Simpler Constructions of Asymmetric Primitives from Obfuscation

Pooya Farshim[1,2], Georg Fuchsbauer[1,2], and Alain Passelègue[3]

[1] DI/ENS, CNRS, PSL University, Paris, France
[2] Inria, Paris, France
[3] UCLA, Los Angeles, USA

pooya.farshim@ens.fr georg.fuchsbauer@ens.fr alapasse@gmail.com

Abstract. We revisit constructions of asymmetric primitives from obfuscation and give simpler alternatives. We consider public-key encryption, (hierarchical) identity-based encryption ((H)IBE), and predicate encryption. Obfuscation has already been shown to imply PKE by Sahai and Waters (STOC'14) and full-fledged functional encryption by Garg et al. (FOCS'13). We simplify all these constructions and reduce the necessary assumptions on the class of circuits that the obfuscator needs to support. Our PKE scheme relies on just a PRG and does not need any puncturing. Our IBE and bounded HIBE schemes convert natural key-delegation mechanisms from (recursive) applications of puncturable PRFs to IBE and HIBE schemes. Our most technical contribution is an *unbounded* HIBE, which uses (public-coin) differing-inputs obfuscation for circuits and whose proof relies on a recent pebbling-based hybrid argument by Fuchsbauer et al. (ASIACRYPT'14). All our constructions are anonymous, support arbitrary inputs, and have compact keys and ciphertexts.

Keywords. Obfuscation, public-key encryption, identity-based encryption, hierarchical IBE, predicate encryption, puncturable PRF, pebbling.

1 Introduction

Indistinguishability obfuscation [BGI+12, GGH+13] has rekindled interest in the search for a unified approach to constructions of cryptographic primitives. It has been proven sufficient to build almost every primitive, from the most basic such as public-key encryption or short signatures, to powerful ones such as functional encryption for all circuits. Indistinguishability obfuscation has been used extensively as a building block for many other constructions [GGH+13, SW14, HSW14, BZ14, BLR+15, HW15, CLP15, PPS15, AS16].

1.1 Motivation

In this paper, we take a second look at constructions of basic cryptographic primitives and try to obtain the simplest possible constructions. Devising simple constructions based on indistinguishability obfuscation (iO) presents multiple

interests. iO for NC^1, and even for smaller classes, can be extended to iO for P/poly via bootstrapping theorems [GGH+13, CLTV15, Lin16] at the cost of strong assumptions such as sub-exponentially secure pseudorandom generators in NC^0 or the sub-exponential hardness of the learning with errors problem. On the other hand, certain simple circuits, such as those with logarithmic input lengths or point functions, can be easily obfuscated.

Therefore, depending on the type of circuits that need to be obfuscated, assuming the existence of iO can be either a weak or an extremely strong assumption. In particular, there seems to be an enormous gap between assuming the existence of iO for P/poly or "bootstrappable" classes of circuits and those for small sets of circuits. Hence, understanding the minimal class of circuits that one needs to obfuscate in order to build various cryptographic primitives is an interesting theoretical measure of their "cryptographic complexity." Indeed, constructing primitives using indistinguishability obfuscation for a small enough class of circuits could lead to efficient constructions of them based on standard assumptions.

1.2 Contributions

In this work, we study public-key encryption (PKE), (hierarchical) identity-based encryption (IBE) and predicate encryption with the goal of minimizing the required class of obfuscatable circuits.

Our PKE scheme, which we sketch below, will demonstrate the core (conceptual) idea behind all schemes that we study in this work: using iO we can convert a (trivial) key-delegation mechanism for a single user (namely a single secret key for that user) to an asymmetric encryption scheme that supports this key delegation.

We show that this idea lends itself to natural generalizations. The key delegation underlying an IBE can be seen as a tree of height one with the trusted authority sitting at the root and the users at its leaves. Keys are derived by applying a pseudorandom function (PRF) to leaf identities under the root key. A hierarchical delegation generalizes this to trees of arbitrary height with a PRF iteratively applied to identities along a path. We show the technique underlying our PKE can convert these key-delegation schemes to encryption schemes that support them respectively.

The HIBE construction just sketched supports a bounded number of levels. As we shall discuss shortly, a number of obstacles prevent us from directly extending the bounded HIBE to an unbounded one. Nevertheless we show that these obstacles can be overcome by introducing a somewhat more involved key delegation (which ensures the compactness of the key delegation operation) and at the expense of introducing stronger primitives.[4] We see this result as the main technical contribution of the paper.

[4] Although the proof of this construction is somewhat complex, the construction itself is relatively simple given that it achieves an unbounded number of levels.

Public-key encryption. Following [GGSW13, BNPW16], we start by constructing a simple public-key encryption scheme. The secret key of the scheme is a random seed $sk \in \{0,1\}^\lambda$ and the corresponding public key is $G(sk)$ where G is a length-doubling pseudorandom generator (PRG). Encrypting a message M under pk consists of obfuscating a circuit that takes as input some $sk' \in \{0,1\}^\lambda$, and checks if $G(sk') = pk$, in which case it outputs the message M, otherwise outputs $\perp$. To decrypt, a user runs the ciphertext (which is an obfuscated circuit) on input the secret key sk. If sk corresponds to the public key, the check passes and the message M is returned; the scheme is thus correct.

Security follows from the security of the PRG, which ensures the public key $G(sk)$ is indistinguishable from a random bit string of the same length. Now, if pk is outside of the range of G, there is no sk' for which $G(sk') = pk$ and encryptions under pk are (obfuscated) circuits that *always* output $\perp$. By the security of the obfuscator, this circuit reveals no more information than a circuit that is independent of the message and always returns $\perp$. This arguably very simple construction can be also shown to be anonymous, meaning that its ciphertexts do not reveal their intended recipients.

Due to its simplicity, we show that our construction can be *efficiently* instantiated based on the standard DDH assumption. In particular, we give a PRG and an indistinguishability obfuscator based on DDH, that when used within the construction result in an anonymous (and lossy) version of ElGamal.

Identity-based encryption. We now apply the idea to key-delegation mechanism for IBEs. In our scheme the secret key of an identity id is the output of a pseudorandom function $F(K, id)$, where K is the master secret key. The master public key is an obfuscated circuit that on input id outputs its public key $pk_{id} := G(F(K, id))$. Encryption and decryption work *as in the PKE scheme.*

To prove the security of the scheme, similarly to our PKE, we need to switch the public key of the challenge identity id^* from $G(F(K, id^*))$ to a random value. Intuitively, this should follow from the pseudorandomness of F. But the IBE game also contains an obfuscated circuit depending on the master key K, and furthermore key extraction for identities $id \neq id^*$ should be simulated. Luckily, this restriction in the IBE security game matches the functionality and security of *puncturable* PRFs: from K we can derive a punctured key K^* that allows evaluating the PRF everywhere except at id^* and $F(K, id^*)$ is indistinguishable from random even in the presence of K^*. (Such PRFs can be based on the GGM [GGM86] construction only assuming the existence of PRGs [BW13, BGI14, KPTZ13].) As for our PKE, since ciphertexts are indistinguishable from obfuscations of a constant circuit, they hide both their underlying plaintexts and recipient identities, giving rise to an IBE that is *anonymous* and can therefore be used, for example, in the context of public-key encryption with keyword search [ABC+05].

We leave it as an interesting open problem to instantiate the IBE construction based on standard assumptions (as we did for PKEs via ElGamal). Doing so could shed further light on the construction of these primitives and conceptually clarify the rationale behind them through their lens of obfuscation.

Bounded HIBE. In an HIBE, identities take the form $id = (id_1, \ldots, id_i)$ and a key for $(id_1, \ldots, id_i)$ allows deriving keys for $(id_1, \ldots, id_{i+1})$ for any id_{i+1}. By recursively applying the PRF to identities along a path we arrive at a hierarchical key-delegation mechanism. Following the PKE construction, we can then convert this to an HIBE scheme as follows. Our scheme defines the master secret key as a random value $sk_0 = K$, and sets the key for id_1 to $sk_1 := F(K, id_1)$, that for (id_1, id_2) to $sk_2 := F(sk_1, id_2)$, and so on. We follow the encryption and decryption procedures of the PKE scheme to arrive at a full scheme. When the puncturable GGM PRF is used, it is easy to see how the puncturability of the PRF maps to the ability to extract keys for identities that are not parents of the challenge identity.

This HIBE only supports an a priori bounded number of levels. To prove security, we need to replace the public key for the challenge identity id^*, computed as

$$pk^* = G(F(\ldots F(F(K, id_1^*), id_2^*) \ldots, id_n^*)) \ ,$$

by a random value. To this end, we first define a game where we change the master circuit (used to compute public keys) and hardwire all values that we will change later in the proof. In more detail, we replace K by K^* punctured at id_1^*, then from $sk_1^* = F(K, id_1^*)$ we derive a key punctured at id_2^* (which allows to derive public keys for all $(id_1^*, id_2, \ldots)$ with $id_2 \neq id_2^*$) and so on until we finally hardwire the public key $pk^* = G(sk_n^*)$ at the lowest level. As we did not change the functionality of the circuit, its obfuscation is indistinguishable from an honestly generated master public key. Note, however, that we *did* change the size of the circuit, and thus need to pad the master circuit *in the scheme* to the maximum size of the circuits that will be used in the proof. (This issue will reappear when extending the construction to an unbounded HIBE.) The rest of the proof proceeds via a sequence of hybrids where we iteratively replace sk_1^* by a random value, then sk_2^*, etc., until we arrive at a random sk_n^*, which together with PRG security allows us to replace $pk^* = G(sk_n^*)$ by a random value.

Unbounded HIBE. There are a number of challenges when extending the HIBE to support an unbounded number of levels (analogously to direct constructions of unbounded HIBEs [LW11]). An obvious one is that circuits cannot take identities of arbitrary lengths as inputs. We could, however, define a Turing machine (TM) that accepts identities of arbitrary lengths and carries out the recursive PRF evaluation to derive its public key. Using the TM-obfuscator of Ishai, Pandey, and Sahai [IPS15], we could then obfuscate this TM. However, when using puncturing, we still need to *hardwire* random values to the circuit in the proof, which would result in a TM whose description size is not a priori bounded. The problem seems to stem from the fact that in order to replace key sk_n^* corresponding to $(id_1^*, \ldots, id_n^*)$ with a random value, we first need to replace *all* intermediate values $sk_1^* := F(K, id_1^*)$, $sk_2^* := F(sk_1^*, id_2^*)$, $\ldots$ with random.

While it is true that we can only replace a value $sk_i^* = F(sk_{i-1}^*, id_i^*)$ by random if sk_{i-1}^* is already random (since the reduction would not know sk_{i-1}^*), the *previous values* need not be random for this game hop. Let $\mathcal{S}_i \subseteq \{1, \ldots, n\}$ denote the set of indices j for which sk_j^* is random in hybrid i. In the proof of

the bounded HIBE, the i-th hybrid replaces sk_i^* by a random value, so we have $\mathcal{S}_i := \{1, \ldots, i\}$. For the proof to go through, the sequence of sets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_\ell$ needs to have the following properties: (A) $\mathcal{S}_0 = \emptyset$ and $n \in \mathcal{S}_\ell$, and (B) every two consecutive sets differ in exactly one element j, and $j-1$ belongs to both sets. Set $\mathcal{S}_0$ with no random values corresponds to the original game and $\mathcal{S}_\ell$ makes the last value corresponding to sk_n^* random. To go from one hybrid to the next, we can replace an additional value by random, since its ancestor is already random (by property (B)).

Fuchsbauer et al. [FKPR14] give such a sequence where $\ell = n^{\log_2 3}$ and with the property that $|\mathcal{S}_i| \leq \log n$. (This corresponds to a pebbling of a path in the GGM tree with logarithmically many pebbles.) Letting n be an upper bound on length of the challenge identity the adversary chooses (which is polynomial in the security parameter λ), we would have a polynomial number ℓ of hybrids, while in each hybrid step we only need to hardwire a *logarithmic* number of values. It seems that we can thus pad the actual scheme to support an exponential number of levels in the hierarchy.

Alas, this approach does not work: we also need to hardwire the *positions* $(id_1^*, \ldots, id_n^*)$ of these values for which there is no upper bound. We therefore use a different approach and *hash* the identities to obtain compact representations of them. We define the secret key for id as $F(K, H(id))$. The problem now is that hashing completely destroys the hierarchical relationship between the identities and we thus need to implement key delegation differently. We do this by adding a second functionality to the public parameters that given a valid secret key sk for a identity id and a value id' returns a secret key $sk' = F(K, H(id, id'))$ for (id, id').

Rather than relying on TM-obfuscators, which can be constructed from public-coin differing-input obfuscation (pc-diO) for circuits [IPS15], we devise *circuits* for computing public keys and delegating secret keys so that we directly rely on pc-diO for circuits.[5] One benefit of this approach is that any (natural) iO for circuits is conjecturally *also a public-coin differing-input obfuscator* for circuits. Hence we avoid the overheads associated with the complex constructions of [IPS15]. Computing public keys can easily be done by defining a circuit $\mathbf{C}^{\text{gen}}(h)$ that returns $G(F(K, h))$ and to get a public key for id we simply run $\mathbf{C}^{\text{gen}}(H(id))$.

The delegation circuit $\mathbf{C}^{\text{del}}$, on the other hand, needs to ensure it only works for descendants. If we relied on pc-diO for *Turing machines*, we could define $\mathbf{C}^{\text{del}}$ on input (sk, id, id') to check whether $G(sk) = G(F(K, H(id)))$, and if so return $F(K, H(id, id'))$. This results in a fairly simple construction of an unbounded HIBE (whose *proof* is nevertheless somewhat involved). At the expense of slightly increasing the complexity of the scheme, we modify this construction so as to only rely on pc-diO for circuits (as discussed above). To do this, we require a user to compactly prove (to the circuit) that two hash values (h, h') correspond to (id, id') with $h = H(id)$ and $h' = H(id, id')$. For this we use a succinct non-

[5] We note that public-coin diO is not known to suffer from impossibility results that apply to its private-coin counterpart [GGHW14, BSW16].

interactive argument of knowledge (SNARK) system and define $\mathbf{C}^{\text{del}}$ as follows: On input (sk, h, h', π) the delegation circuit checks the validity of the proof π for (h, h'), then checks (internally) that $G(sk) = G(F(K, h))$, and if so returns $F(K, h')$.

To bound the number of hardwired values, our proof strategy follows the above ideas. For a set system $\mathcal{S}_0, \ldots, \mathcal{S}_\ell$, we define game G_i as one that replaces the values $pk_j^* = G(F(K, H(id_1^*, \ldots, id_j^*)))$ by random values for all $j \in \mathcal{S}_i$. That is, these values are hardwired into the circuits $\mathbf{C}^{\text{gen}}$ and $\mathbf{C}^{\text{del}}$. Games G_i and G_{i+1} differ in that pk_j^* is random in only one of them, with j defined as $\{j\} := \mathcal{S}_i \triangle \mathcal{S}_{i+1}$ (where $\triangle$ denotes the symmetric difference). To show that these games are indistinguishable, we puncture K at $H(id_1^*, \ldots, id_j^*)$ and use this K^* in $\mathbf{C}^{\text{gen}}$ and $\mathbf{C}^{\text{del}}$. Since we can first hardwire pk_j^*, this does not change the behavior of $\mathbf{C}^{\text{gen}}$, and thus we can rely on iO security.

The behavior of $\mathbf{C}^{\text{del}}$, on the other hand, *does* change, which is why we require (public-coin) differing-inputs obfuscation and must show that it is hard to actually find a differing input. Suppose an adversary finds an input (sk, h, h', π) for which $\mathbf{C}^{\text{del}}$ behaves differently when using the punctured key. This means $h' = H(id_1^*, \ldots, id_j^*)$, but π (together with collision-resistance of H) guarantees that $h = H(id_1^*, \ldots, id_{j-1}^*)$. By property (B) of our set system: $j-1 \in \mathcal{S}_i$; thus the value pk_{j-1}^* corresponding to $G(F(K, H(id_1^*, \ldots, id_{j-1}^*)))$, and thus $G(F(K, h))$, was replaced by random, and hence no sk will satisfy $G(sk) = pk_{j-1}^*$. But this means that $\mathbf{C}^{\text{del}}$ returns $\perp$, which contradicts (sk, h, h', π) being a differing input. Consistently, we require H to be public-coin and collision-resistant [HR04] and the SNARK to be in the common-random-string model [IPS15].[6]

We observe that if identities are restricted to be of bounded length, the hash function can be replaced with the identity function, and the SNARK with a trivial one that simply outputs the witness. These in turn allow us to rely on iO (for circuits) in the $\mathbf{C}^{\text{del}}$ circuit.

Predicate encryption. Predicate encryption (PE) [BW07, KSW08] is a powerful type of public-key encryption that can support, for example, searching for complex queries on encrypted data and includes other forms of PKE such as attribute-based encryption [GPSW06, LOS+10]. A natural construction of a PE scheme also falls out from our techniques. Intuitively, we identify attributes with identities. A secret key $sk_{\mathbf{P}}$ for a predicate $\mathbf{P}$ is then an obfuscated circuit with K and $\mathbf{P}$ hardwired that on input an attribute γ outputs $F(K, \gamma)$ if $\mathbf{P}(\gamma) = 1$ and $\perp$ otherwise. We defer the details of the construction to Appendix 5 under supplementary material.

Complexity of circuits. All of our constructions except PKE obfuscate a puncturable PRF. Thus, the classes of circuits used in these constructions are potentially amenable to further simplifications. However, it is worth noting that

[6] Since our construction relies on public-coin diO, it must be hard to find a differing input even when given the coins used to sample the circuits (whose obfuscations should be indistinguishable). Since a hash collision results in a differing input, it must be hard to find one even given the coins used to sample the hash function.

the obfuscated circuits corresponding to a ciphertext in all of our constructions (specifically, a circuit that evaluates a PRG and possibly a puncturable PRF and outputs a message if the evaluation matches a certain value z) fall into the class of "compute-and-compare" functionalities[7] that can be obfuscated assuming the learning with errors assumption, as revealed by recent works by Wichs and Zirdelis [WZ17] and Goyal, Koppula, and Waters [**?**]. Unfortunately, in our case, the string z is a public key, and the latter results do not apply as they require z to be secret. Yet, these recent works prove that certain classes of programs that appear non-trivial to obfuscate can actually be obfuscated under standard assumptions. This strengthens our belief that simpler constructions based on special-purpose obfuscation is an interesting direction to pursue.

We finally note that puncturing is only required in the proofs. Thus, it would be interesting to devise new PRFs that are easily obfuscatable even though neither puncturing them nor obfuscating their punctured evaluation procedures are efficient. For example, assuming one can obfuscate AES, we either obtain a secure AES-based IBE, or deduce that either AES cannot be punctured or its punctured circuit cannot be obfuscated.

1.3 Related work

Canetti, Rothblum, and Varia [CRV10] adopt a similar approach for building leakage-resilient one-time signatures by first constructing a special-purpose obfuscator for testing hyperplane membership. Similarly, Matsuda and Hanaoka [MH14] propose a construction of CCA-secure encryption based on rerandomizable and composable point function obfuscators.

Garg et al. [GGSW13] show that witness encryption (which is implied by iO) and unique signatures imply identity-based encryption. Such unique signatures can be based on iO and injective one-way functions (or just one-way functions for constructions in the CRS model) [BP15]. Combining these results, one obtains an IBE from (injective) one-way functions and iO, but the construction is fairly complex. In particular, encrypting consists of obfuscating a circuit that runs the verification circuit of a signature scheme, which is itself an obfuscated circuit. Our construction relies on iO and one-way functions and is much simpler. Garg et al. also show that witness encryption combined with a perfectly binding non-interactive commitment scheme implies (key-policy) attribute-based encryption. In particular, their scheme does not guarantee privacy of the attributes, while our predicate encryption scheme does. Garg et al. [GGH+13] also show that full-fledged functional encryption is implied by iO and non-interactive zero-knowledge proofs (and hence so is predicate encryption). Our construction, however, is conceptually simpler, avoids NIZKs, and relies on iO and one-way functions only.

A simple construction of a (semi-adaptive) functional encryption from indistinguishability obfuscation is also described in [Wat15], but it relies on the full

[7] That is a program of the form "if $P(x) = z$ then output m," where x is the input, P a circuit, m a string and z a long pseudorandom secret string.

power of obfuscation, since a functional secret key for f is an obfuscated circuit that evaluates f (amongst other things). Our PE construction follows a different path and obfuscates constrained PRFs. Obfuscating the latter could imply the existence of iO that can be bootstrapped to full iO. Thus our PE construction should be seen as an alternative route to more expressive primitives.

Finally, Brakerski et al. in recent work [BCG+17] construct a hierarchical notion of functional encryption. This implies bounded HIBEs from LWE or low-complexity PRGs and public-key encryption (for their constant-depth and polynomially bounded width delegation structure), and a construction of unbounded HIBE based on unbounded-collusion functional encryption (which implies iO for P/poly assuming subexponential security). Their construction is fairly simple, which is not surprising as functional encryption is powerful enough to subsume IBEs. Since their construction relies on re-encryption techniques, it is inherently public-key. Our construction takes a completely different route and relies on potentially weaker special-purpose obfuscation, while re-encryption techniques lead to indistinguishability obfuscation for any circuit supported by the underlying functional encryption scheme [BV15].

Paper Organization. The rest of the paper is organized as follows. We present general cryptographic tools used throughout the paper in Section 2. Then we successively introduce our constructions of PKE, IBE, PE, and unbounded HIBE in Sections 3,4,5, and 7 respectively. We present our construction of bounded HIBE in Appendix D.

2 Preliminaries

2.1 Notation

We denote the security parameter by $\lambda \in \mathbb{N}$ and assume it is implicitly given to all algorithms in the unary representation 1^λ. By an algorithm we mean a stateless Turing machine. Algorithms are randomized unless stated otherwise and PPT stands for "probabilistic polynomial-time" in the (unary) security parameter. Given a randomized algorithm $\mathcal{A}$ we denote the action of running $\mathcal{A}$ on input(s) $(x_1, \ldots)$ with randomness r and assigning the output(s) to $y_1, \ldots$ by $(y_1, \ldots) \leftarrow \mathcal{A}(x_1, \ldots; r)$, or with fresh randomness if r is not specified. For a finite set X, we denote its cardinality by $|X|$ and the action of sampling a uniformly random element x from X by $x \twoheadleftarrow X$. We denote by $X_1 \triangle X_2$ the symmetric difference (a.k.a. disjunctive union) of two sets X_1, X_2, defined as $X_1 \triangle X_2 = (X_1 \cup X_2) - (X_1 \cap X_2)$. Throughout the paper $\perp$ denotes a special error symbol, ε denotes the empty string and $|$ denotes concatenation. A real-valued function $\mathrm{negl}(\lambda)$ is negligible if $\mathrm{negl}(\lambda) \in \mathcal{O}(\lambda^{-\omega(1)})$. We denote the set of all negligible functions by Negl. Lists are denoted $L_1, L_2, \ldots$ and are implicitly initialized empty.

2.2 Basic tools

We introduce basic cryptographic primitives that we use as tools throughout the paper. In particular, we recall the notions of pseudorandom generator, con-

$\underline{\text{GAME IND}^{\mathcal{A}}_{\text{CPRF}}(\lambda)}$:	$\underline{\text{PROC. PRF}(x)}$:	$\underline{\text{PROC. CHAL}(x)}$:
$b \twoheadleftarrow \{0,1\}$	$L_1 \leftarrow L_1 \cup \{x\}$	$L_3 \leftarrow L_3 \cup \{x\}$
$\text{K} \twoheadleftarrow \{0,1\}^\lambda$	$y \leftarrow \mathbf{CPRF}(\text{K}, x)$	$y \leftarrow \mathbf{CPRF}(\text{K}, x)$
$b' \twoheadleftarrow \mathcal{A}^{\text{PRF},\text{KEY},\text{CHAL}}(1^\lambda)$	return y	if $T[x] = \bot$
for $x' \in L_1, \mathbf{C} \in L_2, x \in L_3$	$\underline{\text{PROC. KEY}(\mathbf{C})}$:	$T[x] \twoheadleftarrow \{0,1\}^{n(\lambda)}$
if $\mathbf{C}(x) = 1 \vee x = x'$	$L_2 \leftarrow L_2 \cup \{\mathbf{C}\}$	if $b = 0$
$b' \leftarrow 0$	$\text{K}_{\mathbf{C}} \leftarrow \mathbf{Cons}(\text{K}, \mathbf{C})$	$y \leftarrow T[x]$
return $(b' = b)$	return $\text{K}_{\mathbf{C}}$	return y

Fig. 1. Security of a constrained PRF.

strained (and puncturable) pseudorandom functions, and indistinguishability obfuscation. Definitions of various public-key primitives are deferred in the corresponding sections.

PRG. A pseudorandom generator is a deterministic algorithm $\mathbf{PRG}(x)$ that takes as input an $x \in \{0,1\}^\lambda$ and outputs a string $y \in \{0,1\}^{\lambda+s(\lambda)}$ for a (positive) polynomial $s(\cdot)$. We say $\mathbf{PRG}$ is secure if for all PPT $\mathcal{A}$

$$\mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\text{IND}^{\mathcal{A}}_{\mathbf{PRG}}(\lambda)\right] - 1 \in \text{NEGL}.$$

$\underline{\text{GAME IND}^{\mathcal{A}}_{\mathbf{PRG}}(\lambda)}$:
$b \twoheadleftarrow \{0,1\}$; $x \twoheadleftarrow \{0,1\}^\lambda$
$y \twoheadleftarrow \{0,1\}^{\lambda+s(\lambda)}$
if $b = 1$, $y \leftarrow \mathbf{PRG}(x)$
$b' \twoheadleftarrow \mathcal{A}(1^\lambda, y)$
return $(b' = b)$

CONSTRAINED PRFS. A constrained pseudorandom function (C-PRF) for a boolean circuit family $\mathsf{CSp} := \{\mathsf{CSp}_\lambda\}_{\lambda\in\mathbb{N}}$, where each $\mathbf{C} \in \mathsf{CSp}_\lambda$ has input space $\{0,1\}^{m(\lambda)}$ for a polynomial $m(\cdot)$, is a tuple of algorithms $(\mathbf{CPRF}, \mathbf{Cons})$ as follows. Algorithm $\mathbf{CPRF}(\text{K}, x)$ is deterministic and on input a key $\text{K} \in \{0,1\}^\lambda$ and $x \in \{0,1\}^{m(\lambda)}$ outputs a string $y \in \{0,1\}^{n(\lambda)}$ for a polynomial $n(\cdot)$. Algorithm $\mathbf{Cons}(\text{K}, \mathbf{C})$ takes a key K and a boolean circuit $\mathbf{C} \in \mathsf{CSp}_\lambda$ and outputs a constrained key $\text{K}_{\mathbf{C}}$. We require that for any $\text{K} \in \{0,1\}^\lambda$, any $\mathbf{C} \in \mathsf{CSp}_\lambda$ and any $x \in \{0,1\}^{m(\lambda)}$

$$\mathbf{C}(x) = 1 \implies \mathbf{CPRF}(\text{K}, x) = \mathbf{CPRF}(\mathbf{Cons}(\text{K}, \mathbf{C}), x) \ .$$

We say that CPRF is a secure constrained PRF if for all PPT $\mathcal{A}$

$$\mathbf{Adv}^{\text{ind}}_{\text{CPRF},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\text{IND}^{\mathcal{A}}_{\text{CPRF}}(\lambda)\right] - 1 \in \text{NEGL} \ ,$$

where game $\text{IND}^{\mathcal{A}}_{\text{CPRF}}(\lambda)$ is shown in Figure 1[8]. We say a constrained PRF is selectively secure if CHAL is queried at the onset and before PRF and KEY.

PUNCTURABLE PRFS. A puncturable pseudorandom function (P-PRF) for a family of sets $\mathsf{SSp} := \{\mathsf{SSp}_\lambda\}_{\lambda\in\mathbb{N}}$, where SSp_λ is a set of polynomial-size sets (i.e., a set system), is a constrained PRF that supports the circuit family $\mathsf{CSp} :=$

[8] For simplicity, table T in Figure 1 is formally defined as a table of size $2^{m(\lambda)}$ which initially contains only $\bot$ but it can easily be reduced to a polynomial-sized data structure by storing the queried inputs/outputs lazily.

$\{\mathsf{CSp}_\lambda\}_{\lambda\in\mathbb{N}}$ where CSp_λ consists of all circuits of the form $\mathbf{C}[S](x)$ for $S \in \mathsf{SSp}_\lambda$ that return 1 iff $x \notin S$. The class of circuits consisting of singleton sets $\mathsf{SSp}_\lambda := \{\{x\} : x \in \{0,1\}^{m(\lambda)}\}$ correspond to singly puncturable PRFs and those consisting of $\mathsf{SSp}_\lambda := \{\{x_1, x_2\} : x_1, x_2 \in \{0,1\}^{m(\lambda)}\}$ to doubly puncturable PRFs. In this work, we will rely on selectively secure singly and doubly puncturable PRFs. Such puncturable PRFs can be based on a PRG via the GGM construction [BW13, BGI14, KPTZ13].

PUBLIC-COIN HASHING. A public-coin hash function with key space $\{0,1\}^\lambda$ and message space $\{0,1\}^*$ is an algorithm $\mathbf{H}(k,x)$ that takes a key $k \in \{0,1\}^\lambda$ and an input $x \in \{0,1\}^*$ and outputs a string $y \in \{0,1\}^{n(\lambda)}$ for some polynomial $n(\cdot)$. The public-coin hash function is collision resistant if for all PPT $\mathcal{A}$

GAME $\mathrm{CR}^{\mathcal{A}}_{\mathbf{H}}(\lambda)$:
$k \twoheadleftarrow \{0,1\}^\lambda$
$(x_0, x_1) \twoheadleftarrow \mathcal{A}(1^\lambda, k)$
return $(x_0 \neq x_1 \wedge$
$\mathbf{H}(k, x_0) = \mathbf{H}(k, x_1))$

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathbf{H},\mathcal{A}}(\lambda) := \Pr\left[\mathrm{CR}^{\mathcal{A}}_{\mathbf{H}}(\lambda)\right] \in \text{NEGL} .$$

OBFUSCATORS. A PPT algorithm **Obf** is called an obfuscator for the (deterministic) circuit class $\mathsf{CSp} := \{\mathsf{CSp}_\lambda\}_{\lambda\in\mathbb{N}}$, where each $\mathbf{C} \in \mathsf{CSp}_\lambda$ has input space $\{0,1\}^\lambda$, if **Obf** on input the security parameter 1^λ and the description of a circuit $\mathbf{C} \in \mathsf{CSp}_\lambda$ outputs a (deterministic) circuit $\overline{\mathbf{C}}$. We require **Obf** to be perfectly correct in the sense that for all $\lambda \in \mathbb{N}$, all $\mathbf{C} \in \mathsf{CSp}_\lambda$, all $\overline{\mathbf{C}} \twoheadleftarrow \mathbf{Obf}(1^\lambda, \mathbf{C})$ and all $x \in \{0,1\}^\lambda$ we have that $\mathbf{C}(x) = \overline{\mathbf{C}}(x)$.

In order to formalize security, we say an adversary $\mathcal{A}$ is *diO-legitimate* if for every PPT extractor $\mathcal{E}$ we have

$$\mathbf{Adv}^{\mathrm{eq}}_{\mathbf{Obf},\mathcal{A},\mathcal{E}}(\lambda) := \Pr\left[\mathrm{EQ}^{\mathcal{E}}_{\mathbf{Obf},\mathcal{A}}(\lambda)\right] \in \text{NEGL} ,$$

where game $\mathrm{EQ}^{\mathcal{E}}_{\mathcal{A}}(\lambda)$ is shown in Figure 2 (right). Algorithm $\mathcal{A}$ is said to be iO-legitimate if the same holds for every *unbounded extractor* $\mathcal{E}$.[9] An obfuscator **Obf** is a public-coin differing-input obfuscator if any diO-legitimate PPT adversary $\mathcal{A}$ we have

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{IND}^{\mathcal{A}}_{\mathbf{Obf}}(\lambda)\right] - 1 \in \text{NEGL} ,$$

where game $\mathrm{IND}^{\mathcal{A}}_{\mathbf{Obf}}(\lambda)$ gives $\mathcal{A}$ access to a left/right (LR) oracle and is shown in Figure 2. We denote such an obfuscator by **diO**. An obfuscator **Obf** is an indistinguishability obfuscator if $\mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{A}}(\lambda)$ is negligible for any iO-legitimate PPT adversary $\mathcal{A}$. We denote such an obfuscator by **iO**.

[9] This is equivalent to requiring that for every query $(\mathbf{C}_0, \mathbf{C}_1)$ queried to LR we have $\mathbf{C}_0(x) = \mathbf{C}_1(x)$ for all $x \in \{0,1\}^\lambda$, i.e., that $\mathbf{C}_0$ and $\mathbf{C}_1$ are functionally equivalent.

GAME $\mathrm{IND}^{\mathcal{A}}_{\mathbf{Obf}}(\lambda)$:	GAME $\mathrm{EQ}^{\mathcal{E}}_{\mathbf{Obf},\mathcal{A}}(\lambda)$:
$b \twoheadleftarrow \{0,1\}$	$r_{\mathcal{A}} \twoheadleftarrow \{0,1\}^{t_{\mathcal{A}}(\lambda)}$; $\mathcal{A}^{\mathrm{LR}}(1^\lambda; r_{\mathcal{A}})$
$b' \twoheadleftarrow \mathcal{A}^{\mathrm{LR}}(1^\lambda)$	$x \twoheadleftarrow \mathcal{E}(1^\lambda, r_{\mathcal{A}}, L_2)$
for $(\mathbf{C}_0, \mathbf{C}_1) \in L_1$	for $(\mathbf{C}_0, \mathbf{C}_1) \in L_1$
if $\lvert\mathbf{C}_0\rvert \neq \lvert\mathbf{C}_1\rvert$ then $b' \leftarrow 0$	if $\mathbf{C}_0(x) \neq \mathbf{C}_1(x)$ return 1
return $(b = b')$	return 0
PROC. $\mathrm{LR}(\mathbf{C}_0, \mathbf{C}_1)$:	PROC. $\mathrm{LR}(\mathbf{C}_0, \mathbf{C}_1)$:
$\overline{\mathbf{C}} \twoheadleftarrow \mathbf{Obf}(1^\lambda, \mathbf{C}_b)$	$r \twoheadleftarrow \{0,1\}^{\mathrm{rl}(\lambda)}$; $\overline{\mathbf{C}} \leftarrow \mathbf{Obf}(1^\lambda, \mathbf{C}_b; r)$
$L_1 \leftarrow L_1 \cup \{(\mathbf{C}_0, \mathbf{C}_1)\}$	$L_1 \leftarrow L_1 \cup \{(\mathbf{C}_0, \mathbf{C}_1)\}$; $L_2 \leftarrow L_2 \cup \{r\}$
return $\overline{\mathbf{C}}$	return $\overline{\mathbf{C}}$

Fig. 2. Left: Indistinguishability game for an obfuscator. **Right**: The functional-equivalence game. Here $t_{\mathcal{A}}$ denotes a polynomial upper-bounding the runtime of $\mathcal{A}$ and $\mathrm{rl}(\cdot)$ denotes the length of the random input of **Obf**.

3 Public-Key Encryption

We start by presenting our simple PKE construction based on iO and a pseudo-random generator with sparse image. Our construction uses an obfuscator that only needs to support simple circuits. In particular, we show this class is simple enough to be instantiated based on the DDH assumption. We will prove the scheme IND-CPA as well as anonymous, meaning that ciphertexts do not reveal their intended recipients.

3.1 Syntax and security

PUBLIC-KEY ENCRYPTION. A public-key encryption (PKE) scheme for message space $\mathsf{MSp} := \{\mathsf{MSp}_\lambda\}_{\lambda\in\mathbb{N}}$ is a tuple of PPT algorithms $\mathrm{PKE} := (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$.

$\mathbf{Gen}(1^\lambda)$: This is the key-generation algorithm, which outputs a key pair $(\mathrm{SK}, \mathrm{PK})$.
$\mathbf{Enc}(\mathrm{PK}, \mathrm{M})$: This is the encryption algorithm, which on input a public key PK and a message M outputs a ciphertext CT.
$\mathbf{Dec}(\mathrm{SK}, \mathrm{CT})$: This is the deterministic decryption algorithm, which on input a secret key SK and a ciphertext CT outputs a message M or the error symbol $\perp$.

The correctness of a PKE scheme requires that for any $\lambda \in \mathbb{N}$, any $(\mathrm{SK}, \mathrm{PK}) \twoheadleftarrow \mathbf{Gen}(1^\lambda)$, any $\mathrm{M} \in \mathsf{MSp}_\lambda$, and $\mathrm{CT} \twoheadleftarrow \mathbf{Enc}(\mathrm{PK}, \mathrm{M})$ we have $\mathbf{Dec}(\mathrm{SK}, \mathrm{CT}) = \mathrm{M}$.

The anonymous and indistinguishable security of a PKE scheme under chosen-plaintext attacks (AI-CPA) [BBDP01] requires that for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\text{ai-cpa}}_{\mathrm{PKE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\text{AI-CPA}^{\mathcal{A}}_{\mathrm{PKE}}(\lambda)\right] - 1 \in \mathrm{NEGL}\ ,$$

where game $\text{AI-CPA}^{\mathcal{A}}_{\mathrm{PKE}}(\lambda)$ is shown in Figure 3. Besides anonymity, this definition implies the standard IND-CPA definition. To see this note that by AI-CPA security, $\mathbf{Enc}(\mathrm{PK}_0, \mathrm{M}_0)$ is indistinguishable from $\mathbf{Enc}(\mathrm{PK}_1, \mathrm{M}_1)$. Furthermore

$\underline{\text{GAME AI-CPA}^{\mathcal{A}}_{\text{PKE}}(\lambda)}$:	$\underline{\text{PROC. LR}(\text{M}_0, \text{M}_1)}$:
$b \twoheadleftarrow \{0,1\}$	$\text{CT} \twoheadleftarrow \mathbf{Enc}(\text{PK}_b, \text{M}_b)$
$(\text{SK}_0, \text{PK}_0) \twoheadleftarrow \mathbf{Gen}(1^\lambda)$	$L_1 \leftarrow L_1 \cup (\text{M}_0, \text{M}_1)$
$(\text{SK}_1, \text{PK}_1) \twoheadleftarrow \mathbf{Gen}(1^\lambda)$	return CT
$b' \twoheadleftarrow \mathcal{A}^{\text{LR}}(\text{PK}_0, \text{PK}_1)$	
for $(\text{M}_0, \text{M}_1) \in L_1$	
if $\lvert\text{M}_0\rvert \neq \lvert\text{M}_1\rvert$ then $b' \leftarrow 0$	
return $(b = b')$	

Fig. 3. The AI-CPA security of a PKE scheme.

$\underline{\text{ALGO. } \mathbf{Gen}(1^\lambda)}$	$\underline{\text{ALGO. } \mathbf{Enc}(\text{PK}, \text{M})}$
$\text{SK} \twoheadleftarrow \{0,1\}^\lambda$	$\text{CT} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}[\text{PK}, \text{M}])$
$\text{PK} \leftarrow \mathbf{PRG}(\text{SK})$	return CT
return (SK, PK)	
$\underline{\text{ALGO. } \mathbf{Dec}(\text{SK}, \text{CT})}$	$\underline{\text{CIRC. } \mathbf{C}[\text{PK}, \text{M}](\text{SK})}$
return $\text{CT}(\text{SK})$	if $\mathbf{PRG}(\text{SK}) = \text{PK}$ return M
	return $\perp$

Fig. 4. Public-key encryption scheme from a PRG and an obfuscator.

$\mathbf{Enc}(\text{PK}_1, \text{M}_1)$ is also indistinguishable $\mathbf{Enc}(\text{PK}_0, \text{M}_1)$ as the adversary can set $\text{M}_1 = \text{M}_0$. Combining the two we get that $\mathbf{Enc}(\text{PK}_0, \text{M}_0)$ is indistinguishable from $\mathbf{Enc}(\text{PK}_0, \text{M}_1)$, as required.

3.2 Construction

We now formalize our public-key encryption scheme. We refer the reader to the introduction for a high-level overview of it. The details of the construction are shown in Figure 4. The proof of the following theorem follows the ideas outlaid in the introduction and can be found in Appendix **??**.

Theorem 1. *Scheme* PKE *in Figure 4 is* AI-CPA *secure if its underlying pseudorandom generator* **PRG** *and obfuscator* **Obf** *are secure. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against* PKE *there are* PPT *adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *against* **PRG** *and* **Obf**, *respectively, such that*

$$\mathbf{Adv}^{\text{ai-cpa}}_{\text{PKE},\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}_2}(\lambda) \ .$$

Algorithm $\mathcal{B}_2$ *places the same number of queries to its (obfuscation)* LR *oracle as* $\mathcal{A}$ *does to its (encryption)* LR *oracle.*

Proof. We prove the theorem via a sequence of games as follows.

G_0**:** This is the AI-CPA game with respect to PKE in Figure 4. By definition we have

$$2 \cdot \Pr[\text{G}^{\mathcal{A}}_0] - 1 = \mathbf{Adv}^{\text{ai-cpa}}_{\text{PKE},\mathcal{A}}(\lambda) \ .$$

G_1**:** This game replaces PK_0 with a truly random bit string of length $\{0,1\}^{2\lambda}$.

We justify this transition based on the security of **PRG**. Let $\mathcal{A}$ be any adversary that distinguishes G_0 from G_1. Consider the following PRG adversary $\mathcal{B}$. Algorithm $\mathcal{B}$ receives a challenge PRG value y, sets $PK_0 := y$, generates a key pair (SK_1, PK_1), picks a random bit b, and runs $\mathcal{A}(PK_0, PK_1)$. When $\mathcal{A}$ queries LR on (M_0, M_1) algorithm $\mathcal{B}$ uses PK_b to encrypt M_b and obtains a ciphertext CT, which it sends to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. When y is a PRG output, $\mathcal{A}$ is run with respect to G_0; when y is truly random, $\mathcal{A}$ is run with respect to G_1. Furthermore, $\mathcal{B}$ outputs a bit that matches the output of G_0 or G_1 when run with $\mathcal{A}$. Hence

$$\Pr[G_0^{\mathcal{A}}] - \Pr[G_1^{\mathcal{A}}] \leq \mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_2**:** This game replaces PK_1 with a truly random bit string of length $\{0,1\}^{2\lambda}$.

Using a virtually identical reduction to that in G_1 for PK_1 we get that for any $\mathcal{A}$ distinguishing G_1 from G_2 there is an adversary $\mathcal{B}$ such that

$$\Pr[G_1^{\mathcal{A}}] - \Pr[G_2^{\mathcal{A}}] \leq \mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_3**:** This game generates the challenge ciphertexts by obfuscating $\mathbf{C}[PK_0, M_0]$ (irrespective of the bit b).

We justify this transition based on the security of **Obf** as follows. Suppose an adversary $\mathcal{A}$ distinguishes game G_2 from G_3. Consider adversary $\mathcal{B}$ against **Obf** as follows. Adversary $\mathcal{B}$ generates two random $y_0, y_1 \twoheadleftarrow \{0,1\}^{2\lambda}$, sets $PK_i := y_i$, for $i = 0, 1$, and runs $\mathcal{A}(PK_0, PK_1)$ (as in G_2). $\mathcal{B}$ picks a random bit b and whenever $\mathcal{A}$ queries its LR on two messages (M_0, M_1), it queries its own (obfuscation) LR oracle on $(\mathbf{C}[PK_b, M_b], \mathbf{C}[PK_0, M_0])$ and receives an obfuscated circuit CT, which it sends to $\mathcal{A}$. When $\mathcal{A}$ terminates outputting a bit b', algorithm $\mathcal{B}_2$ returns $(b = b')$. It is clear that according to the challenge bit in the obfuscation game algorithm $\mathcal{B}$ runs $\mathcal{A}$ according to either game G_2 or G_3. Furthermore, $\mathcal{B}$ outputs a bit that matches the output of the G_2 or G_3 when run with $\mathcal{A}$. Hence

$$\Pr[G_2^{\mathcal{A}}] - \Pr[G_3^{\mathcal{A}}] \leq \mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

It remains to show that the circuits $\mathbf{C}[PK_b, M_b]$ and $\mathbf{C}[PK_0, M_0]$ that $\mathcal{B}$ queries to its obfuscation LR oracle are functionally equivalent and of the same size. Since PK_0 and PK_1 are truly random bit strings of length 2λ, for each value of SK and b we have that $\Pr[\mathbf{PRG}(SK) = PK_b] = 1/2^{2\lambda}$. Hence by the union bound

$$\Pr[\exists\, SK \in \{0,1\}^{\lambda}, b \in \{0,1\} : \mathbf{PRG}(SK) = PK_b] \leq 2^{\lambda+1}/2^{2\lambda} = 2/2^{\lambda} \ .$$

This means with probability at least $1 - 2/2^{\lambda}$ over the choice of PK_b the circuits $\mathbf{C}[PK_b, M_b]$ both implement a circuit always returning $\perp$. The circuits are of equal size as long as the two messages are of equal size.

Finally note that G_3 is independent of the bit b as $\mathbf{C}[PK_0, M_0]$ is always obfuscated. Hence for any (even unbounded) adversary $\mathcal{A}$ we have that $\Pr[G_3^{\mathcal{A}}] = 0$. The theorem follows from the above inequalities. □

Circ. $\mathbf{C}[X, Z, \mathrm{M}](x)$	Algo. $\mathbf{Obf}(\mathbf{C}[X, Z, \mathrm{M}])$	Circ. $\overline{\mathbf{C}}[\{h_{ij}\}_{i,j=1}^{3,2}](x)$
if $(g^x, Y^x) = (X, Z)$	$r_1, r_2, r_3, s_1, s_2, s_3 \twoheadleftarrow \mathbb{Z}_q$	if $h_{11}^x = h_{12} \wedge h_{21}^x = h_{22}$
return M	$h_{11} \leftarrow g^{r_1} Y^{s_1}$; $h_{12} \leftarrow X^{r_1} Z^{s_1}$	return $h_{32} \cdot h_{31}^{-x}$
return 1	$h_{21} \leftarrow g^{r_2} Y^{s_2}$; $h_{22} \leftarrow X^{r_2} Z^{s_2}$	return 1
	$h_{31} \leftarrow g^{r_3} Y^{s_3}$; $h_{32} \leftarrow \mathrm{M} \cdot X^{r_3} Z^{s_3}$	
	return $\overline{\mathbf{C}}[h_{11}, h_{12}, h_{21}, h_{22}, h_{31}, h_{32}]$	

Fig. 5. Left: Circuits that are obfuscated in encryption. **Middle**: An obfuscator for this class. **Right**: The obfuscated circuits.

3.3 A DDH-based instantiation

In this section, we show that the above construction can be instantiated under the standard DDH assumption. That is, we provide a DDH-based PRG and an indistinguishability obfuscator for the class of circuits used in the above construction. The resulting PKE scheme is defined over the message space $\mathbb{G} \setminus \{1\}$ where we identify the element 1 with the error symbol $\perp$.

The DDH assumption. Let $\mathcal{G}(1^\lambda)$ be a group generation algorithm that outputs the description of a cyclic group $\mathbb{G}$ of order q generated by g. Let $\mathbb{G}^* := \mathbb{G} \setminus \{1\}$. The decisional Diffie–Hellman (DDH) assumption for $\mathcal{G}$ requires that for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{DDH}^{\mathcal{A}}_{\mathcal{G}}(\lambda)\right] - 1 \in \mathrm{Negl}\ .$$

Game $\mathrm{DDH}^{\mathcal{A}}_{\mathcal{G}}(\lambda)$:
$(\mathbb{G}, q, g) \twoheadleftarrow \mathcal{G}(1^\lambda)$
$b \twoheadleftarrow \{0, 1\}$
$Y, Z \twoheadleftarrow \mathbb{G}^*$; $x \twoheadleftarrow \mathbb{Z}_q^*$; $X \leftarrow g^x$
$T_0 \leftarrow Y^x$; $T_1 \leftarrow Z$
$b' \twoheadleftarrow \mathcal{A}(\mathbb{G}, q, g, X, Y, T_b)$
return $(b = b')$

We now describe our construction, which is basically a *lossy* version of the El-Gamal PKE scheme [ElG84], meaning that one can define lossy public keys, which are indistinguishable from regular ones, but for which ciphertexts statistically hide all information about their underlying plaintexts.

A DDH-based PRG. Let $\mathbb{G}$ denote a cyclic group of order q generated by g. The public parameters are $(\mathbb{G}, q, g, Y)$ with Y a random element in $\mathbb{G}^*$. Algorithm $\mathbf{PRG}\colon \mathbb{Z}_q^* \longrightarrow (\mathbb{G}^*)^2$ is defined as $\mathbf{PRG}\colon x \mapsto (g^x, Y^x)$. Under the DDH assumption, the outputs of $\mathbf{PRG}$ are computationally indistinguishable from uniformly random elements in $(\mathbb{G}^*)^2$. Plugging this into our PKE construction in Section 3, secret keys take the form $x \in \mathbb{Z}_q^*$ and public keys $\mathrm{PK} = (g^x, Y^x)$.

The circuit class. We next define an indistinguishability obfuscator for the class of circuits $\mathbf{C}[\mathrm{PK}, \mathrm{M}]$ associated with the PRG above. This class consists of circuits defined in Figure 5 (left) for $\mathrm{PK} = (X, Z) \in (\mathbb{G}^*)^2$ and $\mathrm{M} \in \mathbb{G}^*$, where we identify the identity group element 1 with the error symbol $\perp$.

The obfuscator. Given a description (X, Z, M) of the circuit $\mathbf{C}[X, Z, \mathrm{M}]$, we define our obfuscator as shown in Figure 5 (middle), which takes randomness $(r_1, r_2, r_3, s_1, s_2, s_3) \twoheadleftarrow \mathbb{Z}_q^6$ and returns a circuit $\overline{\mathbf{C}}[h_{11}, h_{12}, h_{21}, h_{22}, h_{31}, h_{32}]$ as described on the right of Figure 5. Intuitively, (h_{31}, h_{32}) is an encryption of M if

(X, Z) is in the range of **PRG**, whereas it is a random pair otherwise. Similarly, (h_{11}, h_{12}) and (h_{21}, h_{22}) are encryptions of 1, which let $\overline{\mathbf{C}}$ check whether (X, Z) is in the range of **PRG**. If this is the case then the obfuscated circuit is functionally equivalent to $\mathbf{C}[X, Z, \mathrm{M}]$. However, if (X, Z) is not in the range of **PRG** then $\mathbf{C}[X, Z, \mathrm{M}]$ always returns 1, whereas with negligible probability over the coins of **Obf**, for $\overline{\mathbf{C}}$ there is one input on which it returns a different value. (Since $h_{11}, h_{12}, h_{21}, h_{22}$ are uniformly random, the probability that there exists x with $h_{11}^x = h_{12}$ and $h_{21}^x = h_{22}$ is negligible.)

Lemma 1. *The obfuscator described in Figure 5 is an indistinguishability obfuscator with statistical correctness and statistical indistinguishability.*

Consider an adversary outputting two functionally equivalent circuits $\mathbf{C}[X, Z, \mathrm{M}]$ and $\mathbf{C}[X', Z', \mathrm{M}']$. Both of them always return 1 iff (X, Z) and (X', Z') are not in the range of **PRG**, in which case the obfuscator returns a random tuple $(h_{11}, h_{12}, h_{21}, h_{22}, h_{31}, h_{32})$. If $X = g^x$ and $Z = Y^x$ then $\mathbf{C}[X, Z, \mathrm{M}]$ returns M on input x and 1 otherwise. For $\mathbf{C}[X', Z', \mathrm{M}']$ to be functionally equivalent to $\mathbf{C}[X, Z, \mathrm{M}]$, we must thus have $X' = X$, $Z' = Z$ and $\mathrm{M}' = \mathrm{M}$. This shows the iO security of the obfuscator. A formal proof is given in Appendix A.

4 Identity-Based Encryption

We now present a simple construction of identity-based encryption, again based on an indistinguishability obfuscator and a pseudorandom generator. In addition, we will use a pseudorandom function, which we assume to be puncturable. We prove that our scheme is both IND-CPA secure and anonymous in the selective-ID model. Anonymity means that ciphertexts do not leak the identity of the intended recipients; a property that can be leveraged to construct public-key encryption with keyword search [ABC+05].

4.1 Syntax and security

Identity-based encryption. An identity-based encryption (IBE) scheme for message space $\mathsf{MSp} := \{\mathsf{MSp}_\lambda\}_{\lambda\in\mathbb{N}}$ and identity space $\mathsf{IDSp} := \{\mathsf{IDSp}_\lambda\}_{\lambda\in\mathbb{N}}$ is a tuple of PPT algorithms IBE := (**IBSetup**, **IBEnc**, **IBGen**, **IBDec**) as follows:

IBSetup(1^λ)**:** This is the master key generation outputs a key pair (MSK, MPK).
IBGen(MSK, ID)**:** This is the key-generation algorithm, which on input a master secret key MSK and an identity ID outputs a (user) key SK.
IBEnc(MPK, ID, M)**:** This is the encryption algorithm, which on input a master public key MPK, an identity ID, and a message M outputs a ciphertext CT.
IBDec(SK, CT)**:** This is the deterministic decryption algorithm that on input an SK and a ciphertext CT outputs a message M or the error symbol $\perp$.

Correctness requires that for any $\lambda \in \mathbb{N}$, any $(\mathrm{MSK}, \mathrm{MPK}) \twoheadleftarrow \mathbf{IBSetup}(1^\lambda)$, any $\mathrm{ID} \in \mathsf{IDSp}_\lambda$, any $\mathrm{SK} \twoheadleftarrow \mathbf{IBGen}(\mathrm{MSK}, \mathrm{ID})$, any $\mathrm{M} \in \mathsf{MSp}_\lambda$ and any $\mathrm{CT} \twoheadleftarrow \mathbf{IBEnc}(\mathrm{MPK}, \mathrm{ID}, \mathrm{M})$, we have that $\mathbf{IBDec}(\mathrm{SK}, \mathrm{CT}) = \mathrm{M}$.

GAME Sel-AI-CPA$^{\mathcal{A}}_{\mathrm{IBE}}(\lambda)$:	PROC. LR$(\mathrm{M}_0, \mathrm{M}_1)$:
$(\mathrm{ID}_0, \mathrm{ID}_1, st) \twoheadleftarrow \mathcal{A}(1^\lambda)$	$\mathrm{CT} \twoheadleftarrow \mathbf{IBEnc}(\mathrm{MPK}, \mathrm{ID}_b, \mathrm{M}_b)$
$b \twoheadleftarrow \{0,1\}$	$L_1 \leftarrow L_1 \cup \{(\mathrm{M}_0, \mathrm{M}_1)\}$
$(\mathrm{MSK}, \mathrm{MPK}) \twoheadleftarrow \mathbf{IBSetup}(1^\lambda)$	return CT
$b' \twoheadleftarrow \mathcal{A}^{\mathrm{LR},\mathrm{EXT}}(\mathrm{MPK}, st)$	
for $(\mathrm{M}_0, \mathrm{M}_1) \in L_1, \mathrm{ID} \in L_2$	PROC. EXT(ID):
if $\mathrm{ID} \in \{\mathrm{ID}_0, \mathrm{ID}_1\}$ then $b' \leftarrow 0$	$\mathrm{SK} \twoheadleftarrow \mathbf{IBGen}(\mathrm{MSK}, \mathrm{ID})$
if $\lvert\mathrm{M}_0\rvert \neq \lvert\mathrm{M}_1\rvert$ then $b' \leftarrow 0$	$L_2 \leftarrow L_2 \cup \{\mathrm{ID}\}$
return $(b = b')$	return SK

Fig. 6. Selective AI-CPA security of an IBE scheme.

The selective anonymous and indistinguishable security of IBE under chosen-plaintext attacks [ABC+05] (Sel-AI-CPA) requires that for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\text{sel-ai-cpa}}_{\mathrm{IBE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\text{Sel-AI-CPA}^{\mathcal{A}}_{\mathrm{IBE}}(\lambda)\right] - 1 \in \text{NEGL} \ ,$$

where game Sel-AI-CPA$^{\mathcal{A}}_{\mathrm{IBE}}(\lambda)$ is shown in Figure 6. This notion implies IND-CPA security by setting $\mathrm{ID}_0 = \mathrm{ID}_1$.

4.2 Construction

Our IBE construction generalizes the PKE scheme as follows. The master secret key MSK is a random PRF key. The secret key for user ID is simply the PRF evaluated at the identity, that is $\mathbf{PRF}(\mathrm{MSK}, \mathrm{ID})$. The master public key MPK is an obfuscation of a circuit $\mathbf{C}^{\mathrm{gen}}$ that computes a public key $\mathrm{PK}_{\mathrm{ID}}$ for ID by computing $\mathbf{PRG}(\mathbf{PRF}(\mathrm{MSK}, \mathrm{ID}))$, as in our PKE scheme. Hence we use a PRF to "compress" exponentially many public keys into the master public key MPK (by generating their secret key using the PRF). A message is encrypted by first deriving $\mathrm{PK}_{\mathrm{ID}}$ for the identity using MPK and then obfuscating the same circuit as the PKE scheme. Decryption is identical to the PKE. The details of the scheme are shown in Figure 7.

In our security analysis we will rely on the puncturability of the PRF to simulate key-extraction queries. That is, we use a PRF key punctured at $\{\mathrm{ID}_0, \mathrm{ID}_1\}$ to derive keys for all other users. We will use the indistinguishability security of the obfuscator in conjunction with this to form the master public key. Hence we need to pad the circuit $\mathbf{C}^{\mathrm{gen}}$ so that its size matches that of the modified circuit used in the proof. We define $S^{\mathrm{gen}}(\lambda)$ as (an upper bound on) the maximum size of circuits defined by

$$\widetilde{\mathbf{C}}^{\mathrm{gen}}[\mathrm{K}, r, x_0, x_1](x) := \begin{cases} \mathbf{PRG}(\mathbf{PPRF}(\mathbf{Cons}(\mathrm{K}, \mathbf{C}[\{x_0, x_1\}]; r), x)) & \\ \qquad\qquad\qquad\qquad\qquad\qquad & \text{if } x \notin \{x_0, x_1\} \ ; \\ \mathbf{PRG}(\mathbf{PPRF}(\mathrm{K}, x_0)) & \text{if } x = x_0 \ ; \\ \mathbf{PRG}(\mathbf{PPRF}(\mathrm{K}, x_1)) & \text{if } x = x_1 \ . \end{cases}$$

Here the maximum is taken over $(\mathrm{K}, r, x_0, x_1) \in \{0,1\}^\lambda \times \{0,1\}^{t_{\mathbf{Cons}}(\lambda)} \times \{0,1\}^{m(\lambda)} \times \{0,1\}^{m(\lambda)}$, where $t_{\mathbf{Cons}}$ is the polynomial bounding the runtime of **Cons**. We recall that **Cons** denote the puncturing algorithm of the puncturable PRF.

ALGO. $\mathbf{IBSetup}(1^\lambda)$	ALGO. $\mathbf{IBEnc}(\mathrm{MPK}, \mathrm{ID}, \mathrm{M})$
$\mathrm{MSK} \twoheadleftarrow \{0,1\}^\lambda$	$\mathrm{PK} \leftarrow \mathrm{MPK}(\mathrm{ID})$
pad $\mathbf{C}^{\mathrm{gen}}$ so that $\lvert\mathbf{C}^{\mathrm{gen}}\rvert = S^{\mathrm{gen}}(\lambda)$	$\mathrm{CT} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}^{\mathrm{enc}}[\mathrm{PK}, \mathrm{M}])$
$\mathrm{MPK} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}^{\mathrm{gen}}[\mathrm{MSK}])$	return CT
return $(\mathrm{MSK}, \mathrm{MPK})$	
CIRC. $\mathbf{C}^{\mathrm{gen}}[\mathrm{MSK}](\mathrm{ID})$	CIRC. $\mathbf{C}^{\mathrm{enc}}[\mathrm{PK}, \mathrm{M}](\mathrm{SK})$
$\mathrm{PK} \leftarrow \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}, \mathrm{ID}))$	if $\mathbf{PRG}(\mathrm{SK}) = \mathrm{PK}$ return M
return PK	return $\perp$
ALGO. $\mathbf{IBGen}(\mathrm{ID})$	ALGO. $\mathbf{IBDec}(\mathrm{SK}, \mathrm{CT})$
$\mathrm{SK} \leftarrow \mathbf{PPRF}(\mathrm{MSK}, \mathrm{ID})$	$\mathrm{M} \leftarrow \mathrm{CT}(\mathrm{SK})$
return SK	return M

Fig. 7. Identity-based encryption from a PRG, a puncturable PRF, and an iO.

Theorem 2. *Scheme* IBE *in Figure 7 is* Sel-AI-CPA *secure as long as its underlying pseudorandom generator* **PRG**, *the puncturable pseudorandom function* PPRF, *and the obfuscator* **Obf** *are secure. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against* IBE *there are* PPT *adversaries* $\mathcal{B}_1$, $\mathcal{B}_2$, *and* $\mathcal{B}_3$ *against* **PRG**, PPRF, *and* **Obf** *respectively such that*

$$\mathbf{Adv}^{\text{sel-ai-cpa}}_{\text{IBE},\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{ind}}_{\text{PPRF},\mathcal{B}_2}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}_3}(\lambda) \ .$$

Algorithm $\mathcal{B}_1$ *places at most the same number of queries to its (obfuscation)* LR *oracle that* $\mathcal{A}$ *does to its (encryption)* LR *oracle.*

Proof. We give a detailed outline here and refer the reader to Appendix B for the missing details.

We prove the theorem via a sequence of games as follows.

G_0**:** This is the Sel-AI-CPA game with respect to scheme IBE in Figure 7:

$$2 \cdot \Pr\left[\mathrm{G}_0^{\mathcal{A}}\right] - 1 = \mathbf{Adv}^{\text{sel-ai-cpa}}_{\text{IBE},\mathcal{A}}(\lambda) \ .$$

G_1**:** This game modifies the generation of the master public key as follows. After $\mathcal{A}$ outputs $(\mathrm{ID}_0, \mathrm{ID}_1)$ the game generates a key MSK, computes a key MSK^* punctured at the challenge identity set $\{\mathrm{ID}_0, \mathrm{ID}_1\}$ and defines $y_i := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}, \mathrm{ID}_i))$ for $i = 0, 1$. The master public key is generated by obfuscating the circuit

$$\widetilde{\mathbf{C}}^{\mathrm{gen}}[\mathrm{MSK}^*, y_0, y_1](\mathrm{ID}) := \begin{cases} \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}^*, \mathrm{ID})) & \text{if } \mathrm{ID} \notin \{\mathrm{ID}_0, \mathrm{ID}_1\} \ ; \\ y_0 & \text{if } \mathrm{ID} = \mathrm{ID}_0 \ ; \\ y_1 & \text{if } \mathrm{ID} = \mathrm{ID}_1 \ . \end{cases}$$

and padding it so its size is $S^{\mathrm{gen}}(\lambda)$. Note that this circuit is functionally equivalent to and has the same size as the original circuit $\mathbf{C}^{\mathrm{gen}}[\mathrm{MSK}]$.

As in the proof of Theorem 1, indistinguishability of G_0 and G_1 reduces to the security of the obfuscator. For any distinguisher $\mathcal{A}$ there is a $\mathcal{B}$ such that

$$\Pr\left[G_0^{\mathcal{A}}\right] - \Pr\left[G_1^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

G_2**:** This game no longer sets $y_i := \mathbf{PRG}(\mathbf{PPRF}(\text{MSK}, \text{ID}_i))$, but instead computes it by first sampling two $S_i \twoheadleftarrow \{0,1\}^\lambda$ and then setting $y_i := \mathbf{PRG}(S_i)$.

We can justify this step based on the security of puncturable PRF. Here we answer the extraction queries using the punctured key MSK^*. By the rules of the Sel-AI-CPA game, key extraction will never be queried on ID_0 or ID_1, and hence MSK^* can be used to perfectly simulate the key-extraction oracle. This raises the question whether or not G_2 should use MSK^* and S_i in key-extraction queries as well. This *can* be done at this stage, and as a result we would not need to rely on the legitimacy rules of the IBE game. However, in the next two games we will use the security of the PRG to replace y_i with truly random values. To this end, we must ensure that the S_i's are not used in MPK and only their PRG values are. Hence we rely on IBE legitimacy rules to avoid using S_i later on. Note also that IBE legitimacy makes it inconsequential whether MSK^* or MSK is used. Hence,

$$\Pr\left[G_1^{\mathcal{A}}\right] - \Pr\left[G_2^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\text{ind}}_{\text{PPRF},\mathcal{B}}(\lambda) \ .$$

G_3**:** This game modifies y_0 further as follows. Instead of setting it to $\mathbf{PRG}(S_0)$ it directly samples $y_0 \twoheadleftarrow \{0,1\}^{2\lambda}$.

The secret key for the challenge identity ID_0 at this stage is replaced by a random value that is independent of the (punctured) master key. We can invoke the security of PRG, treating the secret key for ID_0 as the PRG seed, and replace its public key with a truly random value. That is, for any $\mathcal{A}$ that distinguishes G_2 from G_3 we show there is an adversary $\mathcal{B}$ against the security of PRG such that

$$\Pr\left[G_2^{\mathcal{A}}\right] - \Pr\left[G_3^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_4**:** This game modifies y_1 analogously to G_3. That is, instead of setting it to $\mathbf{PRG}(S_1)$ it directly samples $y_1 \twoheadleftarrow \{0,1\}^{2\lambda}$.

As in G_3 we have that for any $\mathcal{A}$ that distinguishes G_3 from G_4, there is a $\mathcal{B}$ such that

$$\Pr\left[G_3^{\mathcal{A}}\right] - \Pr\left[G_4^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_5**:** This game generates the challenge ciphertexts by always encrypting M_0 for identity ID_0 (independently of b).

As in the proof of Theorem 1 we can argue that with overwhelming probability the replaced public keys for both challenge identities lie outside the range of the PRG. This means they are both functionally equivalent to the $\perp$ circuit. This

allows us to perform a final iO hop and conclude that for any $\mathcal{A}$ that distinguishes G_4 from G_5 there is a $\mathcal{B}$ such that

$$\Pr\left[G_4^{\mathcal{A}}\right] - \Pr\left[G_5^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

Finally note that G_5 is independent of the bit b and hence for any (even unbounded) adversary $\mathcal{A}$ we have that $\Pr[G_4^{\mathcal{A}}] = 0$. The theorem follows from this and the above inequalities. □

ID SPACE. The ID space IDSp_λ of the construction matches the input space $\{0,1\}^{m(\lambda)}$ of **PPRF**. An IBE with $\mathsf{IDSp}_\lambda = \{0,1\}^*$ can be obtained in the standard way by hashing the identities using a collision-resistant hash function. An alternative route would be to use a variable-input-length puncturable **PRF**. A statistically correct puncturable PRF is needed when used in conjunction with an indistinguishability obfuscator. However, such PRFs with unbounded domain cannot have compact punctured keys (as a punctured keyed information-theoretically determines the punctured point). This means that there is no a priori bound on the size of the circuit $\widetilde{\mathbf{C}}^{\text{gen}}$ used in the security proof in G_1. We note, however, that the GGM PRF can be turned into a PRF with unbounded input length [Gol01], whose keys can be punctured in the same way as those of the original construction.

ADAPTIVE SECURITY. We have established the *selective* security of the IBE based on the selective security of the puncturable PRF. This raises the question if the theorem naturally extends to the adaptive-ID model under the *adaptive* security of the puncturable PRF. This turns out not to be the case: during the proof we need to puncture the MPK, which is given to the adversary at the onset. It is unclear at which point the underlying MSK should be punctured, even with a reduction down to the adaptive security of the PPRF. On the other hand, adaptive security can be obtained with a sub-exponential loss (on all underlying primitives) through complexity leveraging, or with only a polynomial loss in the random-oracle model. The recent work of Zhandry [Zha16] on extremely lossy functions (ELFs) proposes a generic selective-to-adaptive transformation for IBEs in the standard model that also results in a polynomial loss in security from a *single* exponential assumption. This transformation can be also applied to our IBE and we obtain an adaptive IBE with only a polynomial loss in security.

5 Predicate Encryption

Predicate encryption [BW07, KSW08] is a powerful type of public-key encryption that, for example, supports searching on encrypted data. It includes other forms of PKEs, such as Attribute-Based Encryption [GPSW06, LOS+10] and IBEs. (See [SBC+07, GVW15] for further motivating applications.) We show our approach is flexible enough to also lend itself to constructions of predicate encryption schemes. This construction, which generalizes our IBE scheme, relies on a puncturable PRF, a PRG and an indistinguishability obfuscator.

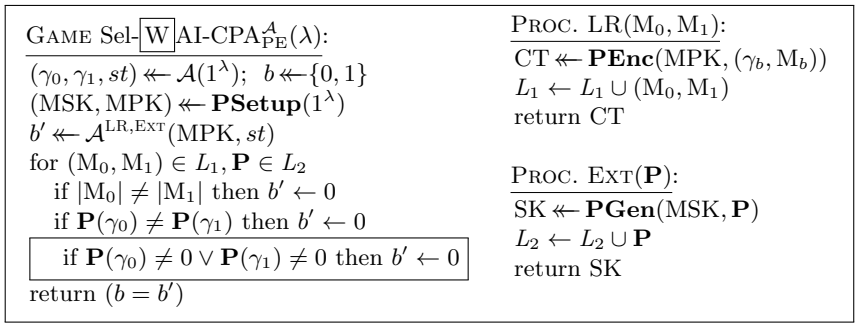

Fig. 8. The Sel-[W]AI-CPA security of a predicate encryption scheme. The Sel-WAI-CPA game includes the boxed statement.

5.1 Definition

PREDICATE ENCRYPTION. A predicate encryption (PE) scheme for message space $\mathsf{MSp} := \{\mathsf{MSp}_\lambda\}_{\lambda\in\mathbb{N}}$ and predicate space $\mathsf{PSp} := \{\mathsf{PSp}_\lambda\}_{\lambda\in\mathbb{N}}$ is a tuple of PPT algorithms $\mathrm{PE} := (\mathbf{PSetup}, \mathbf{PGen}, \mathbf{PEnc}, \mathbf{PDec})$ as follows:

PSetup(1^λ)**:** The master key-generation algorithm outputs a pair $(\mathrm{MSK}, \mathrm{MPK})$.
PGen$(\mathrm{MSK}, \mathbf{P})$**:** The key-generation algorithm, on input a master secret key MSK and a boolean circuit (predicate) $\mathbf{P}$, outputs a key SK.
PEnc$(\mathrm{PK}, \gamma, \mathrm{M})$**:** The encryption algorithm, on input a master public key MPK, an attribute γ and a message M, outputs a ciphertext CT.
PDec$(\mathrm{SK}, \mathrm{CT})$**:** The deterministic decryption algorithm, on input an SK and a ciphertext CT, outputs a message M or the error symbol $\perp$.

The correctness of a PE scheme requires that for any $\lambda \in \mathbb{N}$, any $(\mathrm{MSK}, \mathrm{MPK}) \twoheadleftarrow \mathbf{PGen}(1^\lambda)$, any $\mathbf{P} \in \mathsf{PSp}_\lambda$ and any $\mathrm{SK} \twoheadleftarrow \mathbf{PGen}(\mathrm{MSK}, \mathbf{P})$, any $\mathrm{M} \in \mathsf{MSp}_\lambda$, any $\gamma \in \mathsf{ASp}_\lambda$, and any $\mathrm{CT} \twoheadleftarrow \mathbf{PEnc}(\mathrm{MPK}, \gamma, \mathrm{M})$, we have that if $\mathbf{P}(\gamma) = 1$ then $\mathbf{PDec}(\mathrm{SK}, \mathrm{CT}) = \mathrm{M}$.

In an attribute-hiding PE scheme, on top of plaintexts, the ciphertexts should not reveal the attributes used in encryption. Formally, the selective attribute-hiding (Sel-AI-CPA) security of an PE scheme requires that for any PPT $\mathcal{A}$,

$$\mathbf{Adv}^{\text{sel-ai-cpa}}_{\mathrm{PE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\text{Sel-AI-CPA}^{\mathcal{A}}_{\mathrm{PE}}(\lambda)\right] - 1 \in \mathrm{NEGL}\ ,$$

where game Sel-AI-CPA$^{\mathcal{A}}_{\mathrm{PE}}(\lambda)$ is shown in Figure 8. We define the selective and *weak* attribute-hiding security similarly through game Sel-WAI-CPA that is also shown in Figure 8. In this game the adversary is restricted to only query keys for predicates that evaluate to 0 on the challenge attributes.

5.2 Construction

A natural way to build a PE scheme would be to modify the IBE construction so that instead of a puncturable PRF it uses a *constrained* PRF. The idea is

that a constrained key for a circuit $\mathbf{C}$ can evaluate a PRF value on γ correctly if $\mathbf{C}(\gamma) = 1$. This approach does not work directly. Recall that we used the puncturability of the PRF to simulate the key-extraction queries in the IBE game. Hence, an analogous form of puncturing would be also needed to derive constrained keys. Luckily, a natural construction of a constrained PRF from a puncturable PRF due Boneh and Waters [BW13] already enjoys this property, which we exploit in our construction in Figure 9.

We now show that our predicate encryption scheme is *weakly* and selectively attribute hiding. The reason that we can only achieve a weak level of attribute hiding is that keys for constrained keys for circuits that evaluate to 0 at the punctured pointed can be obtained under a punctured key. We prove the following theorem in Appendix C. We note that the attribute space of our PE scheme matches the input space of the puncturable PRF, and hence it supports a large universe of attributes.

Theorem 3. *Scheme* PE *in Figure 9 is* Sel-WAI-CPA *secure if its underlying pseudorandom generator* $\mathbf{PRG}$*, two-point puncturable pseudorandom function* PPRF*, and obfuscator* $\mathbf{Obf}$ *are secure. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against* PE *there are* PPT *adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$*, and* $\mathcal{B}_3$ *against* $\mathbf{Obf}$*,* PPRF*, and* $\mathbf{PRG}$ *respectively such that*

$$\mathbf{Adv}^{\text{sel-wai-cpa}}_{\text{PE},\mathcal{A}}(\lambda) \leq (2 + Q(\lambda)) \cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{ind}}_{\text{PPRF},\mathcal{B}_2}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}_3}(\lambda) \ ,$$

where $Q(\lambda)$ *denotes the number of queries of* $\mathcal{A}$ *to its key extraction oracle.*

ALGO. $\mathbf{PSetup}(1^\lambda)$
$\text{MSK} \twoheadleftarrow \{0,1\}^\lambda$
$\text{MPK} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}^{\text{gen}}[\text{MSK}])$
return (MSK, MPK)

CIRC. $\mathbf{C}^{\text{gen}}[\text{MSK}](\gamma)$
$\text{PK} \leftarrow \mathbf{PRG}(\mathbf{PPRF}(\text{MSK}, \gamma))$
return PK

ALGO. $\mathbf{PGen}(\text{MSK}, \mathbf{P})$
$\text{SK} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}^{\text{sk}}[\text{MSK}, \mathbf{P}])$
return SK

CIRC. $\mathbf{C}^{\text{sk}}[\text{MSK}, \mathbf{P}](\gamma)$
if $\mathbf{P}(\gamma) = 1$ return $\mathbf{PPRF}(\text{MSK}, \gamma)$
return $\perp$

ALGO. $\mathbf{PEnc}(\text{MPK}, \gamma, \text{M})$
$\text{PK} \leftarrow \text{MPK}(\gamma)$
$\text{CT} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}^{\text{enc}}[\text{PK}, \gamma, \text{M}])$
return CT

CIRC. $\mathbf{C}^{\text{enc}}[\text{PK}, \gamma, \text{M}](\text{SK})$
if $\mathbf{PRG}(\mathbf{PPRF}(\text{SK}, \gamma)) = \text{PK}$
  return M
return $\perp$

ALGO. $\mathbf{PDec}(\text{SK}, \text{CT})$
return $\text{CT}(\text{SK})$

Fig. 9. Predicate encryption from a PRG, a constrained PRF, and an iO.

6 Proof of Theorem 3: Security of Predicate Encryption

Theorem 4 (Theorem 3, restated). *Scheme* PE *in Figure 9 is* Sel-WAI-CPA *secure if its underlying pseudorandom generator* **PRG***, two-point puncturable pseudorandom function* PPRF*, and obfuscator* **Obf** *are secure. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against the* Sel-WAI-CPA *security of* PE *there are* PPT *adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$*, and* $\mathcal{B}_3$ *against* **Obf**, PPRF*, and* **PRG** *respectively such that*

$$\mathbf{Adv}^{\text{sel-wai-cpa}}_{\text{PE},\mathcal{A}}(\lambda) \leq (2+Q(\lambda))\cdot\mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}_1}(\lambda)+\mathbf{Adv}^{\text{ind}}_{\text{PPRF},\mathcal{B}_2}(\lambda)+2\cdot\mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}_3}(\lambda)\ .$$

where $Q(\lambda)$ *denotes the number of queries of* $\mathcal{A}$ *to its key extraction oracle.*

Proof. We prove the theorem via a sequence of games as follows:

G_0**:** This is the weak Sel-WAI-CPA game (Figure 8) with respect to scheme PE in Figure 9:

$$2\cdot\Pr[\mathrm{G}_0^{\mathcal{A}}]-1=\mathbf{Adv}^{\text{sel-wai-cpa}}_{\text{PE},\mathcal{A}}(\lambda)\ .$$

G_1**:** This game modifies the generation of the master public key as follows. After the first phase of $\mathcal{A}$ when the adversary outputs (γ_0,γ_1,st) the game generates a key MSK, computes a punctured key MSK^* at the challenge attribute set $\{\gamma_0,\gamma_1\}$ and defines $y_i:=\mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK},\gamma_i))$ for $i=0,1$. The master public key is generated by obfuscating the circuit

$$\tilde{\mathbf{C}}^{\text{gen}}[\mathrm{MSK}^*,y_0,y_1](\gamma):=\begin{cases}\mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}^*,\gamma)) & \text{if } \gamma\notin\{\gamma_0,\gamma_1\}\\ y_0 & \text{if } \gamma=\gamma_0\\ y_1 & \text{if } \gamma=\gamma_1\ .\end{cases}$$

Note that this is functionally equivalent to the original circuit $\mathbf{C}^{\text{gen}}[\mathrm{MSK}]$.

We show that G_0 and G_1 are indistinguishable. Let $\mathcal{A}$ be an adversary that distinguishes game G_0 from game G_1. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against **Obf** as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs (γ_0,γ_1,st). It then samples a P-PRF key MSK and punctures it at $\{\gamma_0,\gamma_1\}$ to compute MSK^*. It also computes $y_i:=\mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK},\gamma_i))$ for $i=0,1$. Algorithm $\mathcal{B}$ uses these values to generate two circuits $\mathbf{C}^{\text{gen}}$ and $\tilde{\mathbf{C}}^{\text{gen}}$ as above and queries them to its (obfuscation) LR oracle to receive an obfuscated circuit which $\mathcal{B}$ defines to be the MPK. Algorithm $\mathcal{B}$ answers the encryption queries of $\mathcal{A}$ on $(\mathrm{M}_0,\mathrm{M}_1)$ by computing a ciphertext using MPK and M_b. Key-extraction queries on $\mathbf{P}$ are handled using the (un-punctured) key MSK. When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b=b')$. It is easily to see that according to the bit chosen by $\mathcal{B}$'s (obfuscation) game, algorithm $\mathcal{A}$ is run either with respect to game G_0 or G_1 and the output of $\mathcal{B}$ matches the output of these games when run with $\mathcal{A}$. Hence

$$\Pr[\mathrm{G}_0^{\mathcal{A}}]-\Pr[\mathrm{G}_1^{\mathcal{A}}]\leq\mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda)\ .$$

G_2**:** Instead of obfuscating the circuit $\mathbf{C}^{\mathrm{sk}}[\mathrm{MSK}, \mathbf{P}]$ at key extraction, this game uses the punctured key MSK^* at $\{\gamma_0, \gamma_1\}$ and predicate $\mathbf{P}$ to form the circuit below and return an obfuscation of it.

$$\tilde{\mathbf{C}}^{\mathrm{sk}}[\mathrm{MSK}^*, \mathbf{P}](\gamma) := \begin{cases} \mathbf{PPRF}(\mathrm{MSK}^*, \gamma) & \text{if } \mathbf{P}(\gamma) = 1 \\ \perp & \text{otherwise.} \end{cases}$$

Note that by the legitimacy rules of the *weak* AI-CPA game, $\mathbf{P}(\gamma_0) = \mathbf{P}(\gamma_1) = 0$ for all extracted $\mathbf{P}$. Hence the above circuit is functionally equivalent to $\mathbf{C}^{\mathrm{sk}}[\mathrm{MSK}, \mathbf{P}]$ for any extracted $\mathbf{P}$.[10]

Let $\mathcal{A}$ be an adversary that distinguishes G_1 from G_2. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the multi-challenge **Obf** as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs (γ_0, γ_1). It then samples a P-PRF key MSK and punctures it at $\{\gamma_0, \gamma_1\}$ to get MSK^*. It also computes $y_b := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}, \gamma_b))$ for $b = 0, 1$. Algorithm $\mathcal{B}$ uses these values to generate $\tilde{\mathbf{C}}^{\mathrm{gen}}[\mathrm{MSK}^*, y_0, y_1]$ and obfuscates it to obtain MPK (as in G_1). For each key-extraction query $\mathbf{P}$, algorithm $\mathcal{B}$ generates $\mathbf{C}^{\mathrm{sk}}[\mathrm{MSK}, \mathbf{P}]$ and $\tilde{\mathbf{C}}^{\mathrm{sk}}[\mathrm{MSK}^*, \mathbf{P}]$ as above and submits them to its (obfuscation) LR oracle to obtain an obfuscated circuit SK. It answers the query using SK. Algorithm $\mathcal{B}$ answers the encryption query of $\mathcal{A}$ on $(\mathrm{M}_0, \mathrm{M}_1)$ by choosing a bit b at random and computing a ciphertext using MPK and (γ_b, M_b). When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easily to see that according to the bit chosen by $\mathcal{B}$'s (obfuscation) game, algorithm $\mathcal{A}$ is run either with respect to game G_1 or G_2 and the output of $\mathcal{B}$ matches the output of these games when run with $\mathcal{A}$. Hence

$$\Pr[G_1^{\mathcal{A}}] - \Pr[G_2^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \leq Q(\lambda) \cdot \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}'}(\lambda) \ ,$$

where $\mathcal{B}'$ is a single-challenge adversary against the obfuscator.

G_3**:** This game no longer sets $y_b := \mathbf{PRG}(\mathbf{CPRF}(\mathrm{MSK}, \gamma_b))$ but rather computes these values by sampling $\mathrm{S}_b \twoheadleftarrow \{0,1\}^\lambda$ and setting $y_b := \mathbf{PRG}(\mathrm{S}_b)$.

Let $\mathcal{A}$ be an adversary that distinguishes G_2 from G_3. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of the doubly puncturable PRF as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs (γ_0, γ_1). It requests a punctured key MSK^* at $\{\gamma_0, \gamma_1\}$ and also queries its challenge oracle twice on γ_0 and γ_1 to receive two values S_0 and S_1. (Note that these queries are valid.) Algorithm $\mathcal{B}$ sets $y_b := \mathbf{PRG}(\mathrm{S}_b)$, forms the circuit $\tilde{\mathbf{C}}^{\mathrm{gen}}[\mathrm{MSK}^*, y_0, y_1]$ and obfuscates it to get MPK. It resumes $\mathcal{A}$ on MPK and answers its key-extraction queries on $\mathbf{P}$ by retuning an obfuscation of the circuit $\tilde{\mathbf{C}}^{\mathrm{sk}}[\mathrm{MSK}^*, \mathbf{P}]$. The encryption queries are answered by choosing a bit b at random and encrypting (γ_b, M_b) under MPK. When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easily to see that according to the bit chosen by $\mathcal{B}$'s game, algorithm $\mathcal{A}$ is run either with respect to game G_1 or G_2, and the output of $\mathcal{B}$ matches the output of these games when run with $\mathcal{A}$. Hence

$$\Pr[G_2^{\mathcal{A}}] - \Pr[G_3^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{PPRF},\mathcal{B}}(\lambda) \ .$$

[10] As in the IBE reduction we avoid hardwiring any PRF values that will be used as PRG seeds in the later games.

G_4**:** This game modifies the generation of y_0 further: instead of computing it as $\mathbf{PRG}(S_0)$, it directly samples $y_0 \twoheadleftarrow \{0,1\}^{2\lambda}$.

Let $\mathcal{A}$ be an adversary that distinguishes G_3 from G_4. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of PRG as follows. Algorithm $\mathcal{B}$ receives a value y_0 which is either $\mathbf{PRG}(S_0)$ or is truly random. Algorithm $\mathcal{B}$ picks a bit b and runs $\mathcal{A}$ until it outputs an identity (ID_0, ID_1). It generates a PRF key MSK, computes a punctured key MSK^* at $\{\gamma_0, \gamma_1\}$, and uses MSK^*, y_0 and $y_1 := \mathbf{PRG}(S_1)$ for a random S_1 to compute MPK. Algorithm $\mathcal{B}$ resumes $\mathcal{A}$ on MPK and answers its extraction queries by obfuscating $\tilde{\mathbf{C}}^{sk}[MSK^*, \mathbf{P}]$ as in G_3. For the LR query (M_0, M_1), $\mathcal{B}$ chooses a random bit b and encrypts (γ_b, M_b) using MPK. When $\mathcal{A}$ terminates with a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easy to see that if y_0 is $\mathbf{PRG}(S_0)$, algorithm $\mathcal{A}$ is run in game G_2 and when it is truly random it is run with respect to G_3. Hence,

$$\Pr[G_3^{\mathcal{A}}] - \Pr[G_4^{\mathcal{A}}] \leq \mathbf{Adv}^{ind}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_5**:** This game modifies the generation of y_1 analogously to that of y_0 in G_4.

Once again, using a reduction similar to that for the IBE scheme we deduce that for any PPT $\mathcal{A}$ there is a PPT $\mathcal{B}$ such that

$$\Pr[G_4^{\mathcal{A}}] - \Pr[G_5^{\mathcal{A}}] \leq \mathbf{Adv}^{ind}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_6**:** This game generates the challenge ciphertexts by always encrypting M_0 for attribute γ_0 (independently of b).

Let $\mathcal{A}$ be an adversary that distinguishes G_5 and G_6. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of **Obf** as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ answers until it outputs (γ_0, γ_1). It generates a PRF key MSK and computes a punctured key MSK^* at $\{\gamma_0, \gamma_1\}$. It then samples random $y_0, y_1 \twoheadleftarrow \{0,1\}^{2\lambda}$ and computes MPK as obfuscation of $\tilde{\mathbf{C}}^{gen}[MSK^*, y_0, y_1]$. It answers the key extraction queries on $\mathbf{P}$ by returning obfuscations of the circuits $\tilde{\mathbf{C}}^{sk}[MSK^*, y_0, y_1, \mathbf{P}]$. It answers the encryption queries (M_0, M_1) by calling its own (obfuscation) LR oracle on $(\mathbf{C}^{enc}[y_b, M_b], \mathbf{C}^{enc}[y_b, M_b])$ and receives CT. It resumes $\mathcal{A}$ on CT and answers its extraction queries as before. When $\mathcal{A}$ outputs a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easy to see that according to the challenge bit in the obfuscation game, $\mathcal{A}$ is run in game G_5 or G_6. Hence

$$\Pr[G_5] - \Pr[G_6] \leq \mathbf{Adv}^{ind}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

It remains to show that the circuits queried by $\mathcal{B}$ to its challenge oracle are functionally equivalent. The argument is as before: Since y_0 and y_1 are truly random bit strings of length 2λ, for each value of SK and b we have that $\Pr[\mathbf{PRG}(SK) = y_b] = 1/2^{2\lambda}$. Hence by the union bound

$$\Pr\left[\exists\, SK \in \{0,1\}^{\lambda}, b \in \{0,1\} : \mathbf{PRG}(SK) = y_b\right] \leq 2^{\lambda+1}/2^{2\lambda} = 2/2^{\lambda} \ .$$

This means with probability at least $1 - 2/2^\lambda$ over the choice of y_b the circuits $\mathbf{C}^{\text{gen}}[y_b, \mathrm{M}_b]$ both implement the all-zero circuit. Note also that the two circuits are of equal size as long as the two messages are of equal size.

Finally note that the MPK and all oracle responses in G_6 are independent of the bit b. Hence the for any (even unbounded) IND-CPA adversary $\Pr[\mathrm{G}_6^{\mathcal{A}}] = 0$. The theorem follows. □

Through standard generic constructions, our scheme gives rise to a weakly secure functional encryption scheme (where the adversary is confined to keys SK_f for f which satisfy $f(\mathrm{M}_0) = f(\mathrm{M}_1) = 0$). A fully attribute-hiding PE scheme, on the other hand, is sufficient to obtain a full-fledged functional encryption scheme. Although iO is sufficient to obtain functional encryption [GGH+13], it requires the double-encryption technique of Naor and Yung [NY90] and thus also non-interactive zero-knowledge proofs. We leave it as open problem to see if simpler and analogous constructions for functional encryption from iO also exist.

7 Unbounded HIBE

A hierarchal IBE (HIBE) [GS02] generalizes IBEs to a setting where a user with secret key for identity ID can derive secret keys for child identities with prefix ID. In this section we present a construction of an *unbounded* HIBE, where key delegation can be carried out for an exponential number of prefixes. Let us start with the syntax and security of HIBE schemes.

7.1 Syntax and security

Hierarchical IBE. In HIBE schemes, identities are of the form $\overline{\mathrm{ID}} = (\mathrm{ID}_1, \ldots, \mathrm{ID}_n) \in \mathsf{IDSp}_\lambda^n$ for some $n \in \mathbb{N}$. For $m \leq n$ we write $\overline{\mathrm{ID}}_{[m]}$ for $(\mathrm{ID}_1, \ldots, \mathrm{ID}_m)$. An (unbounded) HIBE scheme for message space $\mathsf{MSp} := \{\mathsf{MSp}_\lambda\}_{\lambda\in\mathbb{N}}$, identity space $\mathsf{IDSp} := \{\mathsf{IDSp}_\lambda\}_{\lambda\in\mathbb{N}}$ and depth $d(\cdot)$ is a tuple of PPT algorithms $\mathrm{HIBE} := (\mathbf{HSetup}, \mathbf{HEnc}, \mathbf{HDel}, \mathbf{HDec})$ as follows.

HSetup(1^λ)**:** The master key-generation algorithm outputs a pair $(\mathrm{SK}_0, \mathrm{MPK})$.

HEnc$(\mathrm{MPK}, \overline{\mathrm{ID}}, \mathrm{M})$**:** The encryption algorithm, on input a master public key MPK, a vector of identities $\overline{\mathrm{ID}}$ and a message M, outputs a ciphertext CT.

HDel$(\mathrm{SK}, \mathrm{ID})$**:** The delegation algorithm, on input a secret key SK and an identity ID, outputs a (user) key SK′.

HDec$(\mathrm{SK}, \mathrm{CT})$**:** The deterministic decryption algorithm, on input a secret key SK and a ciphertext CT, outputs a message M or the error symbol $\perp$.

We require correctness of the HIBE scheme: For any $\lambda \in \mathbb{N}$, $d := d(\lambda)$, any $(\mathrm{SK}_0, \mathrm{MPK}) \twoheadleftarrow \mathbf{HSetup}(1^\lambda)$, any $\mathrm{M} \in \mathsf{MSp}_\lambda$, any $\overline{\mathrm{ID}} = (\mathrm{ID}_1, \ldots, \mathrm{ID}_d) \in \mathsf{IDSp}_\lambda^d$, any $\mathrm{CT} \twoheadleftarrow \mathbf{HEnc}(\mathrm{MPK}, \overline{\mathrm{ID}}, \mathrm{M})$, any $\mathrm{SK}_i \twoheadleftarrow \mathbf{HDel}(\mathrm{SK}_{i-1}, \mathrm{ID}_i)$ for $i = 1, \ldots, d$ we have that $\mathbf{HDec}(\mathrm{SK}_d, \mathrm{CT}) = \mathrm{M}$. We call a HIBE *bounded* if correctness (and security; see below) hold for a polynomial $d(\lambda)$. We call the HIBE *unbounded* if this holds for $d(\lambda) = 2^\lambda$.

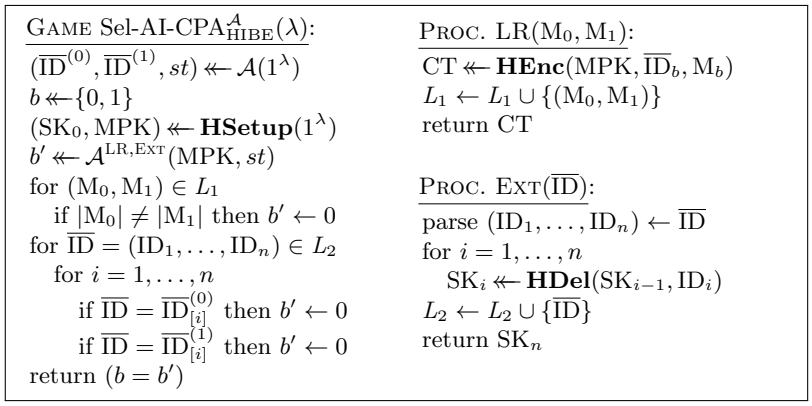

Fig. 10. The selective AI-CPA security of a (possibility unbounded) HIBE scheme.

The Sel-AI-CPA security of HIBE requires that for any legitimate PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\text{sel-ai-cpa}}_{\text{HIBE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\text{Sel-AI-CPA}^{\mathcal{A}}_{\text{HIBE}}(\lambda)\right] - 1 \in \text{NEGL} ,$$

where game Sel-AI-CPA$^{\mathcal{A}}_{\text{HIBE}}(\lambda)$ is shown in Figure 10. We note that the adversary might choose two identities of different depths, and hence an anonymous HIBE, according to our definition, also hides the receipt's level in the hierarchy. In a Sel-AI-CPA-secure bounded HIBE scheme we relax the model and require the adversary to always output a tuple of challenge vectors $(\overline{\text{ID}}_0, \overline{\text{ID}}_1)$ of depth exactly $d(\lambda)$.

7.2 Construction

Our IBE construction generalizes that for PKEs by using a PRF, instead of a single secret corresponding to the PRG seed, to derive many secret keys. Similarly, we can further generalize the IBE construction to a *hierarchal* IBE by considering a *tree* of (puncturable) PRFs (as in the GGM construction). The secret key for a vector of identities $\overline{\text{ID}} := (\text{ID}_1, \ldots, \text{ID}_n)$ corresponds to the iterative PRF applications via $\text{SK}_i \leftarrow \textbf{PRF}(\text{SK}_{i-1}, \text{ID}_i)$ where SK_0 is the master secret key. We can then use the puncturing property of the PRF to replace, via a sequence of hybrids, the PRF value corresponding to any identity $(\text{ID}_1, \ldots, \text{ID}_n)$ with random values one at a time. This then allows us to apply the PRG game hop (as for IBEs) at the leaf nodes to conclude the proof. We formalize this construction and prove its security in Appendix D. We refer the reader to the introduction why this construction does not yield an unbounded HIBE.

We now show how to construct a HIBE scheme with arbitrary ID space and an exponential number of levels. As before we use a PRG and a puncturable PRF. We however also rely on a public-coin differing-inputs obfuscator (pc-diO) for *circuits*, an indistinguishability obfuscator, a public-coin collision-resistant

$\underline{\text{GAME } \text{Ext}^{\mathcal{E}_{\mathcal{A}}}_{\text{SNARK},\mathcal{A}}(\lambda)}$:
$\text{crs} \twoheadleftarrow \{0,1\}^{p_S(\lambda)}$; $r \twoheadleftarrow \{0,1\}^{p_{\mathcal{A}}(\lambda)}$
$(\eta, \pi) \leftarrow \mathcal{A}(\text{crs}; r)$; $w \twoheadleftarrow \mathcal{E}_{\mathcal{A}}(1^\lambda, \text{crs}, r)$
return $(\mathbf{Ver}(\text{crs}, \eta, \pi) = 1 \wedge \mathbf{Rel}(1^\lambda, \eta, w) = 0)$

Fig. 11. The extraction game for a SNARK.

(pc-CR) hash function, and a SNARK proof system with a common random string as defined below.

RELATIONS. A relation $\mathbf{Rel}(1^\lambda, \eta, w)$ is a PPT Turing machine in the length of its inputs (rather than the security parameter) which on input the security parameter, a statement $\eta \in \{0,1\}^*$, and a witness $w \in \{0,1\}^*$ outputs 0 or 1. We say $\eta \in \mathbf{Rel}_\lambda$ if there exists $w \in \{0,1\}^*$ such that $\mathbf{Rel}(1^\lambda, \eta, w) = 1$.

SNARKS. A succinct non-interactive argument of knowledge (SNARK) is a non-interactive proof-of-knowledge system whose proofs lengths and verification time are independent of the witness size and only bounded by an a priori fixed polynomial in the statement length. Formally a SNARK for a relation **Rel** is a pair of PPT algorithms $(\mathbf{Prove}, \mathbf{Ver})$ such that $\mathbf{Prove}(\text{crs}, \eta, w)$ outputs a proof π and $\mathbf{Ver}(\text{crs}, \eta, \pi)$ outputs 0 or 1. We call the SNARK *complete* (or correct) if for any $\lambda \in \mathbb{N}$, any (η, w) with $\mathbf{Rel}(1^\lambda, \eta, w) = 1$, any $\text{crs} \twoheadleftarrow \{0,1\}^{p_S(\lambda)}$ and any $\pi \twoheadleftarrow \mathbf{Prove}(\text{crs}, \eta, w)$ we have that $\mathbf{Ver}(\text{crs}, \eta, \pi) = 1$. We call the SNARK *succinct* if there exist polynomials $p(\cdot)$, $l(\cdot)$ and $q(\cdot)$ such that for any $\lambda \in \mathbb{N}$, any $\text{crs} \twoheadleftarrow \{0,1\}^{p_S(\lambda)}$, any $(\eta, w) \in \{0,1\}^* \times \{0,1\}^*$ with $\mathbf{Rel}(1^\lambda, \eta, w) = 1$, and any $\pi \twoheadleftarrow \mathbf{Prove}(\text{crs}, \eta, w)$ we have that $|\pi| \leq l(\lambda + |\eta|)$, $\mathbf{Prove}(\text{crs}, \eta, w)$ runs in time $q(\lambda + |\eta|)$, and $\mathbf{Ver}(\text{crs}, \eta, \pi)$ runs in time $p(\lambda + |\eta|)$.

We call the SNARK (adaptively) *extractable* if for any PPT $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ such that $\mathbf{Adv}^{\text{ext}}_{\text{SNARK},\mathcal{A},\mathcal{E}_{\mathcal{A}}}(\lambda) := \Pr\left[\text{Ext}^{\mathcal{E}_{\mathcal{A}}}_{\text{SNARK},\mathcal{A}}(\lambda)\right] \in$ NEGL , where game $\text{Ext}^{\mathcal{A}}_{\text{SNARK}}(\lambda)$ is shown in Figure 11.

Given a public-coin hash function **H**, we rely on SNARKs for the relation $\mathbf{Rel}(1^\lambda, (k, h_1, h_2), (w_1, w_2))$ that checks

$$\left(|k| = \lambda \ \wedge \ \mathbf{H}(k, w_1) = h_1 \ \wedge \ \mathbf{H}(k, w_1 | w_2) = h_2 \ \wedge \ w_2 \neq \varepsilon\right).$$

In our construction, the obfuscation of a delegation circuit $\mathbf{C}^{\text{del}}$ with a hard-wired master key will be made public. This circuit takes as input a user secret key SK, two hash values h', h, and a SNARK proof π. The circuit checks that the proof π verifies (and hence proves knowledge of $\overline{\text{ID}}, \text{ID}$ such that $h = \mathbf{H}(k, \overline{\text{ID}})$ and $h' = \mathbf{H}(k, \overline{\text{ID}}|\text{ID})$). This is done explicitly by computing a *public key* for h using the master key and checking if the result matches the one derived from SK. (Comparison with the public key rather than the secret key is needed as in the security proof we have to forget the SK at some point). If all checks pass, a secret key for h' (i.e., the hash of child identity $\overline{\text{ID}}|\text{ID}$) is returned. The other algorithms operate as in the IBE scheme on hashed identities. The details of the construction are shown in Figure 12. As for our IBE construction in Figure 7, we

ALGO. $\mathbf{HSetup}(1^\lambda)$
$\mathrm{K} \twoheadleftarrow \{0,1\}^\lambda$; $k \twoheadleftarrow \{0,1\}^\lambda$
$\mathrm{crs} \twoheadleftarrow \{0,1\}^{r_S(\lambda)}$
$\overline{\mathbf{C}}^{\mathrm{gen}} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}^{\mathrm{gen}}[\mathrm{K}])$
$\overline{\mathbf{C}}^{\mathrm{del}} \twoheadleftarrow \mathbf{diO}(1^\lambda, \mathbf{C}^{\mathrm{del}}[\mathrm{K}])$
$\mathrm{MPK} \leftarrow (\mathrm{crs}, k, \overline{\mathbf{C}}^{\mathrm{gen}}, \overline{\mathbf{C}}^{\mathrm{del}})$
$\mathrm{SK}_0 \leftarrow \mathbf{PPRF}(\mathrm{K}, \mathbf{H}(k, \varepsilon))$
return $((\mathrm{SK}_0, \varepsilon), \mathrm{MPK})$

CIRC. $\mathbf{C}^{\mathrm{gen}}[\mathrm{K}](h)$
$\mathrm{PK} \leftarrow \mathbf{PRG}(\mathbf{PPRF}(\mathrm{K}, h))$
return PK

CIRC. $\mathbf{C}^{\mathrm{del}}[\mathrm{K}, k, \mathrm{crs}](\mathrm{SK}, h, h', \pi)$
if $\mathbf{Ver}(\mathrm{crs}, (k, h, h'), \pi) = 0$ return $\perp$
$\mathrm{PK} \leftarrow \mathbf{PRG}(\mathbf{PPRF}(\mathrm{K}, h))$
if $\mathbf{PRG}(\mathrm{SK}) \neq \mathrm{PK}$ return $\perp$
return $\mathbf{PPRF}(\mathrm{K}, \mathbf{H}(k, h'))$

ALGO. $\mathbf{HEnc}(\mathrm{MPK}, \overline{\mathrm{ID}}, \mathrm{M})$
$\mathrm{PK} \leftarrow \overline{\mathbf{C}}^{\mathrm{gen}}(\mathbf{H}(k, \overline{\mathrm{ID}}))$
$\mathrm{CT} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}^{\mathrm{enc}}[\mathrm{PK}, \mathrm{M}])$
return CT

CIRC. $\mathbf{C}^{\mathrm{enc}}[\mathrm{PK}, \mathrm{M}](\mathrm{SK})$
if $\mathbf{PRG}(\mathrm{SK}) = \mathrm{PK}$ return M
return $\perp$

ALGO. $\mathbf{HDel}((\mathrm{SK}, \overline{\mathrm{ID}}), \mathrm{ID}')$
$h \leftarrow \mathbf{H}(k, \overline{\mathrm{ID}})$; $h' \leftarrow \mathbf{H}(k, \overline{\mathrm{ID}}|\mathrm{ID}')$
$\pi \twoheadleftarrow \mathbf{Prove}(\mathrm{crs}, (k, h, h'), (\overline{\mathrm{ID}}, \mathrm{ID}'))$
$\mathrm{SK}' \twoheadleftarrow \overline{\mathbf{C}}^{\mathrm{del}}(\mathrm{SK}, h, h', \pi)$
return $(\mathrm{SK}', \overline{\mathrm{ID}} \,|\, \mathrm{ID}')$

ALGO. $\mathbf{HDec}((\mathrm{SK}, \overline{\mathrm{ID}}), \mathrm{CT})$
$\mathrm{M} \leftarrow \mathrm{CT}(\mathrm{SK})$
return M

Fig. 12. Unbounded HIBE from a PRG, a puncturable PRF, a pc-CR hash function and a pc-diO.

pad the circuits $\mathbf{C}^{\mathrm{gen}}$ and $\mathbf{C}^{\mathrm{del}}$ respectively to the maximum size of any circuit $\widetilde{\mathbf{C}}^{\mathrm{gen}}$ and $\widetilde{\mathbf{C}}^{\mathrm{del}}$ used in the proof (and argue that these sizes are polynomially bounded).

We now prove the security of the construction. We refer the reader to the introduction for a high-level overview of the proof.

Theorem 5. *Scheme* HIBE *in Figure 12 is* Sel-AI-CPA *secure if* **PRG** *and* **PPRF** *are secure,* **H** *is public-coin collision-resistant, the obfuscator* **iO** *is an indistinguishability obfuscation, and the obfuscator* **diO** *is public-coin differing-input secure. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against* HIBE *there exist* PPT *adversaries* $\mathcal{B}_1$ *against* **iO**, *diO-legitimate* $\mathcal{B}_2$ *against* **diO**, $\mathcal{B}_3$ *against* **H**, $\mathcal{B}_4$ *against* PPRF *and* $\mathcal{B}_5$ *against* **PRG** *such that*

$$\begin{aligned}\mathbf{Adv}^{\text{sel-ai-cpa}}_{\text{HIBE},\mathcal{A}}(\lambda) &\leq (4p(\lambda)+1)\cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{iO},\mathcal{B}_1}(\lambda) + 4p(\lambda)\cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{diO},\mathcal{B}_2}(\lambda) \\ &+ 4p(\lambda)\cdot \mathbf{Adv}^{\text{cr}}_{\mathbf{H},\mathcal{B}_3}(\lambda) + 2p(\lambda)\cdot \mathbf{Adv}^{\text{ind}}_{\text{PPRF},\mathcal{B}_4}(\lambda) + 2p(\lambda)\cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}_5}(\lambda)\ ,\end{aligned}$$

where $p(\lambda) = (n(\lambda))^{\log 3}+1$ *and* $n(\lambda)$ *is such* $|\overline{\mathrm{ID}}^{(0)}|, |\overline{\mathrm{ID}}^{(1)}| \leq n(\lambda)$. *Furthermore,* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *are iO and diO-legitimate respectively.*

Proof. For a fixed n we will use a sequence of sets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{\ell(n)}$ with $\mathcal{S}_i \subseteq \{0, \ldots, n\}$, where two consecutive sets only differ in exactly one element j, and $j-1$ is contained in both sets. Formally, we require

(A) $\mathcal{S}_0 = \{0\}$ and $n \in \mathcal{S}_{\ell(n)}$;
(B) $\forall\, i \in \{0, \ldots, \ell(n)\}\ \exists\, j : \mathcal{S}_i \triangle \mathcal{S}_{i+1} = \{j\}$ and $j-1 \in \mathcal{S}_i \cap \mathcal{S}_{i+1}$

(where "$\triangle$" denotes symmetric difference) For notational convenience, we also define $\mathcal{S}_{-1} := \emptyset$. A simple example of such a sequence would be $\{\mathcal{S}_i\}_{i=-1}^{n}$ with $\mathcal{S}_i := \{0, \ldots, i\}$. Fuchsbauer et al. [FKPR14] give a sequence $\mathcal{S}_0, \ldots, \mathcal{S}_{\ell(n)}$ with $\ell(n) = n^{\log 3}$ and $|\mathcal{S}_i| \leq \log n + 2$ (defined in Appendix E). This logarithmic bound on $\mathcal{S}_i$ will be used to guarantee that the total size of the hardwired values grows slowly and hence when preparing the master public key circuit, it only needs to be padded to a logarithmic length in the maximum depth of identities.

We prove the theorem via a sequence of games illustrated in Figure 13. Game $\mathrm{G}_{i,\delta,L}$ for $i = -1, \ldots, \ell(n)$, $\delta = \pm 1$, and $L = 1, \ldots, 5$ contains all boxed statements that have a label $\leq \boxed{\mathrm{L}}$. Fix an adversary $\mathcal{A}$ and its coins, and let $n^{(0)}$ and $n^{(1)}$ be the lengths of the identity vectors $\overline{\mathrm{ID}}^{(0)}$ and $\overline{\mathrm{ID}}^{(1)}$ that $\mathcal{A}$ outputs.

G_i **for** $-1 \leq i \leq \ell(n_0)$**:** These games are defined as the original game, except that the public keys for all identities $\overline{\mathrm{ID}}^{(0)}_{[j]}$ (i.e., the length-j prefix of challenge identity $\overline{\mathrm{ID}}^{(0)}$) for $j \in \mathcal{S}_i$ are replaced by random values. These values $y_j^{(0)}$ are hardwired into the circuits $\mathbf{C}^{\mathrm{gen}}$ and $\mathbf{C}^{\mathrm{del}}$. The games are formally defined in Figure 13 (ignoring all boxes). Since $\mathcal{S}_{-1} = \emptyset$, we have that G_{-1} is the Sel-AI-CPA game and thus

$$2 \cdot \Pr\left[\mathrm{G}_{-1}^{\mathcal{A}}\right] - 1 = \mathbf{Adv}^{\text{sel-ai-cpa}}_{\mathrm{HIBE},\mathcal{A}}(\lambda) \ .$$

We now show that G_i and G_{i+1} are indistinguishable. Suppose $\mathcal{S}_i \subset \mathcal{S}_{i+1} = \mathcal{S}_{i+\delta}$ with $\delta = 1$. We show this by defining intermediate hybrid games $\mathrm{G}_{i,1,1}, \ldots, \mathrm{G}_{i,1,5}$ and showing that in the following sequence, two consecutive games are indistinguishable.

$$\mathrm{G}_i, \quad \mathrm{G}_{i,1,1}, \quad \mathrm{G}_{i,1,2}, \quad \mathrm{G}_{i,1,3}, \quad \mathrm{G}_{i,1,4}, \quad \mathrm{G}_{i,1,5}, \quad \mathrm{G}_{i+1}$$

If $\mathcal{S}_{i+1} \subset \mathcal{S}_i$ then again we have $\mathcal{S}_{i+1} \subset \mathcal{S}_{(i+1)+\delta}$ with $\delta = -1$ and we simply invert the sequence of hybrids and show that in the following sequence all neighboring games are indistinguishable.

$$\mathrm{G}_{i+1}, \quad \mathrm{G}_{i+1,-1,1}, \quad \mathrm{G}_{i+1,-1,2}, \quad \mathrm{G}_{i+1,-1,3}, \quad \mathrm{G}_{i+1,-1,4}, , \quad \mathrm{G}_{i+1,-1,5} \quad \mathrm{G}_i$$

$\mathrm{G}_{i,\delta,1}$**:** Let j^* be the element that distinguishes $\mathcal{S}_i$ and $\mathcal{S}_{i+\delta}$, i.e., $\mathcal{S}_{i+\delta} = \mathcal{S}_i \cup \{j^*\}$. This game modifies the generation of MPK by replacing the obfuscation of $\mathbf{C}^{\mathrm{gen}}$ by an obfuscation of $\widetilde{\mathbf{C}}^{\mathrm{gen}}$, which computes the public key for identities $\overline{\mathrm{ID}}$ with $\mathbf{H}(k, \overline{\mathrm{ID}}) = h^{(0)}_{j^*} := \mathbf{H}(k, \overline{\mathrm{ID}}^{(0)}_{[j^*]})$ by using a hard-coded value $y^{(0)}_{j^*}$ (in addition to the hard-coded (and also random) values $y_j^{(0)}$ with $j \in \mathcal{S}_i$). It uses a key K^* punctured at $h^{(0)}_{j^*}$ for all other identities. Circuit $\widetilde{\mathbf{C}}^{\mathrm{gen}}$ is also padded to the maximum size of any such circuit. (See Figure 13.) Note that $\widetilde{\mathbf{C}}^{\mathrm{gen}}$ and $\mathbf{C}^{\mathrm{gen}}$ are functionally equivalent and have the same size.

Suppose there exists an adversary $\mathcal{A}$ that distinguishes game G_i from game $\mathrm{G}_{i,\delta,1}$. We build an adversary $\mathcal{B}$ against **iO** as follows: $\mathcal{B}$ simulates $\mathrm{G}_{i,\delta,1}$ for

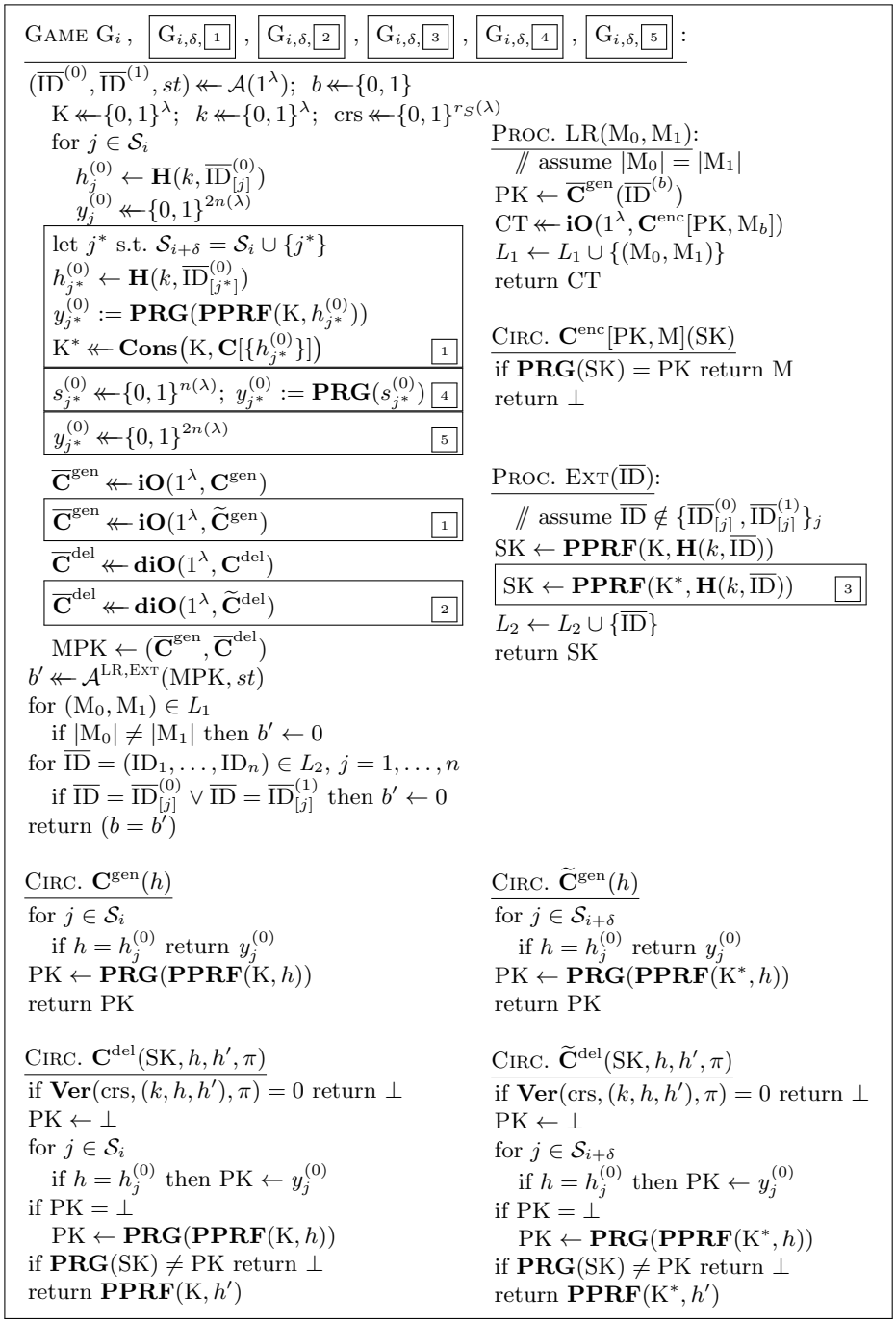

Fig. 13. The selective AI-CPA security of the HIBE scheme from Figure 12 (where we have simplified procedure EXT).

$\mathcal{A}$, constructs both $\mathbf{C}^{\text{gen}}$ and $\widetilde{\mathbf{C}}^{\text{gen}}$, queries them to its (obfuscation) LR oracle to receive an obfuscated circuit which $\mathcal{B}$ defines to be $\overline{\mathbf{C}}^{\text{gen}}$. $\mathcal{B}$ continues the simulation of the game and when $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. Depending on the bit chosen by $\mathcal{B}$'s (obfuscation) game, $\mathcal{A}$ is run either in game G_i or $\mathrm{G}_{i,\delta,1}$ and the output of $\mathcal{B}$ matches the output of these games. Hence

$$\Pr\left[\mathrm{G}_i^{\mathcal{A}}\right] - \Pr\left[\mathrm{G}_{i,\delta,1}^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\text{ind}}_{\mathbf{iO},\mathcal{B}}(\lambda) \ .$$

$\mathrm{G}_{i,\delta,2}$**:** This game again modifies the generation of MPK by replacing the obfuscation of $\mathbf{C}^{\text{del}}$ by an obfuscation of $\widetilde{\mathbf{C}}^{\text{del}}$. The latter differs from $\mathbf{C}^{\text{del}}$ in two aspects: First, if $h = h^{(0)}_{j^*}$, it uses the hard-coded value $y^{(0)}_{j^*}$ to compute PK instead of using the key K. Since $h^{(0)}_{j^*}$ is never evaluated in the 6th line of $\widetilde{\mathbf{C}}^{\text{del}}$, K can be replaced by a punctured key K^* without affecting behavior. Second, if all checks pass, $\widetilde{\mathbf{C}}^{\text{del}}$ now returns $\mathbf{PPRF}(\mathrm{K}^*, h')$ (i.e., using a punctured key instead of K). Circuit $\widetilde{\mathbf{C}}^{\text{del}}$ is also padded to the maximum size of any such circuit.

The second change modifies the behavior of $\widetilde{\mathbf{C}}^{\text{del}}$, in particular $\mathbf{C}^{\text{del}}$ and $\widetilde{\mathbf{C}}^{\text{del}}$ differ on inputs $(\mathrm{SK}, h, h', \pi)$ of the following form:

1. Proof π is valid for the statement (h, h');
2. If $h = h^{(0)}_j$ for some $j \in \mathcal{S}_{i+\delta}$ then $\mathbf{PRG}(\mathrm{SK}) = y^{(0)}_j$;
3. Otherwise $\mathbf{PRG}(\mathrm{SK}) = \mathbf{PRG}(\mathbf{PPRF}(\mathrm{K}, h))$;
4. $h' = h^{(0)}_{j^*}$ (i.e., K^* is punctured at h')

Suppose there exists an adversary $\mathcal{A}$ that distinguishes game $\mathrm{G}_{i,\delta,1}$ from game $\mathrm{G}_{i,\delta,2}$. We build an adversary $\mathcal{B}$ against **diO**, which simulates $\mathrm{G}_{i,\delta,1}$ for $\mathcal{A}$, except that it constructs both $\mathbf{C}^{\text{del}}$ and $\widetilde{\mathbf{C}}^{\text{del}}$ and queries its (obfuscation) LR oracle to receive an obfuscated circuit which $\mathcal{B}$ defines to be $\overline{\mathbf{C}}^{\text{del}}$. When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. Depending on the bit chosen by $\mathcal{B}$'s (obfuscation) game, $\mathcal{A}$ is run either in $\mathrm{G}_{i,\delta,1}$ or $\mathrm{G}_{i,\delta,2}$ and the output of $\mathcal{B}$ matches the output of these games when run with $\mathcal{A}$. Hence

$$\Pr\left[\mathrm{G}_{i,\delta,1}^{\mathcal{A}}\right] - \Pr\left[\mathrm{G}_{i,\delta,2}^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\text{ind}}_{\mathbf{diO},\mathcal{B}}(\lambda) \ .$$

In order to argue that $\mathbf{Adv}^{\text{ind}}_{\mathbf{diO},\mathcal{B}}(\lambda)$ is negligible, it remains to show that $\mathcal{B}$ is diO-legitimate; that is, for every PPT extractor $\mathcal{E}$ we have $\mathbf{Adv}^{\text{eq}}_{\mathcal{B},\mathcal{E}}(\lambda) \in \text{NEGL}$.

Suppose there exists an extractor $\mathcal{E}$ that given $\mathcal{B}$'s coins returns an input on which $\mathbf{C}^{\text{del}}$ and $\widetilde{\mathbf{C}}^{\text{del}}$ differ, i.e., $\mathcal{E}$ returns $(\mathrm{SK}, h, h', \pi)$ that satisfies items (1)–(4) above. Then we show that $\mathcal{E}$ can be used to construct a $\mathcal{B}_S$ that breaks the extractability of SNARK or a $\mathcal{B}_{\mathbf{H}}$ that breaks the collision-resistance of $\mathbf{H}$.

We define $\mathcal{B}_S$ to be an algorithm that samples coins for $\mathcal{B}$, runs $\mathcal{E}$ on them to obtain $(\mathrm{SK}, h, h', \pi)$ and returns statement (h, h') and π. By the extractability property of the SNARK, for this $\mathcal{B}_S$ there exists an extractor $\mathcal{E}_S$ that on input the coins of $\mathcal{B}_S$ outputs a witness $(\overline{\mathrm{ID}}, \mathrm{ID}')$. In particular if item (1) above holds, we have

5. $h = \mathbf{H}(k, \overline{\mathrm{ID}})$ and $h' = \mathbf{H}(k, \overline{\mathrm{ID}}|\mathrm{ID}')$ and $\mathrm{ID}' \neq \varepsilon$.

By definition we have $h^{(0)}_{j^*} = \mathbf{H}(k, \overline{\mathrm{ID}}^{(0)}_{[j^*]})$; combining this with items (4) and (5) yields

$$\mathbf{H}(k, \overline{\mathrm{ID}}|\mathrm{ID}') = \mathbf{H}(k, \overline{\mathrm{ID}}^{(0)}_{[j^*]}) \ .$$

We now argue that with overwhelming probability we have that

6. $\overline{\mathrm{ID}}|\mathrm{ID}' = \overline{\mathrm{ID}}^{(0)}_{[j^*]}$.

If this was not the case then we could build an adversary $\mathcal{B}_{\mathbf{H}}$ that breaks the public-coin collision resistance of $\mathbf{H}$ as follows. Algorithm $\mathcal{B}_{\mathbf{H}}$ receives k, which it uses to define $k \in \{0,1\}^\lambda$ for the scheme. It samples the remaining randomness used by $\mathcal{B}$ and combines it with the received r to form coins $r_{\mathcal{B}}$ for $\mathcal{B}$. It samples coins $r_{\mathcal{E}}$ for $\mathcal{E}$, which together define coins $r_{\mathcal{B}_S}$ for $\mathcal{B}_S$, on which $\mathcal{B}_{\mathbf{H}}$ runs $\mathcal{E}_S$ to obtain $(\overline{\mathrm{ID}}, \mathrm{ID}')$. Finally, $\mathcal{B}_{\mathbf{H}}$ returns $(\overline{\mathrm{ID}}|\mathrm{ID}', \overline{\mathrm{ID}}^{(0)}_{[j^*]})$.

If $j^* = 0$, we have $\overline{\mathrm{ID}}^{(0)}_{[j^*]} = \varepsilon$, so (6) contradicts (5) and thus we have shown that no differing-input extractor exists.

If $j^* \geq 1$, then (6) implies $\overline{\mathrm{ID}} = \overline{\mathrm{ID}}^{(0)}_{[j^*-1]}$, thus $h = \mathbf{H}(k, \overline{\mathrm{ID}}) = \mathbf{H}(k, \overline{\mathrm{ID}}^{(0)}_{[j^*-1]}) = h^{(0)}_{j^*-1}$. By property (B) of the set system $\{\mathcal{S}_0, \ldots\}$ (see beginning of the proof), we have $j^* - 1 \in \mathcal{S}_i$, meaning $y^{(0)}_{j^*-1}$ is a random value for which with overwhelming probability there is no pre-image SK under $\mathbf{PRG}$, i.e., with $\mathbf{PRG}(\mathrm{SK}) = y^{(0)}_{j^*-1}$. This however contradicts item (2), showing no differing-input extractor exists.

$\mathrm{G}_{i,\delta,3}$**:** This game modifies the previous game by using the key K^* punctured at $\mathbf{H}(k, \overline{\mathrm{ID}}^{(0)}_{[j^*]})$ instead of K when answering $\mathcal{A}$'s queries to Ext.

Since $\mathcal{A}$ is not allowed to query a key for $\overline{\mathrm{ID}}^{(0)}_{[j^*]}$, the two games only differ if $\mathcal{A}$ queries the oracle for an identity $\overline{\mathrm{ID}}$ with $\mathbf{H}(k, \overline{\mathrm{ID}}) = \mathbf{H}(k, \overline{\mathrm{ID}}^{(0)}_{[j^*]})$. We can thus construct an adversary $\mathcal{B}_{\mathbf{H}}$, using the given randomness for k, simulates the rest of the game and outputs a collision $(\overline{\mathrm{ID}}, \overline{\mathrm{ID}}^{(0)}_{[j^*]})$. Hence

$$\Pr\left[\mathrm{G}^{\mathcal{A}}_{i,\delta,2}\right] - \Pr\left[\mathrm{G}^{\mathcal{A}}_{i,\delta,3}\right] \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathbf{H},\mathcal{B}}(\lambda) \ .$$

$\mathrm{G}_{i,\delta,4}$**:** This game no longer sets $y^{(0)}_{j^*} := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{K}, h^{(0)}_{j^*}))$, but instead computes it by sampling $s^{(0)}_{j^*} \twoheadleftarrow \{0,1\}^{n(\lambda)}$ and then setting $y^{(0)}_{j^*} := \mathbf{PRG}(s^{(0)}_{j^*})$.

Let $\mathcal{A}$ be an adversary that distinguishes $\mathrm{G}_{i,\delta,3}$ and $\mathrm{G}_{i,\delta,4}$. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the selective security of the puncturable PRF as follows. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ until it outputs a pair $(\overline{\mathrm{ID}}^{(0)}, \overline{\mathrm{ID}}^{(1)})$. It then queries a punctured key K^* at $\mathbf{H}(k, \overline{\mathrm{ID}}^{(0)}_{[j^*]})$ and it queries its PPRF challenge oracle on $h^{(0)}_{j^*} = \mathbf{H}(k, \overline{\mathrm{ID}}^{(0)}_{j^*})$ to receive $s^{(0)}_{j^*}$ (which is either $\mathbf{PPRF}(\mathrm{K}, h^{(0)}_{j^*})$ or truly random.) Note that $\mathcal{B}$'s query is legitimate. Algorithm $\mathcal{B}$ then simulates the rest

of $\mathrm{G}_{i,\delta,3}$. Note that $\mathcal{B}$ can do this because K, the un-punctured key, is not used anywhere. When $\mathcal{A}$ terminates with a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. If $s^{(0)}_{j^*}$ is $\mathbf{PPRF}(\mathrm{K}, h^{(0)}_{j^*})$ then $\mathcal{A}$ is run in game $\mathrm{G}_{i,\delta,3}$, and when it is random then $\mathcal{A}$ is run with respect to $\mathrm{G}_{i,\delta,4}$. Hence,

$$\Pr\left[\mathrm{G}^{\mathcal{A}}_{i,\delta,3}\right] - \Pr\left[\mathrm{G}^{\mathcal{A}}_{i,\delta,4}\right] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{PPRF},\mathcal{B}}(\lambda) \ .$$

$\mathrm{G}_{i,\delta,5}$**:** This game modifies $y^{(0)}_{j^*}$ further: instead of setting it to $\mathbf{PRG}(s^{(0)}_{j^*})$ it directly samples $y^{(0)}_{j^*} \twoheadleftarrow \{0,1\}^{2n(\lambda)}$.

Let $\mathcal{A}$ be an adversary that distinguishes $\mathrm{G}_{i,\delta,4}$ from $\mathrm{G}_{i,\delta,5}$. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of PRG as follows. Algorithm $\mathcal{B}$ receives a value $y^{(0)}_{j^*}$ which is either $\mathbf{PRG}(s^{(0)}_{j^*})$ for a uniformly random $s^{(0)}_{j^*}$, or is a truly random value. Algorithm $\mathcal{B}$ simulates $\mathrm{G}_{i,\delta,4}$ for $\mathcal{A}$, except that it uses the given $y^{(0)}_{j^*}$ (instead of running the code in boxes 4 and 5 in Figure 13). When $\mathcal{A}$ terminates with a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. If $y^{(0)}_{j^*}$ is $\mathbf{PRG}(s^{(0)}_{j^*})$, algorithm $\mathcal{A}$ is run in game $\mathrm{G}_{i,\delta,4}$ and when it is truly random it is run in $\mathrm{G}_{i,\delta,5}$. Hence,

$$\Pr\left[\mathrm{G}^{\mathcal{A}}_{i,\delta,4}\right] - \Pr\left[\mathrm{G}^{\mathcal{A}}_{i,\delta,5}\right] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

We have thus arrived at a game, $\mathrm{G}_{i,\delta,5}$, in which all values $y^{(0)}_j$ for $j \in \mathcal{S}_{i+\delta}$ are random. The only difference to game $\mathrm{G}_{i+\delta}$ is that the latter uses circuits $\mathbf{C}^{\mathrm{gen}}$ and $\mathbf{C}^{\mathrm{del}}$ and oracle Ext, which have an un-punctured key K for the P-PRF evaluations. Analogously to game hops $0 \to 1$, $1 \to 2$. and $2 \to 3$, we can show that there exist iO adversary $\mathcal{B}_1$, diO adversary $\mathcal{B}_2$, and **H**-adversary $\mathcal{B}_3$ with

$$\Pr\left[\mathrm{G}^{\mathcal{A}}_{i,\delta,5}\right] - \Pr\left[\mathrm{G}^{\mathcal{A}}_{i+\delta}\right] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{iO},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{diO},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathrm{cr}}_{\mathbf{H},\mathcal{B}_3}(\lambda) \ .$$

So far we have thus shown that the Sel-AI-CPA game is indistinguishable from game $\mathrm{G}_{\ell(n^{(0)})}$, in particular there exist $\mathcal{B}_1, \ldots, \mathcal{B}_5$ with

$$\begin{aligned}\Pr\left[\mathrm{G}^{\mathcal{A}}(\lambda)\right] - \Pr\left[\mathrm{G}_{\ell(n^{(0)})}\right] \leq 2(\ell(n^{(0)})+1) \cdot \big(&\mathbf{Adv}^{\mathrm{ind}}_{\mathbf{iO},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{diO},\mathcal{B}_2}(\lambda) \\ &+ \mathbf{Adv}^{\mathrm{cr}}_{\mathbf{H},\mathcal{B}_3}(\lambda)\big) + (\ell(n^{(0)})+1) \cdot \big(\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{PPRF},\mathcal{B}_4}(\lambda) + \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{PRG},\mathcal{B}_5}(\lambda)\big) \ .\end{aligned}$$

By property (A) of our set system $\mathcal{S}_{-1}, \ldots, \mathcal{S}_{\ell(n^{(0)})}$, in $\mathrm{G}_{\ell(n^{(0)})}$ the value $y^{(0)}_{\ell(n^{(0)})}$, which corresponds to the public key for challenge identity $\overline{\mathrm{ID}}^{(0)}$, is random.

We now define a similar sequence of games $\mathrm{G}^{(1)}_{-1} := \mathrm{G}_{\ell(n^{(0)})}, \mathrm{G}^{(1)}_0, \ldots, \mathrm{G}^{(1)}_{\ell(n^{(1)})}$ where $\mathrm{G}^{(1)}_i$ is defined by setting all values $y^{(0)}_j$ for $j \in \mathcal{S}_{\ell(n^{(0)})}$ and all values $y^{(1)}_j$ for $j \in \mathcal{S}_j$ to be random. An analogous argument to the one above now shows that $\mathrm{G}^{(1)}_{-1}$ is indistinguishable from $\mathrm{G}^{(1)}_{\ell(n^{(1)})}$. Together, this shows that the original game is indistinguishable from $\mathrm{G}^{(1)}_{\ell(n^{(1)})}$, a game in which the public keys corresponding the challenge identities, i.e., the vales $y^{(0)}_{\ell(n^{(0)})}$ and $y^{(1)}_{\ell(n^{(1)})}$ with overwhelming probability do not have preimages under $\mathbf{PRG}$.

$\mathrm{G}^{(2)}$**:** This game is defined like $\mathrm{G}^{(1)}_{\ell(n^{(1)})}$ but generates the challenge ciphertext by always encrypting M_0 for identity $\overline{\mathrm{ID}}^{(0)}$.

Let $\mathcal{A}$ be an adversary that distinguishes $\mathrm{G}^{(1)}_{\ell(n^{(1)})}$ and $\mathrm{G}^{(2)}$. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of **iO** as follows. $\mathcal{B}$ simulates $\mathrm{G}^{(1)}_{\ell(n^{(1)})}$ for $\mathcal{A}$ answering its LR queries on $(\mathrm{M}_0, \mathrm{M}_1)$ by setting $\mathrm{PK}^{(i)} \leftarrow y^{(i)}_{n^{(i)}}$ for $i = 0, 1$ (which implements $\mathrm{PK}^{(i)} \leftarrow \overline{\mathbf{C}}^{\mathrm{gen}}(\overline{\mathrm{ID}}^{(i)})$) and calling its (obfuscation) LR oracle on $\mathbf{C}^{\mathrm{enc}}[\mathrm{PK}^{(b)}, \mathrm{M}_b]$ and $\mathbf{C}^{\mathrm{enc}}[\mathrm{PK}^{(0)}, \mathrm{M}_0]$ and returning the received CT to $\mathcal{A}$. When $\mathcal{A}$ outputs b' then $\mathcal{B}$ returns $(b = b')$. According to the challenge bit in the obfuscation game, $\mathcal{A}$ is run in game $\mathrm{G}^{(1)}_{\ell(n^{(1)})}$ or $\mathrm{G}^{(2)}$. Hence

$$\Pr\left[(\mathrm{G}^{(1)}_{\ell(n^{(1)})})^{\mathcal{A}}\right] - \Pr\left[(\mathrm{G}^{(2)})^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{iO},\mathcal{B}}(\lambda) \ .$$

It remains to show that the circuits queried by $\mathcal{B}$ to its challenge oracle are functionally equivalent. Since $y^{(0)}_{n^{(0)}}$ and $y^{(1)}_{n^{(1)}}$ are truly random bit strings of length $2n(\lambda)$, with overwhelming probability there do not exist $\mathrm{SK}^{(0)}$, $\mathrm{SK}^{(1)}$ with $\mathbf{PRG}(\mathrm{SK}^{(i)}) = y^{(i)}_{n^{(i)}}$. With overwhelming probability over the choice of $y^{(0)}_{n^{(0)}}$ and $y^{(1)}_{n^{(1)}}$ the circuits $\mathbf{C}^{\mathrm{enc}}[\mathrm{PK}^{(b)}, \mathrm{M}_b]$ and $\mathbf{C}^{\mathrm{enc}}[\mathrm{PK}^{(0)}, \mathrm{M}_0]$ are both functionally equivalent to a circuit that always outputs $\perp$. (And they are of the same size, since $|\mathrm{M}_b| = |\mathrm{M}_0|$.)

Finally note that $\mathrm{G}^{(2)}$ is independent of the bit b and hence for any (even unbounded) adversary $\mathcal{A}$ we have $\Pr[(\mathrm{G}^{(2)})^{\mathcal{A}}] = 0$. The theorem follows. □

Remark. Suppose we define $\mathbf{H}(k, \mathrm{ID}) := \mathrm{ID}$ for all k, ID and $\mathbf{Prove}(\mathrm{crs}, (h, h'), w) := w$. Then there are no collisions for $\mathbf{H}$ and SNARK proofs are perfectly extractable. This means that the diO adversary we constructed when showing that $\mathrm{G}_{i,\delta,1}$ and $\mathrm{G}_{i,\delta,2}$ are indistinguishable is actually *iO-legitimate*. It then suffices to use an indistinguishability obfuscator to obtain a *bounded*-depth HIBE from the above construction.

References

ABC+05. Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, Heidelberg, August 2005.

AS16. Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, January 2016.

BBDP01. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, Heidelberg, December 2001.

BCG+17. Zvika Brakerski, Nishanth Chandran, Vipul Goyal, Aayush Jain, Amit Sahai, and Gil Segev. Hierarchical functional encryption. In *Proceedings of the 2017 ACM Conference on Innovations in Theoretical Computer Science, Berkeley, CA, USA, January 9-11, 2017*, 2017.

BGI+12. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, May 2012.

BGI14. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.

BLR+15. Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 563–594. Springer, Heidelberg, April 2015.

BNPW16. Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 391–418. Springer, Heidelberg, October / November 2016.

BP15. Nir Bitansky and Omer Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 401–427. Springer, Heidelberg, March 2015.

BSW16. Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 792–821. Springer, Heidelberg, May 2016.

BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

BW07. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, Heidelberg, February 2007.

BW13. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

BZ14. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014.

CLP15. Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 287–307. Springer, Heidelberg, August 2015.

CLTV15. Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2015.

CRV10. Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 72–89. Springer, Heidelberg, February 2010.

ElG84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

FKPR14. Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, December 2014.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GGHW14. Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 518–535. Springer, Heidelberg, August 2014.

GGM86. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

GGSW13. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

Gol01. Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

GS02. Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, Heidelberg, December 2002.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015.

HR04. Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105. Springer, Heidelberg, August 2004.

HSW14. Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 201–220. Springer, Heidelberg, May 2014.

HW15. Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, January 2015.

IPS15. Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 668–697. Springer, Heidelberg, March 2015.

KPTZ13. Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, November 2013.

KSW08. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, April 2008.

Lin16. Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 28–57. Springer, Heidelberg, May 2016.

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 Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May 2010.

LW11. Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Heidelberg, May 2011.

MH14. Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via point obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 95–120. Springer, Heidelberg, February 2014.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

PPS15. Omkant Pandey, Manoj Prabhakaran, and Amit Sahai. Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 638–667. Springer, Heidelberg, March 2015.

SBC+07. Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *2007 IEEE Symposium on Security and Privacy*, pages 350–364. IEEE Computer Society Press, May 2007.

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

Wat15. Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Heidelberg, August 2015.

WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. Cryptology ePrint Archive, Report 2017/276, 2017. http://eprint.iacr.org/2017/276.

Zha16. Mark Zhandry. The magic of ELFs. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 479–508. Springer, Heidelberg, August 2016.

Supplementary Material

A Proof of Lemma 1

Correctness. Correctness of the obfuscator means that circuits $\mathbf{C}[X, Z, \mathrm{M}]$ and $\overline{\mathbf{C}}[h_{11}, h_{12}, h_{21}, h_{22}, h_{31}, h_{32}] = \mathbf{Obf}(\mathbf{C}[X, Z, \mathrm{M}])$ are functionally equivalent. Our obfuscator only achieves statistical correctness, meaning that with negligible probability over the choice of $r_1, r_2, r_2, s_1, s_2, s_3$ the obfuscated circuit might differ from the original one. This comes from the fact that there might exist x' such that $h_{11}^{x'} = h_{12}$ and $h_{21}^{x'} = h_{22}$, even though there does not exist any x' such that $(g^{x'}, Y^{x'}) = (X, Z)$ in the original circuit. Letting x, y, z denote the respective discrete logarithms of X, Y, Z, $h_{11}^{x'} = h_{12}$ is equivalent to $(r_1 + ys_1)x' = xr_1 + zs_1$, so $x_1 = \frac{xr_1+zs_1}{r_1+ys_1}$ always satisfies this equation (assuming $r_1 + ys_1 \neq 0$). Similarly, we have $h_{21}^{x'} = h_{22}$ if and only if $(r_2 + ys_2)x' = xr_2 + zs_2$, so assuming $r_1 + ys_1 \neq 0$, we have that $x_2 = \frac{xr_2+zs_2}{r_2+ys_2}$ satisfies this equation.

Note that if $z = xy$, then $x_1 = x_2 = x$ and we have perfect correctness, but if $z \neq xy$, $x_1 = x_2$ and $x_1 \neq x$, the obfuscated circuit differs on this point. Thus the statistical correctness of our obfuscator, since the obfuscated circuit might not be functionally equivalent, but this only happens with negligible probability (when $x_1 = x_2$ and $x_1 \neq x$).

This is not an issue for our construction. In particular, since obfuscator is perfectly correct when the public key is well-formed ($z = xy$), our encryption scheme satisfies perfect correctness.

Indistinguishability. Let us now prove the indistinguishability security of the obfuscator. Let $\mathbf{C}_1[X, Z, \mathrm{M}]$ and $\mathbf{C}_2[X', Z', \mathrm{M}']$ be two functionally equivalent circuits. It is clear that if (X, Z) is a well-formed public-key (meaning that $Z = Y^{\log(X)}$) then one needs to have $X' = X$, $Z' = Z$, and $\mathrm{M}' = \mathrm{M}$, otherwise circuits $\mathbf{C}_1$ and $\mathbf{C}_2$ would differ on input $x = \log(X)$, since $\mathbf{C}_1$ would output M (and we have by definition $\mathrm{M} \neq 1$) and $\mathbf{C}_2$ would output either a different message M' or 1 (if $(X', Z') \neq (g^x, Y^x)$). Hence, if (X, Z) is a well-formed public-key and $\mathbf{C}_2$ is functionally equivalent to $\mathbf{C}_1$, then $\mathbf{C}_2 = \mathbf{C}_1$ and there obfuscations are (perfectly) indistinguishable.

Let us now assume (X, Z) is not a well-formed key (so $Z \neq Y^{\log(X)}$). Then, any circuit $\mathbf{C}_2[X', Z', \mathrm{M}']$ such that (X', Z') is also not a well-formed key is functionally equivalent to $\mathbf{C}_1[X, Z, \mathrm{M}]$, since they both always output 1 (and any circuit $\mathbf{C}_2[X', Z', \mathrm{M}']$ with X', Z' well-formed *is not* functionally equivalent to $\mathbf{C}_1[X, Z, \mathrm{M}]$ since the message space is $\mathbb{G}^* = \mathbb{G} \setminus \{1\}$). Thus, we just need to show that the obfuscations of any such two circuits are indistinguishable and the indistinguishability of our obfuscator will follow. To do so, it is sufficient to prove that the two distributions

$$\begin{matrix}\big(g^{r_1}Y^{s_1}, & X^{r_1}Z^{s_1}, & g^{r_2}Y^{s_2}, & X^{r_2}Z^{s_2}, & g^{r_3}Y^{s_3}, & \mathrm{M}\cdot X^{r_3}Z^{s_3}\big)\\ \big(g^{r_1}Y^{s_1}, & X'^{r_1}Z'^{s_1}, & g^{r_2}Y^{s_2}, & X'^{r_2}Z'^{s_2}, & g^{r_3}Y^{s_3}, & \mathrm{M}'\cdot X'^{r_3}Z'^{s_3}\big)\end{matrix}$$

for $(r_1, r_2, r_3, s_1, s_2, s_3) \twoheadleftarrow \mathbb{Z}_q^6$ are identical. Letting x, y, z, m, x', z', m' denote the respective discrete logarithms of $X, Y, Z, \mathrm{M}, X', Z', \mathrm{M}'$, we thus just want to prove that the distributions

$$\begin{pmatrix} r_1 + ys_1, & xr_1 + zs_1, & r_2 + ys_2, & xr_2 + zs_2, & r_3 + ys_3, & m + xr_3 + zs_3 \end{pmatrix}$$
$$\begin{pmatrix} r_1 + ys_1, & x'r_1 + z's_1, & r_2 + ys_2, & x'r_2 + z's_2, & r_3 + ys_3, & m' + x'r_3 + z's_3 \end{pmatrix}$$

for $(r_1, r_2, r_3, s_1, s_2, s_3) \twoheadleftarrow \mathbb{Z}_q^6$ are identical, when $z \neq xy$ and $z' \neq x'y$. This immediately follows from the facts that:

$$\begin{pmatrix} r_1 + ys_1 \\ xr_1 + zs_1 \\ r_2 + ys_2 \\ xr_2 + zs_2 \\ r_3 + ys_3 \\ m + xr_3 + zs_3 \end{pmatrix} = \begin{pmatrix} 1 & y & 0 & 0 & 0 & 0 \\ x & z & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & y & 0 & 0 \\ 0 & 0 & x & z & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & y \\ 0 & 0 & 0 & 0 & x & z \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ s_1 \\ r_2 \\ s_2 \\ r_3 \\ s_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ m \end{pmatrix}$$

$$\begin{pmatrix} r_1 + ys_1 \\ x'r_1 + z's_1 \\ r_2 + ys_2 \\ x'r_2 + z's_2 \\ r_3 + ys_3 \\ m' + x'r_3 + z's_3 \end{pmatrix} = \begin{pmatrix} 1 & y & 0 & 0 & 0 & 0 \\ x' & z' & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & y & 0 & 0 \\ 0 & 0 & x' & z' & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & y \\ 0 & 0 & 0 & 0 & x' & z' \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ s_1 \\ r_2 \\ s_2 \\ r_3 \\ s_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ m' \end{pmatrix}$$

and that the above two matrices are invertible if and only if $z \neq xy$ and $z' \neq x'y$. This concludes the proof of security of our obfuscator. □

B Proof of Theorem 2: Security of IBE

Theorem 6 (Theorem 2, restated). *Scheme* IBE *in Figure 7 is* Sel-AI-CPA *secure as long as its underlying pseudorandom generator* **PRG**, *the puncturable pseudorandom function* PPRF, *and the obfuscator* **Obf** *are secure. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against* IBE *there are* PPT *adversaries* $\mathcal{B}_1$, $\mathcal{B}_2$, *and* $\mathcal{B}_3$ *against* **PRG**, PPRF, *and* **Obf** *respectively such that*

$$\mathbf{Adv}^{\text{sel-ai-cpa}}_{\text{IBE},\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{ind}}_{\text{PPRF},\mathcal{B}_2}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}_3}(\lambda) \ .$$

Algorithm $\mathcal{B}_1$ *places at most the same number of queries to its (obfuscation)* LR *oracle that* $\mathcal{A}$ *does to its (encryption)* LR *oracle.*

Proof. We prove the theorem via a sequence of games as follows.

G_0**:** This is the Sel-AI-CPA game with respect to scheme IBE in Figure 7:

$$2 \cdot \Pr\left[\mathrm{G}_0^{\mathcal{A}}\right] - 1 = \mathbf{Adv}^{\text{sel-ai-cpa}}_{\text{IBE},\mathcal{A}}(\lambda) \ .$$

G_1**:** This game modifies the generation of the master public key as follows. After $\mathcal{A}$ outputs $(\mathrm{ID}_0, \mathrm{ID}_1)$ the game generates a key MSK, computes a

punctured key MSK^* at the challenge identity set $\{\mathrm{ID}_0, \mathrm{ID}_1\}$ and defines $y_i := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}, \mathrm{ID}_i))$ for $i = 0, 1$. The master public key is generated by obfuscating the circuit

$$\widetilde{\mathbf{C}}^{\mathrm{gen}}[\mathrm{MSK}^*, y_0, y_1](\mathrm{ID}) := \begin{cases} \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}^*, \mathrm{ID})) & \text{if } \mathrm{ID} \notin \{\mathrm{ID}_0, \mathrm{ID}_1\} \ ; \\ y_0 & \text{if } \mathrm{ID} = \mathrm{ID}_0 \ ; \\ y_1 & \text{if } \mathrm{ID} = \mathrm{ID}_1 \ . \end{cases}$$

and padding it so its size is $S^{\mathrm{gen}}(\lambda)$. Note that this circuit is functionally equivalent to and has the same size as the original circuit $\mathbf{C}^{\mathrm{gen}}[\mathrm{MSK}]$.

We show that G_0 and G_1 are indistinguishable. Let $\mathcal{A}$ be an adversary that distinguishes game G_0 from game G_1. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against **Obf** as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs $(\mathrm{ID}_0, \mathrm{ID}_1, st)$. It then samples a P-PRF key MSK and punctures it at $\{\mathrm{ID}_0, \mathrm{ID}_1\}$ to compute MSK^*. It also computes $y_i := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}, \mathrm{ID}_i))$ for $i = 0, 1$. Algorithm $\mathcal{B}$ uses these values to generate two circuits $\mathbf{C}^{\mathrm{gen}}$ and $\widetilde{\mathbf{C}}^{\mathrm{gen}}$ as above and queries them to its (obfuscation) LR oracle to receive an obfuscated circuit which $\mathcal{B}$ defines to be MPK. Algorithm $\mathcal{B}$ answers the encryption queries of $\mathcal{A}$ on $(\mathrm{M}_0, \mathrm{M}_1)$ by computing a ciphertext using MPK and $(\mathrm{ID}_b, \mathrm{M}_b)$. Key-extraction queries on ID are handled using the (un-punctured) key MSK. When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easily seen that according to the bit chosen by $\mathcal{B}$'s (obfuscation) game, algorithm $\mathcal{A}$ is run either with respect to game G_0 or G_1 and the output of $\mathcal{B}$ matches the output of these games when run with $\mathcal{A}$. Hence

$$\Pr\left[\mathrm{G}_0^{\mathcal{A}}\right] - \Pr\left[\mathrm{G}_1^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

G_2: This game no longer sets $y_i := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}, \mathrm{ID}_i))$, but instead computes it by first sampling two $\mathrm{S}_i \twoheadleftarrow \{0,1\}^\lambda$ and then setting $y_i := \mathbf{PRG}(\mathrm{S}_i)$.

Let $\mathcal{A}$ be an adversary that distinguishes G_1 and G_2. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the selective security of the puncturable PRF as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs $(\mathrm{ID}_0, \mathrm{ID}_1, st)$. It then queries a punctured key MSK^* at $\{\mathrm{ID}_0, \mathrm{ID}_1\}$. Algorithm $\mathcal{B}$ also queries the P-PRF challenge oracle on ID_0 and ID_1 to receive S_0 and S_1 which are either $\mathbf{PPRF}(\mathrm{MSK}, \mathrm{ID}_0)$ and $\mathbf{PPRF}(\mathrm{MSK}, \mathrm{ID}_1)$ or truly random values. (Note that $\mathcal{B}$'s query is legitimate.) Algorithm $\mathcal{B}$ uses MSK^* and $y_i := \mathbf{PRG}(\mathrm{S}_i)$ to compute MPK. It resumes $\mathcal{A}$ on (MPK, st) and answers its extraction queries using the punctured key MSK^*. By the rules of the Sel-AI-CPA game the key extraction oracle will be never queried on ID_0 or ID_1, and hence MSK^* can be used to perfectly simulate the key extraction oracle.[11] For an LR query by $\mathcal{A}$ on $(\mathrm{M}_0, \mathrm{M}_1)$,

[11] This raises the question whether or not G_2 should use MSK^* and S_i in key-extraction queries as well. This *can* be done at this stage, and as a result we would not need to rely on the legitimacy rules of the IBE game. However, in the next two games we will use the security of the PRG to replace y_i with truly random values. To this end, we must ensure that the S_i's are not used in MPK and only their PRG values are. Hence we rely on IBE legitimacy rules to avoid using S_i later on. Note also that IBE legitimacy makes it inconsequential whether MSK^* or MSK is used.

algorithm $\mathcal{B}$ encrypts $(\mathrm{ID}_b, \mathrm{M}_b)$ using MPK. When $\mathcal{A}$ terminates with a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easy to see that if S_i is $\mathbf{PPRF}(\mathrm{MSK}, \mathrm{ID}_i)$, algorithm $\mathcal{A}$ is run in game G_1, and when S_i are random, $\mathcal{A}$ is run with respect to G_2. Hence,

$$\Pr\left[\mathrm{G}_1^{\mathcal{A}}\right] - \Pr\left[\mathrm{G}_2^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{PPRF},\mathcal{B}}(\lambda) \ .$$

G_3**:** This game modifies y_0 further as follows. Instead of setting it to $\mathbf{PRG}(\mathrm{S}_0)$ it directly samples $y_0 \twoheadleftarrow \{0,1\}^{2\lambda}$.

Let $\mathcal{A}$ be an adversary that distinguishes G_2 from G_3. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of PRG as follows. Algorithm $\mathcal{B}$ receives a value y_0 which is either $\mathbf{PRG}(\mathrm{S}_0)$, for a uniformly random S_0 or y_0 is truly random. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs $(\mathrm{ID}_0, \mathrm{ID}_1, st)$. It generates a PRF key MSK, computes a punctured key MSK^* at $\{\mathrm{ID}_0, \mathrm{ID}_1\}$, and uses MSK^*, y_0 and $y_1 := \mathbf{PRG}(\mathrm{S}_1)$ for a random S_1 to compute MPK. Algorithm $\mathcal{B}$ resumes $\mathcal{A}$ on (MPK, st) and answers its extraction queries. For the LR query $(\mathrm{M}_0, \mathrm{M}_1)$, $\mathcal{B}$ encrypts $(\mathrm{ID}_b, \mathrm{M}_b)$ using MPK. When $\mathcal{A}$ terminates with a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easy to see that if y_0 is $\mathbf{PRG}(\mathrm{S}_0)$, algorithm $\mathcal{A}$ is run in game G_2 and when it is truly random it is run with respect to G_3. Hence,

$$\Pr\left[\mathrm{G}_2^{\mathcal{A}}\right] - \Pr\left[\mathrm{G}_3^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_4**:** This game modifies y_1 analogously to G_3. That is, instead of setting it to $\mathbf{PRG}(\mathrm{S}_1)$ it directly samples $y_1 \twoheadleftarrow \{0,1\}^{2\lambda}$.

As in G_3 we have that for any $\mathcal{A}$ that distinguishes G_3 from G_4, there is a $\mathcal{B}$ such that

$$\Pr\left[\mathrm{G}_3^{\mathcal{A}}\right] - \Pr\left[\mathrm{G}_4^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_5**:** This game generates the challenge ciphertexts by always encrypting M_0 for identity ID_0 (independently of b).

Let $\mathcal{A}$ be an adversary that distinguishes G_4 and G_5. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of **Obf** as follows. Algorithm $\mathcal{B}$ picks a random bit b, generates MSK and runs $\mathcal{A}$ answers until it outputs $(\mathrm{ID}_0, \mathrm{ID}_1, st)$. It uses MSK to compute a key MSK^* punctured at these identities. It then samples $y_0, y_1 \twoheadleftarrow \{0,1\}^{2\lambda}$ and computes MPK using all these values. It resumes $\mathcal{A}$ on MPK and answers $\mathcal{A}$'s extraction queries. The LR query on $(\mathrm{M}_0, \mathrm{M}_1)$ is answered by choosing a bit b at random and calling the (obfuscation) LR oracle on $(\mathbf{C}^{\mathrm{enc}}[y_b, \mathrm{M}_b], \mathbf{C}^{\mathrm{enc}}[y_0, \mathrm{M}_0])$. Algorithm $\mathcal{B}$ continues in this manner until $\mathcal{A}$ outputs a bit b' and $\mathcal{B}$ returns $(b = b')$. It is easy to see that according to the challenge bit in the obfuscation game, $\mathcal{A}$ is run in game G_4 or G_5. Hence

$$\Pr\left[\mathrm{G}_4^{\mathcal{A}}\right] - \Pr\left[\mathrm{G}_5^{\mathcal{A}}\right] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

It remains to show that the circuits queried by $\mathcal{B}$ to its challenge oracle are functionally equivalent. Since y_0 and y_1 are truly random bit strings of length

2λ, for each value of SK and b we have that $\Pr[\mathbf{PRG}(\mathrm{SK}) = y_b] = 1/2^{2\lambda}$. Hence by the union bound

$$\Pr[\exists \mathrm{SK} \in \{0,1\}^\lambda, b \in \{0,1\} : \mathbf{PRG}(\mathrm{SK}) = y_b] \leq 2^{\lambda+1}/2^{2\lambda} = 2/2^\lambda \ .$$

This means with probability at least $1 - 2/2^\lambda$ over the choice of y_b the circuits $\mathbf{C}^{\mathrm{gen}}[y_b, \mathrm{M}_b]$ both are functionally equivalent to a circuit that always outputs $\perp$. Note also that the two circuits are of equal size as long as the two messages are of equal size.

Finally note that G_5 is independent of the bit b and hence for any (even unbounded) adversary $\mathcal{A}$ we have $\Pr[\mathrm{G}_4^{\mathcal{A}}] = 0$. The theorem follows from this and the above inequalities. □

Avoiding double puncturing. Although in the above proof we relied on puncturing at two points, this can be avoided at the cost of introducing additional game hops. Roughly speaking, we can proceed along the above sequence of games for each b until we reach game G_5. In these games (for the two bits) we switch the master circuit back to one that uses the original master secret key, keeping the public key for ID_b random. We would then puncture the key at ID_{1-b} and replace its public key with a random value, and the switch back to a master circuit that uses the original master secret key, keeping the public key for ID_{1-b} random. These modifications result in a game where the master circuit is no longer punctured for any particular identity and hence is independent of the challenge bit.

C Proof of Theorem 3: Security of Predicate Encryption

Theorem 7 (Theorem 3, restated). *Scheme* PE *in Figure 9 is* Sel-WAI-CPA *secure if its underlying pseudorandom generator* **PRG***, two-point puncturable pseudorandom function* PPRF*, and obfuscator* **Obf** *are secure. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against the* Sel-WAI-CPA *security of* PE *there are* PPT *adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$*, and* $\mathcal{B}_3$ *against* **Obf**, PPRF*, and* **PRG** *respectively such that*

$$\mathbf{Adv}^{\text{sel-wai-cpa}}_{\mathrm{PE},\mathcal{A}}(\lambda) \leq (2+Q(\lambda))\cdot\mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}_1}(\lambda)+\mathbf{Adv}^{\text{ind}}_{\mathrm{PPRF},\mathcal{B}_2}(\lambda)+2\cdot\mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}_3}(\lambda) \ .$$

where $Q(\lambda)$ *denotes the number of queries of* $\mathcal{A}$ *to its key extraction oracle.*

Proof. We prove the theorem via a sequence of games as follows:

G_0: This is the weak Sel-WAI-CPA game (Figure 8) with respect to scheme PE in Figure 9:

$$2 \cdot \Pr[\mathrm{G}_0^{\mathcal{A}}] - 1 = \mathbf{Adv}^{\text{sel-wai-cpa}}_{\mathrm{PE},\mathcal{A}}(\lambda) \ .$$

G_1: This game modifies the generation of the master public key as follows. After the first phase of $\mathcal{A}$ when the adversary outputs (γ_0, γ_1, st) the game generates a key MSK, computes a punctured key MSK^* at the challenge

attribute set $\{\gamma_0, \gamma_1\}$ and defines $y_i := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}, \gamma_i))$ for $i = 0, 1$. The master public key is generated by obfuscating the circuit

$$\tilde{\mathbf{C}}^{\mathrm{gen}}[\mathrm{MSK}^*, y_0, y_1](\gamma) := \begin{cases} \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}^*, \gamma)) & \text{if } \gamma \notin \{\gamma_0, \gamma_1\} \\ y_0 & \text{if } \gamma = \gamma_0 \\ y_1 & \text{if } \gamma = \gamma_1 \ . \end{cases}$$

Note that this is functionally equivalent to the original circuit $\mathbf{C}^{\mathrm{gen}}[\mathrm{MSK}]$.

We show that G_0 and G_1 are indistinguishable. Let $\mathcal{A}$ be an adversary that distinguishes game G_0 from game G_1. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against **Obf** as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs (γ_0, γ_1, st). It then samples a P-PRF key MSK and punctures it at $\{\gamma_0, \gamma_1\}$ to compute MSK^*. It also computes $y_i := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}, \gamma_i))$ for $i = 0, 1$. Algorithm $\mathcal{B}$ uses these values to generate two circuits $\mathbf{C}^{\mathrm{gen}}$ and $\tilde{\mathbf{C}}^{\mathrm{gen}}$ as above and queries them to its (obfuscation) LR oracle to receive an obfuscated circuit which $\mathcal{B}$ defines to be the MPK. Algorithm $\mathcal{B}$ answers the encryption queries of $\mathcal{A}$ on $(\mathrm{M}_0, \mathrm{M}_1)$ by computing a ciphertext using MPK and M_b. Key-extraction queries on $\mathbf{P}$ are handled using the (un-punctured) key MSK. When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easily to see that according to the bit chosen by $\mathcal{B}$'s (obfuscation) game, algorithm $\mathcal{A}$ is run either with respect to game G_0 or G_1 and the output of $\mathcal{B}$ matches the output of these games when run with $\mathcal{A}$. Hence

$$\Pr[\mathrm{G}_0^{\mathcal{A}}] - \Pr[\mathrm{G}_1^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

G_2: Instead of obfuscating the circuit $\mathbf{C}^{\mathrm{sk}}[\mathrm{MSK}, \mathbf{P}]$ at key extraction, this game uses the punctured key MSK^* at $\{\gamma_0, \gamma_1\}$ and predicate $\mathbf{P}$ to form the circuit below and return an obfuscation of it.

$$\tilde{\mathbf{C}}^{\mathrm{sk}}[\mathrm{MSK}^*, \mathbf{P}](\gamma) := \begin{cases} \mathbf{PPRF}(\mathrm{MSK}^*, \gamma) & \text{if } \mathbf{P}(\gamma) = 1 \\ \perp & \text{otherwise.} \end{cases}$$

Note that by the legitimacy rules of the *weak* AI-CPA game, $\mathbf{P}(\gamma_0) = \mathbf{P}(\gamma_1) = 0$ for all extracted $\mathbf{P}$. Hence the above circuit is functionally equivalent to $\mathbf{C}^{\mathrm{sk}}[\mathrm{MSK}, \mathbf{P}]$ for any extracted $\mathbf{P}$.[12]

Let $\mathcal{A}$ be an adversary that distinguishes G_1 from G_2. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the multi-challenge **Obf** as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs (γ_0, γ_1). It then samples a P-PRF key MSK and punctures it at $\{\gamma_0, \gamma_1\}$ to get MSK^*. It also computes $y_b := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}, \gamma_b))$ for $b = 0, 1$. Algorithm $\mathcal{B}$ uses these values to generate $\tilde{\mathbf{C}}^{\mathrm{gen}}[\mathrm{MSK}^*, y_0, y_1]$ and obfuscates it to obtain MPK (as in G_1). For each key-extraction query $\mathbf{P}$, algorithm $\mathcal{B}$ generates $\mathbf{C}^{\mathrm{sk}}[\mathrm{MSK}, \mathbf{P}]$ and $\tilde{\mathbf{C}}^{\mathrm{sk}}[\mathrm{MSK}^*, \mathbf{P}]$ as above and submits them to its (obfuscation) LR oracle to obtain an obfuscated

[12] As in the IBE reduction we avoid hardwiring any PRF values that will be used as PRG seeds in the later games.

circuit SK. It answers the query using SK. Algorithm $\mathcal{B}$ answers the encryption query of $\mathcal{A}$ on $(\mathrm{M}_0, \mathrm{M}_1)$ by choosing a bit b at random and computing a ciphertext using MPK and (γ_b, M_b). When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easily to see that according to the bit chosen by $\mathcal{B}$'s (obfuscation) game, algorithm $\mathcal{A}$ is run either with respect to game G_1 or G_2 and the output of $\mathcal{B}$ matches the output of these games when run with $\mathcal{A}$. Hence

$$\Pr[\mathrm{G}_1^{\mathcal{A}}] - \Pr[\mathrm{G}_2^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \leq Q(\lambda) \cdot \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}'}(\lambda) \ ,$$

where $\mathcal{B}'$ is a single-challenge adversary against the obfuscator.

G_3**:** This game no longer sets $y_b := \mathbf{PRG}(\mathbf{CPRF}(\mathrm{MSK}, \gamma_b))$ but rather computes these values by sampling $\mathrm{S}_b \twoheadleftarrow \{0,1\}^\lambda$ and setting $y_b := \mathbf{PRG}(\mathrm{S}_b)$.

Let $\mathcal{A}$ be an adversary that distinguishes G_2 from G_3. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of the doubly puncturable PRF as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs (γ_0, γ_1). It requests a punctured key MSK^* at $\{\gamma_0, \gamma_1\}$ and also queries its challenge oracle twice on γ_0 and γ_1 to receive two values S_0 and S_1. (Note that these queries are valid.) Algorithm $\mathcal{B}$ sets $y_b := \mathbf{PRG}(\mathrm{S}_b)$, forms the circuit $\tilde{\mathbf{C}}^{\mathrm{gen}}[\mathrm{MSK}^*, y_0, y_1]$ and obfuscates it to get MPK. It resumes $\mathcal{A}$ on MPK and answers its key-extraction queries on $\mathbf{P}$ by retuning an obfuscation of the circuit $\tilde{\mathbf{C}}^{\mathrm{sk}}[\mathrm{MSK}^*, \mathbf{P}]$. The encryption queries are answered by choosing a bit b at random and encrypting (γ_b, M_b) under MPK. When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easily to see that according to the bit chosen by $\mathcal{B}$'s game, algorithm $\mathcal{A}$ is run either with respect to game G_1 or G_2, and the output of $\mathcal{B}$ matches the output of these games when run with $\mathcal{A}$. Hence

$$\Pr[\mathrm{G}_2^{\mathcal{A}}] - \Pr[\mathrm{G}_3^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{PPRF},\mathcal{B}}(\lambda) \ .$$

G_4**:** This game modifies the generation of y_0 further: instead of computing it as $\mathbf{PRG}(\mathrm{S}_0)$, it directly samples $y_0 \twoheadleftarrow \{0,1\}^{2\lambda}$.

Let $\mathcal{A}$ be an adversary that distinguishes G_3 from G_4. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of PRG as follows. Algorithm $\mathcal{B}$ receives a value y_0 which is either $\mathbf{PRG}(\mathrm{S}_0)$ or is truly random. Algorithm $\mathcal{B}$ picks a bit b and runs $\mathcal{A}$ until it outputs an identity $(\mathrm{ID}_0, \mathrm{ID}_1)$. It generates a PRF key MSK, computes a punctured key MSK^* at $\{\gamma_0, \gamma_1\}$, and uses MSK^*, y_0 and $y_1 := \mathbf{PRG}(\mathrm{S}_1)$ for a random S_1 to compute MPK. Algorithm $\mathcal{B}$ resumes $\mathcal{A}$ on MPK and answers its extraction queries by obfuscating $\tilde{\mathbf{C}}^{\mathrm{sk}}[\mathrm{MSK}^*, \mathbf{P}]$ as in G_3. For the LR query $(\mathrm{M}_0, \mathrm{M}_1)$, $\mathcal{B}$ chooses a random bit b and encrypts (γ_b, M_b) using MPK. When $\mathcal{A}$ terminates with a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easy to see that if y_0 is $\mathbf{PRG}(\mathrm{S}_0)$, algorithm $\mathcal{A}$ is run in game G_2 and when it is truly random it is run with respect to G_3. Hence,

$$\Pr[\mathrm{G}_3^{\mathcal{A}}] - \Pr[\mathrm{G}_4^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_5**:** This game modifies the generation of y_1 analogously to that of y_0 in G_4.

Once again, using a reduction similar to that for the IBE scheme we deduce that for any PPT $\mathcal{A}$ there is a PPT $\mathcal{B}$ such that

$$\Pr[\mathrm{G}_4^{\mathcal{A}}] - \Pr[\mathrm{G}_5^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_6**:** This game generates the challenge ciphertexts by always encrypting M_0 for attribute γ_0 (independently of b).

Let $\mathcal{A}$ be an adversary that distinguishes G_5 and G_6. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the security of **Obf** as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ answers until it outputs (γ_0, γ_1). It generates a PRF key MSK and computes a punctured key MSK^* at $\{\gamma_0, \gamma_1\}$. It then samples random $y_0, y_1 \twoheadleftarrow \{0,1\}^{2\lambda}$ and computes MPK as obfuscation of $\tilde{\mathbf{C}}^{\mathrm{gen}}[\mathrm{MSK}^*, y_0, y_1]$. It answers the key extraction queries on $\mathbf{P}$ by returning obfuscations of the circuits $\tilde{\mathbf{C}}^{\mathrm{sk}}[\mathrm{MSK}^*, y_0, y_1, \mathbf{P}]$. It answers the encryption queries $(\mathrm{M}_0, \mathrm{M}_1)$ by calling its own (obfuscation) LR oracle on $(\mathbf{C}^{\mathrm{enc}}[y_b, \mathrm{M}_b], \mathbf{C}^{\mathrm{enc}}[y_b, \mathrm{M}_b])$ and receives CT. It resumes $\mathcal{A}$ on CT and answers its extraction queries as before. When $\mathcal{A}$ outputs a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easy to see that according to the challenge bit in the obfuscation game, $\mathcal{A}$ is run in game G_5 or G_6. Hence

$$\Pr[\mathrm{G}_5] - \Pr[\mathrm{G}_6] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

It remains to show that the circuits queried by $\mathcal{B}$ to its challenge oracle are functionally equivalent. The argument is as before: Since y_0 and y_1 are truly random bit strings of length 2λ, for each value of SK and b we have that $\Pr[\mathbf{PRG}(\mathrm{SK}) = y_b] = 1/2^{2\lambda}$. Hence by the union bound

$$\Pr\left[\exists\, \mathrm{SK} \in \{0,1\}^{\lambda}, b \in \{0,1\} : \mathbf{PRG}(\mathrm{SK}) = y_b\right] \leq 2^{\lambda+1}/2^{2\lambda} = 2/2^{\lambda} \ .$$

This means with probability at least $1 - 2/2^{\lambda}$ over the choice of y_b the circuits $\mathbf{C}^{\mathrm{gen}}[y_b, \mathrm{M}_b]$ both implement the all-zero circuit. Note also that the two circuits are of equal size as long as the two messages are of equal size.

Finally note that the MPK and all oracle responses in G_6 are independent of the bit b. Hence the for any (even unbounded) IND-CPA adversary $\Pr[\mathrm{G}_6^{\mathcal{A}}] = 0$. The theorem follows. □

D Bounded HIBE

The GGM construction, on top of offering puncturing of keys permits a delegation mechanism whereby a punctured key for a set S of points can be used to derive a more restricted punctured key for subset of S. We leverage this property to extend our IBE construction from Section 4 to a HIBE for a priori *bounded* number of polynomial levels in the hierarchies. The reason that our construction only supports only a bounded-depth hierarchy is that the master secret key underlying the generation of the master public key will be punctured at a number of points that is proportional to the depth of challenge identities. Therefore, to

ALGO. $\mathbf{HSetup}(1^\lambda)$	ALGO. $\mathbf{HEnc}(\mathrm{MPK}, \overline{\mathrm{ID}}, \mathrm{M})$
$\mathrm{SK}_0 \twoheadleftarrow \{0,1\}^\lambda$	$\mathrm{PK} \leftarrow \mathrm{MPK}(\overline{\mathrm{ID}})$
$\mathrm{MPK} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}^{\mathrm{gen}}[\mathrm{SK}_0])$	$\mathrm{CT} \twoheadleftarrow \mathbf{iO}(1^\lambda, \mathbf{C}^{\mathrm{enc}}[\mathrm{PK}, \mathrm{M}])$
return $(\mathrm{SK}_0, \mathrm{MPK})$	return CT
CIRC. $\mathbf{C}^{\mathrm{gen}}[\mathrm{SK}_0](\mathrm{ID}_1, \ldots, \mathrm{ID}_d)$	CIRC. $\mathbf{C}^{\mathrm{enc}}[\mathrm{PK}, \mathrm{M}](\mathrm{SK})$
for $i = 1, \ldots, d$:	if $\mathbf{PRG}(\mathrm{SK}) = \mathrm{PK}$ return M
$\quad \mathrm{SK}_i \leftarrow \mathbf{PPRF}(\mathrm{SK}_{i-1}, \mathrm{ID}_i)$	return $\perp$
$\mathrm{PK} \leftarrow \mathbf{PRG}(\mathrm{SK}_d)$	
return PK	
ALGO. $\mathbf{HDel}(\mathrm{SK}, \mathrm{ID})$	ALGO. $\mathbf{HDec}(\mathrm{SK}, \mathrm{CT})$
$\mathrm{SK}' \leftarrow \mathbf{PPRF}(\mathrm{SK}, \mathrm{ID})$	$\mathrm{M} \leftarrow \mathrm{CT}(\mathrm{SK})$
return SK'	return M

Fig. 14. A $d(\cdot)$-bounded HIBE from a PRG, a puncturable PRF, and an iO. In the above $d := d(\lambda)$.

use indistinguishability obfuscation the circuit for the original master public key should be padded to the maximum depth of identities queried.

We present our construction based on (a tree of) punctured PRFs that allows directly handling large identities at each level. Our construction is shown in Figure 14. For this scheme, the PPRF needs to have identical key space and domain, i.e., $m(\lambda) = \lambda$ using the notation from Section 2.

Theorem 8. *The $d(\cdot)$-bounded HIBE scheme in Figure 14 is* Sel-AI-CPA *secure as long as its underlying pseudorandom generator* **PRG***, puncturable PRF* PPRF*, and obfuscator* **Obf** *are secure. More precisely, for any* PPT *adversary $\mathcal{A}$ against* HIBE *there are* PPT *adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$ against* **Obf**, PPRF*, and* **PRG** *respectively such that*

$$\mathbf{Adv}^{\text{sel-ai-cpa}}_{\text{HIBE},\mathcal{A}}(\lambda) \leq 2{\cdot}\mathbf{Adv}^{\text{ind}}_{\mathbf{Obf},\mathcal{B}_1}(\lambda) + (2{\cdot}d(\lambda){-}1){\cdot}\mathbf{Adv}^{\text{ind}}_{\text{PPRF},\mathcal{B}_2}(\lambda) + 2{\cdot}\mathbf{Adv}^{\text{ind}}_{\mathbf{PRG},\mathcal{B}_3}(\lambda) \ .$$

Proof. We prove the theorem via a sequence of games as follows.

G_0**:** This is the Sel-AI-CPA game with respect to scheme HIBE in Figure 14. By definition we have

$$2 \cdot \Pr[\mathrm{G}_0^{\mathcal{A}}] - 1 = \mathbf{Adv}^{\text{sel-ai-cpa}}_{\text{HIBE},\mathcal{A}}(\lambda) \ .$$

G_1**:** This game modifies the generation of the master public key as follows. Let $\overline{\mathrm{ID}}_0 = (\mathrm{ID}_{0,1}, \ldots, \mathrm{ID}_{0,d})$ and $\overline{\mathrm{ID}}_1 = (\mathrm{ID}_{1,1}, \ldots, \mathrm{ID}_{1,d})$ be the identity vectors that $\mathcal{A}$ chooses. Let $(\mathrm{ID}_1, \ldots, \mathrm{ID}_\ell)$ be the longest common prefix of $\overline{\mathrm{ID}}_0$ and $\overline{\mathrm{ID}}_1$. That is, $\mathrm{ID}_{0,i} = \mathrm{ID}_{1,i} = \mathrm{ID}_i$ for $i = 1, \ldots, \ell$. (Note that ℓ might be 0.) The game generates a master key SK_0, it defines $\mathrm{SK}_{0,0} := \mathrm{SK}_{1,0} := \mathrm{SK}_0$, and for $i = 1, \ldots, d$ and $b = 0, 1$ generates correct PRF values for internal

nodes associated to $\mathrm{ID}_{b,i}$:[13]

$$\mathrm{SK}_{b,i} := \mathbf{PRF}(\mathrm{SK}_{b,i-1}, \mathrm{ID}_{b,i}) \ . \tag{1}$$

Note that $\mathrm{SK}_{0,i} = \mathrm{SK}_{1,i}$ for $i = 1, \ldots, \ell$ and there are $2d - \ell$ such values in total. For $i = 0, \ldots, d-1$ and $b = 0, 1$ the game also generates punctured keys

$$\mathrm{SK}^*_{b,i} := \begin{cases} \mathbf{Cons}(\mathrm{SK}_{b,i}, \{\mathrm{ID}_{b,i+1}\}) & \text{if } i \neq \ell \ ; \\ \mathbf{Cons}(\mathrm{SK}_{b,i}, \{\mathrm{ID}_{0,i+1}, \mathrm{ID}_{1,i+1}\}) & \text{if } i = \ell \ . \end{cases} \tag{2}$$

Note that $\mathrm{SK}^*_{0,i} = \mathrm{SK}^*_{1,i}$ for $i = 0, \ldots, \ell$ and that there are $2d - \ell - 1$ punctured keys.

The keys $\{\mathrm{SK}^*_{b,i}\}_{i=0}^{d-1}$ and $\{\mathrm{SK}_{b,i}\}_{i=1}^{d}$ for $b = 0, 1$ contain sufficient information to iteratively compute $\mathbf{PRG}(\mathbf{PPRF}(\mathrm{SK}_0, \overline{\mathrm{ID}}))$ for any $\overline{\mathrm{ID}}$. Roughly speaking, we can do this as follows. Given $\overline{\mathrm{ID}} = (\mathrm{ID}_1, \ldots, \mathrm{ID}_d)$, we start with ID_1 and check if ID_1 is one of the level-one challenge identities $\mathrm{ID}_{0,1}$ or $\mathrm{ID}_{1,1}$. If not, then we can use $\mathrm{SK}^*_{0,0} = \mathrm{SK}^*_{1,0}$ to successfully compute the PRF at this point, and continue iteratively until we reach level d, where we return $\mathrm{PK} = \mathbf{PRG}(\mathrm{SK}_d)$. Else we use the hardwired values $\mathrm{SK}_{b,1}$ for the correct value of b according to whether $\mathrm{ID}_1 = \mathrm{ID}_{0,1}$ or $\mathrm{ID}_1 = \mathrm{ID}_{1,1}$ to compute the PRF once. Suppose $b = 0$ at this stage. In the stage, if ID_2 is not one of $\mathrm{ID}_{0,2}$ or $\mathrm{ID}_{1,2}$, then once gain we use the punctured key $\mathrm{SK}^*_{0,1}$ (which is computed from the correct value $\mathrm{SK}_{0,1}$ at $\{\mathrm{ID}_{0,2}, \mathrm{ID}_{1,2}\}$) to compute the PRF and continue iteratively to level n. Otherwise, we use the correct hardwired values $\mathrm{SK}_{0,2}$ to compute the PRF. Case $b = 1$ is dealt with similarly. We continue in this manner for all ID_i until we reach level d, where we return $\mathbf{PRG}(\mathrm{SK}_d)$. Note that the *intermediate* correct values of the PRF at the challenge identities are not really used in this computation as public keys at two leaf nodes need to be computed. Hence we do not need to hardwire these values. Furthermore, at the two leaf nodes, instead of the PRF values we directly hardwire the public keys, i.e, $y_b := \mathbf{PRG}(\mathbf{PPRF}(\mathrm{MSK}_{d-1}, \mathrm{ID}_d))$ for $b = 0, 1$. The details of this procedure are shown in Figure 15.
Game G_1 returns an obfuscation of this circuit as MPK.

We show that games G_0 and G_1 are indistinguishable. Let $\mathcal{A}$ be an adversary that distinguishes G_0 from G_1. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against **Obf** as follows. Algorithm $\mathcal{B}$ picks a bit b at random runs $\mathcal{A}$ until it outputs $(\overline{\mathrm{ID}}_0, \overline{\mathrm{ID}}_1)$. It then samples a P-PRF key SK_0 and computes $\{\mathrm{SK}^*_{b,i-1}\}_{i=1,b=0,1}^{d}$ as per Equations (1) and (2) and the public key values y_0 and y_1. It then forms the functionally equivalent circuits $\mathbf{C}^{\mathrm{gen}}[\mathrm{SK}_0]$ and $\widetilde{\mathbf{C}}^{\mathrm{gen}}[\{\mathrm{SK}^*_{b,i-1}, \mathrm{ID}_{b,i}, y_b\}_{i=1,b=0,1}^{d}]$ as above, pads then to the appropriate size, and queries them to its LR oracle to get an obfuscated circuit MPK. Algorithm $\mathcal{B}$ answers the encryption queries

[13] Formally a node is associated to a vector of identities. Since the challenge identity vectors are fixed, we simply speak of the final identity in this vector to simplify notation.

$\textsc{Circ}.\ \widetilde{\mathbf{C}}^{\mathrm{gen}}[\{\mathrm{SK}^*_{b,i-1}, \mathrm{ID}_{b,i}, y_b\}_{i=1,b=0,1}^{d}](\mathrm{ID}_1, \ldots, \mathrm{ID}_d)$

$\mathrm{SK}^*_0 \leftarrow \mathrm{SK}^*_{0,0}$
for $i = 1, \ldots, d$:
  if $\mathrm{ID}_i \notin \{\mathrm{ID}_{0,i}, \mathrm{ID}_{1,i}\}$ then
    for $j = i, \ldots, d$:
      $\mathrm{SK}^*_j \leftarrow \mathbf{PRF}(\mathrm{SK}^*_{j-1}, \mathrm{ID}_j)$
    return $\mathbf{PRG}(\mathrm{SK}^*_n)$
  if $\mathrm{ID}_i = \mathrm{ID}_{0,i}$
    if $i = d$ then return y_0
    else $\mathrm{SK}^*_i \leftarrow \mathrm{SK}^*_{0,i}$
  if $\mathrm{ID}_i = \mathrm{ID}_{1,i}$
    if $i = d$ then return y_1
    else $\mathrm{SK}^*_i \leftarrow \mathrm{SK}^*_{1,i}$

Fig. 15. Alternative circuit used in the generation of MPK.

of $\mathcal{A}$ on $(\mathrm{M}_0, \mathrm{M}_1)$ by computing a ciphertext using MPK and $(\overline{\mathrm{ID}}_b, \mathrm{M}_b)$. Key-extraction queries on $\overline{\mathrm{ID}}$ are handled using the (un-punctured) key SK_0. When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. It is easily to see that according to the bit chosen by $\mathcal{B}$'s (obfuscation) game, algorithm $\mathcal{A}$ is run either with respect to game G_0 or G_1 and the output of $\mathcal{B}$ matches the output of these games when run with $\mathcal{A}$. Hence

$$\Pr[\mathrm{G}_0^{\mathcal{A}}] - \Pr[\mathrm{G}_1^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

We will now define games $\mathrm{G}_{2,i}$ for $i = 1, \ldots, 2d - \ell$, which gradually replace the iteratively computed PRF values $\mathrm{SK}_{b,i}$ with truly random values. In doing so, we will rely on the security of **PPRF**. (At a single stage we will rely on "double" puncturing.) Starting from the root of the tree, we will replace the level-one values $\mathrm{SK}_{b,1}$ with truly random values. Next, relying on these already replaced values and P-PRF security, we replace the level-two $\mathrm{SK}_{b,2}$ values with random, etc., until all values, including those at the leaf nodes, are replaced by random values.

$\mathrm{G}_{2,i}$ **for** $i = 0, \ldots, \ell$**:** We define $\mathrm{G}_{2,0}$ as G_1. For $i > 0$, game $\mathrm{G}_{2,i}$ modifies $\mathrm{G}_{2,i-1}$ and replaces $\mathrm{SK}_{0,i} = \mathrm{SK}_{1,i}$ with a truly random value. (Note that $\mathrm{ID}_{0,i} = \mathrm{ID}_{1,i}$ for $i = 1, \ldots, \ell$.)

We will use the security of PPRF with puncturing at $\mathrm{ID}_{0,i} = \mathrm{ID}_{1,i}$ to justify these transitions. Let $\mathcal{A}$ be an adversary that distinguishes $\mathrm{G}_{2,i-1}$ from $\mathrm{G}_{2,i}$. We build an adversary $\mathcal{B}$ against **PPRF** as follows. Algorithm $\mathcal{B}$ picks a random bit b and runs $\mathcal{A}$ until it outputs the challenge identity vectors $(\overline{\mathrm{ID}}_0, \overline{\mathrm{ID}}_1)$. Algorithm $\mathcal{B}$ simulates the game implicitly setting $\mathrm{SK}_{0,i-1}$ to the key of its challenger. $\mathcal{B}$ uses its $\textsc{Key}$ oracle to obtain a key SK^*_{i-1} punctured at $\mathrm{ID}_{0,i}$. It also requests a challenge at $\mathrm{ID}_{0,i}$, which it defines as $\mathrm{SK}_{0,i}$ (which is either $\mathbf{PPRF}(\mathrm{SK}_{0,i-1}, \mathrm{ID}_{0,i})$ or random). PRF values for nodes that are descendants of node $\mathrm{ID}_{0,i}$ (i.e., those

at levels $\geq i+1$) can be computed using $\mathrm{SK}_{0,i}$. PRF values for the siblings of $\mathrm{ID}_{0,i}$ (i.e., those at level i) can be computed using the punctured key SK^*_{i-1}. To compute the PRF values for nodes at levels $\leq i-1$ algorithm $\mathcal{B}$ proceeds as follows. It generates random values $\mathrm{SK}_{0,j}$ for $0 \leq j < i-1$ (level 0 corresponds to the root of the tree). It uses these values to compute PRF values for nodes at levels $\leq i-1$, except for node $\mathrm{ID}_{0,i-1}$; the PRF value for this node, according to the rules of game G_{i-1}, is an independent random value which $\mathcal{B}$ implicitly defines as the (un-punctured) PRF key. We will disallow key extraction for identity $\mathrm{ID}_{0,i-1}$, which is compatible with the legitimacy rules of the HIBE adversary.[14] These computed PRF values and the punctured key SK^*_{i-1} allows $\mathcal{B}$ to also compute the *punctured* keys and PRF values at all nodes that are hardwired into the circuit underlying MPK. Algorithm $\mathcal{B}$ generated these and uses them to form MPK. Algorithm $\mathcal{B}$ answers $\mathcal{A}$'s single LR query by encrypting $(\overline{\mathrm{ID}}_b, \mathrm{M}_b)$ under the generated MPK. When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. According to the challenge bit in the P-PRF game, algorithm $\mathcal{B}$ runs $\mathcal{A}$ either in game $\mathrm{G}_{2,i-1}$ or game $\mathrm{G}_{2,i}$. Hence for any $i = 1, \ldots, \ell$ and any PPT $\mathcal{A}$, there is a PPT $\mathcal{B}$ such that

$$\Pr[\mathrm{G}^{\mathcal{A}}_{2,i-1}] - \Pr[\mathrm{G}^{\mathcal{A}}_{2,i}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{PPRF},\mathcal{B}}(\lambda) \ .$$

$\mathrm{G}_{2,\ell+1}$**:** At level $\ell+1$, the identity vectors differ for the first time. This game modifies $\mathrm{G}_{2,\ell}$ and replaces both $\mathrm{SK}_{0,\ell+1}$ and $\mathrm{SK}_{1,\ell+1}$ with truly random values.

We will use puncturing at $\{\mathrm{ID}_{0,\ell+1}, \mathrm{ID}_{1,\ell+1}\}$ to justify this transition. The reduction is similar to the previous game except that a doubly punctured key $\mathrm{SK}^*_{0,\ell} = \mathrm{SK}^*_{1,\ell}$ at $\{\mathrm{ID}_{0,\ell+1}, \mathrm{ID}_{1,\ell+1}\}$ is requested and the challenge P-PRF oracle is queried on $\mathrm{ID}_{0,\ell+1}$ and $\mathrm{ID}_{0,\ell+1}$ to define $\mathrm{SK}_{0,\ell+1}$ and $\mathrm{SK}_{1,\ell+1}$. Algorithm $\mathcal{A}$ is also barred from querying the key at $\mathrm{ID}_{0,\ell} = \mathrm{ID}_{1,\ell}$. Hence for any PPT $\mathcal{A}$ there is a PPT $\mathcal{B}$ such that

$$\Pr[\mathrm{G}^{\mathcal{A}}_{2,\ell}] - \Pr[\mathrm{G}^{\mathcal{A}}_{2,\ell+1}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{PPRF},\mathcal{B}}(\lambda) \ .$$

$\mathrm{G}_{2,\ell+1+j}$ **for** $j = 1, \ldots, d-\ell-1$**:** Game $\mathrm{G}_{2,\ell+1+j}$ modifies $\mathrm{G}_{2,\ell+1+j-1}$ and replaces $\mathrm{SK}_{0,\ell+1+j}$ with a truly random value.

We use puncturing at $\mathrm{ID}_{0,\ell+1+j}$ as before to justify these transitions. Using a reduction similar to above we obtain that for $j = 1, \ldots, d-\ell-1$ and any PPT $\mathcal{A}$ there is a PPT $\mathcal{B}$ such that

$$\Pr[\mathrm{G}^{\mathcal{A}}_{2,\ell+1+j-1}] - \Pr[\mathrm{G}^{\mathcal{A}}_{2,\ell+1+j}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{PPRF},\mathcal{B}}(\lambda) \ .$$

$\mathrm{G}_{2,d+j}$ **for** $j = 1, \ldots, d-\ell-1$**:** These games replace $\mathrm{SK}_{1,\ell+1+j}$ with random values and as before for any PPT $\mathcal{A}$ there is a PPT $\mathcal{B}$ such that

$$\Pr[\mathrm{G}^{\mathcal{A}}_{2,d+j-1}] - \Pr[\mathrm{G}^{\mathcal{A}}_{2,\ell+1+j}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{PPRF},\mathcal{B}}(\lambda) \ .$$

[14] We emphasize, however, that we are not relying on the full legitimacy of HIBE adversary in these games. In particular a key for $\mathrm{ID}_{0,i}$ *can* be extracted by $\mathcal{A}$.

Note that in game $G_{2,2d-\ell}$ all PRF values, includes those at the leaf nodes, are replaced by truly random values. Overall, the above transitions disallow $\mathcal{A}$ to extract keys for any identity that is a prefix of $(ID_{b,1}, \ldots, ID_{b,d-1})$ for $b = 0, 1$.

$G_{3,0}$**:** Instead of hardwiring $\mathbf{PRG}(SK_{0,d})$ in the MPK circuit in Figure 15, this game samples a truly random value $y_{0,d}$, hardwires it to the MPK circuit.

We rely on the security of **PRG** to bound this transition. Let $\mathcal{A}$ be an adversary that distinguishes $G_{2,2d-\ell}$ from $G_{3,0}$. We build an adversary $\mathcal{B}$ against **PRG** as follows. Algorithm $\mathcal{B}$ receives a PRG challenge y. It chooses a bit b at random and runs $\mathcal{A}$ until it outputs $(\overline{ID}_0, \overline{ID}_1)$. Algorithm $\mathcal{B}$ then generates random values for all nodes $ID_{b,i}$ for $i = 1, \ldots, d-1$ and a final random value for node $ID_{1,d}$, which it passes through **PRG** to compute y_1. These values can also be used to generated all information required to form the MPK circuit in Figure 15 Algorithm $\mathcal{B}$ also hardwires $y_0 := y$ and y_1 into the circuit. Therefore, $\mathcal{B}$ implicitly defines the PRF value at $ID_{0,d}$ as the **PRG** seed. It resumes $\mathcal{A}$ on MPK, an obfuscation of this circuit. It answers the key-extraction queries, except the disallowed one at $ID_{0,d}$, using its internally generated values. Algorithm $\mathcal{B}$ answers $\mathcal{A}$'s LR queries by encrypting $(\overline{ID}_b, M_b)$ under the generated MPK. When $\mathcal{A}$ returns a bit b', algorithm $\mathcal{B}$ returns $(b = b')$. According to whether y is a real or fake PRG value, algorithm $\mathcal{B}$ runs $\mathcal{A}$ in an environment identical to either $G_{2,2d-\ell}$ or G_3. Hence

$$\Pr[G^{\mathcal{A}}_{2,2d-\ell}] - \Pr[G^{\mathcal{A}}_{3,0}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

$G_{3,1}$**:** Instead of hardwiring $\mathbf{PRG}(SK_{1,d})$ in the MPK circuit in Figure 15 this game also samples a truly random value $y_{1,d}$, hardwires it to the MPK circuit.

A reduction similar to that for game $G_{3,0}$ shows that for any PPT $\mathcal{A}$ there is a PPT $\mathcal{B}$ such that

$$\Pr[G^{\mathcal{A}}_{3,0}] - \Pr[G^{\mathcal{A}}_{3,1}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{PRG},\mathcal{B}}(\lambda) \ .$$

G_4**:** This game replaces responds to $\mathcal{A}$'s LR queries (M_0, M_1) by always encrypting M_0 for identity $\overline{ID}_0$ under MPK.

Using the fact that **PRG** has a sparse image, with overwhelming probability both $PK_{\overline{ID}_0}$ and $PK_{\overline{ID}_1}$ are not in the image of **PRG**. Thus the circuits $\mathbf{C}^{\mathrm{enc}}[PK_{\overline{ID}_1}, M_1]$ and $\mathbf{C}^{\mathrm{enc}}[PK_{\overline{ID}_0}, M_0]$ always return $\perp$. By iO-security, their obfuscations are thus indistinguishable and a reduction similar to that for the IBE scheme shows that for any PPT $\mathcal{A}$ there is a PPT iO-legitimate $\mathcal{B}$ such that

$$\Pr[G^{\mathcal{A}}_{3,1}] - \Pr[G^{\mathcal{A}}_{4}] \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathbf{Obf},\mathcal{B}}(\lambda) \ .$$

Finally note that G_4 is independent of the bit b and hence the for any (even unbounded) adversary $\mathcal{A}$ we have $\Pr[G^{\mathcal{A}}_4] = 0$. The theorem follows. □

Fig. 16. Example illustrating some of the set from $\mathcal{S}_{\langle 0\rangle}, \ldots, \mathcal{S}_{\langle 1000\rangle} = \mathcal{S}_{27}$ where $n = 3$.

E A Low-Weight Sequence of Sets

We describe the pebbling sets from [FKPR14] that we use in the proof of Theorem 5. For ease of notation we will work with ternary numbers, which we represent as strings of digits from $\{0, 1, 2\}$ within angular brackets $\langle\cdot\rangle$. We denote repetition of digits as $0^n = 0 \ldots 0$ (n times). Addition will also be in ternary, e.g., $\langle 202\rangle + \langle 1\rangle = \langle 210\rangle$.

Let $N = 2^n$ be a power of 2. In the proof of Theorem 5 we will construct sets $\mathcal{S}_{\langle 0\rangle}, \ldots, \mathcal{S}_{\langle 1\,0^n\rangle} \subset \{0, \ldots, N\}$. These sets will define the positions in the path to the challenge where we replace real values by random ones. The following definition measures how "close" sets (that differ in one element) are.

Neighboring sets. Let $k > 0$. Two distinct sets $\mathcal{S}, \mathcal{S}' \subset \{0, \ldots, N\}$ are called k-neighboring if

1. $\mathcal{S} \triangle \mathcal{S}' := (\mathcal{S} \cup \mathcal{S}') \setminus (\mathcal{S} \cap \mathcal{S}') = \{j\}$ for some $j \in \{0, \ldots, N\}$, i.e., they differ in exactly one element j.
2. $j - k \in \mathcal{S}$.
3. $\forall\, i \in [k-1] : \; j - i \notin \mathcal{S}$.

We define the first set (with index $0 = \langle 0\rangle$) and the last set (with index $3^n = \langle 10^n\rangle$) as

$$\mathcal{S}_{\langle 0\rangle} := \{0\} \quad \text{and} \quad \mathcal{S}_{\langle 1\,0^n\rangle} := \{0, N\}\ .$$

These will correspond to the real game, where only the root (at depth '0') is random, and the random game, where the value at depth N is random too. The remaining (intermediate) sets are defined recursively as follows. For $\ell = 0, \ldots, n$, we define the ℓ-th level of sets to be all the sets of the form $\mathcal{S}_{\langle ?\,0^{n-\ell}\rangle}$ (i.e., whose index in ternary ends with $(n - \ell)$ zeros). Thus, $\mathcal{S}_{\langle 0\rangle}$ and $\mathcal{S}_{\langle 1\,0^n\rangle}$ are the (only) level-0 sets.

Let $\mathcal{S}_I, \mathcal{S}_{I'}$ be two consecutive level-ℓ sets, by which we mean that $I' = I + \langle 1\, 0^{n-\ell} \rangle$. By construction, these sets will differ in exactly one element $\{j\}$ (i.e., $\mathcal{S}_I \neq \mathcal{S}_{I'}$; and $\mathcal{S}_I \cup \{j\} = \mathcal{S}_{I'}$ or $\mathcal{S}_{I'} \cup \{j\} = \mathcal{S}_I$). Then the two level-$(\ell+1)$ sets between the level-ℓ sets $\mathcal{S}_I, \mathcal{S}_{I'}$ are defined as

$$\mathcal{S}_{I+\langle 1\, 0^{n-(\ell+1)} \rangle} := \mathcal{S}_I \cup \{j - \tfrac{N}{2^{\ell+1}}\} \quad \text{and} \quad \mathcal{S}_{I'-\langle 1\, 0^{n-(\ell+1)} \rangle} := \mathcal{S}_{I'} \cup \{j - \tfrac{N}{2^{\ell+1}}\} \ .$$

A concrete example for $N = 2^n = 2^3 = 8$ is illustrated in Figure 16 (where the $\mathcal{S}_I$ is defined by the dark nodes).

An important fact we will use is that consecutive level-ℓ sets are $(N/2^\ell)$-neighboring; in particular, consecutive level-n sets (the 4 lines in the box in Figure 16 illustrate 4 consecutive sets) are thus 1-neighboring, i.e.,

$$\forall\, I \in \{\langle 0 \rangle, \ldots, \langle 2^n \rangle\} :\ \mathcal{S}_I \triangle \mathcal{S}_{I+\langle 1 \rangle} = \{j\} \quad \text{and} \quad j - 1 \in \mathcal{S}_I \ ,$$

meaning they satisfy the desired property.