphertext to the adversary. The post-challenge queries are handled as the pre-challenge queries. Finally, the adversary sends its guess β'. If $\beta \neq \beta'$, the reduction algorithm guesses that all $r^*_{b,i}$ are uniformly random. This reduction algorithm has advantage ϵ in the hinting PRG security game. □

Lemma 4.9. For any adversary $\mathcal{A}$, $\mathsf{adv}^8_{\mathcal{A}} = 0$.

Proof. Note that in hybrid H_9, there is no information about m_β in the challenge ciphertext, since c^*_0 is uniformly random and the $\left\{c^*_{0,i}\right\}$ components are encryptions to attribute x^*_0. Hence, there is no information about β in the challenge ciphertext, and any adversary has zero advantage in H_9. □

5 CCA Secure Predicate Encryption Scheme

Let $\mathsf{PredE} = (\mathsf{PredE.Setup}, \mathsf{PredE.Enc}, \mathsf{PredE.KeyGen}, \mathsf{PredE.Dec})$ be a predicate encryption scheme with randomness-decryptable ciphertexts and one-sided security against chosen plaintext attacks, $\mathsf{PKE} = (\mathsf{CPA.Setup}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec})$ an $\mathsf{IND\text{-}CPA}$ secure public key encryption scheme with randomness-decryptable ciphertexts, $\mathsf{S} = (\mathsf{ss.Setup}, \mathsf{ss.Sign}, \mathsf{ss.Verify})$ a strongly unforgeable one time signature scheme and $\mathsf{HPRG} = (\mathsf{HPRG.Setup}, \mathsf{HPRG.Eval})$ a hinting PRG scheme. We will assume both our encryption schemes have message space $\{0,1\}^{\lambda+1}$. Let $\ell_{\mathsf{PredE}}(\cdot)$ be a polynomial repesenting the number of bits of randomness used by $\mathsf{PredE.Enc}$, $\ell_{\mathsf{PKE}}(\cdot)$ the number of random bits used by $\mathsf{CPA.Enc}$, and $\ell_{\mathsf{vk}}(\cdot)$ the size of verification keys output by $\mathsf{ss.Setup}$. For simplicity of notation, we will assume $\ell(\cdot) = \ell_{\mathsf{PredE}}(\cdot) = \ell_{\mathsf{PKE}}(\cdot)$,[8] $\ell_{\mathrm{out}}(\lambda) = \ell_{\mathsf{vk}}(\lambda) + 3\lambda$ and $\mathsf{PRG}_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell_{\mathrm{out}}(\lambda)}$ a family of secure pseudorandom generators.

[8]Alternatively, we could set ℓ to be max of these two polynomials.

We will now describe our CCA-one-sided secure predicate encryption scheme $\mathsf{PredE}_{\mathsf{CCA}} = (\mathsf{PredE}_{\mathsf{CCA}}.\mathsf{Setup}, \mathsf{PredE}_{\mathsf{CCA}}.\mathsf{Enc}, \mathsf{PredE}_{\mathsf{CCA}}.\mathsf{KeyGen}, \mathsf{PredE}_{\mathsf{CCA}}.\mathsf{Dec})$ with message space $\mathcal{M}_\lambda = \{0,1\}^{\ell(\lambda)}$. For simplicity of notation, we will skip the dependence of ℓ and ℓ_{out} on λ.

$\mathsf{PredE}_{\mathsf{CCA}}.\mathsf{Setup}(1^\lambda)$: The setup algorithm first chooses $(\mathsf{HPRG.pp}, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$. Next it chooses n different PredE keys and PKE keys. Let $(\mathsf{pred.msk}_i, \mathsf{pred.pk}_i) \leftarrow \mathsf{PredE.Setup}(1^\lambda)$, $(\mathsf{cpa.sk}_i, \mathsf{cpa.pk}_i) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$ for each $i \in [n]$. It then chooses $a_i \leftarrow \{0,1\}^{\ell_{\text{out}}}$ for each $i \in [n]$ and $B \leftarrow \{0,1\}^{\ell_{\text{out}}}$. It sets $\mathsf{pe.cca.pk} = \Big(\mathsf{HPRG.pp}, B, (a_i, \mathsf{pred.pk}_i, \mathsf{cpa.pk}_i)_{i\in[n]}\Big)$ and $\mathsf{pe.cca.msk} = (\mathsf{pred.msk}_i, \mathsf{cpa.sk}_i)_{i\in[n]}$.

$\mathsf{PredE}_{\mathsf{CCA}}.\mathsf{Enc}(\mathsf{pe.cca.pk}, m, x)$: Let $\mathsf{pe.cca.pk} = \Big(\mathsf{HPRG.pp}, B, (a_i, \mathsf{pred.pk}_i, \mathsf{cpa.pk}_i)_{i\in[n]}\Big)$. The encryption algorithm first chooses $s \leftarrow \{0,1\}^n$. It sets $c_0 = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m$. Next, it chooses signature keys $(\mathsf{ss.sk}, \mathsf{ss.vk}) \leftarrow \mathsf{ss.Setup}(1^\lambda)$. For each $i \in [n]$, it chooses $v_i \leftarrow \{0,1\}^\lambda$ and $r_i \leftarrow \{0,1\}^\ell$, sets $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i)$ and does the following:

- If $s_i = 0$, it sets $c_{0,i} = \mathsf{PredE.Enc}(\mathsf{pred.pk}_i, 1|v_i, x; \widetilde{r_i})$, $c_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 0^{\lambda+1}; r_i)$ and $c_{2,i} = \mathsf{PRG}(v_i)$.
- If $s_i = 1$, it sets $c_{0,i} = \mathsf{PredE.Enc}(\mathsf{pred.pk}_i, 0^{\lambda+1}, x; r_i)$, $c_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|v_i; \widetilde{r_i})$ and $c_{2,i} = \mathsf{PRG}(v_i) + a_i + B \cdot \mathsf{ss.vk}$.[9].

Finally, it sets $M = \Big(c_0, (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\Big)$, computes $\sigma \leftarrow \mathsf{ss.Sign}(\mathsf{ss.sk}, M)$ and outputs $(\mathsf{ss.vk}, M, \sigma)$ as the ciphertext.

$\mathsf{PredE}_{\mathsf{CCA}}.\mathsf{KeyGen}(\mathsf{pe.cca.msk}, C)$: Let $\mathsf{pe.cca.msk} = (\mathsf{pred.msk}_i, \mathsf{cpa.sk}_i)_{i\in[n]}$. The key generation algorithm computes $\mathsf{pred.sk}_i \leftarrow \mathsf{PredE.KeyGen}(\mathsf{pred.msk}_i, C)$ and outputs $\mathsf{pe.cca.sk} = \Big(C, (\mathsf{pred.sk}_i)_{i\in[n]}\Big)$.

$\mathsf{PredE}_{\mathsf{CCA}}.\mathsf{Dec}(\mathsf{pe.cca.sk}, \mathsf{pe.cca.pk}, \mathsf{pe.cca.ct})$: Let the ciphertext $\mathsf{pe.cca.ct}$ be parsed as $\Big(\mathsf{ss.vk}, \Big(c_0, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\Big), \sigma$ and $\mathsf{pe.cca.sk} = \Big(C, (\mathsf{pred.sk}_i)_{i\in[n]}\Big)$. The decryption algorithm first verifies the signature σ. It checks if $\mathsf{ss.Verify}(\mathsf{ss.vk}, M, \sigma) = 1$, else it outputs $\perp$.

Next, the decryption algorithm computes $d = \mathsf{Find}(\mathsf{pe.cca.pk}, \mathsf{pe.cca.sk}, \mathsf{pe.cca.ct})$ (where Find is defined in Figure 5), and outputs $\mathsf{Check}(\mathsf{pe.cca.pk}, \mathsf{pe.cca.ct}, C, d)$ (where Check is defined in Figure 6).

$\mathsf{Find}(\mathsf{pe.cca.pk}, \mathsf{pe.cca.sk}, \mathsf{pe.cca.ct})$

Inputs: Public Key $\mathsf{pe.cca.pk} = \Big(\mathsf{HPRG.pp}, B, (a_i, \mathsf{pred.pk}_i, \mathsf{cpa.pk}_i)_{i\in[n]}\Big)$
Secret Key $\mathsf{pe.cca.sk} = (\mathsf{pred.sk}_i)_{i\in[n]}$
Ciphertext $\mathsf{pe.cca.ct} = \Big(\mathsf{ss.vk}, \Big(c_0, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\Big), \sigma\Big)$

Output: $d \in \{0,1\}^n$

- For each $i \in [n]$, do the following:
  1. Let $m_i = \mathsf{PredE.Dec}(\mathsf{pred.sk}_i, c_{0,i})$.
  2. If $m_i = 1|v_i$ and $\mathsf{PRG}(v_i) = c_{2,i}$, set $d_i = 0$. Else set $d_i = 1$.
- Output $d = d_1 d_2 \dots d_n$.

Figure 5: Routine Find

[9]Here, we assume the verification key is embedded in $\mathbb{F}_{2^{\ell_{\text{out}}}}$, and the addition and multiplication are performed in $\mathbb{F}_{2^{\ell_{\text{out}}}}$.

```
Check(pe.cca.pk, pe.cca.ct, $C$, $d$)

Inputs: Public Key pe.cca.pk $= \left(\mathsf{HPRG.pp}, B, (a_i, \mathsf{pred.pk}_i, \mathsf{cpa.pk}_i)_{i\in[n]}\right)$
        Ciphertext pe.cca.ct $= \left(\mathsf{ss.vk}, \left(c_0, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\right), \sigma\right)$
        Circuit $C \in \mathcal{C}_\lambda$
        $d \in \{0,1\}^n$
Output: msg $\in \{0,1\}^\ell$

• Let flag = true. For $i = 1$ to $n$, do the following:
    1. Let $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, i)$.
    2. If $d_i = 0$, let $y \leftarrow \mathsf{PredE.Recover}(\mathsf{pred.pk}_i, c_{0,i}, \widetilde{r_i})$. Perform the following checks. If any of the
       checks fail, set flag = false and exit loop.
        – $y = (m, x) \neq \perp$.
        – $C(x) = 1$.
        – $\mathsf{PredE.Enc}(\mathsf{pred.pk}_i, m, x; \widetilde{r_i}) = c_{0,i}$.
        – $m = 1|v$ and $\mathsf{PRG}(v) = c_{2,i}$.
    3. If $d_i = 1$, let $m \leftarrow \mathsf{CPA.Recover}(\mathsf{cpa.pk}_i, c_{1,i}, \widetilde{r_i})$. Perform the following checks. If any of the
       checks fail, set flag = false and exit loop.
        – $m \neq \perp$.
        – $\mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, m; \widetilde{r_i}) = c_{1,i}$.
        – $m = 1|v$ and $c_{2,i} = \mathsf{PRG}(v) + a_i + B \cdot \mathsf{ss.vk}$.
• If flag = true, output $c_0 \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, 0)$. Else output $\perp$.
```

Figure 6: Routine Check

5.1 Security Proof

The security proof works via a sequence of hybrid experiments, and the hybrid experiments are very similar to the ones used for the PKE construction.

We will now show that the above construction satisfies Definition 2.2. Our proof proceeds via a sequence of hybrids. First, we will describe all hybrids, and then show that the hybrids are computationally indistinguishable.

5.1.1 Hybrids

Hybrid H_0 : This corresponds to the original security game (as described in Definition 2.2).

- **Setup Phase**
  1. The challenger first chooses $(\mathsf{HPRG.pp}, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
  2. Next it chooses n different PredE keys and PKE keys. Let $(\mathsf{pred.msk}_i, \mathsf{pred.pk}_i) \leftarrow \mathsf{PredE.Setup}(1^\lambda)$, $(\mathsf{cpa.sk}_i, \mathsf{cpa.pk}_i) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$ for each $i \in [n]$.
  3. The challenger chooses $s^* \leftarrow \{0,1\}^n$, $v_i^* \leftarrow \{0,1\}^\lambda$ for each $i \in [n]$, and $(\mathsf{ss.sk}^*, \mathsf{ss.vk}^*) \leftarrow \mathsf{ss.Setup}(1^\lambda)$. It sets $\widetilde{r_i}^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$. (These components will be used during the challenge phase.)
  4. It then chooses $a_i \leftarrow \{0,1\}^{\ell_{\text{out}}}$ for each $i \in [n]$ and $B \leftarrow \{0,1\}^{\ell_{\text{out}}}$.
  5. It sends $\mathsf{pe.cca.pk} = \left(\mathsf{HPRG.pp}, B, (a_i, \mathsf{pred.pk}_i, \mathsf{cpa.pk}_i)_{i\in[n]}\right)$ to $\mathcal{A}$, and sets the master secret key $\mathsf{pe.cca.msk} = (\mathsf{pred.msk}_i, \mathsf{cpa.sk}_i)_{i\in[n]}$.

- **Pre-challenge Query Phase**
  - *Decryption Queries*
    1. For each query $\left(\mathsf{ct} = \left(\mathsf{ss.vk}, M = \left(c_0, (c_{0,i}, c_{1,i}, c_{2,i})_i\right), \sigma\right), C\right)$, the challenger first checks the signature σ.
    2. Next, it computes $\mathsf{pe.cca.sk}_C \leftarrow \mathsf{PredE_{CCA}.KeyGen}(\mathsf{msk}, C)$. Let $\mathsf{sk}_C = (\mathsf{sk}_{C,i})_{i\in[n]}$.
    3. The challenger first computes $d = \mathsf{Find}\left(\mathsf{pe.cca.pk}, \mathsf{pe.cca.sk}_C, \mathsf{pe.cca.ct}\right)$.
    4. Next, it outputs $\mathsf{Check}\left(\mathsf{pe.cca.pk}, \mathsf{pe.cca.ct}, C, d\right)$.
  - *Key Generation Queries*: For each query C, the challenger outputs the secret key $\mathsf{pe.cca.sk}_C = \mathsf{PredE_{CCA}.KeyGen}(\mathsf{pe.cca.msk}, C)$.

- **Challenge Phase**
  1. The adversary sends two tuples $(x_0^*, m_0^*), (x_1^*, m_1^*)$ such that for all key queries C in the pre-challenge query phase, $C(x_0^*) = C(x_1^*) = 0$.
  2. The challenger chooses a bit $\beta \in \{0,1\}$.
  3. It sets $c^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m_\beta^*$.
  4. It sets $(c_{0,i}^*, c_{1,i}^*, c_{2,i}^*)$ as follows.
     - If $s_i^* = 0$, it sets $c_{0,i}^* = \mathsf{PredE.Enc}(\mathsf{pred.pk}_i, 1|v_i^*, x_\beta^*; \widetilde{r_i}^*)$, $c_{1,i}^* \leftarrow \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 0^{\lambda+1})$ and $c_{2,i}^* = \mathsf{PRG}(v_i^*)$.
     - If $s_i^* = 1$, it sets $c_{0,i}^* \leftarrow \mathsf{PredE.Enc}(\mathsf{pred.pk}_i, 0^{\lambda+1}, x_\beta^*)$, $c_{1,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|v_i^*; \widetilde{r_i}^*)$ and $c_{2,i}^* = \mathsf{PRG}(v_i^*) + a_i + B \cdot \mathsf{ss.vk}^*$.
  5. Finally, it computes a signature σ^* on $M^* = \left(c^*, \left(c_{0,i}^*, c_{1,i}^*, c_{2,i}^*\right)\right)$ using $\mathsf{ss.sk}^*$ and sends $(\mathsf{ss.vk}^*, M, \sigma^*)$ to $\mathcal{A}$.

- **Post-challenge Query Phase**
  - *Decryption Queries* These are handled as in the pre-challenge phase, with the restriction that all queries (ct, C) must satisfy that $\mathsf{ct} \neq \mathsf{ct}^*$ or $C(x_0^*) = C(x_1^*) = 0$.
  - *Key Generation Queries* These are handled as in the pre-challenge phase, with the restriction that all queries C must satisfy $C(x_0^*) = C(x_1^*) = 0$.

- Finally, the adversary sends its guess b.

Hybrid H_1 : This hybrid is similar to the previous one, except that during the decryption queries, the challenger checks if $\mathsf{ss.vk} = \mathsf{ss.vk}^*$. If so, it rejects.

Hybrid H_2 : In this hybrid, the challenger changes Step 4 of the setup phase. It chooses uniformly random $w_i \leftarrow \{0,1\}^\lambda$, and sets $a_i = \mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i) - \mathsf{ss.vk}^* \cdot B$ if $s_i^* = 0$, else $a_i = \mathsf{PRG}(w_i) - \mathsf{PRG}(v_i^*) - \mathsf{ss.vk}^* \cdot B$.

Hybrid H_3 : This hybrid is similar to the previous one, except that the challenger modifies the way decryption queries are handled. Instead of using Find, the challenger uses Find-1 (defined in Figure 7). The Find routine decrypts only the $c_{0,i}$ values. If decryption works, it sets $s_i = 0$, else it sets $s_i = 1$. The Find-1 routine decrypts either $c_{0,i}$ or $c_{1,i}$, depending on the i^{th} bit of s^*. Note that the Check routine is identical in both experiments.

```
Find-1(pe.cca.pk, $(\mathsf{pred.sk}_i, \mathsf{cpa.sk}_i)_{i\in[n]}$, pe.cca.ct, $s^*$)

Inputs: Public Key pe.cca.pk = $\left(\mathsf{HPRG.pp}, B, (a_i, \mathsf{pred.pk}_i, \mathsf{cpa.pk}_i)_{i\in[n]}\right)$
        Secret Keys $(\mathsf{pred.sk}_i, \mathsf{cpa.sk}_i)_{i\in[n]}$
        Ciphertext pe.cca.ct = $\left(\mathsf{ss.vk}, \left(c_0, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\right), \sigma\right)$
        String $s^* \in \{0,1\}^n$
Output: $d \in \{0,1\}^n$

• For each $i \in [n]$, do the following:
    – If $s_i^* = 0$,
        1. Let $m_i = \mathsf{PredE.Dec}(\mathsf{pred.sk}_i, c_{0,i})$.
        2. If $m_i = 1|v_i$ and $\mathsf{PRG}(v_i) = c_{2,i}$, set $d_i = 0$. Else set $d_i = 1$.
    – Else if $s_i^* = 1$,
        1. Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}_i, c_{1,i})$.
        2. If $m_i = 1|v_i$ and $\mathsf{PRG}(v_i) + a_i + B \cdot \mathsf{ss.vk}^* = c_{2,i}$, set $d_i = 1$. Else set $d_i = 0$.
• Output $d = d_1 d_2 \dots d_n$.
```

Figure 7: Routine Find-1

Hybrid H_4 : In this step, the challenger modifies the challenge ciphertext. For all $i \in [n]$ such that $s_i^* = 0$, the challenger sets $c_{1,i}^* \leftarrow \mathsf{CPA.Enc}(\mathsf{cpa.pk}, 1|w_i)$.

Hybrid H_5 : In this step, the challenger modifies the challenge ciphertext. For all $i \in [n]$ such that $s_i^* = 1$, the challenger sets $c_{0,i}^* \leftarrow \mathsf{PredE.Enc}(\mathsf{pred.pk}, 1|w_i, x_\beta^*)$.

Hybrid H_6 : This step is similar to the previous one, except for the decryption queries in the pre-challenge/post-challenge phase. Instead of using Find-1, the challenger uses Find-2 (defined in Figure 8). Note that this routine does not require any of the predicate encryption secret keys.

Hybrid H_7 : This hybrid is identical to the previous one, and the only difference here is change of variable names. In particular, we will swap the variable names v_i^* and w_i if $s_i^* = 1$. This change affects the setup phase (where the a_i values are set), and the challenge phase (where we set $c_{0,i}^*$ and $c_{1,i}^*$). Also, we rename the $\widetilde{r_i}^*$ and r_i^* variables to $r_{0,i}^*$ and $r_{1,i}^*$, depending on s_i^*. For clarity, we will present the entire setup phase and challenge phase below. Note that with this change, the challenger only uses s^* to set $\widetilde{r_i}^*$.

$\mathsf{Find\text{-}2}(\mathsf{pe.cca.pk}, (\mathsf{cpa.sk}_i)_{i\in[n]}, \mathsf{pe.cca.ct})$

Inputs: Public Key $\mathsf{pe.cca.pk} = \Big(\mathsf{HPRG.pp}, B, (a_i, \mathsf{pred.pk}_i, \mathsf{cpa.pk}_i)_{i\in[n]}\Big)$

Secret Keys $(\mathsf{cpa.sk}_i)_{i\in[n]}$

Ciphertext $\mathsf{pe.cca.ct} = \Big(\mathsf{ss.vk}, \Big(c_0, M = (c_{0,i}, c_{1,i}, c_{2,i})_{i\in[n]}\Big), \sigma\Big)$

Output: $d \in \{0,1\}^n$

- For each $i \in [n]$, do the following:
    1. Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}_i, c_{1,i})$.
    2. If $m_i = 1|v_i$ and $\mathsf{PRG}(v_i) + a_i + B\cdot\mathsf{ss.vk}^* = c_{2,i}$, set $d_i = 1$. Else set $d_i = 0$.
- Output $d = d_1 d_2 \dots d_n$.

Figure 8: Routine $\mathsf{Find\text{-}2}$

- **Setup Phase**
    1. The challenger first chooses $(\mathsf{HPRG.pp}, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
    2. Next it chooses n different PredE keys and PKE keys. Let $(\mathsf{pred.msk}_i, \mathsf{pred.pk}_i) \leftarrow \mathsf{PredE.Setup}(1^\lambda)$, $(\mathsf{cpa.sk}_i, \mathsf{cpa.pk}_i) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$ for each $i \in [n]$.
    3. The challenger chooses $s^* \leftarrow \{0,1\}^n$, $v_i^* \leftarrow \{0,1\}^\lambda$, $w_i \leftarrow \{0,1\}^\lambda$ for each $i \in [n]$, and $(\mathsf{ss.sk}^*, \mathsf{ss.vk}^*) \leftarrow \mathsf{ss.Setup}(1^\lambda)$. It sets $r^*_{i,s^*_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$ and $r^*_{i,\overline{s^*_i}} \leftarrow \{0,1\}^\ell$. (These components will be used during the challenge phase.)
    4. It then chooses $B \leftarrow \{0,1\}^{\ell_{\text{out}}}$ and sets $a_i = \mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i) - B\cdot\mathsf{ss.vk}^*$ for each $i \in [n]$.
    5. It sends $\mathsf{pe.cca.pk} = \Big(\mathsf{HPRG.pp}, B, (a_i, \mathsf{pred.pk}_i, \mathsf{cpa.pk}_i)_{i\in[n]}\Big)$ to $\mathcal{A}$, and sets the master secret key $\mathsf{pe.cca.msk} = (\mathsf{pred.msk}_i, \mathsf{cpa.sk}_i)_{i\in[n]}$.
- **Challenge Phase**
    1. The adversary sends two tuples $(x_0^*, m_0^*), (x_1^*, m_1^*)$ such that for all key queries C in the pre-challenge query phase, $C(x_0^*) = C(x_1^*) = 0$.
    2. The challenger chooses a bit $\beta \in \{0,1\}$.
    3. It sets $c^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m^*_\beta$.
    4. It sets $(c^*_{0,i}, c^*_{1,i}, c^*_{2,i})$ as follows.
        – It sets $c^*_{0,i} = \mathsf{PredE.Enc}(\mathsf{pred.pk}_i, 1|v_i^*, x^*_\beta; r^*_{0,i})$, $c^*_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|w_i; r^*_{1,i})$ and $c^*_{2,i} = \mathsf{PRG}(v_i^*) = \mathsf{PRG}(w_i) + a_i + B\cdot\mathsf{ss.vk}^*$.
    5. Finally, it computes a signature σ^* on $M^* = \big(c^*, \big(c^*_{0,i}, c^*_{1,i}, c^*_{2,i}\big)\big)$ using $\mathsf{ss.sk}^*$ and sends $(\mathsf{ss.vk}^*, M, \sigma^*)$ to $\mathcal{A}$.

Hybrid H_8 : In this hybrid, the challenger chooses both $r^*_{b,i}$ uniformly at random from $\{0,1\}^\ell$.

Hybrid H_9 : In this hybrid, the challenger computes $\{c^*_{0,i}\}$ as encryptions for attribute x_0^*. As a result, the ciphertext contains no information about the bit β chosen by the challenger.

Analysis Let $\mathsf{adv}^x_{\mathcal{A}}$ denote the advantage of an adversary $\mathcal{A}$ in Hybrid H_x.

Lemma 5.1. Assuming ss is a strongly unforgeable one-time signature scheme, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^0_{\mathcal{A}} - \mathsf{adv}^1_{\mathcal{A}}| \le \mathsf{negl}(\lambda)$.

Proof. This proof follows from the security of ss. The only difference between these two hybrids is that the challenger, on receiving a decryption query, rejects if it contains $\mathsf{ss.vk}^*$. Suppose there exists a PPT adversary $\mathcal{A}$ such that $|\mathsf{adv}^0_{\mathcal{A}} - \mathsf{adv}^1_{\mathcal{A}}|$ is non-negligible. We can use $\mathcal{A}$ to break the security of ss. The reduction algorithm $\mathcal{B}$ receives a verification key vk^* from the signature scheme's challenger. The reduction algorithm chooses all other components by itself. Next, during the pre-challenge decryption queries, if any decryption query has vk^* in it and the signature verification passes, then the reduction algorithm outputs this as a forgery.

During the challenge phase, the reduction algorithm receives (x_0^*, m_0^*) and (x_1^*, m_1^*). It chooses β, and computes $M^* = \left(c^*, \left(c^*_{0,i}, c^*_{1,i}, c^*_{2,i}\right)\right)$ as in H_0. Finally, it sends M^* to the challenger, and receives signature σ^*. It sends $(\mathsf{vk}^*, M^*, \sigma^*)$ to $\mathcal{A}$.

The adversary then makes polynomially many decryption/key generation queries. If there exists some decryption query with verification key vk^* that verifies, then the reduction algorithm outputs the corresponding message and signature as a forgery.

Clearly, $\mathcal{B}'s$ advantage is at least $\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^2_{\mathcal{A}}$. □

Lemma 5.2. Assuming PRG is a secure pseudorandom generator, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^2_{\mathcal{A}}| \le \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma follows from the security of PRG. The only difference between the two hybrids is the choice of a_i. In H_1, all a_i are chosen uniformly at random. In H_2, the challenger chooses $w_i \leftarrow \{0,1\}^\lambda$ for each i, and sets a_i as either $\mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i) - \mathsf{ss.vk}^* \cdot B$ or $\mathsf{PRG}(w_i) - \mathsf{PRG}(v_i^*) - \mathsf{ss.vk}^* \cdot B$, depending on s_i. Since w_i is not used anywhere else in both these hybrid experiments, we can use PRG security to argue that any PPT adversary has nearly identical advantage in H_1 and H_2. □

Lemma 5.3. Assuming correctness for decryptable ciphertexts for PredE and PKE schemes, for any adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^2_{\mathcal{A}} - \mathsf{adv}^3_{\mathcal{A}}| \le \mathsf{negl}(\lambda)$.

Proof. This is an information-theoretic step, and holds for all adversaries (not necessarily polynomial time). The only difference between these two hybrids is with respect to the decryption queries. In H_2, the challenger uses the routine Find to get a string d, and then checks if d is valid (using Check). In H_3, the challenger uses $\mathsf{Find\text{-}1}$ to compute the string d. In fact, one can prove a more general statement: note that Find corresponds to $\mathsf{Find\text{-}1}$ with last input set to be 0^n. We can show that for any two strings s^* and s', decryption using $\mathsf{Find\text{-}1}(\cdot,\cdot,\cdot,s^*)$ is statistically indistinguishable from decryption using $\mathsf{Find\text{-}1}(\cdot,\cdot,\cdot,s')$. For simplicity, we will present indistinguishability of H_2 and H_3, where in H_2, the challenger uses Find for decryption queries.

We will argue that with overwhelming probability, for any decryption query (ct, C), either Find and $\mathsf{Find\text{-}1}$ output the same d, or they output d and d' respectively but Check rejects both. In particular, it suffices to show that there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $s^* \in [n]$ and $\mathsf{ss.vk}^*$, the following event's probability (denoted by p, parameterized by s^* and $\mathsf{ss.vk}^*$) is at most $\mathsf{negl}(\lambda)$:

$$\left[\begin{array}{c|c} \exists(\mathsf{ct}, C) \text{ s.t.} & \mathsf{HPRG.pp} \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell), B \leftarrow \{0,1\}^{\ell_{\text{out}}} \\ \mathsf{ct} = \left(\mathsf{ss.vk}, \left(c_0, \left(c_{0,i}, c_{1,i}, c_{2,i}\right)\right), \sigma\right) & v_i^*, w_i \leftarrow \{0,1\}^\lambda, \\ \mathsf{ss.vk} \neq \mathsf{ss.vk}^* & a_i = \left(\mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i)\right) \cdot (-1)^{s_i^*} - B \cdot \mathsf{ss.vk}^*, \\ \mathsf{Find}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) = d & (\mathsf{pred.pk}_i, \mathsf{pred.msk}_i) \leftarrow \mathsf{PredE.Enc}(1^\lambda) \\ \mathsf{Find\text{-}1}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}, s^*) = d' & (\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i) \leftarrow \mathsf{CPA.Enc}(1^\lambda) \\ \mathsf{Check}(\mathsf{pk}, \mathsf{ct}, C, d) \neq \mathsf{Check}(\mathsf{pk}, \mathsf{ct}, C, d') & \mathsf{sk} = \left(\mathsf{PredE_{CCA}.KeyGen}(\mathsf{pred.msk}_i, C)\right)_i \end{array}\right]$$

where the probability is over the random coins used in $\mathsf{PredE_{CCA}.Setup}$. Now, $p \le p_0 + p_1$, where p_b is defined as the following event's probability:

$$\left[\begin{array}{c|c} \exists(\mathsf{ct}, C) \text{ s.t.} & \\ \mathsf{ct} = (\mathsf{ss.vk}, (c_0, (c_{0,i}, c_{1,i}, c_{2,i})), \sigma) & \mathsf{HPRG.pp} \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell), B \leftarrow \{0,1\}^{\ell_{\text{out}}} \\ \mathsf{ss.vk} \neq \mathsf{ss.vk}^* & v_i^*, w_i \leftarrow \{0,1\}^\lambda, \\ \mathsf{Find}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) = d & a_i = (\mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*} - B \cdot \mathsf{ss.vk}^*, \\ \mathsf{Find\text{-}1}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}, s^*) = d' & (\mathsf{pred.pk}_i, \mathsf{pred.msk}_i) \leftarrow \mathsf{PredE.Enc}(1^\lambda) \\ i:\ \text{first index s.t. } s_i^* = 1, d_i = b, d_i' = \overline{b} & (\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i) \leftarrow \mathsf{CPA.Enc}(1^\lambda) \\ \mathsf{Check}(\mathsf{pk}, \mathsf{ct}, C, d) \neq \mathsf{Check}(\mathsf{pk}, \mathsf{ct}, C, d') & \mathsf{sk} = (\mathsf{PredE_{CCA}.KeyGen}(\mathsf{pred.msk}_i, C))_i \end{array}\right]$$

We will show that $p_b \leq \mathsf{negl}(\cdot)$ for both $b \in \{0,1\}$. To prove this, let us first consider the following event:

$$p_{\mathsf{PRG}} = \Pr\left[\exists\ \alpha_1, \alpha_2 \in \{0,1\}^\lambda, i \in [n], \mathsf{ss.vk} \text{ s.t. } \mathsf{PRG}(\alpha_1) = \mathsf{PRG}(\alpha_2) + a_i + B \cdot \mathsf{ss.vk}\right]$$

where the probability is over the choice of $B \leftarrow \{0,1\}^{\ell_{\text{out}}}$ and $v_i^*, w_i \leftarrow \{0,1\}^\lambda$. Then $p_b \leq p_{\mathsf{PRG}} + p_b'$, where p_b' is like p_b', except for an additional condition that $\forall \gamma, \delta, \mathsf{PRG}(\gamma) \neq \mathsf{PRG}(\delta) + a_i + B \cdot \mathsf{ss.vk}$. It is formally defined as the following event's probability:

$$\left[\begin{array}{c|c} \exists(\mathsf{ct}, C) \text{ s.t.} & \\ \mathsf{ct} = (\mathsf{ss.vk}, (c_0, (c_{0,i}, c_{1,i}, c_{2,i})), \sigma) & \mathsf{HPRG.pp} \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell), B \leftarrow \{0,1\}^{\ell_{\text{out}}} \\ \mathsf{ss.vk} \neq \mathsf{ss.vk}^* & v_i^*, w_i \leftarrow \{0,1\}^\lambda, \\ \mathsf{Find}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) = d & a_i = (\mathsf{PRG}(v_i^*) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*} - B \cdot \mathsf{ss.vk}^*, \\ \mathsf{Find\text{-}1}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}, s^*) = d' & (\mathsf{pred.pk}_i, \mathsf{pred.msk}_i) \leftarrow \mathsf{PredE.Enc}(1^\lambda) \\ i:\ \text{first index s.t. } s_i^* = 1, d_i = b, d_i' = \overline{b} & (\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i) \leftarrow \mathsf{CPA.Enc}(1^\lambda) \\ \forall \gamma, \delta, \mathsf{PRG}(\gamma) \neq \mathsf{PRG}(\delta) + a_i + B \cdot \mathsf{ss.vk} & \mathsf{sk} = (\mathsf{PredE_{CCA}.KeyGen}(\mathsf{pred.msk}_i, C))_i \\ \mathsf{Check}(\mathsf{pk}, \mathsf{ct}, C, d) \neq \mathsf{Check}(\mathsf{pk}, \mathsf{ct}, C, d') & \end{array}\right]$$

Hence, it suffices to show that $p_{\mathsf{PRG}} \leq \mathsf{negl}(\lambda)$, $p_0' \leq \mathsf{negl}(\lambda)$ and $p_1' \leq \mathsf{negl}(\lambda)$.

Claim 5.1. $p_{\mathsf{PRG}} \leq \mathsf{negl}(\lambda)$.

Proof. We will prove a stronger statement: for all $\mathsf{ss.vk}^*$,s^* and $\{v_i, w_i\}_{i \in [n]}$, the following probability is at most $n \cdot 2^{-\lambda}$:

$$\Pr\left[\begin{array}{l} \exists\ \gamma, \delta \in \{0,1\}^\lambda, i \in [n], \mathsf{ss.vk} \neq \mathsf{ss.vk}^* \text{ s.t.} \\ \mathsf{PRG}(\gamma) = \mathsf{PRG}(\delta) + (\mathsf{PRG}(v_i) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*} + B \cdot \mathsf{ss.vk} \end{array}\right]$$

where the probability is over the choice of B. Fix any integer $i \in [n]$. Consider the following sets.

$$\begin{array}{l} S = \left\{\mathsf{PRG}(x) : x \in \{0,1\}^\lambda\right\} \\ S^- = \left\{\mathsf{PRG}(x) - \mathsf{PRG}(y) - (\mathsf{PRG}(v_i) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*} : x, y \in \{0,1\}^\lambda\right\} \\ S^-_{\mathsf{vk}} = \{\left(\mathsf{PRG}(x) - \mathsf{PRG}(y) - (\mathsf{PRG}(v_i) - \mathsf{PRG}(w_i)) \cdot (-1)^{s_i^*}\right)/(\mathsf{ss.vk} - \mathsf{ss.vk}^*) : \\ x, y \in \{0,1\}^\lambda, \mathsf{ss.vk} \in \{0,1\}^{\ell_{\mathsf{vk}}}\} \end{array}$$

The set S has size at most 2^λ. As a result, the set S^- has size at most $2^{2\lambda}$. Finally, the set S^-_{vk} has size at most $2^{2\lambda + \ell_{\mathsf{vk}}}$. If we choose a uniformly random element from $\{0,1\}^{\ell_{\text{out}}} \equiv \{0,1\}^{3\lambda + \ell_{\mathsf{vk}}}$, then this element falls in S^-_{vk} with probability at most $2^{-\lambda}$. This concludes our proof. □

Claim 5.2. $p_0' = 0$.

Proof. This follows from the definitions of Find, $\mathsf{Find\text{-}1}$ and p_0'. Note that Find sets $d_i = 0$ only if the decrypted value $1|v_i$ satisfies $\mathsf{PRG}(v_i) = c_{2,i}$, and $\mathsf{Find\text{-}1}$ sets $d_i = 1$ only if the decrypted value $1|w_i$ satisfies $\mathsf{PRG}(w_i) + a_i + B \cdot \mathsf{ss.vk} = c_{2,i}$. This, together with the requirement in p_0' that $\forall\ \gamma, \delta$, $\mathsf{PRG}(\gamma) \neq \mathsf{PRG}(\delta) + a_i + B \cdot \mathsf{ss.vk}$, implies that $p_0' = 0$. □

Claim 5.3. Assuming correctness for decryptable ciphertexts , $p_1' = 0$.

Proof. Suppose $s^*_i = 1$, $d_i = 1$, $d'_i = 0$, and Check outputs different value for both d and d'. Let $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, i)$, $\widetilde{r_i}' = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d', i)$, $m \leftarrow \mathsf{PredE.Recover}(\mathsf{cpa.pk}_i, c_{1,i}, \widetilde{r_i})$, $(m', x') \leftarrow \mathsf{PredE.Recover}(\mathsf{pred.pk}_i, c_{0,i}, \widetilde{r_i}')$. Since Check outputs different values for d and d', it does not output $\perp$ for at least one of them in the i^{th} iteration. We will consider two cases.

Case 1: Check does not output $\perp$ for d in the i^{th} iteration: As a result, $m = 1|v$, $c_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, m; \widetilde{r_i})$ and $\mathsf{PRG}(v) + a_i + B \cdot \mathsf{ss.vk} = c_{2,i}$. This means that $\mathsf{CPA.Dec}(\mathsf{sk}_i, c_{1,i}) = 1|v$ (by perfect correctness of the PKE decryption algorithm). However, this means $d'_i = 1$ (by definition of $\mathsf{Find\text{-}1}$). Hence Case 1 cannot occur.

Case 2: Check does not output $\perp$ for d' in the i^{th} iteration: As a result, $m = 1|v$, $c_{0,i} = \mathsf{PredE.Enc}(\mathsf{pred.pk}_i, m, x; \widetilde{r_i})$, $C(x) = 1$ and $\mathsf{PRG}(v) = c_{2,i}$. This means that $\mathsf{PredE.Dec}(\mathsf{pred.sk}_i, c_{0,i}) = 1|v$ (since $C(x) = 1$ and we have perfect correctness for PredE decryption).However, by definition of Find, $d_i = 0$. Hence Case 2 cannot occur. □

□

Lemma 5.4. Assuming PKE is $\mathsf{IND\text{-}CPA}$ secure, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^3_{\mathcal{A}} - \mathsf{adv}^4_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. The only difference in the two hybrids is with respect to the challenge ciphertext. In H_3, the challenger sets $c^*_{1,i}$ to be encryption of $0^{\lambda+1}$ for all $i \in [n]$ such that $s^*_i = 0$. In H_4, the challenger sets $c^*_{1,i}$ to be encryption of $1|w_i$. Note that the decryption queries require $\mathsf{cpa.sk}_i$ only if $s^*_i = 1$. As a result, using the $\mathsf{IND\text{-}CPA}$ security of PKE, it follows that the two hybrids are computationally indistinguishable. □

Lemma 5.5. Assuming PredE satisfies Definition 2.1, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^4_{\mathcal{A}} - \mathsf{adv}^5_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma is similar to the proof of the previous lemma (Lemma 5.4). Note that in hybrids H_3, H_4 and H_5, the challenger does not use the PredE secret key $\mathsf{pred.sk}_{C,i}$ for decryption if $s^*_i = 1$. As a result, we can use the $\mathsf{IND\text{-}CPA}$ security of PredE (as defined in Definition 2.1).Strictly speaking, this proof does not require the 1-sided attribute hiding. It suffices if the scheme PredE satisfies ABE security.

To prove this lemma, we will define $n + 1$ hybrid experiments $H_{4,i}$ for $i \in [n] \cup \{0\}$. In hybrid H_{4,i^*}, all ciphertext components $\{c_{0,i}\}_{i \leq i^*}$ are prepared as in H_5, while all the remaining ones are prepared as in H_4. Note that $H_{4,0}$ corresponds to H_4, while $H_{4,n}$ corresponds to H_5. Also, the string s^* is common for all these intermediate hybrids, and therefore if $s^*_i = 0$, then the experiments $H_{4,i-1}$ and $H_{4,i}$ are identical.

Suppose there exists a PPT adversary $\mathcal{A}$ that can distinguish between H_{4,i^*-1} and H_{4,i^*} with advantage ϵ. We will use $\mathcal{A}$ to break the security of the underlying predicate encryption system. The reduction algorithm receives public key pk^* from the PE challenger. It sets $\mathsf{pred.pk}_{i^*} = \mathsf{pk}^*$, and it chooses $(\mathsf{pred.pk}_i, \mathsf{pred.msk}_i)_{i \neq i^*}$. Next, it receives key generation and decryption queries. For any key generation query C, it receives sk_C from the challenger; it sets $\mathsf{pred.sk}_{i^*} = \mathsf{sk}_C$, and chooses the remaining by itself. For any decryption query (ct, C), note that the reduction algorithm does not need to query the challenger for a secret key corresponding to C. If $s^*_{i^*} = 1$, then the reduction algorithm uses the PKE secret key at the i^* index (and if $s^*_{i^*} = 0$, then H_{4,i^*-1} and H_{4,i^*} are identical). On receiving the challenge tuples $(m^*_0, x^*_0), (m^*_1, x^*_1)$, the reduction algorithm chooses $\beta \leftarrow \{0, 1\}$. It sends $(0^{\lambda+1}, 1|w_i)$ as the challenge messages, and x^*_β as the challenge attribute, and receives $c^*_{i^*}$. It computes the remaining challenge ciphertext by itself, and sends it to the adversary. The post challenge queries are handled similar to the pre-challenge ones. Finally, the adversary sends its guess, which is forwarded to the challenger.

□

Lemma 5.6. Assuming correctness for decryptable ciphertexts for PredE and PKE schemes, for any adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^5_{\mathcal{A}} - \mathsf{adv}^6_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. This proof is similar to the proof of Lemma 5.3. In particular, recall that the proof of Lemma 5.3 works for any s^*, s', and note that $\mathsf{Find\text{-}2}$ simply corresponds to $\mathsf{Find\text{-}1}(\cdot, \cdot, \cdot, 1^n)$. □

Lemma 5.7. $\mathsf{adv}^6_{\mathcal{A}} = \mathsf{adv}^7_{\mathcal{A}}$.

Proof. This follows from the definition of the two hybrids. The only difference between H_6 and H_7 is that the variable names v_i^* and w_i are swapped if $s_i^* = 1$. As a result, any adversary has identical advantage in both hybrids. $\square$

Lemma 5.8. Assuming HPRG satisfies Definition 3.1, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^7_{\mathcal{A}} - \mathsf{adv}^8_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $|\mathsf{adv}^6_{\mathcal{A}} - \mathsf{adv}^7_{\mathcal{A}}| = \epsilon$. We will use $\mathcal{A}$ to build a PPT reduction algorithm $\mathcal{B}$ that breaks the security of HPRG.

The reduction algorithm first receives $\mathsf{HPRG.pp}$ and $\left(r_0^*, \left(r_{i,b}^*\right)_{i\in[n],b\in\{0,1\}}\right)$ from the challenger. It chooses $\{v_i^*, w_i\}$, $(\mathsf{ss.sk}^*, \mathsf{ss.vk}^*)$, sets $\{a_i\}$, chooses $B \leftarrow \{0,1\}^{\ell_{\text{out}}}$, $\left\{(\mathsf{pred.pk}_i, \mathsf{pred.msk}_i) \leftarrow \mathsf{PredE.Setup}(1^\lambda)\right\}$, $\left\{(\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i) \leftarrow \mathsf{CPA.Setup}(1^\lambda)\right\}$ and sends $(\mathsf{HPRG.pp}, B, (a_i, \mathsf{pred.pk}_i, \mathsf{cpa.pk}_i))$ to $\mathcal{A}$. Next, it receives either decryption queries or key generation queries. Key generation and decryption queries can be handled using $\{\mathsf{pred.msk}_i\}$ and $\{\mathsf{cpa.sk}_i\}$ (as in H_6/H_7). For the challenge ciphertext, it chooses $\beta \leftarrow \{0,1\}$, sets $c^* = m_b^* \oplus r_0^*$, computes $c_{0,i}^* = \mathsf{PredE.Enc}(\mathsf{pred.pk}_i, 1|v_i^*, x_\beta; r_{i,0}^*)$, $c_{1,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|w_i; r_{i,1}^*)$, $c_{2,i}^* = \mathsf{PRG}(v_i^*) = \mathsf{PRG}(w_i^*) + a_i + B \cdot \mathsf{ss.vk}^*$ and finally computes a signature on $\left(c^*, \left(c_{0,i}^*, c_{1,i}^*, c_{2,i}^*\right)\right)$. It sends the ciphertext to the adversary. The post-challenge queries are handled as the pre-challenge queries. Finally, the adversary sends its guess β'. If $\beta \neq \beta'$, the reduction algorithm guesses that all $r_{i,b}^*$ are uniformly random. This reduction algorithm has advantage ϵ in the hinting PRG security game. $\square$

Lemma 5.9. Assuming PredE satisfies Definition 2.1, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^8_{\mathcal{A}} - \mathsf{adv}^9_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. Using the security of PredE, we can argue that any PPT adversary has nearly same advantage in H_8 and H_9.

$\square$

Lemma 5.10. For any adversary $\mathcal{A}$, $\mathsf{adv}^9_{\mathcal{A}} = 0$.

Proof. Note that in hybrid H_9, there is no information about m_β in the challenge ciphertext, since c^* is uniformly random and the $\left\{c_{0,i}^*\right\}$ components are encryptions to attribute x_0^*. Hence, there is no information about β in the challenge ciphertext, and any adversary has zero advantage in H_9. $\square$

5.2 Proving Security of Attribute-Based Encryption Systems

We remark that our above proof technique will apply to an Attribute-Based Encryption system that does not guarantee hiding of the attribute string x (i.e. is not a one-sided predicate encryption system.) Recall, that the IND-CPA and IND-CCA functionality and security definitions of ABE correspond exactly to those of one-sided PE of Section 2 with the exception that in the challenge ciphertext attributes given by the attacker we have $x_0^* = x_1^*$.

We observe none of the steps in the above proof use the fact that the scheme is one-sided PE, except for the last step that changes the $c_{0,i}$ ciphertext from encrypting under the attribute x_β^* to x_0^*. In an ABE chosen ciphertext security game, this last step in unnecessary since we already have $x_\beta^* = x_0^*$ by fiat.

Acknowledgements

We thank Amit Sahai for fruitful discussions. And we thank Rishab Goyal for pointing out the simple argument for separating hinting security from standard security in PRGs.

References

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 90–108, 2011.

[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.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *STOC*, pages 103–112, 1988.

[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.

[BL16] Johannes Blömer and Gennadij Liske. Construction of fully cca-secure predicate encryptions from pair encoding schemes. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 431–447, 2016.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, pages 535–564, 2018.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Proceedings of the 4th conference on Theory of cryptography*, TCC'07, pages 535–554, Berlin, Heidelberg, 2007. Springer-Verlag.

[CC09] Melissa Chase and Sherman S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2009.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 595–624, 2015.

[Cha07] Melissa Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from Identity Based Encryption. In *EUROCRYPT '04*, volume 3027 of LNCS, pages 207–222, 2004.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 542–552, 1991.

[DG17a] Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 537–569, 2017.

[DG17b] Nico Dttling and Sanjam Garg. From selective ibe to full ibe and selective hibe. TCC, 2017.

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I*, pages 3–31, 2018.

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 342–360, 2004.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 480–491, 1994.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO '99*, volume 1666 of LNCS, pages 537–554. Springer, 1999.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 416–426, 1990.

[GGH18] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. Cryptology ePrint Archive, Report 2018/872, 2018. `https://eprint.iacr.org/2018/872`.

[GH18] Sanjam Garg and Mohammad Hajiabadi. Trapdoor functions from the computational diffie-hellman assumption. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 362–391, 2018.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 612–621, 2017.

[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.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, 2008.

[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.

[Nao89] Moni Naor. Bit commitment using pseudo-randomness. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 128–136, 1989.

[NP15] Mridul Nandi and Tapas Pandit. Generic conversions from cpa to cca secure functional encryption. Cryptology ePrint Archive, Report 2015/457, 2015. `https://eprint.iacr.org/2015/457`.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437, 1990.

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

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for np from (plain) learning with errors. Cryptology ePrint Archive, Report 2019/158, 2019. `https://eprint.iacr.org/2019/158`.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 187–196, 2008.

[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.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 433–444, 1991.

[RS10] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. *SIAM J. Comput.*, 39(7):3058–3088, 2010.

[Sho98] Victor Shoup. Why chosen ciphertext security matters, 1998.

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

[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.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 600–611, 2017.

[YAHK11] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 71–89, 2011.

A Constructions of Hinting PRG

A.1 Construction Based on the CDH Assumption

A.1.1 Computational Diffie Hellman Assumption

Assumption 1. Let $\mathcal{G}$ be any group of prime order p. The Computational Diffie-Hellman (CDH) assumption holds on $\mathcal{G}$ if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following probability is at most $\mathsf{negl}(\lambda)$:

$$\Pr[g^{ab} \leftarrow \mathcal{A}(g, g^a, g^b) \ : \ g \leftarrow \mathcal{G}, a, b \leftarrow \mathbb{Z}_p]$$

A.1.2 Construction

We will now describe our CDH based hinting PRG scheme.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses a prime p of λ bits. Next, it sets $n = \log p + 2\lambda$. It then chooses a group $\mathcal{G}$ of size p. Next, it chooses $2 \cdot n$ uniformly random group elements $\{g_{i,b} \leftarrow \mathcal{G}\}_{i \in [n], b \in \{0,1\}}$, $2 \cdot (n+1) \cdot \ell$ uniformly random integers $\{\rho_{i,j,b} \leftarrow \mathbb{Z}_p\}_{i \in [n] \cup \{0\}, j \in [\ell], b \in \{0,1\}}$. For each $i \in [n], j \in [\ell], b \in \{0,1\}$, it sets a $2 \times n$ matrix $\mathbf{H}_{i,j,b}$ as follows:

$$\forall (\beta, k) \in \{0,1\} \times [n], \ \mathbf{H}_{i,j,b}[\beta, k] = \begin{cases} \perp & \text{if } (k, \beta) = (i, \bar{b}) \\ g_{k,\beta}^{\rho_{i,j,b}} & \text{otherwise} \end{cases}$$

Similarly, for each $j \in [\ell]$, it sets $\mathbf{H}_{0,j}$ as follows:

$$\forall (\beta, k) \in \{0,1\} \times [n], \ \mathbf{H}_{0,j}[\beta, k] = g_{k,\beta}^{\rho_{0,j,0}}$$

Finally, it chooses the randomness for hardcore bit extraction. It chooses $\left\{r_{0,j} \leftarrow \{0,1\}^{\ell}_{\mathsf{hcb}}\right\}_{j \in [\ell]}$ and $\left\{r_{i,j,b} \leftarrow \{0,1\}^{\ell_{\mathsf{hcb}}}\right\}_{i \in [n], j \in [\ell], b \in \{0,1\}}$. The setup algorithm sets $\mathsf{pp} = \left(\{\mathbf{H}_{0,j}, r_{0,j}\}_{j \in [\ell]}, \{\mathbf{H}_{i,j,b}, r_{i,j,b}\}_{i \in [n], j \in [\ell], b \in \{0,1\}} \right)$.

$\mathsf{Eval}(\mathsf{pp}, s, i)$: Let $\mathsf{pp} = \left(\{\mathbf{H}_{0,j}, r_{0,j}\}_{j \in [\ell]}, \{\mathbf{H}_{i,j,b}, r_{i,j,b}\}_{i \in [n], j \in [\ell], b \in \{0,1\}} \right)$. If $i \neq 0$, the evaluation algorithm does the following: For each $j \in [\ell]$, the evaluation algorithm sets $z_j = \mathsf{hcb}\left(\prod_{k=1}^{n} \mathbf{H}_{i,j,s_i}[s_k, k]; r_{i,j,s_i}\right)$ and outputs $z = z_1 z_2 \ldots z_\ell$.

If $i = 0$, the evaluation algorithm does the following: For each $j \in [\ell]$, it sets $z_j = \mathsf{hcb}\left(\prod_{k=1}^{n} \mathbf{H}_{0,j}[s_k, k]; r_{0,j}\right)$ and outputs $z = z_1 z_2 \ldots z_\ell$.

A.1.3 Proof of Security

To prove security, we will first present a sequence of hybrid experiments, and then show that the hybrid experiments are computationally indistinguishable.

Hybrid H_0 : This is the original security game. The challenger chooses a uniformly random string $s \leftarrow \{0,1\}^n$, chooses $\{g_{i,b}\}_{i,b}$, $\{\rho_{i,j,b}\}_{i,j,b}$, sets public parameters $\mathsf{pp} = \left(\{\mathbf{H}_{0,j}\}_j, \{\mathbf{H}_{i,j,b}\}_{i,j,b}\right)$ as in the construction. Next, it sets $y_0 = \mathsf{Eval}(\mathsf{pp}, s, 0)$, $y_{i,s_i} = \mathsf{Eval}(\mathsf{pp}, s, i)$ and $y_{i,\overline{s_i}} \leftarrow \{0,1\}^\ell$. It outputs pp, $\left(y_0, (y_{i,0}, y_{i,1})_i\right)$.

Next, we define $n \cdot \ell$ hybrid experiments $H_{1,i,j}$ for $i \in [n], j \in [\ell]$. For any tuples (i,j) and (i',j'), we say that $(i,j) \preceq (i',j')$ if either $i < i'$ or ($i = i'$ and $j \leq j'$). Also, let $H_0 \equiv H_{1,1,0}$ and $H_{1,i,\ell} \equiv H_{1,i+1,0}$.

Hybrid H_{1,i^*,j^*} : In this hybrid, the challenger does not choose all $\{y_{i,\overline{s_i}}\}_i$ uniformly at random. Instead, for all $(i,j) \preceq (i^*,j^*)$, it sets the j^{th} bit of $y_{i,\overline{s_i}}$ to be $\mathsf{hcb}\left(\prod_{k=1}^n g_{k,s_k}^{\rho_{k,j,\overline{s_i}}}; r_{i,j,\overline{s_i}}\right)$. For all $j > j^*$, the j^{th} bit of $y_{i^*,\overline{s_{i^*}}}$ are chosen uniformly at random. For all $i > i^*$, the string $y_{i,\overline{s_i}}$ is chosen uniformly at random from $\{0,1\}^\ell$.

Hybrid H_2 : This hybrid is identical to $H_{1,n,\ell}$, except for syntactic changes. The challenger chooses $\{g_{i,b} \leftarrow \mathcal{G}\}_{i,b}$, $\{\rho_{i,j,b}\}_{i,j,b}$ and sets pp as in the previous hybrid. Next, it chooses $s \leftarrow \{0,1\}^n$ and computes $h = \prod_{k=1}^n g_{i,s_i}$. It then uses h to compute y_0 and $\{y_i\}_i$. It sets the j^{th} bit of y_0 as $\mathsf{hcb}\left(h^{\rho_{0,j,0}}; r_{0,j}\right)$, and the j^{th} bit of $y_{i,b}$ as $\mathsf{hcb}\left(h^{\rho_{i,j,b}}; r_{i,j,b}\right)$.

Hybrid H_3 : This hybrid is similar to the previous one, except that the challenger chooses $h \leftarrow \mathcal{G}$.

Next, we consider ℓ hybrid experiments H_{4,j^*} for $j^* \in [\ell]$, where we switch each bit of y_0 to be uniformly random. Let $H_{4,0} \equiv H_3$.

Hybrid H_{4,j^*} : This hybrid is similar to H_3, except that some bits of y_0 are uniformly random. For $j \leq j^*$, the challenger chooses the j^{th} bit of y_0 uniformly at random. For all $j > j^*$, the j^{th} bit of y_0 is computed as $\mathsf{hcb}\left(h^{\rho_{0,j,0}}; r_{0,j}\right)$.

Next, we define $\ell \cdot n$ hybrid experiments H_{5,i^*,j^*} for each $i^* \in [n]$, $j^* \in [\ell]$. For each $(i,j) \preceq (i^*,j^*)$, we switch the j^{th} bit of $y_{i,0}$ and $y_{i,1}$ to be uniformly random.

Hybrid H_{5,i^*,j^*} : In this experiment, for all $(i,j) \preceq (i^*,j^*)$, the challenger sets the j^{th} bit of $y_{i,0}$ and $y_{i,1}$ to be a uniformly random bit. For all other (i,j), it sets the j^{th} bit of $y_{i,b}$ to be $\mathsf{hcb}\left(h^{\rho_{i,j,b}}; r_{i,j,b}\right)$.

Analysis We need to show that the hybrids H_0 and $H_{5,n,\ell}$ are computationally indistinguishable. Let $\mathsf{adv}_x^{\mathcal{A}}$ denote the advantage of an adversary $\mathcal{A}$ in hybrid H_x.

First, we will show that the hybrids $H_{1,i,j}$ are computationally indistinguishable for all $i \in [n], j \in [\ell]$.

Lemma A.1. Assuming CDH is hard on $\mathcal{G}$, for any PPT adversary $\mathcal{A}$ and indices $i^* \in [n], j^* \in [\ell]$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}_{1,i^*,j^*}^{\mathcal{A}} - \mathsf{adv}_{1,i^*,j^*-1}^{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $|\mathsf{adv}_{1,i^*,j^*}^{\mathcal{A}} - \mathsf{adv}_{1,i^*,j^*-1}^{\mathcal{A}}| \geq \epsilon$. Using $\mathcal{A}$, one can build a PPT algorithm $\mathcal{B}$ that can distinguish between the following distributions with advantage ϵ:

$$\mathsf{Dist}_1 = \left\{(g, g^a, h, \mathsf{hcb}(h^a; r), r) \ : \ g, h \leftarrow \mathcal{G}, a \leftarrow \mathbb{Z}_p, r \leftarrow \{0,1\}^{\ell_{\mathsf{hcb}}}\right\}$$
$$\mathsf{Dist}_2 = \left\{(g, g^a, h, \beta, r) \ : \ g, h \leftarrow \mathcal{G}, a \leftarrow \mathbb{Z}_p, r \leftarrow \{0,1\}^{\ell_{\mathsf{hcb}}}, \beta \leftarrow \{0,1\}\right\}$$

The algorithm $\mathcal{B}$ receives (g, g_1, g_2, b, r) from the challenger. It chooses $s \leftarrow \{0,1\}^n$, $\{a_{i,b} \leftarrow \mathbb{Z}_p\}_{(i,b)\neq(i^*,s_{i^*})}$ and sets $g_{i,b} = g^{a_{i,b}}$ for all $(i,b) \neq (i^*, s_{i^*})$. Next, it chooses $\rho_{i,j,b} \leftarrow \mathbb{Z}_p$ and $r_{i,j,b}$ for all $(i,j,b) \neq (i^*, j^*, \overline{s_{i^*}})$. It sets $r_{i^*,j^*,\overline{s_{i^*}}} = r$ and sets $g_{i^*,s_{i^*}} = g_1 / \left(\prod_{k\neq i^*} g_{k,s_k}\right)$. It implicitly sets $\rho_{i^*,j^*,\overline{s_{i^*}}}$ to be the discrete log of g_2 with respect to g.

Using g, g_1, g_2, $\{a_{i,b}\}_{(i,b)\neq(i^*,s_{i^*})}$ and $\{\rho_{i,j,b}\}_{(i,j,b)\neq(i^*,j^*,\overline{s_{i^*}})}$, the reduction algorithm can compute $g_{k,\beta}^{\rho_{i,j,b}}$ for all $(k,\beta,i,j,b) \neq (i^*, s_{i^*}, i^*, j^*, \overline{s_{i^*}})$. If $(k,\beta) = (i^*, s_{i^*})$, it sets $g_{k,\beta}^{i,j,b} = g_1^{\rho_{i,j,b}}$. If $(i,j,b) = (i^*, j^*, \overline{s_{i^*}})$, it sets $g_{k,\beta}^{i,j,b} = g_2^{a_{k,\beta}}$. Since $\mathbf{H}_{i^*,j^*,\overline{s_{i^*}}}[s_{i^*}, i^*] = \perp$, the reduction algorithm can compute the entire public parameters. As a result, the reduction algorithm can also compute y_{k,s_k} (for all k) using pp. For $(i,j) \preceq (i^*,j^*)$, $\mathcal{B}$ can compute the j^{th} bit of $y_{i,\overline{s_i}}$ using $\rho_{i,j,\overline{s_i}}$. For the j^* bit of $y_{i^*,\overline{s_{i^*}}}$, $\mathcal{B}$ uses the challenge bit b. All other bits of $y_{i^*,\overline{s_{i^*}}}$ are chosen uniformly at random, and for all $i > i^*$, it sets $y_{i,\overline{s_i}}$ to be a uniformly random ℓ bit string. Finally, it sends the $y_{i,b}$ strings to the adversary, and forwards the adversary's guess to the challenger.

Clearly, if b is a uniformly random bit, then the adversary's view is same as in H_{1,i^*,j^*-1}. If $g_1 = g^a$ and $b = \mathsf{hcb}(h^a; r)$, then the j^* bit of $y_{i^*,\overline{s_{i^*}}}$ is $\mathsf{hcb}\left((\prod_k g_{k,s_k})^{\rho_{i^*,j^*,\overline{s_{i^*}}}}\right)$, which is equal to $\mathsf{hcb}\left(h^a\right)$. Hence, the adversary's view is same as in H_{1,i^*,j^*}.

Finally, using the hardcore-bit's property, if there exists a PPT algorithm $\mathcal{B}$ that can distinguish between Dist_1 and Dist_2 with non-negligible advantage, then there exists another PPT algorithm $\mathcal{B}'$ that can compute h^a with non-negligible probability. This algorithm $\mathcal{B}'$ can be used to break the CDH assumption on $\mathcal{G}$. □

Observation 1. For any adversary $\mathcal{A}$, $\mathsf{adv}_{1,n,\ell}^{\mathcal{A}} = \mathsf{adv}_2^{\mathcal{A}}$.

Proof. The proof of this observation follows directly from the definition of the hybrid experiments. □

Lemma A.2. For any adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}_2^{\mathcal{A}} - \mathsf{adv}_3^{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma follows from the Leftover Hash Lemma. Since we set $n = \log p + 2\lambda$, the following distributions have statistical distance at most $2^{-\lambda}$:

$$\mathsf{Dist}_1 = \left\{ \{a_{b,i}\}_{b\in\{0,1\}, i\in[n]}, a \ : \ a_{b,i} \leftarrow \mathbb{Z}_p, s \leftarrow \{0,1\}^n, a = \sum_i a_{s_i,i} \right\}$$
$$\mathsf{Dist}_2 = \left\{ \{a_{b,i}\}_{b\in\{0,1\}, i\in[n]}, a \ : \ a_{b,i} \leftarrow \mathbb{Z}_p, a \leftarrow \mathbb{Z}_p \right\}$$

Let g be any generator of $\mathcal{G}$. Then, by setting $g_{i,s_i} = g^{a_i}$, $h = \prod_i g_{i,s_i}$ is statistically indistinguishable from g^a, even if g and $\{a_i\}_i$ are given. This concludes our proof. □

Lemma A.3. Assuming CDH is hard on $\mathcal{G}$, for any PPT adversary $\mathcal{A}$ and indices $j^* \in [\ell]$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}_{4,j^*}^{\mathcal{A}} - \mathsf{adv}_{4,j^*-1}^{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. As in the proof of Lemma A.1, we will use such a PPT adversary $\mathcal{A}$ to build a PPT algorithm $\mathcal{B}$ that can distinguish between Dist_1 and Dist_2 (defined in proof of Lemma A.1). The reduction algorithm receives challenge (g, g_1, g_2, b, r). It sets $h = g_1$, implicitly sets $\rho_{0,j^*,0}$ to be the discrete log of g_2 with respect to g. It chooses $\{a_{i,b}\}_{i,b}$, sets $g_{i,b} = g^{a_{i,b}}$ for all i, b. Next, it chooses $\{\rho_{i,j,b}\}_{(i,j,b)\neq(0,j^*,0)}$, and sets $\mathbf{H}_{0,j^*}[\beta, k] = g_2^{a_{k,\beta}}$ for all (k,β). For all $j \neq j^*$, it sets $\mathbf{H}_{0,j}$ using g and $\{a_{i,b}\}_{i,b}$. Finally, the reduction algorithm sets $r_{0,j^*} = r$, and all other components of the public parameters are chosen honestly. The j^* bit of y_0 is set to be b; for

$j < j^*$, the bits are computed using g_1 and $\rho_{0,j}$. The advantage of $\mathcal{B}$ in distinguishing Dist_1 and Dist_2 is same as the difference of $\mathcal{A}'s$ advantage in H_{4,j^*} and H_{4,j^*-1}. Once we have such a PPT algorithm $\mathcal{B}$, it can be used to break the CDH assumption, using the hardcore-bit property. $\square$

Lemma A.4. Assuming CDH is hard on $\mathcal{G}$, for any PPT adversary $\mathcal{A}$ and indices $i^* \in [\ell], j^* \in [\ell]$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}^{\mathcal{A}}_{5,i^*,j^*} - \mathsf{adv}^{\mathcal{A}}_{5,i^*,j^*-1}| \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma is similar to the proof of Lemma A.3. $\square$

A.2 Construction Based on the LWE Assumption

A.2.1 Learning with Errors Assumption

The Learning with Errors (LWE) problem was introduced by Regev [Reg05], who showed that solving LWE on *average* is as hard as quantumly solving several standard lattice based problems in the worst case. The LWE assumption states that no polynomial time adversary can distinguish between the following oracles. In one case, the oracle chooses a uniformly random secret $\mathbf{s}$, and for each query, it chooses a vector $\mathbf{a}$ uniformly at random, scalar e from a noise distribution and outputs $(\mathbf{a}, \mathbf{s}^T \cdot \mathbf{a} + e)$. In the second case, the oracle simply outputs a uniformly random vector $\mathbf{a}$ together with a uniformly random scalar u. Regev showed that if there exists a polynomial time adversary that can break the LWE assumption, then there exists a polynomial time quantum algorithm that can solve some hard lattice problems in the worst case.

We will use a variant of the LWE assumption, where the challenger outputs some LWE samples, followed by a challenge, which is either an LWE sample or a uniformly random element. This version is as hard as the standard version.

Assumption 2 (Learning with Errors)**.** Let n and q be positive integers and χ a noise distribution over $\mathbb{Z}_q$. The Learning with Errors assumption $\mathsf{LWE}_{n,q,\chi}$, parameterized by n, q, χ, states that for any polynomial N, the following distributions are computationally indistinguishable:

$$\left\{ \left\{(\mathbf{a}_i, \mathbf{s}^T \cdot \mathbf{a}_i + e_i)\right\}_i, (\mathbf{a}, \mathbf{s}^T \cdot \mathbf{a} + e) \;:\; \begin{array}{c} \mathbf{a}_i \leftarrow \mathbb{Z}_q^n \text{ for each } i \in [N], \\ \mathbf{a}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \\ e, e_i \leftarrow \chi \text{ for each } i \in [N] \end{array} \right\}$$
$$\approx_c$$
$$\left\{ \left\{(\mathbf{a}_i, \mathbf{s}^T \cdot \mathbf{a}_i + e_i)\right\}_i, (\mathbf{a}, r) \;:\; \begin{array}{c} \mathbf{a}_i \leftarrow \mathbb{Z}_q^n \text{ for each } i \in [N], \\ \mathbf{a}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, r \leftarrow \mathbb{Z}_q \\ e, e_i \leftarrow \chi \text{ for each } i \in [N] \end{array} \right\}$$

A.2.2 Construction

We will now describe our LWE based hinting PRG scheme.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses LWE dimension d, modulus $p = 2^{d^\epsilon}$, noise parameter $\sigma = \mathsf{poly}(\lambda)$, noise distribution $\chi = \mathcal{D}_\sigma$ and $n = d \cdot \log p + 2\lambda$. It then chooses $2 \cdot n$ uniformly random vectors $\left\{\mathbf{a}_{i,b} \leftarrow \mathbb{Z}_p^d\right\}_{i \in [n], b \in \{0,1\}}$, $2 \cdot (n+1) \cdot \ell$ uniformly random vectors $\left\{\mathbf{v}_{i,j,b} \leftarrow \mathbb{Z}_p^d\right\}_{i \in [n] \cup \{0\}, j \in [\ell], b \in \{0,1\}}$. For each $i \in [n], j \in [\ell], b \in \{0,1\}$, it chooses $e^{k,\beta}_{i,j,b} \leftarrow \chi$ for each $i, k \in [n], j \in [\ell], b, \beta \in \{0,1\}$, sets a $2 \times n$ matrix $\mathbf{H}_{i,j,b}$ as follows:

$$\forall (\beta, k) \in \{0,1\} \times [n],\ \mathbf{H}_{i,j,b}[\beta, k] = \begin{cases} \perp \text{ if } (k, \beta) = (i, \bar{b}) \\ \mathbf{a}^T_{k,\beta} \cdot \mathbf{v}_{i,j,b} + e^{k,\beta}_{i,j,b} \text{ otherwise} \end{cases}$$

Similarly, it chooses $e^{k,\beta}_j \leftarrow \chi$ for each $j \in [\ell], k \in [n], b, \beta \in \{0,1\}$. For each $j \in [\ell]$, it sets $\mathbf{H}_{0,j}$ as follows:

$$\forall (\beta, k) \in \{0,1\} \times [n],\ \mathbf{H}_{0,j}[\beta, k] = \mathbf{a}^T_{k,\beta} \cdot \mathbf{v}_{0,j,0} + e^{k,\beta}_j$$

The setup algorithm sets $\mathsf{pp} = \left(\{\mathbf{H}_{0,j}\}_{j \in [\ell]}, \{\mathbf{H}_{i,j,b}\}_{i \in [n], j \in [\ell], b \in \{0,1\}} \right)$.

$\mathsf{Eval}(\mathsf{pp}, s, i)$: Let $\mathsf{pp} = \left(\{\mathbf{H}_{0,j}\}_{j\in[\ell]}, \{\mathbf{H}_{i,j,b}\}_{i\in[n],j\in[\ell],b\in\{0,1\}}\right)$. If $i \neq 0$, the evaluation algorithm does the following: For each $j \in [\ell]$, the evaluation algorithm sets $z_j = \mathsf{msb}\left(\sum_{k=1}^{n} \mathbf{H}_{i,j,s_i}[s_k, k]\right)$ and outputs $z = z_1 z_2 \dots z_\ell$.

If $i = 0$, the evaluation algorithm does the following: For each $j \in [\ell]$, it sets $z_j = \mathsf{msb}\left(\sum_{k=1}^{n} \mathbf{H}_{0,j}[s_k, k]\right)$ and outputs $z = z_1 z_2 \dots z_\ell$.

A.2.3 Proof of Security

To prove security, we will first present a sequence of hybrid experiments, and then show that the hybrid experiments are computationally indistinguishable.

Hybrid H_0 : This is the original security game. The challenger chooses a uniformly random string $s \leftarrow \{0,1\}^n$, chooses $\{\mathbf{a}_{i,b}\}_{i,b}$, $\{\mathbf{v}_{i,j,b}\}_{i,j,b}$, sets public parameters $\mathsf{pp} = \left(\{\mathbf{H}_{0,j}\}_j, \{\mathbf{H}_{i,j,b}\}_{i,j,b}\right)$ as in the construction. Next, it sets $y_0 = \mathsf{Eval}(\mathsf{pp}, s, 0)$, $y_{i,s_i} = \mathsf{Eval}(\mathsf{pp}, s, i)$ and $y_{i,\overline{s_i}} \leftarrow \{0,1\}^\ell$. It outputs pp, $\left(y_0, (y_{i,0}, y_{i,1})_i\right)$.

Next, we define $n \cdot \ell$ hybrid experiments $H_{1,i,j}$ for $i \in [n], j \in [\ell]$. For any tuples (i,j) and (i',j'), we say that $(i,j) \preceq (i',j')$ if either $i < i'$ or ($i = i'$ and $j \leq j'$). Also, let $H_0 \equiv H_{1,1,0}$ and $H_{1,i,\ell} \equiv H_{1,i+1,0}$.

Hybrid H_{1,i^*,j^*} : In this hybrid, the challenger does not choose all $\{y_{i,\overline{s_i}}\}_i$ uniformly at random. Instead, for all $(i,j) \preceq (i^*,j^*)$, it sets the j^{th} bit of $y_{i,\overline{s_i}}$ to be $\mathsf{msb}\left(\sum_{k=1}^{n}\left(\mathbf{a}_{k,s_k}^T \cdot \mathbf{v}_{i,j,\overline{s_i}} + e_{i,j,s_i}^{k,s_k}\right)\right)$. For all $j > j^*$, the j^{th} bit of $y_{i^*,\overline{s_{i^*}}}$ are chosen uniformly at random. For all $i > i^*$, the string $y_{i,\overline{s_i}}$ is chosen uniformly at random from $\{0,1\}^\ell$.

Hybrid H_2 : This hybrid is similar to $H_{1,n,\ell}$, except for the manner in which the $y_{i,b}$ strings are computed. Let χ' be the distribution obtained by adding n samples from the χ distribution. The challenger chooses $\left\{\mathbf{a}_{i,b} \leftarrow \mathbb{Z}_p^d\right\}_{i,b}$, $\{\mathbf{v}_{i,j,b}\}_{i,j,b}$ and sets pp as in $H_{1,n,\ell}$. Next, it chooses $s \leftarrow \{0,1\}^n$ and computes $\mathbf{a} = \sum_{k=1}^{n} \mathbf{a}_{i,s_i}$. It then uses $\mathbf{a}$ to compute y_0 and $\{y_{i,b}\}_i$. It sets the j^{th} bit of y_0 as $\mathsf{msb}\left(\mathbf{a}^T \cdot \mathbf{v}_{0,j,0} + e_{0,j}\right)$, and the j^{th} bit of $y_{i,b}$ as $\mathsf{msb}\left(\mathbf{a}^T \cdot \mathbf{v}_{i,j,b} + e_{i,j,b}\right)$, where all the $\{e_{0,j}\}_j$ and $\{e_{i,j,b}\}_{i,j,b}$ are chosen from χ' distribution.

Hybrid H_3 : This hybrid is similar to the previous one, except that the challenger chooses $\mathbf{a} \leftarrow \mathbb{Z}_p^d$.

Next, we consider ℓ hybrid experiments H_{4,j^*} for $j^* \in [\ell]$, where we switch each bit of y_0 to be uniformly random. Let $H_{4,0} \equiv H_0$.

Hybrid H_{4,j^*} : This hybrid is similar to H_3, except that some bits of y_0 are uniformly random. For $j \leq j^*$, the challenger chooses the j^{th} bit of y_0 uniformly at random. For all $j > j^*$, the j^{th} bit of y_0 is computed as $\mathsf{msb}\left(\mathbf{a}^T \cdot \mathbf{v}_{0,j,0} + e_{0,j}\right)$.

Next, we define $\ell \cdot n$ hybrid experiments H_{5,i^*,j^*} for each $i^* \in [n]$, $j^* \in [\ell]$. For each $(i,j) \preceq (i^*,j^*)$, we switch the j^{th} bit of $y_{i,0}$ and $y_{i,1}$ to be uniformly random.

Hybrid H_{5,i^*,j^*} : In this experiment, for all $(i,j) \preceq (i^*,j^*)$, the challenger sets the j^{th} bit of $y_{i,0}$ and $y_{i,1}$ to be a uniformly random bit. For all other (i,j), it sets the j^{th} bit of $y_{i,b}$ to be $\mathsf{msb}\left(\mathbf{a}^T \cdot \mathbf{v}_{i,j,b} + e_{i,j,b}\right)$.

Analysis We need to show that the hybrids H_0 and $H_{5,n,\ell}$ are computationally indistinguishable. Let $\mathsf{adv}_x^{\mathcal{A}}$ denote the advantage of an adversary $\mathcal{A}$ in hybrid H_x, and let $H_{1,1,0} \equiv H_0$, $H_{1,i,0} \equiv H_{1,i-1,\ell}$.

First, we will show that the hybrids $H_{1,i,j}$ are computationally indistinguishable for all $i \in [n], j \in [\ell]$.

Lemma A.5. Assuming $\mathsf{LWE}_{p,d,\chi}$ is hard, for any PPT adversary $\mathcal{A}$ and indices $i^* \in [n], j^* \in [\ell]$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}_{1,i^*,j^*}^{\mathcal{A}} - \mathsf{adv}_{1,i^*,j^*-1}^{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{adv}_{1,i^*,j^*}^{\mathcal{A}} - \mathsf{adv}_{1,i^*,j^*-1}^{\mathcal{A}} = \epsilon$. We will show a PPT algorithm $\mathcal{B}$ that breaks $\mathsf{LWE}_{p,d,\chi}$ with advantage ϵ.

The reduction algorithm chooses $s \leftarrow \{0,1\}^n$ and vectors $\{\mathbf{v}_{i,j,b}\}_{(i,j,b)\neq(i^*,j^*,\overline{s_{i^*}})}$. It queries the LWE challenger for $2n-1$ queries and one challenge, and receives $\{(\mathbf{a}_{k,\beta}, h_{k,\beta})\}$ where $\mathbf{a}_{i^*,\overline{s_{i^*}}}, h_{i^*,\overline{s_{i^*}}}$ is the challenge tuple and the remaining are LWE samples. The reduction algorithm sets $\mathbf{H}_{i^*,j^*,\overline{s_{i^*}}}[\beta, k] = h_{k,\beta}$ if $(k,\beta) \neq (i^*, \overline{s_{i^*}})$, and $\perp$ otherwise. The remaining matrices $\{\mathbf{H}_{i,j,b}\}_{(i,j,b)\neq(i^*,j^*,\overline{s_{i^*}})}$ and $\{\mathbf{H}_{0,j}\}_j$ are computed using $\{\mathbf{a}_{i,b}\}$ and $\{\mathbf{v}_{i,j,b}\}_{(i,j,b)\neq(i^*,j^*,\overline{s_{i^*}})}$. This completes the public parameters.

For the evaluations, the reduction algorithm computes y_0, all $\{y_{i,b}\}_{(i,b)\neq(i^*,\overline{s_{i^*}})}$ as in hybrids $H_{1,i^*,j^*-1}/H_{1,i^*,j^*}$. It also computes all bits of $y_{i^*,\overline{s_{i^*}}}$, except the j^{th} one. For the j^{th} bit, the reduction algorithm computes $\mathsf{msb}\left(\sum_{k\neq i} \mathbf{H}_{i^*,j^*,\overline{s_{i^*}}}[s_k, k] + h_{i^*,j^*,\overline{s_{i^*}}}\right)$. The adversary $\mathcal{A}$ sends its guess b', and $\mathcal{B}$ forwards the same to the LWE challenger.

If $(\mathbf{a}_{i^*,j^*,\overline{s_{i^*}}}, h_{i^*,j^*,\overline{s_{i^*}}})$ form an LWE sample, then this hybrid corresponds to H_{1,i^*,j^*}. If $h_{i^*,j^*,\overline{s_{i^*}}}$ is truly random, then this corresponds to H_{1,i^*,j^*-1}. Therefore, $\mathcal{B}$ can break $\mathsf{LWE}_{p,d,\chi}$ with advantage ϵ. ☐

Lemma A.6. For any adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}_{1,n,\ell}^{\mathcal{A}} - \mathsf{adv}_2^{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. In hybrid $H_{1,n,\ell}$, for each $i \in [n], j \in [\ell], b \in \{0,1\}$, the j^{th} bit of y_{i,s_i} is set to be $\mathsf{msb}\left(\sum_{k=1}^n \mathbf{H}_{i,j,s_i}[s_k, k]\right)$, and the j^{th} bit of $y_{i,\overline{s_i}}$ is $\mathsf{msb}\left(\sum_{k=1}^n \mathbf{a}_{k,s_k}^T \cdot \mathbf{v}_{i,j,b} + e_{i,j,b}^{k,s_k}\right)$. In hybrid H_2, the j^{th} bit of $y_{i,b}$ is set to be $\mathsf{msb}\left(\mathbf{a}^T \cdot \mathbf{v}_{i,j,b} + e_{i,j,b}\right)$ for all $(i,j,b) \in [n] \times [\ell] \times \{0,1\}$ (where $e_{i,j,b}$ is chosen from χ'). Clearly, the distributions are identical for $\{y_{i,\overline{s_i}}\}_i$, even given pp.

To show that the two hybrids are statistically indistinguishable, it suffices to argue that the statistical distance between the following distributions is negligible in λ:

$$\mathsf{Dist}_1 = \left\{ \left(\{\mathbf{H}_{i,j,b}\}_{i,j,b}, \{z_{i,j}\}_{i,j}\right) \;:\; \begin{array}{c} \mathbf{a}_{i,b}, \mathbf{v}_{i,j,b} \leftarrow \mathbb{Z}_p^d, e_{i,j,b}^{k,\beta} \leftarrow \chi \\ \mathbf{H}_{i,j,b}[\beta, k] = \mathbf{a}_{k,\beta}^T \cdot \mathbf{v}_{i,j,b} + e_{i,j,b}^{k,\beta} \\ z_{i,j} = \mathsf{msb}\left(\sum_k \mathbf{H}_{i,j,s_i}[s_k, k]\right) \end{array} \right\}$$

$$\mathsf{Dist}_2 = \left\{ \left(\{\mathbf{H}_{i,j,b}\}_{i,j,b}, \{z_{i,j}\}_{i,j}\right) \;:\; \begin{array}{c} \mathbf{a}_{i,b}, \mathbf{v}_{i,j,b} \leftarrow \mathbb{Z}_p^d, e_{i,j,b}^{k,\beta} \leftarrow \chi, e_{i,j} \leftarrow \chi' \\ \mathbf{H}_{i,j,b}[\beta, k] = \mathbf{a}_{k,\beta}^T \cdot \mathbf{v}_{i,j,b} + e_{i,j,b}^{k,\beta} \\ z_{i,j} = \mathsf{msb}\left(\left(\sum_k \mathbf{a}_{k,s_k}\right)^T \cdot \mathbf{v}_{i,j,s_i} + e_{i,j}\right) \end{array} \right\}$$

Since $e_{i,j,b}^{k,\beta} \leftarrow \chi$ and $e_{i,j} \leftarrow \chi'$, with all but negligible probability, $|e_{i,j,b}^{k,\beta}| \leq \sqrt{d} \cdot \sigma$, and $|e_{i,j}| \leq \sqrt{d} \cdot \sigma \cdot n$. With overwhelming probability (over the choice of $\{\mathbf{a}_{i,b}\}$ and $\{\mathbf{v}_{i,j,b}\}$), all the $|\mathbf{a}_{k,\beta} \cdot \mathbf{v}_{i,j,b}|$ values are at least $\sqrt{d} \cdot \sigma \cdot n$ and most $p/2 - \sqrt{d} \cdot \sigma \cdot n$. This is because p is exponential in λ while σ is $\mathsf{poly}(\lambda)$. Hence, conditioned on these two events,

$$\mathsf{msb}\left(\sum_k \mathbf{a}_{k,s_k}^T \cdot \mathbf{v}_{i,j,s_i} + e_{i,j,s_i}^{k,s_k}\right) = \mathsf{msb}\left(\left(\sum_k \mathbf{a}_{k,s_k}\right)^T \cdot \mathbf{v}_{i,j,s_i} + e_{i,j}\right)$$

☐

Lemma A.7. For any adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}_2^{\mathcal{A}} - \mathsf{adv}_3^{\mathcal{A}}| \le \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma follows from the Leftover Hash Lemma. Since we set $n = d \cdot \log p + 2\lambda$, the following distributions have statistical distance at most $2^{-\lambda}$:

$$\mathsf{Dist}_1 = \left\{ \{\mathbf{a}_{b,i}\}_{b \in \{0,1\}, i \in [n]}, \mathbf{a} \ : \ \mathbf{a}_{b,i} \leftarrow \mathbb{Z}_p^d, s \leftarrow \{0,1\}^n, \mathbf{a} = \sum_i \mathbf{a}_{s_i,i} \right\}$$

$$\mathsf{Dist}_2 = \left\{ \{\mathbf{a}_{b,i}\}_{b \in \{0,1\}, i \in [n]}, \mathbf{a} \ : \ \mathbf{a}_{b,i} \leftarrow \mathbb{Z}_p^d, \mathbf{a} \leftarrow \mathbb{Z}_p^d \right\}$$

□

Lemma A.8. Assuming $\mathsf{LWE}_{p,d,\chi'}$, for any $j^* \in [\ell]$, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}_{4,j^*}^{\mathcal{A}} - \mathsf{adv}_{4,j^*-1}^{\mathcal{A}}| \le \mathsf{negl}(\lambda)$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $|\mathsf{adv}_{4,j^*}^{\mathcal{A}} - \mathsf{adv}_{4,j^*-1}^{\mathcal{A}}| = \epsilon$. We will build a PPT algorithm $\mathcal{B}$ that breaks $\mathsf{LWE}_{p,d,\chi'}$ with advantage ϵ.

The reduction algorithm queries for n evaluations followed by a challenge, and receives $\{\mathbf{a}_{i,b}, h_{i,b}\}_{i,b}$ as the LWE evaluations, and $(\mathbf{a}, h)$ as the challenge. It sets $\mathbf{H}_{0,j^*}[\beta, k] = h_{k,\beta}$ for all $(k, \beta) \in [n] \times \{0,1\}$. It chooses $\mathbf{v}_{0,j,0}$ for all $j \ne j^*$ and sets $\mathbf{H}_{0,j}$ for all $j \ne j^*$ as in $H_{4,j^*}/H_{4,j^*-1}$. It also chooses $\{\mathbf{v}_{i,j,b}\}_{i>0}$, and sets $\{\mathbf{H}_{i,j,b}\}_{i>0}$. The reduction algorithm computes $\{y_{i,b}\}_{i>0}$ as in $H_{4,j^*}/H_{4,j^*-1}$ using $\mathbf{a}$ and $\{\mathbf{v}_{i,j,b}\}$ vectors. For y_0, it sets the j^* bit as the LWE challenge h, and the other bits are computed as in $H_{4,j^*}/H_{4,j^*-1}$. Finally, the adversary sends its guess, which is forwarded to the LWE challenger. The reduction algorithm $\mathcal{B}$ has advantage ϵ.

□

Lemma A.9. Assuming $\mathsf{LWE}_{p,d,\chi'}$, for any $i^* \in [n], j^* \in [\ell]$, any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{adv}_{5,i^*,j^*}^{\mathcal{A}} - \mathsf{adv}_{5,i^*,j^*-1}^{\mathcal{A}}| \le \mathsf{negl}(\lambda)$.

Proof. This proof is similar to the proof of Lemma A.8. □