Adaptively Secure Puncturable Pseudorandom Functions in the Standard Model

Susan Hohenberger*
Johns Hopkins University
susan@cs.jhu.edu

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

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

November 26, 2015

Abstract

We study the adaptive security of constrained PRFs in the standard model. We initiate our exploration with puncturable PRFs. A puncturable PRF family is a special class of constrained PRFs, where the constrained key is associated with an element x' in the input domain. The key allows evaluation at all points $x \neq x'$.

We show how to build puncturable PRFs with adaptive security proofs in the standard model that involve only polynomial loss to the underlying assumptions. Prior work had either super-polynomial loss or applied the random oracle heuristic. Our construction uses indistinguishability obfuscation and DDH-hard algebraic groups of composite order.

More generally, one can consider a t-puncturable PRF: PRFs that can be punctured at any set of inputs S, provided the size of S is less than a fixed polynomial. We additionally show how to transform any (single) puncturable PRF family to a t-puncturable PRF family, using indistinguishability obfuscation.

*Supported by the National Science Foundation (NSF) CNS-1154035 and CNS-1228443; the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract FA8750-11-C-0080, the Office of Naval Research under contract N00014-14-1-0333, and a Microsoft Faculty Fellowship.

†Supported by NSF CNS-1228599 and CNS-1414082, DARPA SafeWare, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

1 Introduction

Pseudorandom functions (PRFs) are one of the fundamental building blocks in modern cryptography. A PRF system consists of a keyed function F and a set of keys $\mathcal{K}$ such that for a randomly chosen key $k \in \mathcal{K}$, the output of the function $F(k, x)$ for any input x in the input space "looks" random to a computationally bounded adversary, even when given polynomially many evaluations of $F(k, \cdot)$. Recently, the concept of *constrained pseudorandom functions*[1] was proposed in the concurrent works of Boneh and Waters [4], Boyle, Goldwasser and Ivan [6] and Kiayias, Papadopoulos, Triandopoulos and Zacharias [21]. A constrained PRF system is associated with a family of boolean functions $\mathcal{F} = \{f\}$. As in standard PRFs, there exists a set of *master keys* $\mathcal{K}$ that can be used to evaluate the PRF F. However, given a master key k, it is also possible to derive a *constrained* key k_f associated with a function $f \in \mathcal{F}$. This constrained key k_f can be used to evaluate the function $F(k, \cdot)$ at all inputs x such that $f(x) = 1$. Intuitively, we would want that even if an adversary has k_f, the PRF evaluation at an input x not accepted by f looks random. Security is captured by an *adaptive* game between a PRF challenger and an adversary. The adversary is allowed to make multiple constrained key or point evaluation queries before committing to a challenge x^* not equal to any of the evaluation queries or accepted by any of the functions for which he obtained a constrained key. [2] The challenger either sends the PRF evaluation at x^* or an output chosen uniformly at random from the PRF range space, and the adversary wins if he can distinguish between these two cases.

Since their inception, constrained PRFs have found multiple applications. For example, Boneh and Waters [4] gave applications of broadcast encryption with optimal ciphertext length, identity-based key exchange, and policy-based key distribution. Sahai and Waters [24] used constrained PRFs as a central ingredient in their punctured programming methodology for building cryptosystems using indistinguishable obfuscation. Boneh and Zhandry [5] likewise applied constrained PRFs for realizing multi-party key exchange and broadcast systems.

Adaptive Security in Constrained PRFs In their initial work, Boneh and Waters [4] showed constructions of constrained PRFs for different function families, including one for the class of all polynomial circuits (based on multilinear maps). However, all their constructions offer *selective security* - a weaker notion where the adversary must commit to the challenge input x^* *before* making any evaluation/constrained key queries.[3] Using complexity leveraging, one can obtain adaptive security by guessing the challenge input x^* before any queries are made. However, this results in exponential security loss. The works of [6, 21] similarly dealt with selective security.

Recently, Fuchsbauer, Konstantinov, Pietrzak and Rao [11] showed adaptive security for *prefix-fixing* constrained PRFs, but with quasi-polynomial security loss. Also recently, Hofheinz [16] presented a novel construction that achieves adaptive security for *bit-fixing* constrained PRFs, but in the *random oracle model.*

While selective security has been sufficient for some applications of constrained PRFs, including many recent proofs leveraging the punctured programming [24] methodology (e.g., [24, 19, 5, 2]), there are applications that demand adaptive security, where the security game allows the adversary to query the PRF on many inputs before deciding on the point to puncture. For instance, [5] give

[1]These were alternatively called *functional PRFs* [6] and *delegatable PRFs* [21].

[2]This definition can be extended to handle multiple challenge points. See Section 3 for details.

[3]The prefix construction of [6] and [21] were also selective.

a construction for multiparty key exchange that is semi-statically secure, and this construction requires adaptively secure constrained PRFs for circuits. We anticipate that the further realization of adaptively secure PRFs will introduce further applications of them.

Our Objective and Results Our goal is to study adaptive security of constrained PRFs in the standard model. We initiate this exploration with *puncturable PRFs*, first explicitly introduced in [24] as a specialization of constrained PRFs. A puncturable PRF family is a special class of constrained PRFs, where the constrained key is associated with an element x' in the input domain. The key allows evaluation at all points $x \neq x'$. As noted by [4, 6, 21], the GGM tree-based construction of PRFs from one-way functions (OWFs) [14] can be modified to construct a puncturable PRF. [4] A selective proof of security follows via a hybrid argument, where the reduction algorithm uses the pre-determined challenge query x^* to "plant" its OWF challenge. However, such a technique does not seem powerful enough to obtain adaptive security with only a polynomial-factor security loss. The difficulty in proving adaptive security arises due to the fact that the reduction algorithm must respond to the evaluation queries, and then output a punctured key that is consistent with the evaluations. This means that the reduction algorithm must be able to evaluate the PRF at a large set S (so that all evaluation queries lie in S with non-negligible probability). However, S cannot be very large, otherwise the challenge x^* will lie in S, in which case the reduction algorithm cannot use the adversary's output.

In this work, we show new techniques for constructing adaptively-secure puncturable PRFs in the standard model. A central contribution is to overcome the conflict above, by allowing the reduction algorithm to commit to the evaluation queries, and at the same time, ensuring that the PRF output at the challenge point is unencumbered by the commitment.

Our main idea is to execute a delayed commitment to part of the PRF by partitioning. Initially, in our construction all points are tied to a single (Naor-Reingold [23] style) PRF. To prove security we begin by using the admissible hash function of Boneh and Boyen [3]. We partition the inputs into two distinct sets. The *evaluable set* which contains about $(1 - 1/q)$ fraction of inputs, and a *challenge set* which contains about $1/q$ fraction of inputs, where q is the number of point evaluation queries made by the attacker. Via a set of hybrid steps using the computational assumptions of indistinguishability obfuscation and subgroup hiding we modify the construction such that we use one Naor-Reingold PRF function to evaluate points in the evaluable set and a completely independent Naor-Reingold PRF to evaluate points in the challenge set.

After this separation has been achieved, there is a clearer path for our proof of security. At this point the reduction algorithm will create one PRF itself and use it to answer any attacker point query in the evaluable set. If it is asked for a point x in the challenge set, it will simply abort. (The admissible hash function ensures that we get through without abort with some non-negligible probability.) Eventually, the attacker will ask for a punctured key on x^*, which defines x^* as the challenge input. Up until this point the reduction algorithm has made no commitments on what the second challenge PRF is. It then constructs the punctured key using the a freshly chosen PRF for the challenge inputs. However, when constructing this second PRF it now knows what the challenge x^* actually is and can fall back on selective techniques for completing the proof.

At a lower level our core PRF will be the Naor-Reingold PRF [23], but based in composite-order groups. Let $\mathbb{G}$ be a group of order $N = pq$, where p and q are primes. The master key consists of a group element $v \in \mathbb{G}$ and $2n$ exponents $d_{i,b} \in \mathbb{Z}_N$ (for $i = 1$ to n and $b \in \{0,1\}$).

[4] In fact, the GGM PRF construction can be used to construct prefix-fixing constrained PRFs.

The PRF F takes as input a key $k = (v, \{d_{i,b}\})$, an ℓ-bit input x, uses a public admissible hash function $h : \{0,1\}^\ell \rightarrow \{0,1\}^n$ to compute $h(x) = b_1 \dots b_n$ and outputs $v^{\prod_{j=1}^{n} d_{j,b_j}}$. A punctured key corresponding to x' derived from master key k is the obfuscation of a program P which has k, x' hardwired and outputs $F(k,x)$ on input $x \neq x'$, else it outputs $\perp$.

We will use a parameterized problem (in composite groups) to perform some of the separation step. Our assumption is that given $g, g^a, \dots, g^{a^{n-1}}$ for randomly chosen $g \in \mathbb{G}$ and $a \in \mathbb{Z}_N^*$ it is hard to distinguish g^{a^n} from a random group element. While it is somewhat undesirable to base security on a parameterized assumption, we are able to use the recent results of Chase and Meiklejohn [8] to reduce this to the subgroup decision problem in DDH hard composite order groups.

t-puncturable PRFs We also show how to construct t-puncturable PRFs: PRFs that can be punctured at any set of inputs S, provided $|S| \leq t$ (where $t(\cdot)$ is a fixed polynomial). We show how to transform any (single) puncturable PRF family to a t-puncturable PRF family, using indistinguishability obfuscation. In the security game for t-puncturable PRFs, the adversary is allowed to query for multiple t-punctured keys, each corresponding to a set S of size at most t. Finally, the adversary sends a challenge input x^* that lies in all the sets queried, and receives either the PRF evaluation at x^* or a uniformly random element of the range space.

In the construction, the setup and evaluation algorithm for the t-puncturable PRF are the same as those for the puncturable PRF. In order to puncture a key k at set S, the puncturing algorithm outputs the obfuscation of a program P that takes as input x, checks that $x \notin S$, and outputs $F(k,x)$.

For the proof of security, we observe that when the first t-punctured key query S_1 is made by the adversary, the challenger can guess the challenge $\tilde{x} \in S_1$. If this guess is incorrect, then the challenger simply aborts (which results in a $1/t$ factor security loss). However, if the guess is correct, then the challenger can now use the punctured key $K_{\tilde{x}}$ for all future evaluation/t-punctured key queries. From the security of puncturable PRFs, it follows that even after receiving evaluation/t-punctured key queries, the challenger will not be able to distinguish between $F(k, \tilde{x})$ and a random element in the range space.

We detail this transformation and its proof in Section 5.1. We also believe that we can use a similar approach to directly modify our main construction to handle multiple punctured points, however, we choose to focus on the generic transformation.

Related Works Two recent works have explored the problem of adaptive security of constrained PRFs. Fuchsbauer, Konstantinov, Pietrzak and Rao [11] study the adaptive security of the GGM construction for prefix-free constrained PRFs. They show an interesting reduction to OWFs that suffers only a quasi-polynomial factor $q^{O(\log n)}$ loss, where q is the number of queries made by the adversary, and n is the length of the input. This beats the straightforward conversion from selective to adaptive security, which results in $O(2^n)$ security loss.

Hofheinz [16] shows a construction for bit-fixing constrained PRFs that is adaptively secure, assuming indistinguishability obfuscation and multilinear maps in the *random oracle model.* It also makes novel use of the random oracle for dynamically defining the challenge space based on the output of h. It is currently unclear whether such ideas could be adapted to the standard model.

Fuchsbauer et al. also show a negative result for the Boneh-Waters [4] construction of bit-fixing constrained PRFs. They show that any *simple* reduction from a static assumption to the adaptive security of the Boneh-Waters [4] bit-fixing constrained PRF construction must have an exponential factor security loss. More abstractly, using their techniques, one can show that any bit-fixing scheme that has the following properties will face this obstacle: (a) *fingerprinting queries* - By querying for a set of constrained keys, the adversary can obtain a *fingerprint* of the master key. (b) *checkability* - It is possible to efficiently check that any future evaluation/constrained key queries are consistent with the fingerprint. While these properties capture certain constructions, small perturbations to them could potentially circumvent checkability.

Partitioning type proofs have been used in several applications including identity-based encryption [3, 25, 1, 17], verifiable random functions [20], and proofs of certain signature signature schemes [9, 18, 19]. We believe ours is the first to use partitioning for a delayed commitment to parameters. We note that our delayed technique is someway reminiscent to that of Lewko and Waters [22].

Recently, there has been a push to prove security for indistinguishability obfuscation from basic multilinear map assumptions. The recent work of Gentry, Lewko, Sahai and Waters [13] is a step in this direction, but itself requires the use of complexity leveraging. In the future work, we might hope for such reductions with just polynomial loss — perhaps for special cases of functionality. And thus give an end-to-end polynomial loss proof of puncturable PRFs from multilinear maps assumptions.

Two works have explored the notion of constrained verifiable random functions (VRFs). Fuchsbauer [10] and Chandran, Raghuraman and Vinayagamurthy [7] show constructions of selectively secure constrained VRFs for the class of all polynomial sized circuits. The construction in [7] is also delegatable.

Future Directions A natural question is to construct adaptively-secure constrained PRFs for larger classes of functions in the standard model. Given the existing results of [11] and [16], both directions seem possible. While the techniques of [16] are intricately tied to the random oracle model, it is plausible there could be constructions in the standard model that evade the negative result of [11]. On the other hand, maybe the negative result of [11] (which is specific to the [4] construction) can be extended to show a similar lower bound for all constructions of constrained PRFs with respect to function family $\mathcal{F}$.

2 Preliminaries

First, we recall the notion of *admissible hash functions* due to Boneh and Boyen [3]. Here we state a simplified definition from [19]. Informally, an admissible hash function family is a function h with a 'partition sampling algorithm' $\mathsf{AdmSample}$. This algorithm takes as input a parameter Q and outputs a 'random' partition of the outputs domain, where one of the partitions has $1/Q$ fraction of the points. Also, this partitioning has special structure which we will use in our proof.

Definition 2.1 *Let l, n and θ be efficiently computable univariate polynomials, $h : \{0,1\}^{l(\lambda)} \to \{0,1\}^{n(\lambda)}$ an efficiently computable function and $\mathsf{AdmSample}$ a PPT algorithm that takes as input 1^λ and an integer Q, and outputs a string $u \in \{0,1,\perp\}^{n(\lambda)}$. For any $u \in \{0,1,\perp\}^{n(\lambda)}$, define $P_u : \{0,1\}^{l(\lambda)} \to \{0,1\}$ as follows: $P_u(x) = 0$ if for all $1 \leq j \leq n(\lambda), h(x)_j \neq u_j$, else $P_u(x) = 1$.*

We say that $(h, \mathsf{AdmSample})$ is θ-admissible if the following condition holds:

For any efficiently computable polynomial Q, $\forall\ x_1, \ldots, x_{Q(\lambda)}, x^* \in \{0,1\}^{l(\lambda)}$, where $x^* \notin \{x_i\}_i$,

$$Pr[(\forall i \leq Q(\lambda), P_u(x_i) = 1) \wedge P_u(x^*) = 0] \geq \frac{1}{\theta(Q(\lambda))}$$

where the probability is taken over $u \leftarrow \mathsf{AdmSample}(1^\lambda, Q(\lambda))$.

Theorem 2.1 (Admissible Hash Function Family [3], simplified proof in [9]) *For any efficiently computable polynomial* l*, there exist efficiently computable polynomials* n, θ *such that there exist* θ*-admissible function families mapping* l *bits to* n *bits.*

Note that the above theorem is information theoretic, and is not based on any cryptographic assumptions.

Next, we recall the definition of indistinguishability obfuscation from [12, 24]. Let PPT denote probabilistic polynomial time.

Definition 2.2 *(Indistinguishability Obfuscation) A uniform PPT machine* $i\mathcal{O}$ *is called an indistinguishability obfuscator for a circuit class* $\{\mathcal{C}_\lambda\}$ *if it satisfies the following conditions:*

- *(Preserving Functionality) For all security parameters* $\lambda \in \mathbb{N}$*, for all* $C \in \mathcal{C}_\lambda$*, for all inputs* x*, we have that* $C'(x) = C(x)$ *where* $C' \leftarrow i\mathcal{O}(\lambda, C)$.
- *(Indistinguishability of Obfuscation) For any (not necessarily uniform) PPT distinguisher* $\mathcal{B} = (Samp, \mathcal{D})$*, there exists a negligible function* $negl(\cdot)$ *such that the following holds: if for all security parameters* $\lambda \in \mathbb{N}, \Pr[\forall x, C_0(x) = C_1(x) : (C_0; C_1; \sigma) \leftarrow Samp(1^\lambda)] > 1 - negl(\lambda)$*, then*

$$|\Pr[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 : (C_0; C_1; \sigma) \leftarrow Samp(1^\lambda)] - \\ \Pr[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 : (C_0; C_1; \sigma) \leftarrow Samp(1^\lambda)]| \leq negl(\lambda).$$

In a recent work, [12] showed how indistinguishability obfuscators can be constructed for the circuit class *P/poly*. We remark that $(Samp, \mathcal{D})$ are two algorithms that pass state, which can be viewed equivalently as a single stateful algorithm $\mathcal{B}$. In our proofs we employ the latter approach, although here we state the definition as it appears in prior work.

2.1 Assumptions

Let $\mathcal{G}$ be a PPT group generator algorithm that takes as input the security parameter 1^λ and outputs $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2)$ where $p, q \in \Theta(2^\lambda)$ are primes, $N = pq$, $\mathbb{G}$ is a group of order N, $\mathbb{G}_p$ and $\mathbb{G}_q$ are subgroups of $\mathbb{G}$ of order p and q respectively, and g_1 and g_2 are generators of $\mathbb{G}_p$ and $\mathbb{G}_q$ respectively.

Assumption 1 (Subgroup Hiding for Composite Order DDH-Hard Groups) *Let* $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$ *and* $b \leftarrow \{0,1\}$*. Let* $T \leftarrow \mathbb{G}$ *if* $b = 0$*, else* $T \leftarrow \mathbb{G}_p$*. The advantage of algorithm* $\mathcal{A}$ *in solving Assumption 1 is defined as*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{SGH}} = \left| Pr[b \leftarrow \mathcal{A}(N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, T)] - \frac{1}{2} \right|$$

We say that Assumption 1 holds if for all PPT $\mathcal{A}$*,* $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{SGH}}$ *is negligible in* λ.

Note that the adversary $\mathcal{A}$ gets generators for both subgroups $\mathbb{G}_p$ and $\mathbb{G}_q$. This is in contrast to bilinear groups, where, if given generators for both subgroups, the adversary can use the pairing to distinguish a random group element from a random subgroup element.

Analogously, we assume that no PPT adversary can distinguish between a random element of $\mathbb{G}$ and a random element of $\mathbb{G}_q$ with non-negligible advantage. This is essentially Assumption 1, where prime q is chosen instead of p, and $\mathbb{G}_q$ is chosen instead of $\mathbb{G}_p$.

Assumption 2 *This assumption is parameterized with an integer $n \in \mathbb{Z}$. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$, $g \leftarrow \mathbb{G}$, $a \leftarrow \mathbb{Z}_N^*$ and $b \leftarrow \{0,1\}$. Let $D = (N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, g, g^a, \dots, g^{a^{n-1}})$. Let $T = g^{a^n}$ if $b = 0$, else $T \leftarrow \mathbb{G}$. The advantage of algorithm $\mathcal{A}$ in solving Assumption 2 is defined as*

$$\mathsf{Adv}_{\mathcal{A}} = \left| Pr[b \leftarrow \mathcal{A}(D, T)] - \frac{1}{2} \right|$$

We say that Assumption 2 holds if for all PPT $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}$ is negligible in λ.

We will use Assumption 2 for clarity in certain parts of our proof, but we do not give it a name because it is implied by other named assumptions. First, Assumption 2 is implied by the n-Power Decisional Diffie-Hellman Assumption [15]. Second, it is also implied by the non-parameterized Assumption 1. The recent results of Chase and Meiklejohn [8] essentially show this latter implication, but that work focuses on the target groups of bilinear maps, whereas our algebraic focus does not involve bilinear maps.

3 Constrained Pseudorandom Functions

In this section, we define the syntax and security properties of a constrained pseudorandom function family. This definition is similar to the one in Boneh-Waters [4], except that the keys are constrained with respect to a circuit family instead of a set system.

Let $\mathcal{K}$ denote the key space, $\mathcal{X}$ the input domain and $\mathcal{Y}$ the range space. The PRF is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that can be computed by a deterministic polynomial time algorithm. We will assume there is a Setup algorithm $F.\mathsf{setup}$ that takes the security parameter λ as input and outputs a random secret key $k \in \mathcal{K}$.

A PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is said to be *constrained* with respect to a circuit family $\mathcal{C}$ if there is an additional key space $\mathcal{K}_c$, and three algorithms $F.\mathsf{setup}$, $F.\mathsf{constrain}$ and $F.\mathsf{eval}$ as follows:

- $F.\mathsf{setup}(1^\lambda)$ is a PPT algorithm that takes the security parameter λ as input and outputs a description of the key space $\mathcal{K}$, the constrained key space $\mathcal{K}_c$ and the PRF F.
- $F.\mathsf{constrain}(k, C)$ is a PPT algorithm that takes as input a PRF key $k \in \mathcal{K}$ and a circuit $C \in \mathcal{C}$ and outputs a constrained key $k_C \in \mathcal{K}_c$.
- $F.\mathsf{eval}(k_C, x)$ is a deterministic polynomial time algorithm that takes as input a constrained key $k_C \in \mathcal{K}_c$ and $x \in \mathcal{X}$ and outputs an element $y \in \mathcal{Y}$. Let k_C be the output of $F.\mathsf{constrain}(k, C)$. For correctness, we require the following:

$$F.\mathsf{eval}(k_C, x) = \begin{cases} F(k, x) & \text{if } C(x) = 1 \\ \perp & \text{otherwise} \end{cases}$$

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

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

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

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

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

Guess A outputs a guess b' of b.

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

Definition 3.1 *The PRF F is a secure constrained PRF with respect to $\mathcal{C}$ if for all PPT adversaries A $\mathsf{Adv}^F_{\mathcal{A}}(\lambda)$ is negligible in λ.*

3.1 Puncturable Pseudorandom Functions

In this section, we define the syntax and security properties of a puncturable pseudorandom function family. Puncturable PRFs are a special class of constrained pseudorandom functions.

A PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a puncturable pseudorandom function if there is an additional key space $\mathcal{K}_p$ and three polynomial time algorithms $F.\mathsf{setup}$, $F.\mathsf{eval}$ and $F.\mathsf{puncture}$ as follows:

- $F.\mathsf{setup}(1^\lambda)$ is a randomized algorithm that takes the security parameter λ as input and outputs a description of the key space $\mathcal{K}$, the punctured key space $\mathcal{K}_p$ and the PRF F.

- $F.\mathsf{puncture}(k, x)$ is a randomized algorithm that takes as input a PRF key $k \in \mathcal{K}$ and $x \in \mathcal{X}$, and outputs a key $k_x \in \mathcal{K}_p$.

- $F.\mathsf{eval}(k_x, x')$ is a deterministic algorithm that takes as input a punctured key $k_x \in \mathcal{K}_p$ and $x' \in \mathcal{X}$. Let $k \in \mathcal{K}$, $x \in \mathcal{X}$ and $k_x \leftarrow F.\mathsf{puncture}(k, x)$. For correctness, we need the following property:
$$F.\mathsf{eval}(k_x, x') = \begin{cases} F(k, x') & \text{if } x \neq x' \\ \perp & \text{otherwise} \end{cases}$$

Security of Puncturable PRFs: The security game between the challenger and the adversary A consists of the following four phases.

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

Evaluation Query Phase A queries for polynomially many evaluations. For each evaluation query x, the challenger sends $F(k,x)$ to A.

Challenge Phase A chooses a challenge $x^* \in \mathcal{X}$. The challenger computes $k_{x^*} \leftarrow F.\mathsf{puncture}(k, x^*)$. If $b = 0$, the challenger outputs k_{x^*} and $F(k,x^*)$. Else, the challenger outputs k_{x^*} and $y \leftarrow \mathcal{Y}$ chosen uniformly at random.

Guess A outputs a guess b' of b.

Let $E \subset \mathcal{X}$ be the set of evaluation queries. A wins if $b = b'$ and $x^* \notin E$. The advantage of A is defined to be $\mathsf{Adv}_A^F(\lambda) = Pr[A \text{ wins}]$.

Definition 3.2 *The PRF F is a secure puncturable PRF if for all probabilistic polynomial time adversaries A $\mathsf{Adv}_{\mathcal{A}}^F(\lambda)$ is negligible in λ.*

3.1.1 t-Puncturable Pseudorandom Functions

The notion of puncturable PRFs can be naturally extended to that of t-puncturable PRFs, where it is possible to derive a key punctured at any set S of size at most t. A formal definition of t-puncturable PRFs can be found in Section 5.

4 Construction

We now describe our puncturable PRF family. It consists of the PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ and the three algorithms $F.\mathsf{setup}, F.\mathsf{puncture}$ and $F.\mathsf{eval}$. The input domain is $\mathcal{X} = \{0,1\}^\ell$, where $\ell = \ell(\lambda)$. We define the key space $\mathcal{K}$ and range space $\mathcal{Y}$ as part of the setup algorithm described next.

$\mathbf{F.setup(1^\lambda)}$ $F.\mathsf{setup}$, on input 1^λ, runs $\mathcal{G}$ to compute $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Let n, θ be polynomials such that there exists a θ-admissible hash function h mapping $\ell(\lambda)$ bits to $n(\lambda)$ bits. For simplicity of notation, we will drop the dependence of ℓ and n on λ.

The key space is $\mathcal{K} = \mathbb{G} \times \left(\mathbb{Z}_N^2\right)^n$ and the range is $\mathcal{Y} = \mathbb{G}$. The setup algorithm chooses $v \in \mathbb{G}$, $d_{i,b} \in \mathbb{Z}_N$, for $i = 1$ to n and $b \in \{0,1\}$, and sets $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$.

The PRF F for key k on input x is then computed as follows. Let $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))) \in \mathbb{G} \times \left(\mathbb{Z}_N^2\right)^n$ and $h(x) = (b_1, \ldots, b_n)$, where $b_i \in \{0,1\}$. Then,

$$F(k,x) = v^{\prod_{j=1}^{n} d_{j,b_j}}.$$

$\mathbf{F.puncture(k, x')}$ $F.\mathsf{puncture}$ computes an obfuscation of $\mathsf{PuncturedKey}_{k,x'}$ (defined in Figure 1); that is, $K_{x'} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}_{k,x'})$ where $\mathsf{PuncturedKey}_{k,x'}$ is padded to be of appropriate size. [5]

[5]Looking ahead, in the proof of security, the program $\mathsf{PuncturedKey}_{k,x'}$ will be replaced by $\mathsf{PuncturedKey}'_{V,w,D,u,x'}$, $\mathsf{PuncturedKeyAlt}_{u,k,k',x'}$ and $\mathsf{PuncturedKeyAlt}'_{u,W,E,k,x'}$ in subsequent hybrids. Since this transformation relies on $i\mathcal{O}$ being secure, we need that all programs have same size. Hence, all programs are padded appropriately to ensure that they have the same size.

```
PuncturedKey$_{k,x'}$

Input: $x \in \{0,1\}^\ell$

Constants : The group $\mathbb{G}$
            $k = (v, ((d_{1,0}, d_{1,1}) \ldots (d_{n,0}, d_{n,1}))) \in \mathbb{G} \times \left(\mathbb{Z}_N^2\right)^n$
            $x' \in \{0,1\}^\ell$

  Compute $h(x) = b_1 \ldots b_n \in \{0,1\}^n$.
  if $x = x'$ then
      Output $\perp$.
  else
      Output $v^{\prod_{j=1}^n d_{j,b_j}}$.
  end if
```

Figure 1: Program PuncturedKey

F.eval$(\mathbf{K_{x'}}, \mathbf{x})$ The punctured key $K_{x'}$ is a program that takes an ℓ-bit input. We define

$$F.\mathsf{eval}(K_{x'}, x) = K_{x'}(x).$$

4.1 Proof of Security

We will now prove that our construction is a secure puncturable PRF as defined in Definition 3.2. Specifically, the claim we show is:

Theorem 4.1 (Main Theorem) *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and the Subgroup Hiding Assumption holds for groups output by $\mathcal{G}$, the PRF F defined above, together with algorithms F.setup, F.puncture and F.eval, is a secure punctured pseudorandom function as defined in Definition 3.2.*

Proof: In order to prove this, we define the following sequence of games. Assume the adversary $\mathcal{A}$ makes $Q = Q(\lambda)$ evaluation queries (where $Q(\cdot)$ is a polynomial) before sending the challenge input.

4.1.1 Sequence of Games

We underline the primary changes from one game to the next.

Game 0 This game is the original security game from Definition 3.2 between the challenger and $\mathcal{A}$ instantiated by the construction under analysis. Here the challenger first chooses a random PRF key, then $\mathcal{A}$ makes evaluation queries and finally sends the challenge input. The challenger responds by sending a key punctured at the challenge input, and either a PRF evaluation at the challenged point or a random value.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $v \in \mathbb{G}$, $d_{i,b} \in \mathbb{Z}_N$, for $i = 1$ to n and $b \in \{0,1\}$, and set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$.
2. On any evaluation query $x_i \in \{0,1\}^\ell$, compute $h(x_i) = b_1^i \ldots b_n^i$ and output $v^{\prod_{j=1}^n d_{j,b_j^i}}$.

3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i \; \forall\, i \leq Q$. Compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}_{k,x^*})$ and $h(x^*) = b_1^* \ldots b_n^*$. Let $y_0 = v^{\prod_{j=1}^{n} d_{j,b_j^*}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

Game 1 This game is the same as the previous one, except that we simulate a partitioning game while the adversary operates and if an undesirable partition arises, we abort the game and decide whether or not the adversary "wins" by a coin flip. This partitioning game works as follows: the challenger samples $u \in \{0,1,\perp\}^n$ using $\mathsf{AdmSample}$ and aborts if either there exists an evaluation query x such that $P_u(x) = 0$ or the challenge query x^* satisfies $P_u(x^*) = 1$.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $v \in \mathbb{G}, d_{i,b} \in \mathbb{Z}_N$, for $i = 1$ to n and $b \in \{0,1\}$, and set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$.
   Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, Q)$ and let $S_u = \{x : P_u(x) = 1\}$ (recall $P_u(x) = 0$ if $h(x)_j \neq u_j \; \forall 1 \leq j \leq n$).
2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b_1^i \ldots b_n^i$ and output $v^{\prod_{j=1}^{n} d_{j,b_j^i}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i \; \forall\, i \leq Q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}_{k,x^*})$ and $h(x^*) = b_1^* \ldots b_n^*$. Let $y_0 = v^{\prod_{j=1}^{n} d_{j,b_j^*}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β'. $\mathcal{B}$ performs an *artificial abort* step. If artificial abort occurs, $\mathcal{B}$ chooses a uniformly random bit $\gamma' \leftarrow \{0,1\}$, and $\mathcal{A}$ wins if $\gamma' = 1$.
   If artificial abort does not occur, $\mathcal{A}$ wins if $\beta = \beta'$.

Artificial Abort: The information theoretic guarantee of admissible hash functions states that for all queries $x_1, \ldots, x_Q, x^*$, $\Pr[P_u(x_i) = 1 \text{ for all } i,\; P_u(x^*) = 0] \geq 1/\theta(Q)$. However, this probability (over the choice of u) could be different for different tuples $(x_1, \ldots, x_Q, x^*)$. To handle this, Waters [25] introduced the *artificial abort* technique. At a high level, this technique ensures that the experiment aborts with almost identical probability for all input sequences. This is achieved in the following manner: after the adversary submits its guess, the challenger estimates the abort probability corresponding to this input sequence. The experiment then 'artificially' aborts with an additional probability η. This probability η, which depends on the sequence $(x_1, \ldots, x_Q, x^*)$, is computed as follows.

- First, estimate the abort probability corresponding to $(x_1, \ldots, x_Q, x^*)$. This is performed by choosing $\tilde{u} \leftarrow \mathsf{AdmSample}(1^\lambda, Q)$ repeatedly and checking if $P_{\tilde{u}}(x_i) = 1$ for all i and $P_{\tilde{u}}(x^*) = 0$. Using Chernoff bounds, we can argue that this gives us an accurate estimate τ of the abort probability corresponding to $(x_1, \ldots, x_Q, x^*)$.
- Let $\tau^* = 1 - 1/\theta(Q)$. Abort with probability $(\tau^* - \tau)/(1 - \tau)$.

These two steps ensure that the abort probability is always negligibly close to τ^* (independent of the adversary's queries). The formal analysis for this part is identical to the one in [25], [20].

Game 2 In this game, the challenger modifies the punctured key and outputs an obfuscation of $\mathsf{PuncturedKeyAlt}$ defined in Figure 2. On inputs x such that $P_u(x) = 1$, the altered punctured key uses the same master key k as before. However, if $P_u(x) = 0$, the altered punctured key uses a different master key k' that is randomly chosen from the key space.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $v \in \mathbb{G}, d_{i,b} \in \mathbb{Z}_N$, for $i = 1$ to n and $b \in \{0, 1\}$.
   Set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$.
   Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, Q)$.
2. On any evaluation query $x_i \in \{0, 1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0, 1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b_1^i \ldots b_n^i$ and output $v^{\prod_{j=1}^n d_{j,b_j^i}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i\ \forall\ i \leq Q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0, 1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else choose $\underline{w \in \mathbb{G}, e_{i,b} \in \mathbb{Z}_N, \text{ for } i = 1 \text{ to } n \text{ and } b \in \{0, 1\}}$.
   Set $\underline{k' = (w, ((e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1})))}$.
   $\underline{\text{Compute } K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt}_{u,k,k',x^*})}$ and $h(x^*) = b_1^* \ldots b_n^*$. Let $y_0 = w^{\prod_{j=1}^n e_{j,b_j^*}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0, 1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β'. $\mathcal{B}$ performs an *artificial abort* step. If artificial abort occurs, $\mathcal{B}$ chooses a uniformly random bit $\gamma' \leftarrow \{0, 1\}$, and $\mathcal{A}$ wins if $\gamma' = 1$.
   If artificial abort does not occur, $\mathcal{A}$ wins if $\beta = \beta'$.

```
                    $\mathsf{PuncturedKeyAlt}_{u,k,k',x'}$

Input: $x \in \{0,1\}^\ell$

Constants : The group $\mathbb{G}, k = (v, ((d_{1,0}, d_{1,1}) \ldots (d_{n,0}, d_{n,1}))) \in \mathbb{G} \times (\mathbb{Z}_N^2)^n$
            $\underline{k' = (w, ((e_{1,0}, e_{1,1}) \ldots (e_{n,0}, e_{n,1})))} \in \mathbb{G} \times (\mathbb{Z}_N^2)^n$
            $x' \in \{0,1\}^\ell, u \in \{0,1,\perp\}^n$

  Compute $h(x) = b_1 \ldots b_n$.
  if $x = x'$ then
      Output $\perp$.
  else if $\underline{P_u(x) = 0}$ then
      $\underline{\text{output } w^{\prod_{j=1}^n e_{j,b_j}}}$.
  else
      Output $v^{\prod_{j=1}^n d_{j,b_j}}$.
  end if
```

Figure 2: Program PuncturedKeyAlt

Game 3 In this game, the challenger changes how the master key k' is chosen so that some elements contain an a-factor, for use on inputs x where $P_u(x) = 0$.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $v \in \mathbb{G}, d_{i,b} \in \mathbb{Z}_N$, for $i = 1$ to n and $b \in \{0, 1\}$,

and set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$.
Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, Q)$.

2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
Else compute $h(x_i) = b_1^i \ldots b_n^i$ and output $v^{\prod_j d_{j,b_j^i}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i \ \forall\ i \leq Q$. Check if $P_u(x^*) = 0$.
If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
Else choose $w \leftarrow \mathbb{G}$, $a \leftarrow \mathbb{Z}_N^*$ and $e'_{i,b} \leftarrow \mathbb{Z}_N$.
Let $e_{i,b} = e'_{i,b} \cdot a$ if $h(x^*)_i = b$, else $e_{i,b} = e'_{i,b}$.
Let $k' = (w, ((e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1})))$, $K_{x^*} \leftarrow i\mathcal{O}(\mathsf{PuncturedKeyAlt}_{u,k,k',x^*})$.
Let $h(x^*) = b_1^* \ldots b_n^*$ and $y_0 = w^{\prod_j e_{j,b_j^*}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β'. $\mathcal{B}$ performs an *artificial abort* step. If artificial abort occurs, $\mathcal{B}$ chooses a uniformly random bit $\gamma' \leftarrow \{0,1\}$, and $\mathcal{A}$ wins if $\gamma' = 1$.
If artificial abort does not occur, $\mathcal{A}$ wins if $\beta = \beta'$.

Game 4 This game is the same as the previous one, except that the altered punctured program contains the constants $\{w^{a^i}\}_{i=0}^n$ hardwired. These terms are used to compute the output of the punctured program. The punctured key is an obfuscation of $\mathsf{PuncturedKeyAlt}'$ defined in Figure 3.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $v \in \mathbb{G}$, $d_{i,b} \in \mathbb{Z}_N$, for $i = 1$ to n and $b \in \{0,1\}$, and set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$.
Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, Q)$.
2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
Else compute $h(x_i) = b_1^i \ldots b_n^i$ and output $v^{\prod_j d_{j,b_j^i}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i \ \forall\ i \leq Q$. Check if $P_u(x^*) = 0$.
If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
Else choose $w \leftarrow \mathbb{G}$, $a \leftarrow \mathbb{Z}_N^*$ and $e'_{i,b} \leftarrow \mathbb{Z}_N$.
Let $W = (w, w^a, \ldots, w^{a^{n-1}})$, $E = ((e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1}))$.
Let $K''_{x^*} \leftarrow i\mathcal{O}(\mathsf{PuncturedKeyAlt}'_{u,W,E,k,x^*})$, $h(x^*) = b_1^* \ldots b_n^*$, $y_0 = \left(w^{a^n}\right)^{\prod_j e'_{j,b_j^*}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K''_{x^*}, y_β).
5. $\mathcal{A}$ outputs β'. $\mathcal{B}$ performs an *artificial abort* step. If artificial abort occurs, $\mathcal{B}$ chooses a uniformly random bit $\gamma' \leftarrow \{0,1\}$, and $\mathcal{A}$ wins if $\gamma' = 1$.
If artificial abort does not occur, $\mathcal{A}$ wins if $\beta = \beta'$.

Game 5 In this game, we replace the term w^{a^n} with a random element from $\mathbb{G}$. Hence, both y_0 and y_1 are random elements of $\mathbb{G}$, thereby ensuring that any adversary has zero advantage in this game.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $v \in \mathbb{G}$, $d_{i,b} \in \mathbb{Z}_N$, for $i = 1$ to n and $b \in \{0,1\}$, and set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$.
Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, Q)$.

```
                          PuncturedKeyAlt'_{u,W,E,k,x'}

  Input: $x \in \{0,1\}^\ell$

  Constants : The group $\mathbb{G}, k = (v, ((d_{1,0}, d_{1,1}) \ldots (d_{n,0}, d_{n,1}))) \in \mathbb{G} \times (\mathbb{Z}_N^2)^n$
              $W = (w_0, \ldots, w_{n-1}) \in \mathbb{G}^n, E = ((e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1})) \in (\mathbb{Z}_N^2)^n$
              $x' \in \{0,1\}^\ell, u \in \{0,1,\perp\}^n$

  Compute $h(x) = b_1 \ldots b_n$ and $h(x') = b'_1 \ldots b'_n$. Let $t_{x'}(x) = |\{i : b_i = b'_i\}|$.
  if $x = x'$ then
      Output $\perp$.
  else if $P_u(x) = 0$ then
      output $(w_{t_{x'}(x)})^{\prod_{j=1}^n e'_{j,b_j}}$.
  else
      Output $v^{\prod_{j=1}^n d_{j,b_j}}$.
  end if
```

Figure 3: Program PuncturedKeyAlt'

2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b_1^i \ldots b_n^i$ and output $v^{\prod_{j=1}^n d_{j,b_j^i}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i \ \forall\ i \leq Q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else choose $w \leftarrow \mathbb{G}$, $a \leftarrow \mathbb{Z}_N^*$, and $e'_{i,b} \leftarrow \mathbb{Z}_N$. Let $W = (w, w^a, \ldots, w^{a^{n-1}})$, $E = ((e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1}))$ and $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt}'_{u,W,E,k,x^*})$.
   Let $h(x^*) = b_1^* \ldots b_n^*$. Choose $T \leftarrow \mathbb{G}$ and let $y_0 = (T)^{\prod_{j=1}^n e'_{j,b_j^*}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β'. $\mathcal{B}$ performs an *artificial abort* step. If artificial abort occurs, $\mathcal{B}$ chooses a uniformly random bit $\gamma' \leftarrow \{0,1\}$, and $\mathcal{A}$ wins if $\gamma' = 1$.
   If artificial abort does not occur, $\mathcal{A}$ wins if $\beta = \beta'$.

4.1.2 Adversary's Advantage in these Games

Let $\mathsf{Adv}^i_{\mathcal{A}}$ denote the advantage of adversary $\mathcal{A}$ in Game i. We will now show that if an adversary $\mathcal{A}$ has non-negligible advantage in Game i, then $\mathcal{A}$ has non-negligible advantage in Game $i+1$. Finally, we show that $\mathcal{A}$ has advantage 0 in Game 5.

Claim 1 *For any adversary $\mathcal{A}$, $\mathsf{Adv}^1_{\mathcal{A}} \geq \mathsf{Adv}^0_{\mathcal{A}}/\theta(Q)$.*

Proof: This claim follows from the θ-admissibility of the hash function h. Recall h is θ-admissible if for all $x_1, \ldots, x_q, x^*$, $Pr[\ \forall i, P_u(x_i) = 1 \wedge P_u(x^*) = 0] \geq 1/\theta(Q)$, where the probability is only over the choice of $u \leftarrow \mathsf{AdmSample}(1^\lambda, Q)$. Therefore, if $\mathcal{A}$ wins with advantage ϵ in Game 0, then $\mathcal{A}$ wins with advantage at least $\epsilon/\theta(Q)$ in Game 1. ∎

Claim 2 *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and the Subgroup Hiding Assumption holds, for any PPT adversary $\mathcal{A}$,*

$$\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}} \leq negl(\lambda).$$

Clearly, the two programs in Game 1 and Game 2 are functionally different (they differ on 'challenge partition' inputs x where $P_u(x) = 0$), and therefore the proof of this claim involves multiple intermediate experiments. In the first hybrid experiment, we transform the program such that the program computes the output in a different manner, although the output is the same as in the original program. Next, the constants hardwired in the modified program are modified such that the output changes on all 'challenge partition' inputs (this step uses Assumption 2). Essentially, both programs use a different base for the challenge partition inputs. Next, using Subgroup Hiding Assumption and Chinese Remainder Theorem, even the exponents can be changed for the challenge partition, thereby ensuring that the original program and final program use different PRF keys for the challenge partition. The formal proof can be found in Appendix A.

Claim 3 *For any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^3_{\mathcal{A}} = \mathsf{Adv}^2_{\mathcal{A}}$.*

Proof: Game 2 and Game 3 are identical, except for the manner in which the constants $e_{i,b}$ are chosen. In Game 2, $e_{i,b} \leftarrow \mathbb{Z}_N$, while in Game 3, the challenger first chooses $e'_{i,b} \leftarrow \mathbb{Z}_N$, $a \leftarrow \mathbb{Z}^*_N$, and sets $e_{i,b} = e'_{i,b} \cdot a$ if $h(x)_i = b$, else sets $e_{i,b} = e'_{i,b}$. Since $a \in \mathbb{Z}^*_N$ (and therefore is invertible), $e'_{i,b} \cdot a$ is also a uniformly random element in $\mathbb{Z}_N$ if $e'_{i,b}$ is. Hence the two experiments are identical. ∎

Claim 4 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}}$ is non-negligible in λ, then there exists a PPT distinguisher $\mathcal{B}$ that breaks the security of $i\mathcal{O}$ with advantage non-negligible in λ.*

Proof: Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the security of $i\mathcal{O}$ with advantage ϵ by interacting with $\mathcal{A}$. $\mathcal{B}$ first sets up the parameters, including u and k, and answers the evaluation queries of $\mathcal{A}$ exactly as in steps 1 and 2 of Game 3, which are identical to steps 1 and 2 of Game 4. When $\mathcal{A}$ sends $\mathcal{B}$ a challenge input x^*, $\mathcal{B}$ checks that $P_u(x^*) = 0$ and if not aborts (identical in both games).

Next $\mathcal{B}$ chooses further values to construct the circuits: $w \leftarrow \mathbb{G}$, $a \leftarrow \mathbb{Z}^*_N$ and $e'_{i,b} \leftarrow \mathbb{Z}_N$. Let $e_{i,b} = e'_{i,b} \cdot a$ if $h(x^*)_i = b$, else $e_{i,b} = e'_{i,b}$. Let $k' = (w, ((e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1})))$, $W = (w, w^a, \ldots, w^{a^{n-1}})$ and $E = ((e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1}))$.

$\mathcal{B}$ constructs $C_0 = \mathsf{PuncturedKeyAlt}_{u,k,k',x^*}$, $C_1 = \mathsf{PuncturedKeyAlt}'_{u,W,E,k,x^*}$, and sends C_0, C_1 to the $i\mathcal{O}$ challenger. $\mathcal{B}$ receives $K_{x^*} \leftarrow i\mathcal{O}(C_b)$ from the challenger. It computes $h(x^*) = b^*_1 \ldots b^*_n$, $y_0 = w^{\prod_j e_{j,b^*_j}}$, $y \leftarrow \mathbb{G}$, $\beta \leftarrow \{0,1\}$, sends (K_{x^*}, y_β) to $\mathcal{A}$ and receives β' in response. If $\beta = \beta'$, $\mathcal{B}$ outputs 0, else it outputs 1.

We will now prove that the circuits C_0 and C_1 have identical functionality. Consider any ℓ bit string x, and let $h(x) = b_1 \ldots b_n$. Recall $t_{x^*}(x) = |\{i : b_i = b^*_i\}|$.

For any $x \in \{0,1\}^\ell$ such that $x = x^*$, both circuits output $\perp$.

For any $x \in \{0,1\}^\ell$ such that $x \neq x^*$ and $P_u(x) = 1$, both circuits output $v^{\prod_{j=1}^n d_{j,b_j}}$.

For any $x \in \{0,1\}^\ell$ such that $x \neq x^*$ and $P_u(x) = 0$, we have

$$C_0(x) = \mathsf{PuncturedKeyAlt}_{u,k,k',x^*}(x) = w^{\prod_{j=1}^{n} e_{j,b_j}} = w^{a^{t_{x^*}(x)} \prod_{j=1}^{n} e'_{j,b_j}} = \left(w_{t_{x^*}(x)}\right)^{\prod_{j=1}^{n} e'_{j,b_j}} = \mathsf{PuncturedKeyAlt}'_{u,W,E,k,x^*}(x) = C_1(x).$$

As C_0 and C_1 have identical functionality, $Pr[\mathcal{B}$ wins $] = Pr[\mathcal{A}$ wins in Game 3] - $Pr[\mathcal{A}$ wins in Game 4]. If $\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}} = \epsilon$, then $\mathcal{B}$ wins the $i\mathcal{O}$ security game with advantage ϵ. ■

Claim 5 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^4_{\mathcal{A}} - \mathsf{Adv}^5_{\mathcal{A}}$ is non-negligible in λ, then there exists a PPT adversary $\mathcal{B}$ that breaks Assumption 2 with advantage non-negligible in λ.*

Proof: Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^4_{\mathcal{A}} - \mathsf{Adv}^5_{\mathcal{A}} = \epsilon$, then we can build an adversary that breaks Assumption 2 with advantage ϵ. The games are identical except that Game 5 replaces the term w^{a^n} with a random element of $\mathbb{G}$. On input an Assumption 2 instance $(N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, w, w^a, \ldots, w^{a^{n-1}})$ together with challenge value T (which is either w^{a^n} or a random element in $\mathbb{G}$), use these parameters as in Game 5 with $\mathcal{A}$. If $\mathcal{A}$ guesses it was in Game 4, guess that $T = w^{a^n}$, else guess that T was random. ■

Observation 1 *For any adversary $\mathcal{A}$, $\mathsf{Adv}^5_{\mathcal{A}} = 0$.*

Proof: If the challenger aborts either during the evaluation or challenge phase, then $\mathcal{A}$ has 0 advantage, since $\mathcal{A}$ wins with probability 1/2. If the challenger does not abort during both these phases, then $\mathcal{A}$ receives (K_{x^*}, y_β), and $\mathcal{A}$ must guess β. However, both y_0 and y_1 are uniformly random elements in $\mathbb{G}$, and therefore, $\mathsf{Adv}^5_{\mathcal{A}} = 0$. ■

4.1.3 Conclusion of the Main Proof

Given Claims 1-5 and Observation 1, we can conclude that if $i\mathcal{O}$ is a secure indistinguishability obfuscator and Assumption 1 holds (in Appendix B, we show that Assumption 1 implies Assumption 2), then any PPT adversary $\mathcal{A}$ has negligible advantage in the puncturable PRF security game (i.e., Game 0). ■

5 t-Puncturable PRFs

Let $t(\cdot)$ be a polynomial. A PRF $F_t : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a t-puncturable pseudorandom function if there is an additional key space $\mathcal{K}_p$ and three polynomial time algorithms $F_t.\mathsf{setup}$, $F_t.\mathsf{eval}$ and $F_t.\mathsf{puncture}$ defined as follows.

- $F_t.\mathsf{setup}(1^\lambda)$ is a randomized algorithm that takes the security parameter λ as input and outputs a description of the key space $\mathcal{K}$, the punctured key space $\mathcal{K}_p$ and the PRF F_t.
- $F_t.\mathsf{puncture}(k, S)$ is a randomized algorithm that takes as input a PRF key $k \in \mathcal{K}$ and $S \subset \mathcal{X}$, $|S| \leq t(\lambda)$, and outputs a t-punctured key $K_S \in \mathcal{K}_p$.

- $F_t.\mathsf{eval}(k_S, x')$ is a deterministic algorithm that takes as input a t-punctured key $k_S \in \mathcal{K}_p$ and $x' \in \mathcal{X}$. Let $k \in \mathcal{K}$, $S \subset \mathcal{X}$ and $k_S \leftarrow F_t.\mathsf{puncture}(k, S)$. For correctness, we need the following property:
$$F_t.\mathsf{eval}(k_S, x') = \begin{cases} F_t(k, x') & \text{if } x' \notin S \\ \perp & \text{otherwise} \end{cases}$$

The security game between the challenger and adversary is similar to the security game for puncturable PRFs. However, in this case, the adversary is allowed to make multiple challenge queries (as in the security game for constrained PRFs). The game consists of the following three phases.

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

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

- **Evaluation Query** A sends $x \in \mathcal{X}$, and receives $F_t(k, x)$.
- **Key Query** A sends a set $S \subset \mathcal{X}$, and receives $F_t.\mathsf{puncture}(k, S)$.
- **Challenge Query** A sends $x \in \mathcal{X}$ as a challenge query. If $b = 0$, the challenger outputs $F_t(k, x)$. Else, the challenger outputs a random element $y \leftarrow \mathcal{Y}$.

Guess A outputs a guess b' of b.

Let $x_1, \ldots, x_{q_1} \in \mathcal{X}$ be the evaluation queries, $S_1, \ldots, S_{q_2} \subset \mathcal{X}$ be the t-punctured key queries and $x_1^*, \ldots, x_s^*$ be the challenge queries. A wins if $\forall i \leq q_1, j \leq s,\ x_i \neq x_j^*$, $\forall i \leq q_2, j \leq s,\ x_j^* \in S_i$ and $b' = b$. The advantage of A is defined to be $\mathsf{Adv}_A^{F_t}(\lambda) = Pr[A \text{ wins}]$.

Definition 5.1 *The PRF F_t is a secure t-puncturable PRF if for all PPT adversaries $\mathcal{A}$ $\mathsf{Adv}_{\mathcal{A}}^{F_t}(\lambda)$ is negligible in λ.*

5.1 Construction

In this section, we present our construction of t-puncturable PRFs from puncturable PRFs and indistinguishability obfuscation. Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a puncturable PRF, and $F.\mathsf{setup}$, $F.\mathsf{puncture}$, $F.\mathsf{eval}$ the corresponding setup, puncturing and evaluation algorithms. We now describe our t-puncturable PRF F_t, and the corresponding algorithms $F_t.\mathsf{setup}$, $F_t.\mathsf{puncture}$ and $F_t.\mathsf{eval}$.

$\mathbf{F_t.setup(1^\lambda)}$ $F_t.\mathsf{setup}$ is the same as $F.\mathsf{setup}$.

$\mathbf{F_t.puncture(k, S)}$ $F_t.\mathsf{puncture}(k, S)$ computes an obfuscation of the program $\mathsf{PuncturedKey^t}_{k,S}$ defined in Figure 4; that is, $K_S \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey^t}_{k,S})$. As before, the program $\mathsf{PuncturedKey^t}_{k,S}$ is padded to be of appropriate size.

$\mathbf{F_t.eval(K_S, x)}$ The punctured key K_S is a program that takes an input in $\mathcal{X}$. We define
$$F_t.\mathsf{eval}(K_S, x) = K_S(x).$$

```
$\mathsf{PuncturedKey^t}_{k,S}$

Input: $x \in \mathcal{X}$
Constants : The function description $F, k \in \mathcal{K}, S \subset \mathcal{X}$ such that $|S| \leq t$
  if $x \in S$ then
    Output $\perp$.
  else
    Output $F(k, x)$.
  end if
```

Figure 4: Program $\mathsf{PuncturedKey^t}$

5.2 Proof of Security

We will now prove that the above construction is a secure t-puncturable PRF as defined in Definition 5.1.

Theorem 5.1 *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and F, together with F.setup, F.$\mathsf{puncture}$ and F.eval is a secure puncturable PRF, the PRF F_t defined above, together with F_t.setup, F_t.$\mathsf{puncture}$ and F_t.eval, is a secure t-puncturable PRF.*

For simplicity, we will assume that the adversary makes q_1 evaluation queries, q_2 punctured key queries and 1 challenge query. As shown by [4], this can easily be extended to the general case of multiple challenge queries via a hybrid argument. We will first define the intermediate hybrid experiments.

Game 0 This game is the original security game between the challenger and adversary $\mathcal{A}$, where the challenger first chooses a PRF key, then $\mathcal{A}$ makes evaluation/t-punctured key queries and finally sends the challenge input. The challenger responds with either the PRF evaluation at challenge input, or sends a random element of the range space.

1. Choose a key $k \leftarrow \mathcal{K}$.
2. $\mathcal{A}$ makes evaluation/t-punctured key queries.
   (a) If $\mathcal{A}$ sends an evaluation query x_i, then output $F(k, x_i)$.
   (b) If $\mathcal{A}$ sends a t-punctured key query for set S_j, output the key $K_{S_j} \leftarrow i\mathcal{O}(\mathsf{PuncturedKey^t}_{k,S_j})$.
3. $\mathcal{A}$ sends challenge query x^* such that $x^* \neq x_i \; \forall i \leq q_1$ and $x^* \in S_j \; \forall j \leq q_2$. Choose $\beta \leftarrow \{0, 1\}$. If $\beta = 0$, output $y = F(k, x^*)$, else output $y \leftarrow \mathcal{Y}$.
4. $\mathcal{A}$ sends β' and wins if $\beta = \beta'$.

Game 1 This game is the same as the previous one, except that the challenger introduces an abort condition. When the first t-punctured key query S_1 is made, the challenger guesses the challenge query $\tilde{x} \leftarrow S_1$. The challenger aborts if any of the evaluation queries are $\tilde{x}$, if any of the future t-punctured key queries does not contain $\tilde{x}$ or if the challenge query $x^* \neq \tilde{x}$.

1. Choose a key $k \leftarrow \mathcal{K}$.

2. $\mathcal{A}$ makes evaluation/t-punctured key queries.
   Let S_1 be the first t-punctured key query. Choose $\tilde{x} \leftarrow S_1$ and output key $K_{S_1} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey^t}_{k,S_1})$. For all evaluation queries x_i before S_1, output $F(k, x_i)$.
   For all queries after S_1, do the following.
   (a) If $\mathcal{A}$ sends an evaluation query x_i and $x_i = \tilde{x}$, abort.
       Choose $\gamma_i^1 \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $\gamma_i^1 = 1$.
       Else if $x_i \neq \tilde{x}$, output $F(k, x_i)$.
   (b) If $\mathcal{A}$ sends a t-punctured key query for set S_j and $\tilde{x} \notin S_j$, abort.
       Choose $\gamma_i^2 \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $\gamma_i^2 = 1$.
       Else if $\tilde{x} \in S_j$, output $K_{S_j} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey^t}_{k,S_j})$.
3. $\mathcal{A}$ sends challenge query x^* such that $x^* \neq x_i\ \forall i \leq q_1$ and $x^* \in S_j\ \forall j \leq q_2$.
   If $\tilde{x} \neq x^*$, abort. Choose $\gamma^* \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else if $\tilde{x} = x^*$, choose $\beta \leftarrow \{0,1\}$. If $\beta = 0$, output $y = F(k, x^*)$, else output $y \leftarrow \mathcal{Y}$.
4. $\mathcal{A}$ sends β' and wins if $\beta = \beta'$.

Next, we define q_2 games, Game 1_l, $1 \leq l \leq q_2$. Let Game 1_0 = Game 1.

Game 1_l In this game, the first l punctured key queries use $K_{\tilde{x}}$, while the remaining use k.

1. Choose a key $k \leftarrow \mathcal{K}$.
2. $\mathcal{A}$ makes evaluation/t-punctured key queries.
   Let S_1 be the first t-punctured key query. Choose $\tilde{x} \leftarrow S_1$.
   Compute $K_{\tilde{x}} \leftarrow F.\mathsf{puncture}(k, \tilde{x})$.
   Output $K_{S_1} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt^t}_{K_{\tilde{x}},S_1})$ (where $\mathsf{PuncturedKeyAlt^t}$ is defined in Figure 5).
   For all evaluation queries x_i before S_1, output $F(k, x_i)$.
   For all queries after S_1, do the following.
   (a) If $\mathcal{A}$ sends an evaluation query x_i and $x_i = \tilde{x}$, abort. Choose $\gamma_i^1 \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $\gamma_i^1 = 1$.
       Else if $x_i \neq \tilde{x}$, output $F.\mathsf{eval}(K_{\tilde{x}}, x_i) = F(k, x_i)$.
   (b) If $\mathcal{A}$ sends a t-punctured key query for set S_j and $\tilde{x} \notin S_j$, abort. Choose $\gamma_i^2 \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $\gamma_i^2 = 1$.
       Else if $\tilde{x} \in S_j$ and $j \leq l$, output $K_{S_j} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt^t}_{K_{\tilde{x}},S_j})$.
       Else output $K_{S_j} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey^t}_{k,S_j})$.
3. $\mathcal{A}$ sends challenge query x^* such that $x^* \neq x_i\ \forall i \leq q_1$ and $x^* \in S_j\ \forall j \leq q_2$.
   If $\tilde{x} \neq x^*$, abort. Choose $\gamma^* \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else if $\tilde{x} = x^*$, choose $\beta \leftarrow \{0,1\}$. If $\beta = 0$, output $y = F(k, x^*)$, else output $y \leftarrow \mathcal{Y}$.
4. $\mathcal{A}$ sends β' and wins if $\beta = \beta'$.

Game 2 In this game, the challenger outputs a random element as the response to the challenge query.

1. Choose a key $k \leftarrow \mathcal{K}$.

```
$\mathsf{PuncturedKeyAlt^t}_{K_{\tilde{x}},S}$

Input: $x \in \mathcal{X}$
Constants : The function description $F, K_{\tilde{x}}, S \subset \mathcal{X}$ such that $|S| \le t$
  if $x \in S$ then
      Output $\perp$.
  else
      Output $F.\mathsf{eval}(K_{\tilde{x}}, x)$.
  end if
```

Figure 5: Program $\mathsf{PuncturedKeyAlt^t}$

2. $\mathcal{A}$ makes evaluation/t-punctured key queries.
   Let S_1 be the first t-punctured key query. Choose $\tilde{x} \leftarrow S_1$ and compute $K_{\tilde{x}} \leftarrow F.\mathsf{puncture}(k, \tilde{x})$.
   Output $K_{S_1} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt^t}_{K_{\tilde{x}},S_1})$.
   For all evaluation queries x_i before S_1, output $F(k, x_i)$.
   For all queries after S_1, do the following.
   (a) If $\mathcal{A}$ sends an evaluation query x_i and $x_i = \tilde{x}$, abort. Choose $\gamma_i^1 \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $\gamma_i^1 = 1$.
       Else if $x_i \ne \tilde{x}$, output $F.\mathsf{eval}(K_{\tilde{x}}, x_i) = F(k, x_i)$.
   (b) If $\mathcal{A}$ sends a t-punctured key query for set S_j and $\tilde{x} \notin S_j$, abort. Choose $\gamma_i^2 \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $\gamma_i^2 = 1$.
       Else if $\tilde{x} \in S_j$, output $K_{S_j} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt^t}_{K_{\tilde{x}},S_j})$.
3. $\mathcal{A}$ sends challenge query x^* such that $x^* \ne x_i\ \forall i \le q_1$ and $x^* \in S_j\ \forall j \le q_2$.
   If $\tilde{x} \ne x^*$, abort. Choose $\gamma^* \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else if $\tilde{x} = x^*$, choose $\beta \leftarrow \{0,1\}$ and $\underline{\text{output } y \leftarrow \mathcal{Y}}$.
4. $\mathcal{A}$ sends β' and wins if $\beta = \beta'$.

5.2.1 Adversary's Advantage in these Games

Let $\mathsf{Adv}^i_{\mathcal{A}}$ denote the advantage of adversary $\mathcal{A}$ in Game i.

Observation 2 *For any adversary $\mathcal{A}$, $\mathsf{Adv}^1_{\mathcal{A}} \ge \mathsf{Adv}^0_{\mathcal{A}}/t$.*

Proof: Since one of the elements of S_1 will be the challenge input, and $|S_1| \le t$, the challenger's guess is correct with probability $1/|S_1| \ge 1/t$. Hence, $\mathsf{Adv}^1_{\mathcal{A}} \ge \mathsf{Adv}^0_{\mathcal{A}}/t$. ■

We will now show that Game 1_l and Game 1_{l+1} are computationally indistinguishable, assuming $i\mathcal{O}$ is secure.

Claim 6 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{1_l}_{\mathcal{A}} - \mathsf{Adv}^{1_{l+1}}_{\mathcal{A}}$ is non-negligible in λ, then there exists a PPT distinguisher $\mathcal{B}$ that breaks the security of $i\mathcal{O}$ with advantage non-negligible in λ.*

Proof: Note that the only difference between Game 1_l and Game 1_{l+1} is in the response to the $(l+1)th$ t-punctured key query. In Game 1_l, $\mathsf{PuncturedKey^t}_{k,S_{l+1}}$ is used to compute $K_{S_{l+1}}$, while in Game 1_{l+1}, $\mathsf{PuncturedKeyAlt^t}_{K_{\tilde{x}},S_{l+1}}$ is used. Suppose there exists a PPT adversary $\mathcal{A}$ such that

$\mathsf{Adv}_{\mathcal{A}}^{1_l} - \mathsf{Adv}_{\mathcal{A}}^{1_{l+1}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ and breaks the security of $i\mathcal{O}$ with advantage ϵ.

$\mathcal{B}$ chooses $k \leftarrow \mathcal{K}$ and for all evaluation queries x_i before the first t-punctured key query, outputs $F(k, x_i)$. On receiving the first t-punctured key query S_1, $\mathcal{B}$ chooses $\tilde{x} \leftarrow S_1$ and computes $K_{\tilde{x}} \leftarrow F.\mathsf{puncture}(k, \tilde{x})$. The evaluation queries are computed as in Game 1_l and 1_{l+1}. The first l t-punctured key queries are constructed using k, while the last $q_2 - l - 1$ t-punctured keys are constructed using $K_{\tilde{x}}$ (as in Game 1_l and Game 1_{l+1}). For the $(l+1)th$ query, $\mathcal{B}$ does the following. $\mathcal{B}$ sets $C_0 = \mathsf{PuncturedKey^t}_{k,S_{l+1}}$ and $C_1 = \mathsf{PuncturedKeyAlt^t}_{K_{\tilde{x}},S_{l+1}}$, and sends C_0, C_1 to the $i\mathcal{O}$ challenger, and receives $K_{S_{l+1}}$ in response, which it sends to $\mathcal{A}$.

Finally, after all queries, the challenger sends the challenge query x^*. $\mathcal{B}$ checks that $\tilde{x} = x^*$, sets $y_0 = F(k, x^*)$ and chooses $y_1 \leftarrow \mathcal{Y}, \beta \leftarrow \{0, 1\}$. It outputs y_β and receives β' in response. If $\beta = \beta'$, $\mathcal{B}$ outputs 0, else it outputs 1.

From the correctness property of puncturable PRFs, it follows that $F.\mathsf{eval}(K_{\tilde{x}}, x) = F(k, x)$ for all $x \notin S_{l+1}$. Hence, the circuits C_0 and C_1 are functionally identical. This completes our proof. ■

Next, we show that Game 1_{q_2} and Game 2 are computationally indistinguishable.

Claim 7 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{1_{q_2}} - \mathsf{Adv}_{\mathcal{A}}^{2}$ is non-negligible in λ, then there exists a PPT distinguisher $\mathcal{B}$ that breaks the security of puncturable PRF F with advantage non-negligible in λ.*

Proof: We will use $\mathcal{A}$ to construct a PPT algorithm $\mathcal{B}$ that breaks the security of puncturable PRF F with advantage $\mathsf{Adv}_{\mathcal{A}}^{1_{q_2}} - \mathsf{Adv}_{\mathcal{A}}^{2}$. Observe that in Game 1_{q_2}, the challenger requires the master key k only for the evaluation queries before the first t-punctured key query. After the first t-punctured key query S_1, the challenger chooses $\tilde{x} \leftarrow S_1$, computes a punctured key $K_{\tilde{x}}$, and uses this to compute all future evaluation queries and t-punctured keys.

$\mathcal{B}$ begins interacting with $\mathcal{A}$. For each evaluation query x_i before the first t-punctured key query, $\mathcal{B}$ sends x_i to the puncturable PRF challenger, and receives y_i, which it forwards to $\mathcal{A}$. On receiving the first t-punctured key query S_1, $\mathcal{B}$ chooses $\tilde{x} \leftarrow S_1$ and sends $\tilde{x}$ as challenge input to the puncturable PRF challenger. $\mathcal{B}$ receives $K_{\tilde{x}}$ and y. It uses $K_{\tilde{x}}$ for all remaining queries. On receiving challenge x^* from $\mathcal{A}$, $\mathcal{B}$ checks $x^* = \tilde{x}$ and sends y. $\mathcal{B}$ sends $\mathcal{A}$'s response to the PRF challenger.

Note that until the challenge query is made, both games are identical and $\mathcal{B}$ simulates them perfectly. If y is truly random, then $\mathcal{A}$ receives a response as per Game 2, else it receives a response as per Game 1_{q_2}. ■

Finally, we have the following simple observation.

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

From the above claims and observations, we can conclude that if $i\mathcal{O}$ is a secure indistinguishability obfuscator as per Definition 2.2, and F, together with $F.\mathsf{setup}, F.\mathsf{puncture}, F.\mathsf{eval}$ is a secure puncturable PRF as per Definition 3.2, then any PPT adversary $\mathcal{A}$ has negligible advantage in Game 0.

6 Conclusion

Puncturable and t-puncturable PRFs have numerous cryptographic applications. This work provides the first constructions and proofs of adaptive security in the standard model. This is an interesting step forward in its own right, and we believe the techniques used to achieve adaptiveness from indistinguishability obfuscation may be useful elsewhere. Moreover, this work resolves for at least the puncturable PRF space, the larger question of characterizing which classes of functions admit an adaptively-secure constrained PRF in the standard model. As noted earlier, the results of [11] and [16] provide intuition both for and against whether this is indeed possible for many other function families.

References

[1] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (h)ibe in the standard model. In: Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques. pp. 553–572. EUROCRYPT'10, Springer-Verlag, Berlin, Heidelberg (2010), `http://dx.doi.org/10.1007/978-3-642-13190-5_28`

[2] Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: Indistinguishability obfuscation vs. auxiliary-input extractable functions: One must fall. Cryptology ePrint Archive, Report 2013/641 (2013), `http://eprint.iacr.org/`

[3] Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: CRYPTO. pp. 443–459 (2004)

[4] Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: ASIACRYPT. pp. 280–300 (2013)

[5] Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: CRYPTO (2014)

[6] Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings. pp. 501–519 (2014)

[7] Chandran, N., Raghuraman, S., Vinayagamurthy, D.: Constrained pseudorandom functions: Verifiable and delegatable. Cryptology ePrint Archive, Report 2014/522 (2014), `http://eprint.iacr.org/`

[8] Chase, M., Meiklejohn, S.: Déjà q: Using dual systems to revisit q-type assumptions. In: EUROCRYPT. pp. 622–639 (2014)

[9] Freire, E.S.V., Hofheinz, D., Paterson, K.G., Striecks, C.: Programmable hash functions in the multilinear setting. In: CRYPTO. pp. 513–530 (2013)

[10] Fuchsbauer, G.: Constrained verifiable random functions. In: Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings. pp. 95–114 (2014)

[11] Fuchsbauer, G., Konstantinov, M., Pietrzak, K., Rao, V.: Adaptive security of constrained prfs. In: Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II. pp. 82–101 (2014)

[12] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)

[13] Gentry, C., Lewko, A., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309 (2014), `http://eprint.iacr.org/`

[14] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: FOCS. pp. 464–479 (1984)

[15] Golle, P., Jarecki, S., Mironov, I.: Cryptographic primitives enforcing communication and storage complexity. In: Financial Cryptography. pp. 120–135 (2002)

[16] Hofheinz, D.: Fully secure constrained pseudorandom functions using random oracles. IACR Cryptology ePrint Archive 2014, 372 (2014)

[17] Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. J. Cryptology 25(3), 484–527 (2012)

[18] Hohenberger, S., Sahai, A., Waters, B.: Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In: CRYPTO (2013)

[19] Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In: EUROCRYPT. pp. 201–220 (2014)

[20] Hohenberger, S., Waters, B.: Constructing verifiable random functions with large input spaces. In: EUROCRYPT. pp. 656–672 (2010)

[21] Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: ACM Conference on Computer and Communications Security. pp. 669–684 (2013)

[22] Lewko, A.B., Waters, B.: New proof methods for attribute-based encryption: Achieving full security through selective techniques. In: CRYPTO. pp. 180–198 (2012)

[23] Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. J. ACM 51(2), 231–262 (2004)

[24] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC. pp. 475–484 (2014)

[25] Waters, B.: Efficient identity-based encryption without random oracles. In: EUROCRYPT. pp. 114–127 (2005)

A Proof of Claim 2 ($\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}$ is negligible)

Recall that Claim 2 states that ssuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and the Subgroup Hiding Assumption holds, for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}} \leq \text{negl}(\lambda).$$

In order to prove this, we establish a sequence of intermediate experiments Game 1A to Game 1G and show that any two consecutive experiments are computationally indistinguishable. First, we recall Game 1.

Game 1

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$.
   Choose $v \leftarrow \mathbb{G}$, $d_{i,b} \leftarrow \mathbb{Z}_N$. Set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$.
2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b^i_1 \ldots b^i_n$ and output $v^{\prod_j d_{j,b^i_j}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i$ for all $i \leq q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}_{k,x^*})$ and $h(x^*) = b^*_1 \ldots b^*_n$. Let $y_0 = v^{\prod_j d_{j,b^*_j}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

Game 1A We change the sampling procedure for the $d_{i,b}$ values so that some include a factor of a.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$.
   Choose $v \leftarrow \mathbb{G}$, $\underline{d'_{i,b} \leftarrow \mathbb{Z}_N,\ a \leftarrow \mathbb{Z}^*_N \text{ and set } d_{i,b} = d'_{i,b} \text{ if } u_i = b, \text{ else } d_{i,b} = d'_{i,b} \cdot a.}$ [6].
   Set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$.

2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b^i_1 \ldots b^i_n$ and output $v^{\prod_j d_{j,b^i_j}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i$ for all $i \leq q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}_{k,x^*})$ and $h(x^*) = b^*_1 \ldots b^*_n$. Let $y_0 = v^{\prod_j d_{j,b^*_j}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

[6]If $u_i = \perp$, then $d_{i,b} = d'_{i,b} \cdot a$ for $b = 0, 1$.

Game 1B We substitute an alternative method of computing the punctured key program output using a differently formatted input.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$.
   Let $r_u(x) = |\{j : u_j \neq h(x)_j\}|$. Choose $v \leftarrow \mathbb{G}$, $d'_{i,b} \leftarrow \mathbb{Z}_N$, $a \leftarrow \mathbb{Z}_N^*$ and set $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = d'_{i,b} \cdot a$.
   Let $D = ((d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1})), V = (v, v^a, \ldots, v^{a^{n-1}})$ and $w = v^{a^n}$.
2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$. [7]
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b_1^i \ldots b_n^i$. Output $v^{\prod_j d_{j,b_j^i}} = \left(v^{a^{r_u(x_i)}}\right)^{\prod_j d'_{i,b_j^i}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i$ for all $i \leq q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}'_{V,w,D,u,x^*})$ and $h(x^*) = b_1^* \ldots b_n^*$. Let $y_0 = v^{\prod_j d_{j,b_j^*}} = w^{\prod_j d'_{j,b_j^*}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

```
                    $\mathsf{PuncturedKey}'_{V,w,D,u,x'}$

Input: $x \in \{0,1\}^\ell$

Constants : The group $\mathbb{G}$
            $V = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{G}^n, w \in \mathbb{G}$
            $D = \left((d'_{1,0}, d'_{1,1}) \ldots (d'_{n,0}, d'_{n,1})\right) \in \left(\mathbb{Z}_N^2\right)^n$
            $x' \in \{0,1\}^\ell, u \in \{0,1,\perp\}^n$

  Compute $h(x) = b_1 \ldots b_n$.
  **if** $x = x'$ **then**
      Output $\perp$.
  **else if** $P_u(x) = 0$ **then**
      Output $w^{\prod_j d'_{j,b_j}}$.
  **else**
      Compute $r_u(x)$. Output $\left(v_{r_u(x)}\right)^{\prod_i d'_{j,b_j}}$.
  **end if**
```

Figure 6: Program $\mathsf{PuncturedKey}'$

Game 1C In this game, the term v^{a^n} is replaced by a random element of $\mathbb{G}$.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$. Let $r_u(x) = |\{j : u_j \neq h(x)_j\}|$.
   Choose $v \leftarrow \mathbb{G}$, $d'_{i,b} \leftarrow \mathbb{Z}_N$, $a \leftarrow \mathbb{Z}_N^*$ and set $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = d'_{i,b} \cdot a$.
   Let $D = ((d'_{1,0}, d'_{1,1}) \ldots, (d'_{n,0}, d'_{n,1})), V = (v, v^a, \ldots, v^{a^{n-1}})$ and $w \leftarrow \mathbb{G}$.

[7]Note that $r_u(x) < n$ if $P_u(x) = 1$, else $r_u(x) = n$.

2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b^i_1 \dots b^i_n$. Output $v^{\prod_j d_{j,b^i_j}} = \left(v^{a^{r_u(x_i)}}\right)^{\prod_j d'_{i,b^i_j}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i$ for all $i \leq q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}'_{V,w,D,u,x^*})$ and $h(x^*) = b^*_1 \dots b^*_n$. Let $\underline{y_0 = w^{\prod_j d'_{j,b^*_j}}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

Game 1D This game is same as the previous one, except that v and w are chosen from subgroups $\mathbb{G}_p$ and $\mathbb{G}_q$ respectively, instead of $\mathbb{G}$.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$. Let $r_u(x) = |\{j : u_j \neq h(x)_j\}|$.
   Choose $\underline{v \leftarrow \mathbb{G}_p,\ w \leftarrow \mathbb{G}_q}$, $d'_{i,b} \leftarrow \mathbb{Z}_N$, $a \leftarrow \mathbb{Z}^*_N$ and set $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = d'_{i,b} \cdot a$.
   Let $D = ((d'_{1,0}, d'_{1,1}), \dots, (d'_{n,0}, d'_{n,1}))$, $V = (v, v^a, \dots, v^{a^{n-1}})$.
2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b^i_1 \dots b^i_n$. Output $v^{\prod_j d_{j,b^i_j}} = \left(v^{a^{r_u(x_i)}}\right)^{\prod_j d'_{i,b^i_j}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i$ for all $i \leq q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}'_{V,w,D,u,x^*})$ and $h(x^*) = b^*_1 \dots b^*_n$. Let $y_0 = w^{\prod_j d'_{j,b^*_j}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

Game 1E

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$. Choose $v \leftarrow \mathbb{G}_p$, $w \leftarrow \mathbb{G}_q$, $\underline{d_{i,b} \leftarrow \mathbb{Z}_N\text{. Let } d_{i,b,p} = d_{i,b} \mod p \text{ and } d_{i,b,q} = d_{i,b} \mod q}$.
   $\underline{\text{Set } k = (v, ((d_{1,0,p}, d_{1,1,p}), \dots, (d_{n,0,p}, d_{n,1,p})))}$.
   $\underline{\text{Set } k' = (w, ((d_{1,0,q}, d_{1,1,q}), \dots, (d_{n,0,q}, d_{n,1,q})))}$.
2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b^i_1 \dots b^i_n$. Output $v^{\prod_j d_{j,b^i_j}} \underline{= v^{\prod_j d_{j,b^i_j,p}}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i$ for all $i \leq q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $\underline{K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt}_{u,k,k',x^*})}$ and $h(x^*) = b^*_1 \dots b^*_n$. Let $y_0 = w^{\prod_j d_{j,b^*_j}}$ $\underline{= w^{\prod_j d_{j,b^*_j,q}}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

Game 1F

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$. Choose $v \leftarrow \mathbb{G}_p$, $w \leftarrow \mathbb{G}_q$, $\underline{d_{i,b,p} \leftarrow \mathbb{Z}_p,\ e_{i,b,q} \leftarrow \mathbb{Z}_q}$.
   Set $k = (v, ((d_{1,0,p}, d_{1,1,p}), \ldots, (d_{n,0,p}, d_{n,1,p})))$.
   $\underline{\text{Set } k' = (w, ((e_{1,0,q}, e_{1,1,q}), \ldots, (e_{n,0,q}, d_{n,1,q})))}$.
2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b_1^i \ldots b_n^i$. Output $\underline{= v^{\prod_j d_{j,b_j^i,p}}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i$ for all $i \leq q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt}_{u,k,k',x^*})$ and $h(x^*) = b_1^* \ldots b_n^*$. Let $\underline{y_0 = w^{\prod_j e_{j,b_j^*,q}}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

Game 1G This game is same as the previous one, except that the $d_{i,b}$ and $e_{i,b}$ values are uniformly chosen from $\mathbb{Z}_N$ instead of $\mathbb{Z}_p$ and $\mathbb{Z}_q$, respectively.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$. Choose $v \leftarrow \mathbb{G}_p$, $w \leftarrow \mathbb{G}_q$, $\underline{d_{i,b} \leftarrow \mathbb{Z}_N,\ e_{i,b} \leftarrow \mathbb{Z}_N}$.
   Set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$ and $k' = (w, ((e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, d_{n,1})))$.
2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b_1^i \ldots b_n^i$. Output $= v^{\prod_j d_{j,b_j^i}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i$ for all $i \leq q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt}_{u,k,k',x^*})$ and $h(x^*) = b_1^* \ldots b_n^*$. Let $y_0 = w^{\prod_j e_{j,b_j^*}}$ and $y_1 \leftarrow \mathbb{G}$.
4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

Game 2 This is Game 2 from Section 4.1. This game is same as the previous one, except that v and w are chosen from $\mathbb{G}$ instead of the subgroups $\mathbb{G}_p$ and $\mathbb{G}_q$ respectively.

1. Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$.
   $\underline{\text{Choose } v \leftarrow \mathbb{G},\ w \leftarrow \mathbb{G}}$, $d_{i,b} \leftarrow \mathbb{Z}_N$, $e_{i,b} \leftarrow \mathbb{Z}_N$.
   Set $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$ and $k' = (w, ((e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, d_{n,1})))$.
2. On any evaluation query $x_i \in \{0,1\}^\ell$, check if $P_u(x_i) = 1$.
   If not, flip a coin $\gamma_i \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma_i = 1$.
   Else compute $h(x_i) = b_1^i \ldots b_n^i$. Output $= v^{\prod_j d_{j,b_j^i}}$.
3. $\mathcal{A}$ sends challenge input x^* such that $x^* \neq x_i$ for all $i \leq q$. Check if $P_u(x^*) = 0$.
   If not, flip a coin $\gamma^* \leftarrow \{0,1\}$ and abort. $\mathcal{A}$ wins if $\gamma^* = 1$.
   Else compute $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt}_{u,k,k',x^*})$ and $h(x^*) = b_1^* \ldots b_n^*$. Let $y_0 = w^{\prod_j e_{j,b_j^*}}$ and $y_1 \leftarrow \mathbb{G}$.

4. Flip coin $\beta \leftarrow \{0,1\}$. Output (K_{x^*}, y_β).
5. $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

We will now prove that the difference in advantage of any PPT adversary in two consecutive game is negligible in λ. Let $\mathsf{Adv}_{\mathcal{A}}^{1\alpha}$ denote the advantage of adversary $\mathcal{A}$ in Game 1α.

Observation 4 *For any adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{1} = \mathsf{Adv}_{\mathcal{A}}^{1A}$.*

Proof: The only difference between Game 1 and Game 1A is the manner in which the constants $d_{i,b}$ are chosen. In Game 1, the challenger chooses $d_{i,b} \leftarrow \mathbb{Z}_N$. In Game 1A, the challenger first chooses $d'_{i,b} \leftarrow \mathbb{Z}_N$, $a \leftarrow \mathbb{Z}_N^*$ and then sets $d_{i,b}$ as either $d'_{i,b}$ or $d'_{i,b} \cdot a$, depending on u_i. However, note that in either case, $d_{i,b}$ is a uniformly random element in $\mathbb{Z}_N$ since $a \in \mathbb{Z}_N^*$. ■

Claim 8 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{1A} - \mathsf{Adv}_{\mathcal{A}}^{1B}$ is non-negligible in λ, then there exists a PPT adversary $\mathcal{B}$ that breaks the security of $i\mathcal{O}$ with advantage non-negligible in λ.*

Proof: Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{1A} - \mathsf{Adv}_{\mathcal{A}}^{1B} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the security of $i\mathcal{O}$ with advantage ϵ by interacting with $\mathcal{A}$. $\mathcal{B}$ establishes some parameters that will be needed to interact with $\mathcal{A}$ and which will be used as part of the circuits designed by $\mathcal{B}$: first it chooses $v \leftarrow \mathbb{G}$, $a \leftarrow \mathbb{Z}_N^*$, $d'_{i,b} \leftarrow \mathbb{Z}_N$, $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$ and sets $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = d'_{i,b} \cdot a$. On receiving evaluation query x_i, $\mathcal{B}$ first checks that $P_u(x_i) = 1$. If so, it computes $h(x) = b_1^i \dots b_n^i$ and sends $v^{\prod_j d_{j,b_j^i}}$ to $\mathcal{A}$.

On receiving challenge query x^* from $\mathcal{A}$, $\mathcal{B}$ checks $P_u(x^*) = 0$. Now, $\mathcal{B}$ is ready to construct the circuits. It sets $k = (v, ((d_{1,0}, d_{1,1}), \dots, (d_{n,0}, d_{n,1})))$, $V = (v, v^a, \dots, v^{a^{n-1}})$, $w = v^{a^n}$ and $D = ((d'_{1,0}, d'_{1,1}), \dots, (d'_{n,0}, d'_{n,1}))$. It uses k to construct circuit $C_0 = \mathsf{PuncturedKey}_{k,x^*}$, uses V, w, D to construct $C_1 = \mathsf{PuncturedKey}'_{V,w,D,u,x*}$ and sends C_0, C_1 to the $i\mathcal{O}$ challenger. $\mathcal{B}$ receives $K_{x^*} \leftarrow i\mathcal{O}(C_b)$ from the challenger. It computes $h(x^*) = b_1^* \dots b_n^*$, $y_0 = v^{\prod_j d_{j,b_j^*}}$, $y \leftarrow \mathbb{G}$, $\beta \leftarrow \{0,1\}$, sends (K_{x^*}, y_β) to $\mathcal{A}$ and receives β' in response. If $\beta = \beta'$, $\mathcal{B}$ outputs 0, else it outputs 1.

We will now prove that the circuits C_0 and C_1 have identical functionality. Consider any ℓ bit string x, and let $h(x) = b_1 \dots b_n$. Recall $r_u(x) = |\{j : u_j \neq b_j\}|$.

$$\prod_j d_{j,b_j} = \left(\prod_{u_j = b_j} d'_{j,b_j}\right) \cdot \left(\prod_{u_j \neq b_j} (d'_{j,b_j} \cdot a)\right) = \left(\prod_{u_j = b_j} d'_{j,b_j}\right) \cdot a^{r_u(x)} \cdot \left(\prod_{u_j \neq b_j} d'_{j,b_j}\right)$$

Hence, $v^{\prod_j d_{j,b_j}} = \left(v^{a^{r_u(x)}}\right)^{\prod_j d'_{j,b_j}}$. Also, recall $r_u(x) = n$ iff $P_u(x) = 0$. Therefore, to compute $v^{\prod_j d_{j,b_j}}$ for some x such that $P_u(x) = 1$, we only need the constant $\{d'_{i,b}\}$ and $\{v, v^a, v^{a^2}, \dots, v^{a^{n-1}}\}$. Similarly, if $P_u(x) = 0$, we only need the constants $\{d'_{i,b}\}$ and v^{a^n} to compute $v^{\prod_j d_{j,b_j}}$. Using this observation, we can prove that the programs $\mathsf{PuncturedKey}_{k,x^*}$ and $\mathsf{PuncturedKey}'_{V,w,D,u,x^*}$ have identical functionality.

For any $x \in \{0,1\}^\ell$ such that $P_u(x) = 1$,

$$\mathsf{PuncturedKey}'_{V,w,D,u,x^*}(x) = \left(v^{a^{r_u(x)}}\right)^{\prod_j d'_{j,b_j}} = v^{\prod_j d_{j,b_j}} = \mathsf{PuncturedKey}_{k,x^*}(x)$$

For any $x \in \{0,1\}^\ell$ such that $P_u(x) = 0$,

$$\mathsf{PuncturedKey}'_{V,w,D,u,x^*}(x) = w^{\prod_j d'_{j,b_j}} = \left(v^{a^n}\right)^{\prod_j d'_{j,b_j}} = v^{\prod_j d_{j,b_j}}$$
$$= \mathsf{PuncturedKey}_{k,x^*}(x)$$

Since C_0 and C_1 have identical functionality, $Pr[\mathcal{B}$ wins $] = Pr[\mathcal{A}$ wins in Game 1A] - $Pr[\mathcal{A}$ wins in Game 1B]. If $\mathsf{Adv}^{1A}_{\mathcal{A}} - \mathsf{Adv}^{1B}_{\mathcal{A}} = \epsilon$, then $\mathcal{B}$ wins the $i\mathcal{O}$ security game with advantage ϵ. ■

Claim 9 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{1B}_{\mathcal{A}} - \mathsf{Adv}^{1C}_{\mathcal{A}}$ is non-negligible in λ, then there exists a PPT adversary $\mathcal{B}$ that breaks Assumption 2 with advantage non-negligible in λ.*

Proof: Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{1B}_{\mathcal{A}} - \mathsf{Adv}^{1C}_{\mathcal{A}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that uses $\mathcal{A}$ and breaks Assumption 2 with advantage at least ϵ. $\mathcal{B}$ receives the group description $\mathbb{G}$, $(v, v^a, \ldots, v^{a^{n-1}}) \in \mathbb{G}^n$ and $T \in \mathbb{G}$, where $T = v^{a^n}$ or $T \leftarrow \mathbb{G}$. $\mathcal{B}$ chooses $d'_{i,b} \leftarrow \mathbb{Z}_N$, $u \leftarrow \mathsf{AdmSample}(1^\lambda, q)$. $\mathcal{B}$ sets $V = (v, \ldots, v^{a^{n-1}})$, $w = T$, $D = ((d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}))$.

On evaluation query x_i, $\mathcal{B}$ first checks that $P_u(x_i) = 1$. Next, it computes $h(x) = b^i_1 \ldots b^i_n$, and finally outputs $\left(v^{a^{r_u(x_i)}}\right)^{\prod_j d'_{i,b^i_j}}$. Note that $r_u(x) < n$ if $P_u(x) = 0$. Therefore, $\mathcal{B}$ can simulate the evaluation query phase perfectly.

On challenge query x^*, $\mathcal{B}$ first checks that $P_u(x^*) = 0$. Next, it computes $h(x) = b^*_1 \ldots b^*_n$, $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}'_{V,w,D,u,x^*})$. It sets $y_0 = w^{\prod_j d'_{j,b_j}}$ and $y_1 \leftarrow \mathbb{G}$.

Finally, $\mathcal{B}$ chooses $\beta \leftarrow \{0,1\}$, sends (K_{x^*}, y_β) from $\mathcal{A}$, and receives β'. If $\beta = \beta'$, $\mathcal{B}$ outputs 0 (i.e. it guesses $T = v^{a^n}$), else it outputs 1.

Clearly, if $\mathsf{Adv}^{1B}_{\mathcal{A}} - \mathsf{Adv}^{1C}_{\mathcal{A}} = \epsilon$, then $\mathcal{B}$ also wins with advantage ϵ. ■

Claim 10 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{1C}_{\mathcal{A}} - \mathsf{Adv}^{1D}_{\mathcal{A}}$ is non-negligible in λ, then there exists a PPT adversary $\mathcal{B}$ that breaks Assumption 1 with advantage non-negligible in λ.*

Proof: We will prove this claim using an intermediate experiment Hyb. Hyb is the same as Game 1C, except that $v \leftarrow \mathbb{G}_p$. We will prove that Game 1C and Hyb are computationally indistinguishable, and similarly, Hyb and Game 1D are computationally indistinguishable.

Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{1C}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Hyb}}_{\mathcal{A}} = \epsilon$. Using $\mathcal{A}$, we will construct a PPT algorithm $\mathcal{B}$ that breaks Assumption 1. $\mathcal{B}$ receives $\mathbb{G}$, $\mathbb{G}_p$, $\mathbb{G}_q$, $g_1 \leftarrow \mathbb{G}_p$, $g_2 \leftarrow G_q$ and T, where $T \leftarrow \mathbb{G}_p$ or $T \leftarrow \mathbb{G}$. $\mathcal{B}$ sets $v = T$, chooses $a \leftarrow \mathbb{Z}^*_N$, $d'_{i,b} \leftarrow \mathbb{Z}_N$, $u \leftarrow \mathsf{AdmSample}$, $w \leftarrow \mathbb{G}$ and computes $V = (v, v^a, \ldots, v^{a^{n-1}})$ and $d_{i,b}$ as in Game 1C.

On receiving evaluation query x_i from $\mathcal{A}$, $\mathcal{B}$ computes $h(x_i) = b^i_1 \ldots b^i_n$, checks if $P_u(x_i) = 1$ and responds with $\left(v^{a^{r_u(x_i)}}\right)^{\prod_j d'_{j,b^i_j}}$.

On receiving challenge query x^*, $\mathcal{B}$ checks $P_u(x^*) = 0$, and computes $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKey}'_{V,w,D,u,x^*})$, $y_0 = (w)^{\prod_j d'_{j,b^*_j}}$, $y_1 \leftarrow \mathbb{G}$, $\beta \leftarrow \{0,1\}$ and outputs (K_{x^*}, y_β). If $\mathcal{A}$ outputs β, $\mathcal{B}$ guesses $v \in \mathbb{G}$, else guesses $v \in \mathbb{G}_p$. Notice that if $\mathsf{Adv}^{\mathsf{Hyb}}_{\mathcal{A}} - \mathsf{Adv}^{1C}_{\mathcal{A}} = \epsilon$, then $\mathcal{B}$ breaks Assumption 1 with advantage ϵ.

In order to prove our claim, we now need to show that the experiments Hyb and Game 1D are computationally indistinguishable. Note that the only difference between these two experiments is the manner in which w is chosen. In Hyb, $w \leftarrow \mathbb{G}$, while in Game 1D, $w \leftarrow \mathbb{G}_q$. We can show that if Assumption 1 holds, then Hyb and Game 1D are computationally indistinguishable. This argument is exactly similar to the step from Game 1C to Hyb, the only difference being that the assumption used will be that $w \leftarrow \mathbb{G}$ is computationally indistinguishable from $w \leftarrow \mathbb{G}_q$. ■

Claim 11 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{1D} - \mathsf{Adv}_{\mathcal{A}}^{1E}$ is non-negligible in λ, then there exists a PPT adversary $\mathcal{B}$ that breaks the security of $i\mathcal{O}$ with advantage non-negligible in λ.*

Proof: We will use two intermediate experiments Hyb_1 and Hyb_2, and use the security of $i\mathcal{O}$ to prove that (a) Game 1D and Hyb_1 are computationally indistinguishable, (b) Hyb_1 and Hyb_2 are identical experiments, and (c) Hyb_2 and Game 1E are computationally indistinguishable.

First we define the experiments Hyb_1 and Hyb_2. In Hyb_1, the challenger first samples $u, v, w, d'_{i,b}, a$ as in Game 1D, and sets $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = a$ (same as before). The evaluation query responses are also same as in Game 1D. For the challenge query x^*, the challenger first checks that $P_u(x^*) = 0$. Next, it sets $w' = w^{1/a^n}$, computes $y_0 = w'^{\prod_j d_{j,b_j^*}} = w^{\prod_j d_{j,b_j^*}}, y_1 \leftarrow \mathbb{G}$ as in Game 1D. However, the punctured key K_{x^*} is computed using $\mathsf{PuncturedKeyAlt}$. The challenger sets $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$, $k' = (w', ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$ and computes $K_{x^*} \leftarrow i\mathcal{O}(\lambda, \mathsf{PuncturedKeyAlt}_{k,k',u,x^*})$.

Hyb_2 is the same as Hyb_1, except that it chooses $d_{i,b} \leftarrow \mathbb{Z}_N$ and uses w instead of w' in the key k' and for computing y_0.

Proof of (a): Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{1D} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hyb}_1} = \epsilon$. We will first construct a PPT algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the security of $i\mathcal{O}$. $\mathcal{B}$ samples $u, v, w, d'_{i,b}$ and computes $d_{i,b}$ as in Game 1D/Hyb. On receiving evaluation query x_i, it checks $P_u(x_i) = 0$ and outputs $v^{\prod_j d_{j,b_j^i}}$. On receiving challenge query x^*, $\mathcal{B}$ sets $V = (v, v^a, \ldots, v^{a^{n-1}})$, $D = ((d_{1,0}, d_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}))$, k, k' as in Hyb, and constructs circuits $C_0 = \mathsf{PuncturedKey}'_{V,w,D,u,x^*}$ and $C_1 = \mathsf{PuncturedKeyAlt}_{k,k',u,x^*}$. It sends C_0, C_1 to the $i\mathcal{O}$ challenger, and receives $K_{x^*} \leftarrow i\mathcal{O}(\lambda, C_b)$. $\mathcal{B}$ computes y_0, chooses y_1, β, sends (K_{x^*}, y_β) to $\mathcal{A}$ and receives β' in response. If $\beta = \beta'$, $\mathcal{B}$ outputs 0, else outputs 1.

In order to complete this proof, we show that circuits $\mathsf{PuncturedKey}'_{V,w,D,u,x^*}$ and $\mathsf{PuncturedKeyAlt}_{k,k',u,x^*}$ have identical functionality.

For any $x \in \{0,1\}^\ell$ such that $P_u(x) = 1$, $h(x) = b_1 \ldots b_n$,

$$\mathsf{PuncturedKeyAlt}_{k,k',u,x^*}(x) = v^{\prod_j d_{j,b_j}} = \left(v^{a^{r_u(x)}}\right)^{\prod_j d'_{j,b_j}} = \mathsf{PuncturedKey}'_{V,w,D,u,x^*}(x).$$

For any $x \in \{0,1\}^\ell$ such that $P_u(x) = 0$, $x \neq x^*$, $h(x) = b_1 \ldots b_n$,

$$\begin{aligned}\mathsf{PuncturedKey}_{k,k',u,x^*}(x) = w'^{\prod_j d_{j,b_j}} = \left(w'^{a^n}\right)^{\prod_j d'_{j,b_j}} = w^{\prod_j d'_{j,b_j}} \\ = \mathsf{PuncturedKey}'_{V,w,D,u,x^*}(x).\end{aligned}$$

For $x = x^*$, both programs outputs $\perp$. Therefore, the two programs are functionally equivalent. Hence, $\mathcal{B}$ breaks the security of $i\mathcal{O}$ with advantage ϵ.

Proof of (b): Note that in both Hyb_1 and Hyb_2, $d_{i,b}$ is a uniformly random element in $\mathbb{Z}_N$ (since $a \in Z_N^*$). Also, if $w \leftarrow \mathbb{G}_q$, then w^{1/a^n} is a uniformly random element in $\mathbb{G}_q$. From these two observations, it follows that the two experiments Hyb_1 and Hyb_2 are the same.

Proof of (c): Game 1E is similar to Hyb_2, except that the challenger chooses k, k' differently. In Game 1E, the challenger chooses $v \leftarrow \mathbb{G}_p, w \leftarrow \mathbb{G}_q, d_{i,b} \leftarrow \mathbb{Z}_N$. It sets $d_{i,b,p} = d_{i,b} \mod p$, $d_{i,b,q} = d_{i,b} \mod q$, $k = (v, ((d_{1,0,p}, d_{1,1,p}), \ldots, (d_{n,0,p}, d_{n,1,p})))$ and $k' = (w, ((d_{1,0,q}, d_{1,1,q}), \ldots, (d_{n,0,q}, d_{n,1,q})))$. Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hyb}_2} - \mathsf{Adv}_{\mathcal{A}}^{1E} = \epsilon$. Consider the following PPT algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the security of $i\mathcal{O}$. $\mathcal{B}$ chooses $v, w, \{d_{i,b}\}$ and sets $k = (v, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$, $k' = (w, ((d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})))$, $\tilde{k} = (v, ((d_{1,0,p}, d_{1,1,p}), \ldots, (d_{n,0,p}, d_{n,1,p})))$ and $\tilde{k'} = (w, ((d_{1,0,q}, d_{1,1,q}), \ldots, (d_{n,0,q}, d_{n,1,q})))$. $\mathcal{B}$ uses the constants v and $\{d_{i,b}\}_{i,b}$ to respond to $\mathcal{A}$'s evaluation queries. On receiving challenge query x^*, $\mathcal{B}$ constructs circuits $C_0 = \mathsf{PuncturedKeyAlt}_{k,k',u,x^*}$ and $C_1 = \mathsf{PuncturedKeyAlt}_{\tilde{k},\tilde{k'},u,x^*}$ and sends C_0, C_1 to the $i\mathcal{O}$ challenger. It receives K_{x^*} in response. $\mathcal{B}$ computes y_0, y_1, chooses $\beta \leftarrow \{0,1\}$ and sends (K_{x^*}, y_β) to $\mathcal{A}$, and receives β' in response. If $\beta = \beta'$, then $\mathcal{B}$ outputs 0, else it outputs 1.

Since $v \in \mathbb{G}_p$, $v^{\prod_j d_{j,b_j}} = v^{\prod_j d_{j,b_j,p}}$ for any sequence $b_1 \ldots b_n$. Similarly, since $w \in \mathbb{G}_q$, $w^{\prod_j d_{j,b_j}} = w^{\prod_j d_{j,b_j,q}}$. Therefore the circuits C_0, C_1 have identical functionality. Hence, $\mathcal{B}$ breaks the security of $i\mathcal{O}$ with advantage ϵ. ■

Claim 12 *For any adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{1F} = \mathsf{Adv}_{\mathcal{A}}^{1E}$.*

Proof: We will show that Game 1E and Game 1F are identical, and therefore, any adversary has the same advantage in both games. Note that the only difference between Game 1E and Game 1F is the manner in which the keys k, k' are chosen. In Game 1E, the challenger chooses $v \in \mathbb{G}_p$, $w \in \mathbb{G}_q$, $d_{i,b} \in \mathbb{Z}_N$ and sets $k = (v, ((d_{1,0,p}, d_{1,1,p}), \ldots, (d_{n,0,p}, d_{n,1,p})))$ and $k' = (w, ((d_{1,0,q}, d_{1,1,q}), \ldots, (d_{n,0,q}, d_{n,1,q})))$. In Game 1F, the challenger chooses $d_{i,b,p} \leftarrow \mathbb{Z}_p$, $e_{i,b,q} \leftarrow \mathbb{Z}_q$ and sets $k = (v, ((d_{1,0,p}, d_{1,1,p}), \ldots, (d_{n,0,p}, d_{n,1,p})))$ and $k' = (w, ((e_{1,0,q}, e_{1,1,q}), \ldots, (e_{n,0,q}, e_{n,1,q})))$. Using Chinese Remainder Theorem, it follows that $\{(x \mod p, x \mod q) : x \leftarrow \mathbb{Z}_N\} \equiv \{(x, y) : x \leftarrow \mathbb{Z}_p, y \leftarrow Z_q\}$, and hence, Game 1E and Game 1F are identical. ■

Claim 13 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{1F} - \mathsf{Adv}_{\mathcal{A}}^{1G}$ is non-negligible in λ, then there exists a PPT adversary $\mathcal{B}$ that breaks the security of $i\mathcal{O}$ with advantage non-negligible in λ.*

Proof: The proof of this claim is similar to the proof of Claim 11, part (c). ■

Claim 14 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{1G} - \mathsf{Adv}_{\mathcal{A}}^{2}$ is non-negligible in λ, then there exists a PPT adversary $\mathcal{B}$ that breaks Assumption 1 with advantage non-negligible in λ.*

Proof: The proof of this claim is similar to the one for Claim 10. ■

B Reducing Assumption 2 to Subgroup Hiding Assumption in Composite Order DDH-Hard Groups

Chase and Meiklejohn showed that in bilinear groups of composite order, many q-type reductions are implied by subgroup hiding. The paper also states that a similar result holds in composite order DDH-hard groups. For completeness, we include a proof of this implication.

Theorem B.1 *Let $(N, p, q, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$. Then, Assumption 1 implies Assumption 2.*

As in the work by Chase and Meiklejohn, the proof of this theorem uses the *dual system* technique introduced by Waters. First, we define $4n+1$ hybrid experiments $\textsf{Game } 1$, $\textsf{Game } 1_1$, $\textsf{Game } 1_2$, $\textsf{Game } 1_3$, …, $\textsf{Game } n-1$, $\textsf{Game } n-1_1$, $\textsf{Game } n-1_2$, $\textsf{Game } n-1_3$, $\textsf{Game } n$ and then show that (a) no PPT adversary can distinguish between consecutive hybrid experiments (b) any adversary has negligible advantage in $\textsf{Game } n$.

B.1 Sequence of Games

We underline the changes from one game to the next.

Game 0 : Choose $v \leftarrow \mathbb{G}, a \leftarrow \mathbb{Z}_N$, $\beta \in \{0,1\}$ and set $T_0 = v^{a^n}$, $T_1 \leftarrow \mathbb{G}$.
Output $(N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, v, v^a, \dots, v^{a^{n-1}}, T_\beta)$.

Game i : Choose $\underline{v_1 \leftarrow \mathbb{G}_p, a_1, \dots a_i, r_1, \dots, r_i \leftarrow \mathbb{Z}_N}$, $\beta \in \{0,1\}$ and set $\underline{T_0 = v_1^{\sum_{j=1}^{i} r_j a_j^n}}$, $T_1 \leftarrow \mathbb{G}$.
Output $(N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, \underline{v_1^{\sum_{j=1}^{i} r_j}, v_1^{\sum_{j=1}^{i} r_j a_j}, \dots, v_1^{\sum_{j=1}^{i} r_j a_j^{n-1}}, T_\beta})$.

Game i_1 : Choose $v_1 \leftarrow \mathbb{G}_p$, $\underline{v_2 \leftarrow \mathbb{G}_q}$, $a_1, \dots a_i, r_1, \dots, r_i \leftarrow \mathbb{Z}_N$, $\beta \in \{0,1\}$ and set $\underline{T_0 = v_1^{\sum_{j=1}^{i} r_j a_j^n} v_2^{a_i^n}}$, $T_1 \leftarrow \mathbb{G}$.
Output $(N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, \underline{v_1^{\sum_{j=1}^{i} r_j} v_2, v_1^{\sum_{j=1}^{n} r_j a_j} v_2^{a_i}, \dots, v_1^{\sum_{j=1}^{i} r_j a_j^{n-1}} v_2^{a_i^{n-1}}, T_\beta})$

Game i_2 : Choose $v_1 \leftarrow \mathbb{G}_p$, $v_2 \leftarrow \mathbb{G}_q$, $a_1, \dots a_i$, $\underline{a_{i+1}}$, $r_1, \dots, r_i \leftarrow \mathbb{Z}_N$, $\beta \in \{0,1\}$ and set $\underline{T_0 = v_1^{\sum_{j=1}^{i} r_j a_j^n} v_2^{a_{i+1}^n}}$, $T_1 \leftarrow \mathbb{G}$.
Output $(N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, \underline{v_1^{\sum_{j=1}^{i} r_j} v_2, v_1^{\sum_{j=1}^{i} r_j a_j} v_2^{a_{i+1}}, \dots, v_1^{\sum_{j=1}^{i} r_j a_j^{n-1}} v_2^{a_{i+1}^{n-1}}, T_\beta}))$.

Game i_3 : Choose $v_1 \leftarrow \mathbb{G}_p$, $v_2 \leftarrow \mathbb{G}_q$, $a_1, \dots a_i$, a_{i+1}, $r_1, \dots, r_i$, $\underline{r_{i+1}} \leftarrow \mathbb{Z}_N$, $\beta \in \{0,1\}$ and set $\underline{T_0 = v_1^{\sum_{j=1}^{i+1} r_j a_j^n} v_2^{a_{i+1}^n}}$, $T_1 \leftarrow \mathbb{G}$.
Output $(N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, \underline{v_1^{\sum_{j=1}^{i+1} r_j} v_2, v_1^{\sum_{j=1}^{i+1} r_j a_j} v_2^{a_{i+1}}, \dots, v_1^{\sum_{j=1}^{i+1} r_j a_j^{n-1}} v_2^{a_{i+1}^{n-1}}, T_\beta}))$.

B.2 Adversary's Advantage in these Games

Let $\textsf{Adv}_{\mathcal{A}}^{1\alpha}$ denote the advantage of adversary $\mathcal{A}$ in $\textsf{Game } \alpha$. Note that in Game 0, $a \leftarrow \mathbb{Z}_N$, while in Assumption 2, $a \leftarrow \mathbb{Z}_N^*$. This results in security loss that is negligible in λ.

Claim 1 *If there exists a PPT adversary* $\mathcal{A}$ *such that* $\textsf{Adv}_{\mathcal{A}}^{10} - \textsf{Adv}_{\mathcal{A}}^{11} = \epsilon$*, then there exists a PPT algorithm* $\mathcal{B}$ *that can distinguish between a random element of* $\mathbb{G}$ *and a random element of* $\mathbb{G}_p$ *with advantage* ϵ*.*

Proof: The only difference between the two games is that in $\textsf{Game } 0$, the challenger chooses $v \leftarrow \mathbb{G}$, while in $\textsf{Game } 1$, the challenger chooses $v_1 \leftarrow \mathbb{G}_p$. $\mathcal{B}$ receives $N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2$ and w from the challenger, where $w \leftarrow \mathbb{G}$ or $w \leftarrow \mathbb{G}_p$. $\mathcal{B}$ sets $v = w$, chooses $a \in \mathbb{Z}_N$, $\beta \in \{0,1\}$ and sets $T_0 = v^{a^n}$, $T_1 \leftarrow \mathbb{G}$. It sends $(N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, v, v^a, \dots, v^{a^{n-1}}, T_\beta)$ ot $\mathcal{A}$ and receives β' in response. If $\beta' = \beta$, $\mathcal{B}$ sends 1 to the challenger (indicating that $w \in \mathbb{G}$), else it sends 0. Clearly, if $w \in \mathbb{G}$, then this corresponds to $\mathsf{Game}\ 0$, else it corresponds to $\mathsf{Game}\ 1$. If $\mathcal{A}$ wins $\mathsf{Game}\ 0$ with probability p_0, and wins $\mathsf{Game}\ 1$ with probability p_1, then the advantage of $\mathcal{B}$ in breaking Assumption 1 is $p_0 - p_1 = \epsilon$. ■

Claim 2 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{1i} - \mathsf{Adv}_{\mathcal{A}}^{1i_1} = \epsilon$, then there exists a PPT algorithm $\mathcal{B}$ that can distinguish between a random element of $\mathbb{G}$ and a random element of $\mathbb{G}_p$ with advantage ϵ.*

Proof: The PPT algorithm $\mathcal{B}$ is defined as follows. First, $\mathcal{B}$ receives $N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2$ and w from the challenger, where $w \leftarrow \mathbb{G}$ or $w \leftarrow \mathbb{G}_p$. $\mathcal{B}$ chooses $v' \leftarrow \mathbb{G}_p$, $a_1, \ldots, a_{i-1}, r_1, \ldots, r_{i-1} \leftarrow \mathbb{Z}_N$, $\beta \leftarrow \{0,1\}$ and sets $T_0 = v'^{\sum_{j=1}^{i-1} r_j a_j^n} w^{a_i^n}$ and $T_1 \leftarrow \mathbb{G}$. It sends $(N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2, v'^{\sum_{j=1}^{i-1} r_j} w, v'^{\sum_{j=1}^{i-1} r_j a_j} w^{a_i}, \ldots, v'^{\sum_{j=1}^{i-1} r_j a_j^{n-1}} w^{a_i^{n-1}}, T_\beta)$ to $\mathcal{A}$ and receives β' in response. If $\beta' = \beta$, it sends 0 to the challenger, else it sends 1.

If $w \in \mathbb{G}$, then $w = v'^r v_2$ for some $r \in \mathbb{Z}_N, v_2 \in \mathbb{G}_q$. If $w \in \mathbb{G}_p$, then $w = v'^r$ for some $r \in \mathbb{Z}_N$. Therefore, $\mathcal{B}$ implicitly sets $r_i = r$, and hence, if $w \in \mathbb{G}_p$, $\mathcal{A}$ receives the challenge as per $\mathsf{Game}\ i$, otherwise it receives the challenge as per $\mathsf{Game}\ i_1$. ■

Claim 3 *For any adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{1i_1} = \mathsf{Adv}_{\mathcal{A}}^{1i_2}$.*

Proof: This step is information-theoretic. First, let us recall the Chinese Remainder Theorem. For distinct primes p, q and $N = pq$, we have

$$\{(x \mod p, x \mod q) : x \leftarrow \mathbb{Z}_N\} \equiv \{(x \mod p, y \mod q) : x \leftarrow \mathbb{Z}_N, y \leftarrow \mathbb{Z}_N\}.$$

Hence, it follows that distributions D_1 and D_2 are identical, where

$$D_1 = \left\{ \sum_{j=1}^{i} r_j a_j \mod p, \ldots, \sum_{j=1}^{i} r_j a_j^n \mod p, a_i \mod q : r_1, \ldots, r_i, a_1, \ldots, a_i \leftarrow \mathbb{Z}_N \right\}$$

$$D_2 = \left\{ \sum_{j=1}^{i} r_j a_j \mod p, \ldots, \sum_{j=1}^{i} r_j a_j^n \mod p, a_{i+1} \mod q : r_1, \ldots, r_i, a_1, \ldots, a_i, a_{i+1} \leftarrow \mathbb{Z}_N \right\}.$$

Note that since $v \in \mathbb{G}_p$, $v^x = v^{x \bmod p}$. Similarly, $w^y = w^{y \bmod q}$. Hence it follows that any adversary has identical advantage in $\mathsf{Game}\ i_1$ and $\mathsf{Game}\ i_2$.

■

Claim 4 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{1i_2} - \mathsf{Adv}_{\mathcal{A}}^{1i_3} = \epsilon$, then there exists a PPT algorithm $\mathcal{B}$ that can distinguish between a random element of $\mathbb{G}$ and a random element of $\mathbb{G}_q$ with advantage ϵ.*

Proof: The PPT algorithm $\mathcal{B}$ is defined as follows. First, $\mathcal{B}$ receives $N, \mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, g_1, g_2$ and w from the challenger, where $w \leftarrow \mathbb{G}$ or $w \leftarrow \mathbb{G}_q$. $\mathcal{B}$ chooses $v' \leftarrow \mathbb{G}_p$, $a_1, \ldots, a_i, r_1, \ldots, r_i \leftarrow \mathbb{Z}_N$, $\beta \leftarrow \{0,1\}$ and sets $T_0 = v'^{\sum_{j=1}^{i} r_j a_j^n} w^{a_{i+1}^n}$ and $T_1 \leftarrow \mathbb{G}$. It sends $(N, \mathbb{G}, \mathbb{G}_p, G_q, g_1, g_2, v'^{\sum_{j=1}^{i} r_j} w, v'^{\sum_{j=1}^{i} r_j a_j} w^{a_{i+1}}, \ldots, v'^{\sum_{j=1}^{i} r_j a_j^{n-1}} w^{a_{i+1}^{n-1}}, T_\beta)$ to $\mathcal{A}$ and receives β' in response. If $\beta' = \beta$, it sends 0 to the challenger (i.e. $\mathcal{B}$ guesses that $w \leftarrow \mathbb{G}_q$), else it sends 1.

As in the proof of 2, if $w \in \mathbb{G}$, then $w = v'^r v_2$, else $w = v_2$ for some r, v_2. $\mathcal{B}$ therefore implicitly sets $r_{i+1} = r$. If $w \in \mathbb{G}_q$, then $\mathcal{A}$ gets a $\mathsf{Game}\ i_2$ challenge, else a $\mathsf{Game}\ i_3$ challenge. ■

Claim 5 *If there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{1i3}_{\mathcal{A}} - \mathsf{Adv}^{1i+1}_{\mathcal{A}} = \epsilon$, then there exists a PPT algorithm $\mathcal{B}$ that can distinguish between a random element of $\mathbb{G}$ and a random element of $\mathbb{G}_q$ with advantage ϵ.*

Proof: The proof for this step is similar to the proof of Claim 4. ■

Claim 6 *For any adversary $\mathcal{A}$, $\mathsf{Adv}^{1n}_{\mathcal{A}} \leq negl(\lambda)$.*

Proof: Let us consider the following distributions (all operations are modulo p):

$$D_1 = \left\{ \left(\sum_{j=1}^{n+1} r_j, \sum_{j=1}^{n+1} r_j a_j, \ldots, \sum_{j=1}^{n+1} r_j a_j^{n-1}, \sum_{j=1}^{n+1} r_j a_j^{n} \right) : r_1, \ldots, r_{n+1}, a_1, \ldots, a_{n+1} \leftarrow \mathbb{Z}_p \right\}$$

$$D_2 = \left\{ \left(\sum_{j=1}^{n+1} r_j, \sum_{j=1}^{n+1} r_j a_j, \ldots, \sum_{j=1}^{n} r_j a_j^{n-1}, r \right) : r_1, \ldots, r_{n+1}, a_1, \ldots, a_{n+1}, r \leftarrow \mathbb{Z}_p \right\}$$

In order to show that $\mathsf{Adv}^{1n+1}_{\mathcal{A}} \leq \text{negl}(\lambda)$, it suffices to show that D_1 and D_2 are statistically indistinguishable. In fact, one can prove the following stronger statement.

Observation 5 $D_1 \approx_s \{(c_1, \ldots, c_{n+1}) : c_1, \ldots, c_{n+1} \leftarrow \mathbb{Z}_p\}$

Note that D_1 can also be expressed as

$$\left\{ (r_1 \ldots r_{n+1}) \cdot V_{a_1 \ldots a_{n+1}} : r_i, a_j \in \mathbb{Z}_p \right\}$$

where $V_{a_1 \ldots a_{n+1}}$ is a Vandermonde matrix of dimension $(n+1) \times (n+1)$.

$$V_{a_1 \ldots a_{n+1}} = \begin{bmatrix} 1 & a_1 & \cdots & a_1^n \\ 1 & a_2 & \cdots & a_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & a_{n+1} & \cdots & a_{n+1}^n \end{bmatrix}$$

If all a_is are distinct and non-zero, then $V_{a_1 \ldots a_{n+1}}$ is invertible. Hence, if the entries a_i are chosen uniformly at random from $\mathbb{Z}_p$, then with overwhelming probability, $V_{a_1 \ldots a_n}$ is invertible. Thus, there is a bijection between $\{(r_1, \ldots, r_{n+1}) \cdot V : r_i \in \mathbb{Z}_p\}$ and $\mathbb{Z}_p^{n+1}$. This proves our observation. ■