Constrained PRFs for $\mathbf{NC}^1$ in Traditional Groups

Nuttapong Attrapadung[1], Takahiro Matsuda[1], Ryo Nishimaki[2], Shota Yamada[1], Takashi Yamakawa[2]

[1]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
{n.attrapadung,t-matsuda,yamada-shota}@aist.go.jp
[2]NTT Secure Platform Laboratories, Tokyo, Japan
{nishimaki.ryo,yamakawa.takashi}@lab.ntt.co.jp

February 8, 2018

Abstract

We propose new constrained pseudorandom functions (CPRFs) in *traditional groups*. Traditional groups mean cyclic and multiplicative groups of prime order that were widely used in the 1980s and 1990s (sometimes called "pairing free" groups). Our main constructions are as follows.

- We propose a selectively single-key secure CPRF for *circuits with depth $O(\log n)$ (that is, $\mathbf{NC}^1$ circuits) in traditional groups* where n is the input size. It is secure under the L-decisional Diffie-Hellman inversion (L-DDHI) assumption in the group of quadratic residues $\mathbb{QR}_q$ and the decisional Diffie-Hellman (DDH) assumption in a traditional group of order q *in the standard model*.
- We propose a selectively single-key *private bit-fixing* CPRF in *traditional groups*. It is secure under the DDH assumption in any prime-order cyclic group *in the standard model*.
- We propose *adaptively* single-key secure CPRF for $\mathbf{NC}^1$ and private bit-fixing CPRF in the random oracle model.

To achieve the security in the standard model, we develop a new technique using correlated-input secure hash functions.

Contents

1 Introduction

1.1 Background

Pseudorandom functions (PRFs) are one of the most fundamental notions in cryptography [GGM86]. A PRF is a deterministic function $\mathsf{PRF}(\cdot,\cdot) : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ where $\mathcal{K}$, $\mathcal{D}$, and $\mathcal{R}$ are its key space, domain, and range, respectively. Roughly speaking, we say that PRF is a secure PRF if outputs of $\mathsf{PRF}(\mathsf{msk},\cdot)$ look random for any input $x \in \mathcal{D}$ and a randomly chosen key $\mathsf{msk} \in \mathcal{K}$. Not only are PRFs used to construct secure encryption schemes but also they frequently appear in the constructions of various cryptographic primitives.

Constrained PRF. Boneh and Waters introduced the notion of *constrained PRFs (CRPFs)* [BW13] (Kiayias, Papadopoulos, Triandopoulos, and Zacharias [KPTZ13] and Boyle, Goldwasser, and Ivan [BGI14] also proposed the same notion in their concurrent and independent works). CPRFs are an advanced type of PRFs. Specifically, if we have a master secret key msk of a CPRF PRF, then we can generate a "constrained" key sk_f for a function $f : \mathcal{D} \to \{0,1\}$. We can compute the value $\mathsf{PRF}(\mathsf{msk}, x)$ from sk_f and x if $f(x) = 0$ holds; otherwise cannot. For an input x such that $f(x) = 1$, the value $\mathsf{PRF}(\mathsf{msk}, x)$ looks pseudorandom.[1]

CPRFs with various types of function classes have been considered. Here, we explain the classes of *bit-fixing functions* and *circuits* since we present new CPRFs for these functions.

Bit-fixing functions: Let $\{0,1\}^n$ be the domain of a CPRF. Each function in this class is specified by a "constraint vector" $c = (c_1, \dots, c_n) \in \{0,1,*\}^n$, from which a *bit-fixing* function $f_c : \{0,1\}^n \to \{0,1\}$ is defined as follows. If $c_i = *$ or $x_i = c_i$ holds for all $i \in [n]$, then $f_c(x) = 0$; otherwise $f_c(x) = 1$.

Circuits: This class consists of functions $\{f_C\}$ computable by polynomial-sized boolean circuits C, defined by $f_C(\cdot) := C(\cdot)$. We call a CPRF for this function class simply a CPRF for circuits. If a CPRF supports functions computable by polynomial-sized boolean circuits with depth $O(\log n)$, where n is the input-length of the circuits, then we call it a CPRF for $\mathbf{NC}^1$.

The number of constrained keys that can be released (to a potentially malicious party) is one of the important security measures of CPRFs. If a-priori unbounded polynomially many constrained keys could be released (i.e., the number of queries is not a-priori bounded), then a CPRF is called *collusion-resistant*. If only one constrained key can be released, it is call a *single-key secure* CPRF. Boneh and Waters [BW13] showed that (collusion-resistant) CPRFs have many applications such as broadcast encryption with optimal ciphertext length. (See their paper and references therein for more details.)

Private CPRF. Boneh, Lewi, and Wu [BLW17] proposed the notion of *privacy* for CPRFs (Kiayias *et al.* also proposed policy privacy as essentially the same notion [KPTZ13]). Roughly speaking, private CPRFs do not reveal information about constraints embedded in constrained keys beyond what is leaked from the evaluation results using the constrained keys.

Known instantiations. The first papers on CPRFs [BW13, KPTZ13, BGI14] observed that the Goldreich-Goldwasser-Micali [GGM86] PRF yields a puncturable PRF[2] (and a CPRF for related simple functions). However, it turned out that achieving CPRFs for other types of function classes are

[1] We note that the role of the constraining function f is "reversed" from the definition by Boneh and Waters [BW13], in the sense that the evaluation by a constrained key sk_f is possible for inputs x with $f(x) = 1$ in their definition, while it is possible for inputs x for $f(x) = 0$ in our paper. Our treatment is the same as Brakerski and Vaikuntanathan [BV15].

[2] A constrained key in which a set of points is hard-wired enables us to compute an output if an input is not in the specified set.

quite challenging. Here, we review some prior works on CPRFs whose function classes are related to those we focus on in this study (i.e., bit-fixing functions and $\mathbf{NC}^1$ circuits).

Boneh and Waters [BW13] constructed a left-right CPRF[3] in the random oracle model (ROM) from bilinear maps and a collusion-resistant bit-fixing CPRF and collusion-resistant CPRF for circuits from multilinear maps [GGH13] in the standard model. After that, Brakerski and Vaikuntanathan [BV15] constructed a single-key secure CPRF for circuits from standard lattice-based assumptions, without relying on multilinear maps.

Boneh *et al.* [BLW17] constructed a collusion-resistant private CPRF for circuits from indistinguishability obfuscation (IO) [BGI+12, GGH+16], and a single-key private bit-fixing CPRF and puncturable CPRF from multilinear maps [BLW17]. After that, a single-key private puncturable PRF [BKM17], a single-key private CPRF for $\mathbf{NC}^1$ [CC17], and a single-key private CPRF for circuits [BTVW17, PS18] were constructed from standard lattice assumptions.

Our motivation. (Private) CPRFs have been attracting growing attention as above since they are useful tools to construct various cryptographic primitives [BW13, BLW17]. A number of other types of CPRFs have been constructed [HKKW14, HKW15, DKW16, HKW15, HKKW14, BFP+15, AFP16]. However, all of known sufficiently expressive (private) CPRFs (such as bit-fixing, circuits) rely on IO, multilinear maps, or lattices, and there is currently no candidate of secure multilinear maps.

Very recently, Bitansky [Bit17] and Goyal, Hohenberger, Koppula, and Waters [GHKW17] proposed sub-string match[4] CPRFs in *traditional groups* to construct verifiable random functions. In this paper, by traditional groups we mean the multiplicative groups of prime order[5] that have been widely used to construct various cryptographic primitives such as the ElGamal public-key encryption scheme, around two decades before bilinear maps dominate the area of cryptography [BF03]. (Of course, they are still being used for many cryptographic primitives). However, their CPRFs are not expressive enough and do not satisfy the standard security requirements of CPRFs[6]. See Tables 1 and 2 for comparisons. There is no construction of *expressive enough* (private) CPRF in *traditional groups*. This status might be reasonable since lattices and multilinear maps are stronger tools.

Based on the motivation mentioned above, we tackle the following question:

Is it possible to construct sufficiently expressive (private) CPRFs in traditional groups?

In this study, we give affirmative answers to this question and show that traditional groups are quite powerful tools. From the theoretical point of view, the more instantiations of cryptographic primitives are available, the more desirable. One reason is that constructions from different tools can be alternatives when one tool is broken (like multilinear maps). Another reason is that, generally, new instantiations shed light on how to construct the studied primitive, and widen and deepen our insights on it. One remarkable example of this line of research would be the recent work by Döttling and Garg [DG17], who constructed an identity-based encryption (IBE) scheme and a hierarchical IBE scheme in traditional groups. Another example would be the work by Boyle, Isahai, and Gilboa [BGI16], who constructed communication-efficient secure two-party protocols in traditional groups. It is also expected that new instantiations provide us with insights on how to use the studied primitive in applications (in the real world or in the construction of another primitive as a building block).

[3]There are left and right constrained keys in which v_ℓ and v_r are hard-wired, respectively. We can compute outputs by using the left (resp. right) constrained key if the first (resp. last) half of an input is equal to v_ℓ (resp. v_r).

[4]This is the negation of bit-fixing functions, that is, $f_c(x) = 0$ if there exists an index i such that $x_i \neq c_i$ (i-th bit of a constraint) and $c_i \neq *$. It can be seen as and the generalization of punctured predicate.

[5]For example, cyclic group $\mathbb{H} \subset \mathbb{Z}_q^*$ of a prime order p such that $q = 2p + 1$ where q is also a prime.

[6]In their sub-string match CPRFs, adversaries are not given access to the evaluation oracle, which gives outputs of a CPRF for queried inputs. We call such security no-evaluation security in this paper.

1.2 Our Contributions

In this paper, we present new construction of a CPRF and a private CPRF in *traditional groups* as main contributions.

Properties of our CPRFs are summarized as follows.

- Our first CPRF is a *selectively single-key secure*[7] *CPRF for* $\mathbf{NC}^1$ in traditional groups. It is secure under the L-decisional Diffie-Hellman inversion (L-DDHI) assumption[8] in the group of quadratic residues $\mathbb{QR}_q$ and the decisional Diffie-Hellman (DDH) assumption[9] in a traditional group $\mathbb{G}$ of order q *in the standard model*. Here, $\mathbb{QR}_q$ denotes the group of quadratic residue modulo q, where q is a prime such that $q = 2p + 1$ and p is also a prime. We need to use this specific type of group for technical reasons. See Section 1.3 and Section 4 for the details.

- Our second CPRF is a *selectively single-key private bit-fixing CPRF* in traditional groups. Specifically, it is secure under the standard DDH assumption in any prime-order cyclic group *in the standard model*.

- Our third and fourth CPRFs are an *adaptively*[10] *single-key secure CPRF for* $\mathbf{NC}^1$ *circuits* and an *adaptively single-key private bit-fixing CPRF*, both in the ROM. Our standard model and ROM constructions of CPRFs for $\mathbf{NC}^1$, share high-level ideas behind the constructions in common, and the same is true for our bit-fixing CPRFs. These connections are explained in Section 1.3.

The main technique that enables us to achieve the above results, is a novel use of *correlated-input secure hash functions*. We will explain the technical overview in Section 1.3.

As an application of our results, we can obtain a single-key secret-key attributed-based encryption (ABE) scheme with *optimal ciphertext overhead* in traditional groups. A (multi-key) public-key ABE scheme with optimal ciphertext overhead was presented by Zhandry [Zha16], but it is based on multilinear maps. See Section 6 for more details.

1.3 Technical Overview

In this section, we provide an overview of our construction ideas. We ignore many subtle issues in this section and focus on the essential ideas for simplicity.

Basic construction satisfying no-evaluation security. To illustrate our ideas in a modular manner, we start with a no-evaluation secure CPRF for $\mathbf{NC}^1$, that is, adversaries do not have access to the evaluation oracle. We denote the PRF by $\mathsf{PRF}_{\mathsf{NE}}$. It turns out that even in this simple setting, it is non-trivial to construct a CPRF for $\mathbf{NC}^1$ in traditional groups (or bilinear groups) since known constructions use some sort of "fully homomorphic" properties of lattices or multilinear maps, both of which are not available in traditional groups. In the following, let λ be the security parameter.

The first challenge is how to implement an $\mathbf{NC}^1$ circuit constraint in a key. Our idea is to encode an $\mathbf{NC}^1$ circuit f[11] into a bit string $f = (f_1, \ldots, f_z) \in \{0,1\}^z$ and then embed this into a secret key. When evaluating a PRF value on input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, we will "homormorphically" evaluate $U(\cdot, x)$ on the secret key, where $U(\cdot,\cdot)$ is a universal circuit that outputs $U(f, x) = f(x)$ on input (f, x).

[7]Adversaries commit a function to be embedded in a constrained key at the beginning of the security experiment and have access to the evaluation oracle, which gives outputs of CPRFs for queried inputs.

[8]The L-DDHI assumption in a group $\mathbb{H}$ of order p [BB04, CHL05] says that it is hard to distinguish $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^L}, g^{1/\alpha})$ from $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^L}, g^{z})$ where $g \xleftarrow{R} \mathbb{H}, \alpha, z \xleftarrow{R} \mathbb{Z}_p$. See Section 2.1 for the rigorous definition.

[9]The DDH assumption in a group $\mathbb{G}$ of order q says that it is hard to distinguish (g, g^x, g^y, g^{xy}) from (g, g^x, g^y, g^z) where $g \xleftarrow{R} \mathbb{G}, x, y, z \xleftarrow{R} \mathbb{Z}_q$.

[10]Adversaries can decide a function for which it makes the key query at any time.

[11]Here, we identify a circuit that computes a function f with f itself.

Table 1: Comparison of CPRFs (we omit constructions based on multilinear maps or IO). In "Function" column, sub-match is sub-string match. Prefix-fixing means that a constrained key with prefix p enables us to compute outputs for inputs $p\|*$. "# keys" column means the number of issuable constrained keys. "Eval.$\mathcal{O}$" column means the evaluation oracle is available for adversaries or not. "Tool" column means what kind of cryptographic tools are used. GGM, pairing, and group mean the PRF by Goldreich, Goldwasser, and Micali [GGM86], bilinear maps, and traditional groups, respectively. In "Assumptions" column, OWF, BDDH, LWE, and 1D-SIS means one-way function, bilinear Diffie-Hellman, learning with errors, and one-dimensional short integer solution assumptions, respectively. In "Model" column, Std means the standard model. In "Misc" column, key-hom means key-homomorphic property.

Reference	Function	# keys	Eval.$\mathcal{O}$	Tool	Assumptions	Model	Misc
[BW13]	puncture[a]	N/A	N/A	GGM	OWF	Std	
[BW13]	left/right	multi	✓	pairing	BDDH	ROM	
[KPTZ13]	puncture[a]	N/A	N/A	GGM	OWF	Std	
[BGI14]	puncture[a]	N/A	N/A	GGM	OWF	Std	
[BFP$^+$15]	prefix-fixing	multi	✓	lattice	LWE	Std	key-hom
[BV15]	circuit	single	✓	lattice	LWE, 1D-SIS	Std	
[Bit17]	sub-match	single	no	group	DDH	Std	
[GHKW17]	sub-match	single	no	group	L-power DDH	Std	
[GHKW17]	sub-match	single	no	group	Φ-hiding	Std	
Ours	$\mathbf{NC}^1$	single	✓	group	DDH, L-DDHI	Std	

[a] More precisely, they consider slightly different functions, but we write just "puncture" for simplicity since their constructions are based on the GGM PRF. See their papers for details.

Table 2: Comparison of private CPRFs (we omit constructions based on multilinear maps and IO). See Table 1 for terms.

Reference	Predicate	# keys	Eval.$\mathcal{O}$	Tool	Assumptions	Model
[KPTZ13]	puncture[a]	N/A	N/A	GGM	OWF	Std
[BKM17]	puncture	N/A	N/A	lattice	LWE, 1D-SIS	Std
[CC17]	bit-fixing	single	✓	lattice	LWE	Std
[CC17]	$\mathbf{NC}^1$	single	✓	lattice	LWE	Std
[BTVW17]	circuit	single	✓	lattice	LWE	Std
[PS18]	circuit	single	✓	lattice	LWE, 1D-SIS	Std
Ours	bit-fixing	single	✓	group	DDH	Std

[a] Same as in Table 1.

To make the representation of the universal circuit $U(\cdot,\cdot)$ compatible with our algebraic setting, we regard $U(\cdot,\cdot)$ as a degree-D polynomial of the variables $\{f_i\}$ and $\{x_j\}$, such that D is some fixed polynomial of λ.[12] Furthermore, we extend the input space of $U(\cdot,\cdot)$ to be non-binary, where the computation is done over $\mathbb{Z}_p$ using the polynomial representation of $U(\cdot,\cdot)$. Specifically, we allow the input of the form $((b_1,\ldots,b_z),x) \in \mathbb{Z}_p^z \times \{0,1\}^n$.

Now, we give a more detailed description of $\mathsf{PRF}_{\mathsf{NE}}$. A master secret key msk of $\mathsf{PRF}_{\mathsf{NE}}$ is of the form $(b_1,\ldots,b_z,\alpha,g)$, where $b_i \overset{\mathsf{R}}{\leftarrow} \mathbb{Z}_p$ for each $i \in [z]$ and $\alpha \overset{\mathsf{R}}{\leftarrow} \mathbb{Z}_p^*$, and g is a generator of a traditional group $\mathbb{H}$ of order p. (We will turn to the explanation on this group $\mathbb{H}$ later in this subsection.) The evaluation algorithm of $\mathsf{PRF}_{\mathsf{NE}}$ outputs $g^{x'/\alpha}$, where $x' = U((b_1,\ldots,b_z),x) \in \mathbb{Z}_p$. To compute a constrained key sk_f of an $\mathbf{NC}^1$ circuit f, we set $b'_i := (b_i - f_i)\alpha^{-1}$. The constrained key is $\mathsf{sk}_f = (f, b'_1,\ldots,b'_z, g, g^{\alpha}, g^{\alpha^2},\ldots,g^{\alpha^{D-1}})$.

We then look closer at why this construction achieves the constraint defined by the $\mathbf{NC}^1$ circuit f.

[12]We can construct universal circuit U whose depth is only constant times deeper than that of f by the result of Cook and Hoover [CH85]. It is well known that an $\mathbf{NC}^1$ circuit can be represented by a polynomial with polynomial degree (for example, this fact is used for functional encryption for $\mathbf{NC}^1$ [GVW12]).

When we compute $x' := U((b_1,\ldots,b_z),x)$ by using $b_i = \alpha \cdot b'_i + f_i$, we can write the computation of U in the following way:

$$x' = U((\alpha \cdot b'_1 + f_1, \ldots, \alpha \cdot b'_z + f_z), x) = f(x) + \sum_{j=1}^{D} c_j \alpha^j,$$

where the coefficients $\{c_j\}_j$ are efficiently computable from the descriptions of U and f, $\{b'_i\}_i$, and x since the degree D is polynomial in the security parameter. This can be seen by observing that $U((\alpha \cdot b'_1 + f_1, \ldots, \alpha \cdot b'_z + f_z), x)$ should be equal to $f(x)$ when $\alpha = 0$ since we have $U((f_1,\ldots,f_z),x) = f(x)$ by the definition of a universal circuit.

- If $f(x) = 0$, then we can compute $g^{x'/\alpha} = g^{f(x)/\alpha + \sum_{j=0}^{D-1} c_j \alpha^j}$ since the $g^{f(x)/\alpha}$ part disappears and the remaining part is computable from $\mathsf{sk}_f = (f, b'_1, \ldots, b'_z, g, g^{\alpha}, \ldots, g^{\alpha^{D-1}})$ and x.
- If $f(x) = 1$, then $g^{x'/\alpha} = g^{f(x)/\alpha + \sum_{j=0}^{D-1} c_j \alpha^j}$ looks random since $g^{1/\alpha}$ looks random even if $(g, g^{\alpha}, \ldots, g^{\alpha^{D-1}})$ is given, due to the $(D-1)$-DDHI assumption in $\mathbb{H}$.

This is a high-level intuition for why $\mathsf{PRF}_{\mathsf{NE}}$ for $\mathbf{NC}^1$ is no-evaluation secure. This CPRF $\mathsf{PRF}_{\mathsf{NE}}$ is our base construction, and the idea behind our construction here is inspired by the affine partitioning function used in the recent construction of a verifiable random function by Yamada [Yam17].

On the other hand, this construction can be broken by making only one evaluation query: Suppose that $x \neq \widehat{x}$ satisfy $f(x) = f(\widehat{x}) = 1$. Then we can write $\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, x) = g^{1/\alpha + \sum_{j=0}^{D-1} c_j \alpha^j}$ and $\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, \widehat{x}) = g^{1/\alpha + \sum_{j=0}^{D-1} \widehat{c_j} \alpha^j}$ by using $\{c_j\}_j$ and $\{\widehat{c_j}\}_j$ that are efficiently computable by an adversary. Then we have $\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, \widehat{x}) = \mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, x) \cdot g^{\sum_{j=0}^{D-1} (\widehat{c_j} - c_j)\alpha^j}$. Therefore if an adversary obtains $\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, x)$, then it can efficiently compute $\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, \widehat{x})$ and break the security of the PRF.

Single-key secure construction in the ROM. To achieve security against adversaries making a-priori unbounded polynomially many evaluation queries (i.e., the number of queries is polynomial, but not fixed in advance), we consider using a random oracle as an intermediate step. (This construction is denoted by $\mathsf{PRF}^{\mathsf{rom}}$.) $\mathsf{PRF}^{\mathsf{rom}}$ is the same as $\mathsf{PRF}_{\mathsf{NE}}$ except that an output is now computed by $H(g^{x'/\alpha})$, instead of $g^{x'/\alpha}$, where $H : \mathbb{H} \to \{0,1\}^{n'}$ is a cryptographic hash function. In the ROM where H is modeled as a random oracle, adversaries make hash queries and obtain outputs of the hash function H. If $f(x) = 1$, then an adversary cannot compute $g^{x'/\alpha}$ due to the no-evaluation security, and thus $H(g^{x'/\alpha})$ seems uniformly random from the view of the adversary. Therefore evaluation queries from an adversary can be answered with uniformly random strings, and the adversary cannot notice whether this is a correct behavior of the evaluation oracle as long as it does not find a collision (x_1, x_2) such that $g^{x'_1/\alpha} = g^{x'_2/\alpha}$ where $x'_i = U((b_1,\ldots,b_z),x_i)$. Our real construction is slightly modified from the above construction so that such a collision exists only with negligible probability (see Section 4.1 for the detail).

The second challenge is how to remove the random oracle and achieve security against a-priori unbounded polynomially evaluation queries in the standard model.

Replacing a random oracle with a correlated-input secure hash function. We observe that we do not need the full power of random oracles to prove the security of CPRFs. Specifically, we can use a *correlated-input secure hash function* (CIH) [IKNP03, GL10, BC10a, GOR11][13], instead of random oracles.

[13]Several works defined similar notions in different names such as related-key security. We use the name "correlated-input security" since we think it is the most suitable name for our usage.

Here, we briefly recall the definition of a CIH whose definition is associated with a class of functions Ψ. At the beginning, the challenger chooses the challenge bit $\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$, a function description CIH,[14] and a random element r from the domain of CIH. The adversary is given CIH and access to an oracle that, upon a query $\psi_i \in \Psi$ from the adversary, answers $\mathsf{CIH}(\psi_i(r))$ if $\mathsf{coin} = 1$; otherwise the oracle answers the query with $\mathsf{RF}(\psi_i(r))$, where RF is a truly random function. If it is hard for adversaries to distinguish the case $\mathsf{coin} = 1$ from the case $\mathsf{coin} = 0$, we say that CIH is correlated-input pseudorandom for Ψ (or simply, a CIH for Ψ).[15].

If there exists a CIH for *group-induced* functions $\psi_\Delta : \mathbb{H} \to \mathbb{H}$ such that $\Delta \in \mathbb{H}$ and $\psi_\Delta(y) := y \cdot \Delta$ (denoted by CIH_0) where $\cdot$ is the group operation of $\mathbb{H}$, then $\mathsf{CIH}_0(\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, x))$ is a secure CPRF. This can be seen as follows: For x satisfying $f(x) = 1$, $\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, x)$ can be written as $g^{1/\alpha} \cdot g^{\sum_{j=0}^{D-1} c_j \alpha^j}$ where $g^{1/\alpha}$ is pseudorandom and $g^{\sum_{j=0}^{D-1} c_j \alpha^j}$ is efficiently computable from the view of an adversary as discussed above. By applying the security of a CIH by setting $y := g^{1/\alpha}$ and $\Delta = g^{\sum_{j=0}^{D-1} c_j \alpha^j}$, we can see that $\mathsf{CIH}_0(\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, x))$ is computationally indistinguishable from $\mathsf{RF}(\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, x))$. This is computationally indistinguishable from a random function as long as $\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, x)$ has no collision, and the actual construction of $\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}, x)$ is made collision-free as mentioned in the previous paragraph.

However, there is one subtle issue: The only known instantiation of CIH for group induced functions which satisfies our security requirements is the CIH based on the DDH assumption by Bellare and Cash [BC10a] (denoted by $\mathsf{CIH}_{\mathsf{BC}}$). In $\mathsf{CIH}_{\mathsf{BC}}$, we consider the *$m$-dimensional, component-wise group-induced functions* $\Psi_m^{\mathsf{g\text{-}indc}} := \{\psi_{\vec{a}} \mid \vec{a} \in (\mathbb{Z}_q^*)^m\}$, where $\psi_{\vec{a}} : (\mathbb{Z}_q^*)^m \to (\mathbb{Z}_q^*)^m$ is defined by $\psi_{\vec{a}}(\vec{r}) := \vec{a} \star \vec{r}$ and $\star$ denotes the component-wise group operation on $\mathbb{Z}_q^*$. Here, the domain of $\mathsf{CIH}_{\mathsf{BC}}$ is not compatible with the range of $\mathsf{PRF}_{\mathsf{NE}}$ (the output is $g^{x'/\alpha_i} \in \mathbb{H}$). One might think that m-folded parallel running of $\mathsf{PRF}_{\mathsf{NE}}$ on $\mathbb{H} := \mathbb{Z}_q^*$ works, but this is not the case. This is because if $\mathbb{H} := \mathbb{Z}_q^*$, then the L-DDHI assumption can be easily broken by computing the Jacobi symbol.

We observe that the attack based on the Jacobi symbol does not work if we consider the group of quadratic residues modulo q, denoted by $\mathbb{QR}_q$ instead of $\mathbb{Z}_q^*$, and it is reasonable to assume the L-DDHI assumption holds on $\mathbb{QR}_q$. However, if we set $\mathbb{H} := \mathbb{QR}_q$, then we cannot simply use the security of $\mathsf{CIH}_{\mathsf{BC}}$ since it is not obvious if the security of $\mathsf{CIH}_{\mathsf{BC}}$ still holds when we restrict the domain of $\mathsf{CIH}_{\mathsf{BC}}$ to $\mathbb{QR}_q^m$. We resolve the issue by proving that the CIH obtained by restricting the domain of $\mathsf{CIH}_{\mathsf{BC}}$ to $\mathbb{QR}_q^m$ (denoted by $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$) is also secure as a CIH for component-wise group operations on $\mathbb{QR}_q^m$ under the DDH assumption on a group of an order $p = \frac{q-1}{2}$ if p is a prime. See Section 3 for more details of $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$.

It is ready to explain our CRPF PRF for $\mathbf{NC}^1$. It uses multiple instances of $\mathsf{PRF}_{\mathsf{NE}}$ and apply a CIH for m-dimensional component-wise group-induced functions to the outputs from those instances. That is, we define

$$\mathsf{PRF}_{\mathbf{NC}^1}(\mathsf{msk}, x) := \mathsf{CIH}_{\widetilde{\mathsf{BC}}}\Big(\mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}_1, x), \ldots, \mathsf{PRF}_{\mathsf{NE}}(\mathsf{msk}_m, x) \Big).$$

Now, we look closer at why correlated-input pseudorandomness helps us achieve security in the presence of a-priori unbounded polynomially many evaluation queries. In $\mathsf{PRF}_{\mathsf{NE}}$, when the inputs x with $f(x) = 1$ are used, we can view its output as consisting of two separate parts. Specifically, we can write $g^{x'/\alpha} = g^{f(x)/\alpha + \sum_{j=0}^{D-1} c_j \alpha^j} = \mathsf{Aux}(\mathsf{msk}) \cdot \mathsf{SEval}(\mathsf{sk}_f, x)$ if we define $\mathsf{Aux}(\mathsf{msk}) := g^{1/\alpha}$ and $\mathsf{SEval}(\mathsf{sk}_f, x) := g^{\sum_{j=0}^{D-1} c_j \alpha^j}$ (where SEval stands for "semi"-evaluation). The first part is computable only from msk, and the second part is computable from sk_f and x. Thanks to the $(D-1)$-DDHI assumption, it is now easy to see that $\mathsf{Aux}(\mathsf{msk})$ is indistinguishable from a random element even if sk_f is

[14]In the formal security definition, the function is parameterized by a public parameter generated by some setup procedure. We ignore the public parameter in the explanation below for simplicity. See Section 2.5 for the rigorous security definition for CIHs.

[15]The definition of CIHs in this paper can be seen as a hybrid of correlated-input pseudorandom by Goyal. *et al.* [GOR11] and RKA-PRG by Bellare and Cash [BC10a]. See Section 2.5 for the formal definition.

given. Therefore, it holds that

$$\mathsf{PRF}_{\mathbf{NC}^1}(\mathsf{msk}, x) \approx_{\mathsf{c}} \mathsf{CIH}_{\widetilde{\mathsf{BC}}}\Big(r_1 \cdot \mathsf{SEval}(\mathsf{sk}_{f,1}, x), \ldots, r_m \cdot \mathsf{SEval}(\mathsf{sk}_{f,m}, x) \Big),$$

where $r_i \xleftarrow{\mathsf{R}} \mathbb{H}$ for all $i \in [m]$ and $\approx_{\mathsf{c}}$ denotes computational indistinguishability. Furthermore, $\mathsf{sk}_{f,i}$ denotes the secret key associated to f generated from msk_i. (Namely, it corresponds to the i-th instance.) Here, $\phi_i := \mathsf{SEval}(\mathsf{sk}_{f,i}, x) \in \mathbb{H}$ are adversarially chosen correlated values and fall in the component-wise group-induced functions $\Psi_m^{\mathsf{g\text{-}indc}}$ due to $(\phi_1, \ldots, \phi_m) \in \mathbb{H}^m$. Therefore, by applying the correlated-input pseudorandomness of $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$, we obtain

$$\mathsf{CIH}_{\widetilde{\mathsf{BC}}}(r_1 \cdot \phi_1, \ldots, r_m \cdot \phi_m) \approx_{\mathsf{c}} \mathsf{RF}(r_1 \cdot \phi_1, \ldots, r_m \cdot \phi_m).$$

As long as adversaries do not find a collision (x_1, x_2) such that $(\mathsf{SEval}(\mathsf{sk}_{f,1}, x_1), \ldots, \mathsf{SEval}(\mathsf{sk}_{f,m}, x_1)) = (\mathsf{SEval}(\mathsf{sk}_{f,1}, x_2), \ldots, \mathsf{SEval}(\mathsf{sk}_{f,m}, x_2))$, $\mathsf{PRF}_{\mathbf{NC}^1}(\mathsf{msk}, \cdot)$ is pseudorandom since RF is a truly random function. It is not difficult to see that a collision is hard to find by the universality of the modified $\mathsf{PRF}_{\mathsf{NE}}$ (see Lemma 4.15 for the detail). Therefore, we can prove the pseudorandomness of PRF against a-priori unbounded polynomially many evaluation queries in the standard model by using the security of CIH for (m-dimensional, component-wise) group-induced functions.

How to achieve private constraint. Here, we give a brief explanation on how our single-key private CPRF for bit-fixing functions is constructed. The basic strategy is the same as that of our CPRFs for $\mathbf{NC}^1$. That is, we firstly construct a private bit-fixing CPRF in the ROM, and then convert it into a private bit-fixing CPRF in the standard model via a CIH for an appropriate function class.

Our single-key private bit-fixing CPRF in the ROM is very simple. This is slightly different from what we present in Section 5.2, but we stick to the following construction in this section since it is consistent with the standard model construction in Section 5.1. A master secret key is $\mathsf{msk} := \{s_{i,b}\}_{i\in[n],b\in\{0,1\}}$ and a PRF output for input x is $H(\sum_{i=1}^n s_{i,x_i})$ where H is a (standard) hash function. For convenience, we define $\mathsf{PRF}_{\mathsf{bf\text{-}NE}}(\mathsf{msk}, x) := \sum_{i=1}^n s_{i,x_i}$. A constrained key for $c \in \{0,1,*\}^n$ is $\{t_{i,b}\}_{i\in[n],b\in\{0,1\}}$ where $t_{i,b} := s_{i,b}$ if $c_i = *$ or $c_i = b$; otherwise $t_{i,b} \xleftarrow{\mathsf{R}} \mathbb{Z}_p$. If an input does not match the constraint c, then the sum includes completely unrelated values and we can not compute the correct output. Adversaries are given just random values by the random oracle. Moreover, adversaries can not distinguish two different constraints as long as a challenge input does not satisfy the constraints since both $s_{i,b}$ and $t_{i,b}$ are uniformly random values in $\mathbb{Z}_p$. This construction satisfies adaptive single-key privacy in the random oracle model, without relying on any complexity assumption.

Now we replace the cryptographic hash function (random oracle) H with a CIH $\mathsf{CIH}_{\mathsf{aff}}$ for *affine functions* $\Phi^{\mathsf{aff}} = \{\phi_{\vec{u},\vec{v}} : \mathbb{Z}_p^m \to \mathbb{Z}_p^m\}$ where $\vec{u} \in (\mathbb{Z}_p^*)^m$, $\vec{v} \in \mathbb{Z}_p^m$, and $\phi_{\vec{u},\vec{v}}(\vec{x}) := \vec{u} \odot \vec{x} + \vec{v}$ where $\odot$ is component-wise multiplication in $\mathbb{Z}_p$. Our private bit-fixing CPRF is defined by

$$\mathsf{PRF}_{\mathsf{BF}}(\mathsf{msk}, x) := \mathsf{CIH}_{\mathsf{aff}}\Big(\mathsf{PRF}_{\mathsf{bf\text{-}NE}}(\mathsf{msk}_1, x), \ldots, \mathsf{PRF}_{\mathsf{bf\text{-}NE}}(\mathsf{msk}_m, x) \Big).$$

A constrained key sk_c consists of constrained keys for c with respect to msk_j, for all $j \in [m]$. It is easy to see that the correctness holds. For the security, we set $t_{i,b,j} := s_{i,b,j} - \alpha_j$ for $c_i \neq *$ and $b = 1 - c_i$ where $\alpha_j \xleftarrow{\mathsf{R}} \mathbb{Z}_p$. Then, we can write $\sum_{i=1}^n s_{i,x_i,j} = u\alpha_j + v_j$ for some $u \in [n]$ (especially $u \neq 0$) where $v_j = \sum_{i=1}^n t_{i,x_i,j}$ for an evaluation query x from an adversary, since x is not allowed to satisfy the constraint. For two different constraints, the adversary cannot distinguish which constraint is used in a constrained key (that is, $s_{i,b,j} \approx_{\mathsf{c}} t_{i,b,j} + \alpha_j$) since $t_{i,b,j}$ is uniformly random. Here, α_j's are uniformly random and u and v_j are adversarially chosen values. It is easy to see that this falls into the class of affine functions. Thus, we can use the security of the CIH $\mathsf{CIH}_{\mathsf{aff}}$ for affine functions, and obtain

$$\mathsf{CIH}_{\mathsf{aff}}(u\alpha_1 + v_1, \ldots, u\alpha_m + v_m) \approx_{\mathsf{c}} \mathsf{RF}(u\alpha_1 + v_1, \ldots, u\alpha_m + v_m).$$

As long as a collision of $(\mathsf{PRF}_{\mathsf{bf\text{-}NE}}(\mathsf{msk}_1, \cdot), \ldots, \mathsf{PRF}_{\mathsf{bf\text{-}NE}}(\mathsf{msk}_m, \cdot)$ is not found, $\mathsf{RF}(u\alpha_1+v_1, \ldots, u\alpha_m+v_m)$ is indistinguishable from a random value. Furthermore, it is not difficult to show that the condition holds by the universality of $F_t(x) := (u\alpha_1 + v_1, \ldots, u\alpha_m + v_m)$. Therefore, we can prove the security of our private bit-fixing CPRF. See Lemma 5.6 for the details.

1.4 Other Related Works

While we focus on (private) CPRFs without IO and multilinear maps, many expressive (private) CPRFs have been proposed based on IO or multilinear maps: collusion-resistant CPRFs for circuit based on multilinear maps [BW13, BFP+15], adaptively secure CPRFs based on IO [HKKW14, HKW15], collusion-resistant CPRFs for Turing machines based on (differing-input) IO [DKW16, AFP16], collusion-resistant private CPRFs for circuits based on IO [BLW17].

CPRFs and private CPRFs are useful to construct advanced cryptographic primitives. Boneh and Waters showed that we can construct broadcast encryption with optimal ciphertext length, identity-based non-interactive key-exchange, and policy-based key distribution from CPRFs [BW13]. Boneh *et al.* showed that private constrained message authentication code (MAC), watermarking PRF, searchable encryption, and on-line/off-line 2-server private keyword search from private CPRFs [BLW17]. Requirements on security of (private) CPRFs depends on these applications. Boneh, Kim, and Montgomery prove that single-key simulation-based private CPRFs imply single-key simulation-based secure secret-key functional encryption (FE) [BKM17]. Canetti and Chen prove that two-key indistinguishability-based (resp. simulation-based) private CPRFs for circuits imply indistinguishability (resp. virtual black-box) obfuscation (they also prove that private CPRFs imply secret-key FE) [CC17].

Cohen, Goldwasser, and Vaikuntanathan showed a connection between CPRFs for some class of functions and computational learning theory [CGV15].

See the papers and references therein for more details.

Organization. The rest of the paper is organized as follows. After introducing notations, security definitions, and building blocks in Section 2, we present our correlated-input secure hash function in Section 3, our CPRFs for $\mathbf{NC}^1$ and its security proofs in Section 4, and our private bit-fixing CPRF in Section 5.

2 Preliminaries

In this section, we review the basic notation and the definitions for complexity assumptions, tools, and cryptographic primitives.

Basic notation. We denote by $\mathbb{N}$ the set of all natural numbers. If $n \in \mathbb{N}$, then "$[n]$" denotes the set $\{1, \ldots, n\}$. We denote by "$x := y$" that y is deterministically assigned to x. If S is a finite set, then "$x \xleftarrow{\mathsf{R}} S$" denotes that x is chosen uniformly at random from S. If $\mathcal{D}$ and $\mathcal{D}'$ are distributions (over some set), then "$x \xleftarrow{\mathsf{R}} \mathcal{D}$" denotes that x is chosen according to the distribution $\mathcal{D}$, and "$\mathcal{D} \approx_{\mathsf{c}} \mathcal{D}'$" denotes that the two distributions are computationally indistinguishable. If x and y are bit-strings, then we denote by "$x \| y$" the concatenation of x and y, and "$(x \overset{?}{=} y)$" is defined to be 1 if $x = y$ and 0 otherwise. "PPT" stands for *probabilistic polynomial time*. If $\mathcal{A}$ is a probabilistic algorithm, then "$y \xleftarrow{\mathsf{R}} \mathcal{A}(x)$" denotes that $\mathcal{A}$ computes and outputs y by taking x as input and using an internal randomness that is chosen uniformly at random. If furthermore $\mathcal{O}$ is a (possibly probabilistic) function, then "$\mathcal{A}^{\mathcal{O}}$" denotes that $\mathcal{A}$ has oracle access to $\mathcal{O}$. A function $f(\cdot) : \mathbb{N} \to [0, 1]$ is said to be *negligible* if for all polynomials $p(\cdot)$ and all sufficiently large $\lambda \in \mathbb{N}$, we have $f(\lambda) < 1/p(\lambda)$. Throughout the paper, we use "λ" to denote a security parameter (which is given to algorithms always in the unary form 1^λ). We denote by "$\mathrm{poly}(\cdot)$" an unspecified integer-valued positive polynomial of λ and by "$\mathrm{negl}(\lambda)$" an unspecified negligible function of λ. For sets $\mathcal{D}$ and $\mathcal{R}$, "$\mathsf{Func}(\mathcal{D},\mathcal{R})$" denotes the set of all functions with domain $\mathcal{D}$ and range $\mathcal{R}$.

2.1 Complexity Assumptions

Here, we review complexity assumptions on cyclic groups that we use in this paper. For convenience, we introduce the notion of a "group generator". We say that a PPT algorithm GGen is a *group generator*, if it takes a security parameter 1^λ as input and outputs a "group description" $\mathcal{G} := (\mathbb{G}, p)$ where $\mathbb{G}$ is a group with prime order $p = \Omega(2^\lambda)$, from which one can efficiently sample a generator uniformly at random.

Definition 2.1 (Decisional Diffie-Hellman Assumption). *Let* GGen *be a group generator. We say that the* decisional Diffie-Hellman (DDH) assumption *holds with respect to* GGen*, if for all PPT adversaries* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathsf{ddh}}_{\mathsf{GGen},\mathcal{A}}(\lambda)$ *defined below is negligible:*

$$\mathsf{Adv}^{\mathsf{ddh}}_{\mathsf{GGen},\mathcal{A}}(\lambda) := \Big|\Pr[\mathcal{A}(\mathcal{G}, g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{A}(\mathcal{G}, g, g^x, g^y, g^z) = 1]\Big|,$$

where $\mathcal{G} = (\mathbb{G}, p) \xleftarrow{\mathsf{R}} \mathsf{GGen}(1^\lambda)$*,* $g \xleftarrow{\mathsf{R}} \mathbb{G}$*, and* $x, y, z \xleftarrow{\mathsf{R}} \mathbb{Z}_p^*$*.*

Definition 2.2 (L-Diffie-Hellman Inversion Assumption). *Let* GGen *be a group generator and* $L = L(\lambda) = \mathrm{poly}(\lambda)$*. We say that the* L-Diffie-Hellman inversion (DDHI) assumption *holds with respect to* GGen*, if for all PPT adversaries* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{L\text{-}\mathsf{ddhi}}_{\mathsf{GGen},\mathcal{A}}(\lambda)$ *defined below is negligible:*

$$\mathsf{Adv}^{L\text{-}\mathsf{ddhi}}_{\mathsf{GGen},\mathcal{A}}(\lambda) := \Big|\Pr[\mathcal{A}(\mathcal{G}, g, (g^{\alpha^i})_{i\in[L]}, \psi_0) = 1] - \Pr[\mathcal{A}(\mathcal{G}, g, (g^{\alpha^i})_{i\in[L]}, \psi_1) = 1]\Big|,$$

where $\mathcal{G} = (\mathbb{G}, p) \xleftarrow{\mathsf{R}} \mathsf{GGen}(1^\lambda)$*,* $g \xleftarrow{\mathsf{R}} \mathbb{G}$*,* $\alpha \xleftarrow{\mathsf{R}} \mathbb{Z}_p^*$*,* $\psi_0 := g^{1/\alpha}$*, and* $\psi_1 \xleftarrow{\mathsf{R}} \mathbb{G}$*.*

2.2 Pseudorandom Function

In this paper, we will treat a pseudorandom function (PRF) that has a public parameter. Hence, we introduce such a definition here. Since we will later extend the syntax of a PRF here into that of a CPRF, in which a key output from KeyGen is called a master secret key, a key for a PRF is denoted by msk here.

Formally, a PRF PRF consists of the three PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval})$ with the following interfaces:

$\mathsf{Setup}(1^\lambda) \xrightarrow{\mathsf{R}} \mathsf{pp}$**:** This is the setup algorithm that takes a security parameter 1^λ as input, and outputs a public parameter pp, where pp specifies the descriptions of the key space $\mathcal{K}$, the input-length $n = n(\lambda) = \mathrm{poly}(\lambda)$ (that defines the domain $\{0,1\}^n$), and the range $\mathcal{R}$.

$\mathsf{KeyGen}(\mathsf{pp}) \xrightarrow{\mathsf{R}} \mathsf{msk}$**:** This is the key generation algorithm that takes a public parameter pp as input, and outputs a key $\mathsf{msk} \in \mathcal{K}$.

$\mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x) =: y$**:** This is the deterministic evaluation algorithm that takes a public parameter pp, a key $\mathsf{msk} \in \mathcal{K}$, and an element $x \in \{0,1\}^n$ as input, and outputs an element $y \in \mathcal{R}$.

Whenever clear from the context, we will drop pp from the input of Eval. Furthermore, when there is no confusion, we may abuse the notation and use PRF to denote the evaluation algorithm itself, and use the notations such as "$\mathsf{PRF} : \mathcal{K} \times \{0,1\}^n \to \mathcal{R}$" and "$\mathsf{PRF}(\mathsf{msk}, x)$" (where the latter means the execution of $\mathsf{Eval}(\mathsf{msk}, x)$) for enabling easier and more intuitive descriptions.

Definition 2.3 (Security of PRF). *We say that* $\mathsf{PRF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval})$ *is a secure PRF if for all PPT adversaries* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda)$ *defined below is negligible:*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda) := \Big|\Pr[\mathcal{A}^{\mathsf{Eval}(\mathsf{msk},\cdot)}(\mathsf{pp}) = 1] - \Pr[\mathcal{A}^{\mathsf{RF}(\cdot)}(\mathsf{pp}) = 1]\Big|,$$

where $\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda)$*,* $\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp})$*, and* $\mathsf{RF}(\cdot) \xleftarrow{\mathsf{R}} \mathsf{Func}(\{0,1\}^n, \mathcal{R})$*.*

2.3 Constrained Pseudorandom Function

Here, we give the syntax and security definitions for a constrained pseudorandom function (CPRF). For clarity, we will define a CPRF as a primitive that has a public parameter. However, this treatment is compatible with the standard syntax in which there is no public parameter, because it can always be contained as part of a master secret key and constrained secret keys.

Syntax. Let $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$ be a class of functions[16] where each $\mathcal{F}_{\lambda,k}$ is a set of functions with domain $\{0,1\}^k$ and range $\{0,1\}$, and the description size (when represented by a circuit) of every function in $\mathcal{F}_{\lambda,k}$ is bounded by $\mathrm{poly}(\lambda,k)$.

A CPRF for $\mathcal{F}$ consists of the five PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ where $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval})$ constitutes a PRF (where a key msk output by KeyGen is called a *master secret key*), and the last two algorithms $\mathsf{Constrain}$ and CEval have the following interfaces:

$\mathsf{Constrain}(\mathsf{pp}, \mathsf{msk}, f) \xrightarrow{\mathsf{R}} \mathsf{sk}_f$**:** This is the constraining algorithm that takes as input a public parameter pp, a master secret key msk, and a function $f \in \mathcal{F}_{\lambda,n}$, where $n = n(\lambda) = \mathrm{poly}(\lambda)$ is the input-length specified by pp. Then, it outputs a constrained key sk_f.

$\mathsf{CEval}(\mathsf{pp}, \mathsf{sk}_f, x) =: y$**:** This is the deterministic constrained evaluation algorithm that takes a public parameter pp, a constrained key sk_f, and an element $x \in \{0,1\}^n$ as input, and outputs an element $y \in \mathcal{R}$.

As in an ordinary PRF, whenever clear from the context, we will drop pp from the inputs of Eval, $\mathsf{Constrain}$, and CEval, and the executions of them are denoted as "$\mathsf{Eval}(\mathsf{msk}, x)$", "$\mathsf{Constrain}(\mathsf{msk}, f)$", and "$\mathsf{CEval}(\mathsf{sk}_f, x)$", respectively.

Correctness. For correctness of a CPRF for a function class $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$, we require that for all $\lambda \in \mathbb{N}$, $\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda)$ (which specifies the input length $n = n(\lambda) = \mathrm{poly}(\lambda)$), $\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp})$, functions $f \in \mathcal{F}_{\lambda,n}$, and inputs $x \in \{0,1\}^n$ satisfying $f(x) = 0$, we have

$$\mathsf{CEval}\Big(\ \mathsf{Constrain}(\mathsf{msk}, f), x\ \Big) = \mathsf{Eval}(\mathsf{msk}, x).$$

Remark 2.4. We note that in our definition, the role of the constraining functions f is "reversed" from that in [BW13], in the sense that correctness (i.e. the equivalence $\mathsf{Eval}(\mathsf{msk}, \cdot) = \mathsf{CEval}(\mathsf{sk}_f, \cdot)$) is required for inputs x with $f(x) = 0$, while it is required for inputs x with $f(x) = 1$ in [BW13].

Security. Here, we give the security definitions for a CPRF. We only consider CPRFs that are secure in the presence of a single constrained key, for which we consider two flavors of security: *selective single-key security* and *adaptive single-key security*. The former notion only captures security against adversaries $\mathcal{A}$ that decide the constraining function f (and the constrained key sk_f is given to $\mathcal{A}$) before seeing any evaluation result of the CPRF, while the latter notion has no such restriction and captures security against adversaries that may decide the constraining function f at any time. Also, in Section 4, as a security notion for a CPRF used as a building block, we will use the notion of *no-evaluation security*, which captures security against adversaries that have no access to the evaluation oracle. The definition below reflects these differences.

Formally, for a CPRF $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ (with input-length $n = n(\lambda)$) for a function class $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$ and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we define the single-key security experiment $\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda)$ as described in Figure 1 (left).

[16]In this paper, a "class of functions" is a set of "sets of functions". Each $\mathcal{F}_{\lambda,k}$ in $\mathcal{F}$ considered for a CPRF is a set of functions parameterized by a security parameter λ and an input-length k.

```
$\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda)$ :
    $\mathsf{coin} \xleftarrow{R} \{0,1\}$
    $\mathsf{pp} \xleftarrow{R} \mathsf{Setup}(1^\lambda)$
    $\mathsf{msk} \xleftarrow{R} \mathsf{KeyGen}(\mathsf{pp})$
    $\mathsf{RF}(\cdot) \xleftarrow{R} \mathsf{Func}(\{0,1\}^n, \mathcal{R})$
    $\mathcal{O}_{\mathsf{Chal}}(\cdot) := \begin{cases} \mathsf{Eval}(\mathsf{msk},\cdot) & \text{if } \mathsf{coin} = 1 \\ \mathsf{RF}(\cdot) & \text{if } \mathsf{coin} = 0 \end{cases}$
    $(f, \mathsf{st}_{\mathcal{A}}) \xleftarrow{R} \mathcal{A}_1^{\mathcal{O}_{\mathsf{Chal}}(\cdot),\mathsf{Eval}(\mathsf{msk},\cdot)}(\mathsf{pp})$
    $\mathsf{sk}_f \xleftarrow{R} \mathsf{Constrain}(\mathsf{msk}, f)$
    $\widehat{\mathsf{coin}} \xleftarrow{R} \mathcal{A}_2^{\mathcal{O}_{\mathsf{Chal}}(\cdot),\mathsf{Eval}(\mathsf{msk},\cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$
    Return $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$.
```

```
$\mathsf{Expt}^{\mathsf{cprf\text{-}priv}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda)$ :
    $\mathsf{coin} \xleftarrow{R} \{0,1\}$
    $\mathsf{pp} \xleftarrow{R} \mathsf{Setup}(1^\lambda)$
    $\mathsf{msk} \xleftarrow{R} \mathsf{KeyGen}(\mathsf{pp})$
    $(f_0, f_1, \mathsf{st}_{\mathcal{A}}) \xleftarrow{R} \mathcal{A}_1^{\mathsf{Eval}(\mathsf{msk},\cdot)}(\mathsf{pp})$
    $\mathsf{sk}_{f_{\mathsf{coin}}} \xleftarrow{R} \mathsf{Constrain}(\mathsf{msk}, f_{\mathsf{coin}})$
    $\widehat{\mathsf{coin}} \xleftarrow{R} \mathcal{A}_2^{\mathsf{Eval}(\mathsf{msk},\cdot)}(\mathsf{sk}_{f_{\mathsf{coin}}}, \mathsf{st}_{\mathcal{A}})$
    Return $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$.
```

Figure 1: **Left:** The experiment for defining single-key security for a CPRF. **Right:** The experiment for defining single-key privacy for a CPRF.

In the security experiment, the adversary $\mathcal{A}$'s single constraining query is captured by the function f included in the first-stage algorithm $\mathcal{A}_1$'s output. Furthermore, $\mathcal{A}_1$ and $\mathcal{A}_2$ have access to the *challenge* oracle $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ and the *evaluation* oracle $\mathsf{Eval}(\mathsf{msk},\cdot)$, where the former oracle takes $x^* \in \{0,1\}^n$ as input, and returns either the actual evaluation result $\mathsf{Eval}(\mathsf{msk}, x^*)$ or the output $\mathsf{RF}(x^*)$ of a random function, depending on the challenge bit $\mathsf{coin} \in \{0,1\}$.

We say that an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the security experiment $\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},n,\mathcal{A}}(\lambda)$ is *admissible* if $\mathcal{A}_1$ and $\mathcal{A}_2$ are PPT and respect the following restrictions:

- $f \in \mathcal{F}_{\lambda,n}$.
- $\mathcal{A}_1$ and $\mathcal{A}_2$ never make the same query twice.
- All challenge queries x^* made by $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $f(x^*) = 1$, and are distinct from any of the evaluation queries x that they submit to the evaluation oracle $\mathsf{Eval}(\mathsf{msk},\cdot)$.

Furthermore, we say that $\mathcal{A}$ is *selectively admissible* if, in addition to the above restrictions, $\mathcal{A}_1$ makes no challenge or evaluation queries. Finally, we say that $\mathcal{A}$ is a *no-evaluation adversary* if $\mathcal{A}_1$ and $\mathcal{A}_2$ are PPT, and they do not make any queries, except that $\mathcal{A}_2$ is allowed to make only a single challenge query x^* such that $f(x^*) = 1$.

Definition 2.5 (Security of CPRF). *We say that a CPRF* CPRF *for a function class $\mathcal{F}$ is* adaptively single-key secure*, if for all admissible adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda) = 1] - 1/2|$ is negligible.*

We define selective single-key security *(resp.* no-evaluation security*) of* CPRF *analogously, by replacing the phrase "all admissible adversaries $\mathcal{A}$" in the above definition with "all selectively admissible adversaries $\mathcal{A}$" (resp. "all no-evaluation adversaries $\mathcal{A}$").*

Remark 2.6. As noted in [BW13], without loss of generality we can assume that $\mathcal{A}$ makes a challenge query only once, because security for a single challenge query can be shown to imply security for multiple challenge queries via a standard hybrid argument. Hence, in the rest of the paper we only use the security experiment with a single challenge query for simplicity.

Remark 2.7. In some existing works [BW13, FKPR14, DKW16] the term "selective" is used to mean that $\mathcal{A}$ has to make a challenge query at the beginning of the security experiment. On the other hand, in this paper "selective" means that $\mathcal{A}$ has to make a constraining query at the beginning of the security experiment, which is the same definitional approach as in [BV15].

2.4 Private Constrained PRF

Here, we define an additional security notion for a CPRF called *privacy* introduced by Boneh *et al.* [BLW17]. We only consider a CPRF that achieves privacy in the presence of a single constrained key, and as in the case of (ordinary) security in the previous subsection, we consider two flavors: *selective single-key privacy* and *adaptive single-key privacy*.

Formally, for a CPRF $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ (with input-length $n = n(\lambda)$) for a function class $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$ and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we define the single-key privacy experiment $\mathsf{Expt}^{\mathsf{cprf\text{-}priv}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda)$ as described in Figure 1 (right).

In the experiment, the adversary $\mathcal{A}$'s single challenge query is captured by the function pair (f_0, f_1) output by its first-stage algorithm $\mathcal{A}_1$. Note that $\mathcal{A}_1$ and $\mathcal{A}_2$ have access to the *evaluation* oracle $\mathsf{Eval}(\mathsf{msk}, \cdot)$.

We say that an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the privacy experiment $\mathsf{Expt}^{\mathsf{cprf\text{-}priv}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda)$ is *admissible* if $\mathcal{A}_1$ and $\mathcal{A}_2$ are PPT and respect the following restrictions:

- $f_0, f_1 \in \mathcal{F}_{\lambda,n}$, and f_0 and f_1 have the same description size.
- $\mathcal{A}_1$ and $\mathcal{A}_2$ never make the same query twice.
- All evaluations queries x made by $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $f_0(x) = f_1(x)$.

Furthermore, we say that $\mathcal{A}$ is *selectively admissible* if, in addition to the above restrictions, $\mathcal{A}_1$ makes no evaluation query.

Definition 2.8 (Privacy of CPRF). *We say that a CPRF* CPRF *for a class of functions* $\mathcal{F}$ *is* adaptively single-key private*, if for all admissible adversaries* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathsf{cprf\text{-}priv}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}priv}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda) = 1] - 1/2|$ *is negligible.*

We define selective single-key privacy *of* CPRF *analogously, by replacing the phrase "all admissible adversaries* $\mathcal{A}$*" in the above definition with "all selectively admissible adversaries* $\mathcal{A}$*".*

Simpler security notion in the selective single-key setting. So far, we have defined two kinds of security notions: (ordinary) security and privacy. Here, for convenience, we introduce a simple security notion that implies both of the aforementioned security notions in the selective, single-key setting. This simple security notion makes the security analyses of our private CPRF in Section 5 simpler. (See the proof of Theorem 5.2.)

Our security notion is a simulation-based one and involves a simulator $\mathcal{S}$: We call a PPT algorithm $\mathcal{S}$ a *simulator* if it takes a public parameter pp (output by $\mathsf{Setup}(1^\lambda)$) and the description size $1^{|f|}$ of a function $f \in \mathcal{F}_{\lambda,n}$ as input,[17] and outputs some value that "looks like" a constrained key.

For a CPRF $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ (with input-length $n = n(\lambda)$) for a function class $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$, a simulator $\mathcal{S}$, and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we define the "real" single-key experiment $\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda)$ and the "ideal" single-key experiment $\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{A}}(\lambda)$ as described in Figure 2.

Note that in both games, $\mathcal{A}_2$ is given access to an oracle, which is implemented by the actual evaluation algorithm $\mathsf{Eval}(\mathsf{msk}, \cdot)$ in the real experiment, and by a random function $\mathsf{RF}(\cdot)$ in the ideal experiment.

We say that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the real/ideal experiments is *selectively admissible* if $\mathcal{A}_1$ and $\mathcal{A}_2$ are PPT and respect the following restrictions:

- $f \in \mathcal{F}_{\lambda,n}$.

[17]Our proposed private CPRFs in Section 5 are for bit-fixing functions, in which case the description size is determined by the input-length $n = n(\lambda)$ which is in turn specified in pp, and thus $1^{|f|}$ is redundant information for $\mathcal{S}$. The definition here is for a case of general function classes.

```
$\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda)$ :
    $\mathsf{pp} \xleftarrow{R} \mathsf{Setup}(1^\lambda)$
    $\mathsf{msk} \xleftarrow{R} \mathsf{KeyGen}(\mathsf{pp})$
    $(f, \mathsf{st}_\mathcal{A}) \xleftarrow{R} \mathcal{A}_1(\mathsf{pp})$
    $\mathsf{sk}_f \xleftarrow{R} \mathsf{Constrain}(\mathsf{msk}, f)$
    $\widehat{\mathsf{coin}} \xleftarrow{R} \mathcal{A}_2^{\mathsf{Eval}(\mathsf{msk},\cdot)}(\mathsf{sk}_f, \mathsf{st}_\mathcal{A})$
    Return $\widehat{\mathsf{coin}}$.
```

```
$\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{A}}(\lambda)$ :
    $\mathsf{pp} \xleftarrow{R} \mathsf{Setup}(1^\lambda)$
    $\mathsf{RF}(\cdot) \xleftarrow{R} \mathsf{Func}(\{0,1\}^n, \mathcal{R})$
    $(f, \mathsf{st}_\mathcal{A}) \xleftarrow{R} \mathcal{A}_1(\mathsf{pp})$
    $\mathsf{sk}_f \xleftarrow{R} \mathcal{S}(\mathsf{pp}, 1^{|f|})$
    $\widehat{\mathsf{coin}} \xleftarrow{R} \mathcal{A}_2^{\mathsf{RF}(\cdot)}(\mathsf{sk}_f, \mathsf{st}_\mathcal{A})$
    Return $\widehat{\mathsf{coin}}$.
```

Figure 2: Security experiments for defining our simulation-based single-key security for a CPRF. **Left:** The "real" single-key experiment. **Right:** The "ideal" single-key experiment involving a simulator $\mathcal{S}$.

- $\mathcal{A}_2$ never makes the same query twice.
- All oracle queries $x \in \{0,1\}^n$ made by $\mathcal{A}_2$ satisfy $f(x) = 1$. (i.e. $\mathsf{Eval}(\mathsf{msk}, x)$ is not trivially computable even given sk_f).

Definition 2.9 (Simulation-Security of CPRF). *We say that a CPRF* CPRF *for a function class $\mathcal{F}$ is* selectively single-key simulation-secure*, if there exists a PPT simulator $\mathcal{S}$ such that for all selectively admissible adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{cprf\text{-}sim}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{A}}(\lambda)$ defined below is negligible:*

$$\mathsf{Adv}^{\mathsf{cprf\text{-}sim}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{A}}(\lambda) = 1]\right|.$$

The following lemma guarantees that the above defined simulation-based security notion implies (ordinary) security and privacy.

Lemma 2.10. *Let* $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ *be a selectively single-key simulation-secure CPRF for a function class $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$. Then,* CPRF *is selectively single-key secure and selectively single-key private as well.*

Proof of Lemma 2.10. Let $\mathcal{S}$ be a PPT simulator that is guaranteed to exist due to the simulation-security of CPRF, which assures that the advantage is negligible for any selectively admissible adversary. Below, let $n = n(\lambda) = \mathrm{poly}(\lambda)$ denote the input-length of CPRF.

We first show the selective single-key security of CPRF, and then its selectively single-key privacy.

Selective single-key security. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any selectively admissible adversary that attacks the selective single-key security of CPRF. For simplicity, we assume that $\mathcal{A}_2$ makes a challenge query only once (see Remark 2.6), and all evaluation queries x made by $\mathcal{A}_2$ satisfy $f(x) = 1$. The latter assumption is without loss of generality, because $\mathcal{A}_2$ is given the constrained key sk_f and can by itself compute $\mathsf{Eval}(\mathsf{msk}, x)$ for x with $f(x) = 0$, by executing $\mathsf{CEval}(\mathsf{sk}_f, x)$.

Using $\mathcal{A}$, we will show how to construct a selectively admissible adversary $\mathcal{B}$ against the simulation-security of CPRF satisfying

$$\mathsf{Adv}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda) = 2 \cdot \mathsf{Adv}^{\mathsf{cprf\text{-}sim}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{B}}(\lambda), \tag{1}$$

which will complete the proof that CPRF is selectively single-key secure.

The description of $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ is fairly straightforward, and is as follows:

$\mathcal{B}_1(\mathsf{pp})$**:** $\mathcal{B}_1$ is identical to $\mathcal{A}_1$, namely, $\mathcal{B}_1$ runs $(f, \mathsf{st}_\mathcal{A}) \xleftarrow{R} \mathcal{A}_1(\mathsf{pp})$, and terminates with output $(f, \mathsf{st}_\mathcal{B} := \mathsf{st}_\mathcal{A})$.

$\mathcal{B}_2^{\mathcal{O}(\cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{B}})$**:** (where $\mathcal{O}(\cdot)$ is either $\mathsf{Eval}(\mathsf{msk}, \cdot)$ or $\mathsf{RF}(\cdot) \xleftarrow{\mathsf{R}} \mathsf{Func}(\{0,1\}^n, \mathcal{R})$) $\mathcal{B}_2$ picks $\mathcal{A}$'s challenge bit $\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$, and runs $\mathcal{A}_2(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$, where $\mathcal{B}_2$ responds to $\mathcal{A}_2$'s queries as follows:

- For the challenge query x^* from $\mathcal{A}_2$ (which by definition satisfies $f(x^*) = 1$), if $\mathsf{coin} = 1$, then $\mathcal{B}_2$ forwards x^* to its oracle $\mathcal{O}$ and receives the result y from $\mathcal{O}$. Otherwise (i.e., $\mathsf{coin} = 0$), $\mathcal{B}_2$ picks a random element $y \xleftarrow{\mathsf{R}} \mathcal{R}$. In either case, $\mathcal{B}_2$ returns y to $\mathcal{A}_2$.
- For an evaluation query x from $\mathcal{A}_2$ (which by our simplification assumption satisfies $f(x) = 1$ as well), $\mathcal{B}_2$ sends x to its oracle $\mathcal{O}$ and forwards the answer y from $\mathcal{O}$ to $\mathcal{A}_2$.

When $\mathcal{A}_2$ terminates with output $\widehat{\mathsf{coin}}$, $\mathcal{B}_2$ outputs $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$ and terminates.

The above completes the description of $\mathcal{B}$. It is straightforward to see that $\mathcal{B}$ is selectively admissible.

Consider the case that $\mathcal{B}$ runs in the real experiment $\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF},\mathcal{F},\mathcal{B}}(\lambda)$. It is straightforward to see that in this case, $\mathcal{B}$ simulates the security experiment $\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda)$ perfectly for $\mathcal{A}$. Since $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $\widehat{\mathsf{coin}} = \mathsf{coin}$), due to the definition of $\mathcal{A}$'s advantage, we have

$$\mathsf{Adv}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda) = 2 \cdot \left|\Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF},\mathcal{F},\mathcal{B}}(\lambda) = 1] - \frac{1}{2}\right|.$$

On the other hand, note that when $\mathcal{B}$ runs in the ideal experiment $\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF},\mathcal{F},\mathcal{B}}(\lambda)$, $\mathcal{A}$'s view is completely independent of coin. Indeed, since $\mathcal{B}$'s oracle is a random function $\mathsf{RF}(\cdot)$, the answer to $\mathcal{A}$'s challenge query is an output of a random function $\mathsf{RF}(\cdot)$ if $\mathsf{coin} = 1$, and is a random value in $\mathcal{R}$ if $\mathsf{coin} = 0$, and thus it is distributed uniformly over $\mathcal{R}$ regardless of coin. Furthermore sk_f and the answers to $\mathcal{A}$'s evaluation queries obviously do not contain the information of coin. This means that the probability that $\mathcal{A}$'s guess $\widehat{\mathsf{coin}}$ on coin is correct (and consequently $\mathcal{B}$ outputs 1) is exactly $1/2$, i.e., we have

$$\Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{B}}(\lambda) = 1] = \frac{1}{2}.$$

Using the above two equations in the definition of $\mathsf{Adv}^{\mathsf{cprf\text{-}sim}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{B}}(\lambda)$, we obtain Equation (1), as required. This completes the proof for the selective single-key security of CPRF.

Selective single-key privacy. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any selectively admissible adversary that attacks the selective single-key privacy of CPRF. For simplicity, we assume that all evaluation queries x made by $\mathcal{A}_2$ satisfy $f_0(x) = f_1(x) = 1$. This is without loss of generality, because $\mathcal{A}_2$ is given the constrained key sk_f and $\mathcal{A}_2$'s evaluation queries x must satisfy $f_0(x) = f_1(x)$, and thus $\mathcal{A}_2$ can by itself compute $\mathsf{Eval}(\mathsf{msk}, x)$ for x with $f_0(x) = f_1(x) = 0$, by executing $\mathsf{CEval}(\mathsf{sk}_f, x)$.

Using $\mathcal{A}$, we will show how to construct a selectively admissible adversary $\mathcal{B}$ against the simulation-security of CPRF satisfying

$$\mathsf{Adv}^{\mathsf{cprf\text{-}priv}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda) = 2 \cdot \mathsf{Adv}^{\mathsf{cprf\text{-}sim}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{B}}(\lambda), \tag{2}$$

which will complete the proof that CPRF is selectively single-key private.

The description of $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ is again fairly straightforward:

$\mathcal{B}_1(\mathsf{pp})$**:** $\mathcal{B}_1$ runs $(f_0, f_1, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}_1(\mathsf{pp})$. Then $\mathcal{B}_1$ picks $\mathcal{A}$'s challenge bit $\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$, and sets the state information $\mathsf{st}_{\mathcal{B}}$ as all the information known to $\mathcal{B}_1$. Finally, $\mathcal{B}_1$ terminates with output $(f_{\mathsf{coin}}, \mathsf{st}_{\mathcal{B}})$.

$\mathcal{B}_2^{\mathcal{O}(\cdot)}(\mathsf{sk}_{f_{\mathsf{coin}}}, \mathsf{st}_{\mathcal{B}})$**:** (where $\mathcal{O}(\cdot)$ is either $\mathsf{Eval}(\mathsf{msk}, \cdot)$ or $\mathsf{RF}(\cdot) \xleftarrow{\mathsf{R}} \mathsf{Func}(\{0,1\}^n, \mathcal{R})$) $\mathcal{B}_2$ runs $\mathcal{A}_2(\mathsf{sk}_{f_{\mathsf{coin}}}, \mathsf{st}_{\mathcal{A}})$, where $\mathcal{B}_2$ simulates $\mathcal{A}_2$'s evaluation oracle by using its own oracle $\mathcal{O}$. When $\mathcal{A}_2$ terminates with output $\widehat{\mathsf{coin}}$, $\mathcal{B}_2$ outputs $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$ and terminates.

The above completes the description of $\mathcal{B}$. It is straightforward to see that $\mathcal{B}$ is selectively admissible.

Consider the case that $\mathcal{B}$ runs in the real experiment $\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF},\mathcal{F},\mathcal{B}}(\lambda)$. It is straightforward to see that in this case, $\mathcal{B}$ simulates the privacy experiment $\mathsf{Expt}^{\mathsf{cprf\text{-}priv}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda)$ perfectly for $\mathcal{A}$. Since $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $\widehat{\mathsf{coin}} = \mathsf{coin}$), due to the definition of $\mathcal{A}$'s advantage, we have

$$\mathsf{Adv}^{\mathsf{cprf\text{-}priv}}_{\mathsf{CPRF},\mathcal{F},\mathcal{A}}(\lambda) = 2 \cdot \left|\Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF},\mathcal{F},\mathcal{B}}(\lambda) = 1] - \frac{1}{2}\right|.$$

On the other hand, note that when $\mathcal{B}$ runs in the ideal experiment $\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF},\mathcal{F},\mathcal{B}}(\lambda)$, $\mathcal{A}$'s view is completely independent of coin. Indeed, sk_f is generated by $\mathcal{S}$ and thus is independent of coin, and $\mathcal{A}$'s evaluation queries are also answered independently of coin. This means that the probability that $\mathcal{A}$'s guess $\widehat{\mathsf{coin}}$ on coin is correct (and consequently $\mathcal{B}$ outputs 1) is exactly $1/2$, i.e., we have

$$\Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{B}}(\lambda) = 1] = \frac{1}{2}.$$

Using the above two equations in the definition of $\mathsf{Adv}^{\mathsf{cprf\text{-}sim}}_{\mathsf{CPRF},\mathcal{F},\mathcal{S},\mathcal{B}}(\lambda)$, we obtain Equation (2), as required. This completes the proof for the selective single-key privacy of CPRF, and the entire proof of Lemma 2.10. ■

2.5 Correlated-Input Secure Hash Function

Here, we review the definition of a correlated-input secure hash function (CIH) that was originally introduced in Goyal *et al.* [GOR11].

Syntactically, a CIH is an efficiently computable deterministic (hash) function that has a public parameter pp that is generated by using some setup procedure, and we refer to such a pair of function and setup procedure as a *publicly parameterized function*. In this paper, we will consider a CIH that is associated with a group generator GGen. Thus, we model its setup algorithm by a "parameter generation" algorithm PrmGen that takes a group description $\mathcal{G}$ generated by GGen as input, and outputs a public parameter pp.

Formally, a publicly parameterized function CIH with respect to a group generator GGen, consists of the two PPT algorithms $(\mathsf{PrmGen}, \mathsf{Eval})$ with the following interfaces:

$\mathsf{PrmGen}(\mathcal{G}) \xrightarrow{\mathsf{R}} \mathsf{pp}$**:** This is the parameter generation algorithm that takes as input a group description $\mathcal{G}$ output by $\mathsf{GGen}(1^{\lambda})$. Then, it outputs a public parameter pp, where we assume that pp contains $\mathcal{G}$ and the descriptions of the domain $\mathcal{D}$ and the range $\mathcal{R}$.

$\mathsf{Eval}(\mathsf{pp}, x) =: y$**:** This is the deterministic evaluation algorithm that takes a public parameter pp and an element $x \in \mathcal{D}$ as input, and outputs an element $y \in \mathcal{R}$.

When there is no confusion, we will abuse the notation and denote by "$\mathsf{CIH}(\mathsf{pp}, x)$" to mean the execution of $\mathsf{Eval}(\mathsf{pp}, x)$. Furthermore, when pp is clear from the context, we may sometimes drop pp from the input of CIH, and treat as if it is a single function (e.g. "$\mathsf{CIH} : \mathcal{D} \to \mathcal{R}$") for more intuitive descriptions.

Security of CIHs. The security definition of a CIH that we use in this paper is a slightly generalized version of *correlated-input pseudorandomness* defined in [GOR11] (see Remark 2.12 for the differences from related works).

Let GGen be a group generator, and $\mathsf{CIH} = (\mathsf{PrmGen}, \mathsf{Eval})$ be a publicly parameterized function with respect to GGen. Let $\mathcal{F} = \{\mathcal{F}_{\lambda,z}\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$ be a class of functions, where each $\mathcal{F}_{\lambda,z}$ is a set of functions

```
$\mathsf{Expt}^{\mathsf{cih}}_{\mathsf{CIH},\mathsf{GGen},\mathcal{F},\mathcal{A}}(\lambda)$ :
    $\mathsf{coin} \xleftarrow{R} \{0,1\}$
    $\mathcal{G} \xleftarrow{R} \mathsf{GGen}(1^\lambda)$
    $\mathsf{pp} \xleftarrow{R} \mathsf{PrmGen}(\mathcal{G})$
    $\mathsf{RF}(\cdot) \xleftarrow{R} \mathsf{Func}(\mathcal{D},\mathcal{R})$
    $x \xleftarrow{R} \mathcal{D}$
    $\widehat{\mathsf{coin}} \xleftarrow{R} \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pp})$
    Return $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$.
```

```
$\mathcal{O}(f \in \mathcal{F}_{\lambda,\mathsf{pp}})$ :
    $y := \begin{cases} \mathsf{Eval}(\mathsf{pp}, f(x)) & \text{if } \mathsf{coin} = 1 \\ \mathsf{RF}(f(x)) & \text{if } \mathsf{coin} = 0 \end{cases}$
    Return $y$.
```

Figure 3: **Left:** The security experiment for a CIH. **Right:** The definition of the oracle $\mathcal{O}$ in the experiment.

parameterized by $\lambda \in \mathbb{N}$ and $z \in \{0,1\}^*$,[18] and it is required that for all $\lambda \in \mathbb{N}$, if $\mathcal{G} \xleftarrow{R} \mathsf{GGen}(1^\lambda)$ and $\mathsf{pp} \xleftarrow{R} \mathsf{PrmGen}(\mathcal{G})$, then the domain and the range of functions in $\mathcal{F}_{\lambda,\mathsf{pp}}$ are identical to the domain of $\mathsf{Eval}(\mathsf{pp}, \cdot)$.

For the publicly parameterized function CIH, the group generator GGen, the function class $\mathcal{F}$, and an adversary $\mathcal{A}$, we define the security experiment $\mathsf{Expt}^{\mathsf{cih}}_{\mathsf{CIH},\mathcal{F},\mathcal{A}}(\lambda)$ as described in Figure 3.

Note that in the experiment, the oracle $\mathcal{O}(\cdot)$ that $\mathcal{A}$ has access to, takes $f \in \mathcal{F}_{\lambda,\mathsf{pp}}$ as input, and returns either the evaluation result $\mathsf{CIH}(\mathsf{pp}, f(x))$ or the output $\mathsf{RF}(f(x))$ of the random function RF, depending on the challenge bit $\mathsf{coin} \in \{0,1\}$.

Definition 2.11 (Security of CIH). *Let* CIH *be a publicly parameterized function with respect to a group generator* GGen, *and let $\mathcal{F}$ be a function class. We say that* CIH *is a* CIH for $\mathcal{F}$ *(or, $\mathcal{F}$-CIH) with respect to* GGen, *if for all PPT adversaries $\mathcal{A}$, the advantage* $\mathsf{Adv}^{\mathsf{cih}}_{\mathsf{CIH},\mathsf{GGen},\mathcal{F},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{cih}}_{\mathsf{CIH},\mathsf{GGen},\mathcal{F},\mathcal{A}}(\lambda) = 1] - 1/2|$ *is negligible.*

Remark 2.12 (On the difference between CIHs and related-key secure PRFs (or PRGs)). This remark provides additional information for readers who are familiar with related primitives. We note that Definition 2.11 is essentially the same as the definition of a related-key secure pseudorandom generator (RKA-PRG) by Bellare and Cash [BC10a, Section 6, Equation (27)]. A very minor difference is that we explicitly consider public parameters in the syntax. An RKA-PRG can be seen as a generalized version of correlated-input pseudorandomness by Goyal, O'Neill, and Rao [GOR11, Definition 7]. If $\mathcal{A}$ in the security of a CIH must declare functions that will be queried to the oracle at the beginning of the experiment (i.e., selective security) and $\mathsf{RF}(f(x))$ is replaced by a uniformly random element in $\mathcal{R}$, then it is the same as correlated-input pseudorandomness. The reason why we select the name "CIH" is that it is well-suited for our usage.

Moreover, an RKA-PRF implies an RKA-PRG[19]. Therefore, the RKA-PRF (or RKA-PRG) by Bellare and Cash [BC10a, Theorem 4.2] and the RKA-PRF by Abdalla, Benhamouda, Passelègue, and Paterson [ABPP14, Theorem 7] are secure CIHs under our definition. (Of course, supported function classes are the same as theirs.)

In Sections 3 and 5 , we introduce two concrete function classes for CIHs used as building blocks in our proposed CPRFs.

[18]For a class of functions $\mathcal{F}$ considered for CIHs, we allow each member of $\mathcal{F}$ to be parameterized by not only $\lambda \in \mathbb{N}$ but also $z \in \{0,1\}^*$. The role of z is to associate the functions with a public parameter pp generated by $\mathsf{Setup}(1^\lambda)$. See the security experiment in Figure 3.

[19]If we fix an input of a PRF and view its key as a seed of a PRG, then the former can be seen as a latter.

2.6 Collision Resistant Hash Function

We will also use a standard collision resistant hash function (CRHF), and thus we recall the definition here.

Definition 2.13. *We say that a publicly parameterized function* $\mathsf{H}_{\mathsf{cr}} = (\mathsf{Setup}, \mathsf{Eval})$ *is a collision resistant hash function (CRHF) if for all PPT adversaries* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathsf{crh}}_{\mathsf{H}_{\mathsf{cr}},\mathcal{A}}(\lambda)$ *defined below is negligible:*

$$\mathsf{Adv}^{\mathsf{crh}}_{\mathsf{H}_{\mathsf{cr}},\mathcal{A}}(\lambda) := \Pr[\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda); (x, x') \xleftarrow{\mathsf{R}} \mathcal{A}(\mathsf{pp}) : \mathsf{Eval}(\mathsf{pp}, x) = \mathsf{Eval}(\mathsf{pp}, x') \wedge x \neq x']$$

For notational convenience, whenever pp, and in particular the domain $\mathcal{D}$ and range $\mathcal{R}$, are clear from the context, we treat H_{cr} as if it is a single hash function, and use a notation like "$\mathsf{H}_{\mathsf{cr}} : \mathcal{D} \to \mathcal{R}$."

3 Building Block: Correlated-input Secure Hash

In this section, we construct a CIH for group-induced functions on $\mathbb{QR}_q^n$, and prove its security under the DDH assumption. The definition of group-induced functions is given in Section 3.2.

Quadratic Residuosity groups. A safe prime q is a prime such that $q = 2p + 1$ for some p which is also a prime. We denote by $\mathbb{QR}_q$ the subgroup of all quadratic residues in $\mathbb{Z}_q^*$. From an elementary result, we have that $\mathbb{QR}_q$ is a group of prime order p. We denote by $\mathsf{SPGGen}(1^\lambda)$ a group generator that outputs a group description $(\mathbb{G}, q)$ where q is a safe prime and $q = \Omega(2^\lambda)$.

3.1 Naor-Reingold PRF and Our Variant

For constructing a CIH scheme, we will use a slight variant of the Naor-Reingold PRF [NR04]. We first recall their PRF, which can be defined with respect to any group generator GGen. We denote it by NR. The setup takes a security parameter 1^λ as input and outputs a public parameter $\mathsf{pp} = (\mathbb{G}, g, n)$ where $\mathbb{G}$ is a group of prime order q output from $\mathsf{GGen}(1^\lambda)$, g is a generator of $\mathbb{G}$, and $n \in \mathbb{N}$. The evaluation is done as follows.

$$\begin{aligned} \mathsf{NR} : \quad (\mathbb{Z}_q^*)^{n+1} \quad \times \quad \{0,1\}^n \quad &\longrightarrow \mathbb{G} \\ \Big((x_0, \ldots, x_n)\,,\,(u_1, \ldots, u_n)\Big) &\longmapsto g^{\left(x_0 \prod_{i=1}^n x_i^{u_i}\right)} \end{aligned}$$

Our variant is exactly the same as NR but with the key space $(\mathbb{Z}_q^*)^{n+1}$ being replaced by $\mathbb{QR}_q^{n+1}$ and the group generator being confined to SPGGen. In the variant, the key is sampled from the key space $\mathbb{QR}_q^{n+1}$ uniformly at random. More precisely, our PRF is operated on $\mathbb{QR}_q^{n+1} \times \{0,1\}^n \to \mathbb{G}$ with exactly the same evaluation as NR. We denote this PRF as NR'.

Recall that for the security of NR, we have the following lemma.

Proposition 3.1. *([NR04]) If the DDH assumption holds with respect to* GGen*, then* NR *with respect to* GGen *is a secure PRF.*

Regarding the security of NR', we can show the following lemma.

Theorem 3.2. *If the DDH assumption holds with respect to* SPGGen*, then* NR' *with respect to* SPGGen *is a secure PRF.*

The rest of this subsection is devoted to the proof of Theorem 3.2. Before proving the theorem, we prepare two computational assumptions, and prove that both of them are reduced to the DDH assumption.

Definition 3.3 (Quadratic Residuosity in the Exponent Assumption). *We say that the* quadratic residuosity in the exponent (QRE) assumption *holds with respect to* SPGGen*, if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{qre}}_{\mathsf{SPGGen},\mathcal{A}}(\lambda)$ defined below is negligible:*

$$\mathsf{Adv}^{\mathsf{qre}}_{\mathsf{SPGGen},\mathcal{A}}(\lambda) := \left|\Pr[\mathcal{A}(\mathcal{G}, g, g^x) = 1] - \Pr[\mathcal{A}(\mathcal{G}, g, g^{x'}) = 1]\right|,$$

where $\mathcal{G} = (\mathbb{G}, q) \xleftarrow{\mathsf{R}} \mathsf{SPGGen}(1^\lambda)$, $g \xleftarrow{\mathsf{R}} \mathbb{G}$, $x \xleftarrow{\mathsf{R}} \mathbb{QR}_q$, and $x' \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*$.

Definition 3.4 (Quadratic Exponent Decisional Diffie-Hellman Assumption). *We say that the* quadratic exponent decisional Diffie-Hellman (QE-DDH) assumption *holds with respect to* SPGGen*, if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{qe\text{-}ddh}}_{\mathsf{SPGGen},\mathcal{A}}(\lambda)$ defined below is negligible:*

$$\mathsf{Adv}^{\mathsf{qe\text{-}ddh}}_{\mathsf{SPGGen},\mathcal{A}}(\lambda) := \left|\Pr[\mathcal{A}(\mathcal{G}, g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{A}(\mathcal{G}, g, g^x, g^y, g^z) = 1]\right|,$$

where $\mathcal{G} = (\mathbb{G}, q) \xleftarrow{\mathsf{R}} \mathsf{SPGGen}(1^\lambda)$, $g \xleftarrow{\mathsf{R}} \mathbb{G}$, and $x, y, z \xleftarrow{\mathsf{R}} \mathbb{QR}_q$.

We then prove that the above assumptions can be reduced to the DDH assumption.

Lemma 3.5. *If the DDH assumption holds with respect to* SPGGen*, then the QRE assumption holds with respect to* SPGGen*.*

Proof of Lemma 3.5. Let $\mathcal{A}$ be an adversary against the QRE assumption such that $\mathsf{Adv}^{\mathsf{qre}}_{\mathsf{SPGGen},\mathcal{A}}(\lambda)$ is non-negligible. For any group description $\mathcal{G} = (\mathbb{G}, q)$ output by $\mathsf{SPGGen}(1^\lambda)$, we let

$$\epsilon(\mathcal{G}) := \Pr[g \xleftarrow{\mathsf{R}} \mathbb{G}, x \xleftarrow{\mathsf{R}} \mathbb{QR}_q : \mathcal{A}(\mathcal{G}, g, g^x) = 1] - \Pr[g \xleftarrow{\mathsf{R}} \mathbb{G}, x \xleftarrow{\mathsf{R}} \mathbb{Z}_q^* : \mathcal{A}(\mathcal{G}, g, g^x) = 1].$$

Then, by definition, we have

$$\left|\mathbb{E}_{\mathcal{G} \xleftarrow{\mathsf{R}} \mathsf{SPGGen}(1^\lambda)}[\epsilon(\mathcal{G})]\right| = \mathsf{Adv}^{\mathsf{qre}}_{\mathsf{SPGGen},\mathcal{A}}(\lambda).$$

We first construct a PPT algorithm $\mathcal{A}'$ that given $(\mathcal{G}, g, g^x)$ predicts the Legendre symbol $(\frac{x}{q})$ with probability $1/2 + \epsilon(\mathcal{G})/2$. Especially, the probability that it correctly predicts the Legendre symbol does not depend on $g \in \mathbb{G}$ or $x \in \mathbb{Z}_q^*$ and only depends on $\mathcal{G}$. The construction of $\mathcal{A}'$ is as follows.

$\mathcal{A}'(\mathcal{G}, g, X)$**:** It picks $r \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*$ and $x' \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*$, sets $g' := g^r$ and $X' := X^{rx'}$, and runs $\mathsf{coin} \xleftarrow{\mathsf{R}} \mathcal{A}(\mathcal{G}, g', X')$. If $\mathsf{coin} = 1$, then it outputs $(\frac{x'}{q})$, and otherwise it picks $\beta \xleftarrow{\mathsf{R}} \{-1, 1\}$ and outputs β.

The above completes the description of $\mathcal{A}'$. For any group description $\mathcal{G} = (\mathbb{G}, q)$, $g \in \mathbb{G}$ and $x \in \mathbb{Z}_q^*$, we have

$$\begin{aligned}
&\Pr\left[\mathcal{A}'(\mathcal{G}, g, g^x) = \left(\frac{x}{q}\right)\right] \\
&= \Pr\left[r \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*, x' \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*, g' := g^r, X' := (g^x)^{rx'} : \mathcal{A}(\mathcal{G}, g', X') = 1 \wedge \left(\frac{x'}{q}\right) = \left(\frac{x}{q}\right)\right] \\
&\quad + \Pr[r \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*, x' \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*, g' := g^r, X' := (g^x)^{rx'} : \mathcal{A}(\mathcal{G}, g', X') = 0] \cdot \frac{1}{2} \\
&= \frac{1}{2} \cdot \Pr[\widehat{g} \xleftarrow{\mathsf{R}} \mathbb{G}, \widehat{x} \xleftarrow{\mathsf{R}} \mathbb{QR}_q : \mathcal{A}(\mathcal{G}, \widehat{g}, g^{\widehat{x}}) = 1] + \Pr[\mathcal{G}, \widehat{g} \xleftarrow{\mathsf{R}} \mathbb{G}, \widehat{x} \xleftarrow{\mathsf{R}} \mathbb{Z}_q^* : \mathcal{A}(\mathcal{G}, \widehat{g}, g^{\widehat{x}}) = 0] \cdot \frac{1}{2} \\
&= \frac{1}{2} + \frac{\epsilon(\mathcal{G})}{2},
\end{aligned}$$

where in the third equality we set $\widehat{g} := g^r$ and $\widehat{x} := xx'$. We say that $\mathcal{A}'$ succeeds if $\mathcal{A}'(\mathcal{G}, g, g^x)$ outputs $(\frac{x}{q})$.

Next, we construct a PPT adversary $\mathcal{B}$ that breaks the DDH assumption. The construction of $\mathcal{B}$ is as follows.

$\mathcal{B}(\mathcal{G}, g, X, Y, Z)$: It picks $r \overset{\mathsf{R}}{\leftarrow} \mathbb{Z}_q^*$ and executes $a \overset{\mathsf{R}}{\leftarrow} \mathcal{A}'(\mathcal{G}, g, X)$, $b \overset{\mathsf{R}}{\leftarrow} \mathcal{A}'(\mathcal{G}, g, Y)$, $c \overset{\mathsf{R}}{\leftarrow} \mathcal{A}'(\mathcal{G}, g, g^r)$ and $d \overset{\mathsf{R}}{\leftarrow} \mathcal{A}'(\mathcal{G}, g, Z^r)$. If $abc = d$ holds, then it outputs 1 and otherwise it outputs 0.

The above completes the description of $\mathcal{B}$. We remark that each of the 4 executions of $\mathcal{A}'$ called by $\mathcal{B}$ succeeds with probability $1/2 + \epsilon(\mathcal{G})/2$ and these probabilities are independent. If we have $(\frac{x}{q})(\frac{y}{q}) = (\frac{z}{q})$ where $X = g^x$, $Y = g^y$ and $Z = g^z$, then $\mathcal{B}(\mathcal{G}, g, X, Y, Z)$ returns 1 if and only if the number of executions of $\mathcal{A}$ that succeed is even (i.e., that is 4, 2 or 0). This probability can be calculated as follows:

$$\begin{aligned}&\left(\frac{1}{2} + \frac{\epsilon(\mathcal{G})}{2}\right)^4 + 6\left(\frac{1}{2} + \frac{\epsilon(\mathcal{G})}{2}\right)^2\left(\frac{1}{2} - \frac{\epsilon(\mathcal{G})}{2}\right)^2 + \left(\frac{1}{2} - \frac{\epsilon(\mathcal{G})}{2}\right)^4 \\ &= \frac{1}{2} + \frac{\epsilon(\mathcal{G})^4}{2}.\end{aligned}$$

On the other hand, if we have $(\frac{x}{q})(\frac{y}{q}) \neq (\frac{z}{q})$ where $X = g^x, Y = g^y$ and $Z = g^z$, then $\mathcal{B}(\mathcal{G}, g, X, Y, Z)$ returns 1 if and only if the number of executions of $\mathcal{A}'$ that succeed is odd (i.e., that is 3 or 1). This probability can be calculated as:

$$1 - \left(\frac{1}{2} + \frac{\epsilon(\mathcal{G})^4}{2}\right) = \frac{1}{2} - \frac{\epsilon(\mathcal{G})^4}{2}.$$

Note also that we always have $(\frac{x}{q})(\frac{y}{q}) = (\frac{xy}{q})$, while if $x, y, z \overset{\mathsf{R}}{\leftarrow} \mathbb{Z}_p^*$, then $(\frac{x}{q})(\frac{y}{q}) = (\frac{z}{q})$ and $(\frac{x}{q})(\frac{y}{q}) \neq (\frac{z}{q})$ occur each with probability $1/2$. Therefore, we have

$$\Pr[g \overset{\mathsf{R}}{\leftarrow} \mathbb{G}, x, y \overset{\mathsf{R}}{\leftarrow} \mathbb{Z}_q^* : \mathcal{B}(\mathcal{G}, g, g^x, g^y, g^{xy}) = 1] = \frac{1}{2} + \frac{\epsilon(\mathcal{G})^4}{2}$$

and

$$\begin{aligned}&\Pr[g \overset{\mathsf{R}}{\leftarrow} \mathbb{G}, x, y, z \overset{\mathsf{R}}{\leftarrow} \mathbb{Z}_q^* : \mathcal{B}(\mathcal{G}, g, g^x, g^y, g^z) = 1] \\ &= \frac{1}{2} \cdot \left(\left(\frac{1}{2} + \frac{\epsilon(\mathcal{G})^4}{2}\right) + \left(\frac{1}{2} - \frac{\epsilon(\mathcal{G})^4}{2}\right)\right) \\ &= \frac{1}{2}.\end{aligned}$$

Then, we can calculate the DDH advantage of $\mathcal{B}$ as follows:

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{ddh}}_{\mathsf{SPGGen},\mathcal{B}}(\lambda)\\
&= \Big|\Pr[\mathcal{G} \xleftarrow{\mathsf{R}} \mathsf{SPGGen}(1^\lambda), g \xleftarrow{\mathsf{R}} \mathbb{G}, x, y \xleftarrow{\mathsf{R}} \mathbb{Z}_q^* : \mathcal{B}(\mathcal{G}, g, g^x, g^y, g^{xy}) = 1]\\
&\qquad - \Pr[\mathcal{G} \xleftarrow{\mathsf{R}} \mathsf{SPGGen}(1^\lambda), g \xleftarrow{\mathsf{R}} \mathbb{G}, x, y, z \xleftarrow{\mathsf{R}} \mathbb{Z}_q^* : \mathcal{B}(\mathcal{G}, g, g^x, g^y, g^z) = 1]\Big|\\
&= \Big| \mathbb{E}_{\mathcal{G}\xleftarrow{\mathsf{R}}\mathsf{SPGGen}(1^\lambda)}\Big[\Pr[g \xleftarrow{\mathsf{R}} \mathbb{G}, x, y \xleftarrow{\mathsf{R}} \mathbb{Z}_q^* : \mathcal{B}(\mathcal{G}, g, g^x, g^y, g^{xy}) = 1]\\
&\qquad - \Pr[g \xleftarrow{\mathsf{R}} \mathbb{G}, x, y, z \xleftarrow{\mathsf{R}} \mathbb{Z}_q^* : \mathcal{B}(\mathcal{G}, g, g^x, g^y, g^z) = 1] \Big]\Big|\\
&= \mathbb{E}_{\mathcal{G}\xleftarrow{\mathsf{R}}\mathsf{SPGGen}(1^\lambda)}\left[\frac{\epsilon(\mathcal{G})^4}{2}\right]\\
&\geq \frac{\left(\mathbb{E}_{\mathcal{G}\xleftarrow{\mathsf{R}}\mathsf{SPGGen}(1^\lambda)}[\epsilon(\mathcal{G})]\right)^4}{2}\\
&= \frac{\mathsf{Adv}^{\mathsf{qre}}_{\mathsf{SPGGen},\mathcal{A}}(\lambda)^4}{2},
\end{aligned}
$$

where the inequality is due to Jensen's inequality. The above inequality shows that if $\mathsf{Adv}^{\mathsf{qre}}_{\mathsf{SPGGen},\mathcal{A}}(\lambda)$ is non-negligible, then so is $\mathsf{Adv}^{\mathsf{ddh}}_{\mathsf{SPGGen},\mathcal{B}}(\lambda)$, and thus proves the lemma. ■

Remark 3.6. At first glance, it seems strange that $\mathcal{B}$ introduces a "dummy" element g^r. This is to make the advantage of $\mathcal{B}$ quartic in $\epsilon(\mathcal{G})$ so that we can apply Jensen's inequality. (We note that we cannot apply Jensen's inequality if that is cubic since $\epsilon(\mathcal{G})$ may take a negative value.) Though a tighter reduction may be possible using the random self-reducibility of an instance of the QRE problem, we give the above reduction for simplicity.

Lemma 3.7. *If the DDH assumption holds with respect to* SPGGen*, then the QE-DDH assumption holds with respect to* SPGGen.

Proof of Lemma 3.7. We have the following sequence of indistinguishability on quadruples of random variables:

$$
\begin{aligned}
&\big((g, g^x, g^y, g^{xy}) \mid g \xleftarrow{\mathsf{R}} \mathbb{G}, x \xleftarrow{\mathsf{R}} \mathbb{QR}_q, y \xleftarrow{\mathsf{R}} \mathbb{QR}_q\big) &&\\
\approx_c\ &\big((g, g^x, g^y, g^{xy}) \mid g \xleftarrow{\mathsf{R}} \mathbb{G}, x \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*, \ y \xleftarrow{\mathsf{R}} \mathbb{QR}_q\big) && \text{(from QRE, on } x)\\
\approx_c\ &\big((g, g^x, g^y, g^{xy}) \mid g \xleftarrow{\mathsf{R}} \mathbb{G}, x \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*, \ y \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*\big) && \text{(from QRE, on } y)\\
\approx_c\ &\big((g, g^x, g^y, g^{z}) \mid g \xleftarrow{\mathsf{R}} \mathbb{G}, x \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*, \ y \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*, \ z \xleftarrow{\mathsf{R}} \mathbb{Z}_q^*\big) && \text{(from DDH)}\\
\approx_c\ &\big((g, g^x, g^y, g^{z}) \mid g \xleftarrow{\mathsf{R}} \mathbb{G}, x \xleftarrow{\mathsf{R}} \mathbb{QR}_q, y \xleftarrow{\mathsf{R}} \mathbb{QR}_q, z \xleftarrow{\mathsf{R}} \mathbb{QR}_q\big) && \text{(from QRE, on } x, y, z).
\end{aligned}
$$

From this and Lemma 3.5, we can conclude this lemma. ■

Due to Lemma 3.5 and Lemma 3.7, we have that Theorem 3.2 follows directly from the following lemma.

Lemma 3.8. *If the QE-DDH and QRE assumptions hold with respect to* SPGGen*, then* NR′ *with respect to* SPGGen *is a secure PRF.*

The proof can be done in a similar manner to the original proof of NR [NR04], albeit replacing the DDH assumption with the QE-DDH assumption. We note one difference from the original NR proof is that the reduction incurs a multiplicative loss by the number Q of an adversary's evaluation queries, since an instance of the QE-DDH assumption does not have random self-reducibility. That is, the structure of the proof proceeds more similarly to the proof for the Goldreich-Goldwasser-Micali PRF [GGM86].

Proof of Lemma 3.8. Let $\mathcal{A}$ be an adversary that attacks the PRF-security of NR', and let Q be the number of evaluation queries made by $\mathcal{A}$. For $\ell \in [0, n], j \in [1, Q]$, we define an intermediate game as follows.

$\mathsf{Game}_{\ell,j}$**:** At the beginning, the game picks $x_\ell, \ldots, x_n \xleftarrow{\mathsf{R}} \mathbb{QR}_q$ and prepares an empty list L. For the k-th evaluation query from $\mathcal{A}$, say, for $\vec{u} = (u_1, \ldots, u_n) \in \{0,1\}^n$, the game does as follows. Denote $\vec{u}_{|\ell} = (u_1, \ldots, u_\ell)$ and $\vec{u}_{|0} = \epsilon$ (the empty string). It checks if a pair $(\vec{u}_{|\ell}, t_{\vec{u}_{|\ell}})$ has already been in the list L. If not, it does as follows.

- If $k \le j$, it picks $t_{\vec{u}_{|\ell}} \xleftarrow{\mathsf{R}} \mathbb{QR}_q$ and stores $(\vec{u}_{|\ell}, t_{\vec{u}_{|\ell}})$ into L.
- If $k > j$, it picks $t_{\vec{u}_{|\ell-1}} \xleftarrow{\mathsf{R}} \mathbb{QR}_q$ and sets $t_{\vec{u}_{|\ell-1}\|v} = t_{\vec{u}_{|\ell-1}} \times x_\ell^v$ for $v \in \{0,1\}$. It stores $(\vec{u}_{|\ell-1}\|v, t_{\vec{u}_{|\ell-1}\|v})$ for both $v \in \{0,1\}$ into L.

It returns $g^{t_{\vec{u}_{|\ell}} \prod_{i=\ell+1}^{n} x_i^{u_i}}$ to $\mathcal{A}$ as the response to the k-th evaluation query.

$\mathsf{Game}_{\mathsf{final}}$**:** The game simply returns a random element in $\mathbb{G}$ for each evaluation query from $\mathcal{A}$.

It is clear that $\mathsf{Game}_{0,Q}$ is exactly the same as the PRF security game. On the other hand, in $\mathsf{Game}_{\mathsf{final}}$, all the returned values are completely random, and hence, the adversary $\mathcal{A}$ has zero advantage. Let $\mathsf{Adv}_{\ell,j}$ be the advantage of the adversary $\mathcal{A}$ in $\mathsf{Game}_{\ell,j}$. We claim and prove the following.

- For $\ell \in [0, n], j \in [2, Q]$, we have that $\mathsf{Adv}_{\ell,j-1} \approx \mathsf{Adv}_{\ell,j}$ under the QE-DDH assumption. The proof is as follows. We observe that the two games differ at most at the response to the j-th evaluation query, say, for $\vec{u}^\star$. Indeed, they differ only if the pair $(\vec{u}_{|\ell}, t_{\vec{u}_{|\ell}})$ is not in L at the time of the j-th evaluation query. In such a case, we simulate the games by implicitly setting

$$t_{\vec{u}^\star_{|\ell-1}} = x, \qquad x_\ell = y, \qquad t_{\vec{u}^\star_{|\ell}} = z, \tag{3}$$

  where (g, g^x, g^y, g^z) is the QE-DDH challenge. We can see that if $z = xy$ then this simulates $\mathsf{Game}_{\ell,j-1}$, while if $z \xleftarrow{\mathsf{R}} \mathbb{QR}_q$ then this simulates $\mathsf{Game}_{\ell,j}$. Therefore, if the difference of $\mathcal{A}$'s advantage in the two games is non-negligible, it can be used to break the QE-DDH assumption.

- For $\ell \in [1, n]$, we have that $\mathsf{Adv}_{\ell-1,Q} \approx \mathsf{Adv}_{\ell,1}$ under the QE-DDH assumption. We observe that the two games differ only at the response to the first query, say, for $\vec{u}^\star$. We thus simulate the games by again setting exactly as Eq. (3). We can see that if $z = xy$ then this simulates $\mathsf{Game}_{\ell-1,Q}$, while if $z \xleftarrow{\mathsf{R}} \mathbb{QR}_q$ then this simulates $\mathsf{Game}_{\ell,1}$. Therefore, if the difference of $\mathcal{A}$'s advantage in the two games is non-negligible, it can be used to break the QE-DDH assumption.

- We have $\mathsf{Adv}_{n,Q} \approx \mathsf{Adv}_{\mathsf{final}}$ under the QRE assumption. In game $\mathsf{Game}_{n,Q}$, the answer to each evaluation query is of the form g^z where $z \xleftarrow{\mathsf{R}} \mathbb{QR}_q$. This can be modified to a random element in $\mathbb{G}$ using the QRE assumption (applying it Q times query-by-query).

Combining all the hybrids, this concludes the proof. ■

3.2 Bellare-Cash CIH Construction and Our Variant

CIH for group-induced functions. The notion of *(component-wise) group-induced functions with respect to a group generator* GGen is a function class $\Psi^{\mathsf{g\text{-}indc}} = \{\Psi^{\mathsf{g\text{-}indc}}_{\lambda,z}\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$ satisfying the following property for all $(\lambda, z) \in \mathbb{N} \times \{0,1\}^*$: If z can be parsed as a tuple $(\mathcal{G}, n, z')$ so that $\mathcal{G} = (\mathbb{G}, q)$ is a group description output by $\mathsf{GGen}(1^\lambda)$, $n \in \mathbb{N}$, and $z' \in \{0,1\}^*$, then we have $\Psi^{\mathsf{g\text{-}indc}}_{\lambda,z} = \{\psi_{\vec{a}} : (\mathbb{Z}_q^*)^n \to (\mathbb{Z}_q^*)^n \mid \vec{a} \in (\mathbb{Z}_q^*)^n\}$, where for each $\vec{a} \in (\mathbb{Z}_q^*)^n$, $\psi_{\vec{a}}(\vec{x}) := \vec{a} \star \vec{x} \in (\mathbb{Z}_q^*)^n$ and $\star$ denotes the component-wise multiplication in $\mathbb{Z}_q^*$.

CIH Construction. We are now ready to describe our CIH for the (component-wise) group-induced functions with respect to SPGGen. It can be considered as a variant of the hash function by Bellare and Cash [BC10a], denoted as $\mathsf{CIH}_{\mathsf{BC}}$, which we recall as follows. The public parameter consists of the description of $\mathbb{G}$, which is a cyclic group of order q, output from the group generator $\mathsf{GGen}(1^\lambda)$, a generator g of $\mathbb{G}$, and a collision-resistant hash function $\mathsf{H}_{\mathsf{cr}} : \mathbb{G}^{n+1} \to \{0,1\}^{n-2}$. The evaluation is defined as follows.

$$\begin{aligned} \mathsf{CIH}_{\mathsf{BC}} : (\mathbb{Z}_q^*)^{n+1} &\longrightarrow \mathbb{G} \\ \vec{x} &\longmapsto \mathsf{NR}\Big(\vec{x},\ 11\|\mathsf{H}_{\mathsf{cr}}\Big(\mathsf{NR}(\vec{x},e_0),...,\mathsf{NR}(\vec{x},e_n)\Big)\Big) \end{aligned}$$

where $e_0 = 0^n$ and $e_k = 0^{k-1}\|1\|0^{n-k}$ for $k \in [n]$.

Our variant of CIH is exactly the same as $\mathsf{CIH}_{\mathsf{BC}}$ but the domain is restricted. In more detail, our CIH is operated on $\mathbb{QR}_q^{n+1} \to \mathbb{G}$ with exactly the same evaluation as $\mathsf{CIH}_{\mathsf{BC}}$. Note that due to our restriction on the domain, the NR evaluation inside the function is thus restricted to NR'. We denote this CIH as $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$.

Theorem 3.9. *If the DDH assumption holds with respect to* SPGGen *and* H_{cr} *is a CRHF, then* $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$ *is a secure CIH for the (component-wise) group-induced functions with respect to* SPGGen.

Due to Theorem 3.2, which states that NR' is a secure PRF under the DDH assumption, we have that Theorem 3.9 follows directly from the following lemma.

Lemma 3.10. *If* NR' *with respect to* SPGGen *is a secure PRF and* H_{cr} *is a CRHF, then* $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$ *is a secure CIH for the (component-wise) group-induced functions with respect to* SPGGen.

Before proving Lemma 3.10, we recall some useful properties of NR from [BC10a]. Recall that for $\vec{a}, \vec{b} \in (\mathbb{Z}_q^*)^{n+1}$, we use the operator $\star$ to denote the component wise group operation, i.e., $\vec{a} \star \vec{b} = (\, a_0 \cdot b_0,\ ...,\ a_n \cdot b_n\,)$, where we parse $\vec{a} = (a_0, ..., a_n), \vec{b} = (b_0, ..., b_n)$.

Proposition 3.11. *([BC10a]) There exists an efficient algorithm* T *satisfying the following two properties.*

Key-malleability: *For any* $\vec{x}, \vec{a} \in (\mathbb{Z}_q^*)^{n+1}$, $\mathsf{inp} \in \{0,1\}^n$, *we have that* $\mathsf{T}(\vec{a}, \mathsf{inp}, \mathsf{NR}(\vec{x}, \mathsf{inp})) = \mathsf{NR}(\vec{a} \star \vec{x}, \mathsf{inp})$.

Uniformity: *For any* $\vec{a}_1, ..., \vec{a}_m \in (\mathbb{Z}_q^*)^{n+1}$ *and pairwise distinct* $\mathsf{inp}_1, ..., \mathsf{inp}_m \in \{0,1\}^n$, *we have that* $\Big(\mathsf{T}(\vec{a}_1, \mathsf{inp}_1, \mathsf{RF}(\mathsf{inp}_1)), ..., \mathsf{T}(\vec{a}_m, \mathsf{inp}_m, \mathsf{RF}(\mathsf{inp}_m))\Big)$ *is uniformly distributed in* $\mathbb{G}^m$ *where* $\mathsf{RF}(\cdot) \overset{\mathsf{R}}{\leftarrow} \mathsf{Func}(\{0,1\}^n, \mathbb{G})$.

Proposition 3.12. *([BC10a]) Let* $e_0 = 0^n$ *and* $e_k = 0^{k-1}\|1\|0^{n-k}$ *for* $k \in [n]$. *For all* $\vec{a}, \vec{a}', \vec{x} \in (\mathbb{Z}_q^*)^{n+1}$ *such that* $\vec{a} \neq \vec{a}'$, *we have that* $(\mathsf{NR}(\vec{a} \star \vec{x}, e_0), ..., \mathsf{NR}(\vec{a} \star \vec{x}, e_n)) \neq (\mathsf{NR}(\vec{a}' \star \vec{x}, e_0), ..., \mathsf{NR}(\vec{a}' \star \vec{x}, e_n))$.

We are now ready to prove Lemma 3.10.

Proof of Lemma 3.10. Let $\mathcal{A}$ be an adversary that attacks the security of $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$, and let Q be the number of evaluation queries made by $\mathcal{A}$. For simplicity and without loss of generality, we assume that $\mathcal{A}$ does not make the same query twice. We construct an adversary $\mathcal{B}$ that breaks the PRF-security of NR'. The construction of $\mathcal{B}$ is as follows.

$\mathcal{B}^{\mathcal{O}}(1^\lambda)$**:** $\mathcal{B}$ first makes evaluation queries e_k to its own oracle $\mathcal{O}$ to obtain s_k for $k \in \{0,1,...,n\}$. It then runs $\mathcal{A}(1^\lambda)$ and answers $\mathcal{A}$'s queries as follows.

When $\mathcal{A}$ issues the i-th evaluation query $\vec{a}^{(i)}$, $\mathcal{B}$ responds as follows. It computes $t_k^{(i)} := \mathsf{T}(\vec{a}^{(i)}, e_k, s_k)$ for all $k \in \{0,1,...,n\}$ and $u^{(i)} := \mathsf{H}_{\mathsf{cr}}(t_0^{(i)}, ..., t_n^{(i)})$. If there exists $i' < i$ such that

$u^{(i')} = u^{(i)}$, then $\mathcal{B}$ aborts and outputs a random bit. Otherwise $\mathcal{B}$ makes an evaluation query $11\|u^{(i)}$ to its own oracle to obtain $v^{(i)}$. $\mathcal{B}$ then computes $y^{(i)} := \mathsf{T}(\vec{a}^{(i)}, 11\|u^{(i)}, v^{(i)})$ and gives $y^{(i)}$ to $\mathcal{A}$ as the response to $\mathcal{A}$'s k-th evaluation query.

When $\mathcal{A}$ halts, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

The above completes the description of $\mathcal{B}$.

We now prove that $\mathcal{B}$ breaks the PRF-security of NR' if $\mathcal{A}$ breaks the CIH-security of $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$. First, we observe that the probability that $\mathcal{B}$ aborts is negligible. This follows from Proposition 3.12 and the collision resistance of H_{cr}. That is, if $\mathcal{B}$ aborts, we can construct an adversary $\mathcal{B}'$ that breaks the security of CRHF. Hence, in what follows, we assume that $\mathcal{B}$ does not abort. If $\mathcal{B}$'s oracle $\mathcal{O}$ is the actual PRF-evaluation algorithm, i.e., it is $\mathsf{NR}(\vec{x}, \cdot)$ for a randomly chosen $\vec{x} \xleftarrow{\mathsf{R}} \mathbb{QR}_q^{n+1}$, then $\mathcal{B}$'s response $y^{(i)}$ for $\mathcal{A}$'s i-th evaluation query, satisfies the following equality for every $i \in [Q]$:

$$\begin{aligned}
y^{(i)} &= \mathsf{T}(\vec{a}^{(i)}, 11\|u^{(i)}, v^{(i)}) \\
&= \mathsf{T}(\vec{a}^{(i)}, 11\|u^{(i)}, \mathsf{NR}(\vec{x}, 11\|u^{(i)})) \\
&= \mathsf{NR}(\vec{a}^{(i)} \star \vec{x}, 11\|u^{(i)}) \\
&= \mathsf{NR}(\vec{a}^{(i)} \star \vec{x}, 11\|\mathsf{H}_{\mathsf{cr}}(t_0^{(i)}, ..., t_n^{(i)})) \\
&= \mathsf{NR}(\vec{a}^{(i)} \star \vec{x}, 11\|\mathsf{H}_{\mathsf{cr}}(\mathsf{T}(\vec{a}^{(i)}, e_0, s_0), ..., \mathsf{T}(\vec{a}^{(i)}, e_n, s_n))) \\
&= \mathsf{NR}(\vec{a}^{(i)} \star \vec{x}, 11\|\mathsf{H}_{\mathsf{cr}}(\mathsf{T}(\vec{a}^{(i)}, e_0, \mathsf{NR}(\vec{x}, e_0)), ..., \mathsf{T}(\vec{a}^{(i)}, e_n, \mathsf{NR}(\vec{x}, e_n)))) \\
&= \mathsf{NR}(\vec{a}^{(i)} \star \vec{x}, 11\|\mathsf{H}_{\mathsf{cr}}(\mathsf{NR}(\vec{a}^{(i)} \star \vec{x}, e_0), ..., \mathsf{NR}(\vec{a}^{(i)} \star \vec{x}, e_n))) \\
&= \mathsf{CIH}_{\widetilde{\mathsf{BC}}}(\vec{a}^{(i)} \star \vec{x}).
\end{aligned}$$

This means that $\mathcal{B}$ correctly simulates the security experiment for $\mathsf{CIH}'_{\mathsf{BC}}$ for the case $\mathsf{coin} = 1$.

It remains to prove that if $\mathcal{B}$'s oracle $\mathcal{O}$ is a truly random function, then $y^{(i)}$ for all $i \in [Q]$ are independently and uniformly random, and thus $\mathcal{B}$ correctly simulates the security experiment for $\mathsf{CIH}'_{\mathsf{BC}}$ for the case $\mathsf{coin} = 0$. Note that the evaluation queries made by $\mathcal{B}$ to its own oracle are $e_0, e_1, ..., e_n, 11\|u^{(1)}, ..., 11\|u^{(Q)}$. It is clear that they are pairwise distinct when $\mathcal{B}$ does not abort. Therefore, by the uniformity of T we can conclude that $y^{(i)} = \mathsf{T}(\vec{a}^{(i)}, 11\|u^{(i)}, \mathcal{O}(11\|u^{(i)}))$ for $i \in [Q]$ are independently and uniformly random. We showed that $\mathcal{B}$ breaks the PRF security if $\mathcal{A}$ breaks the CIH-security as long as the collision does not happen. That is, we proved that $\mathsf{Adv}^{\mathsf{cih}}_{\mathsf{CIH}_{\widetilde{\mathsf{BC}}}, \mathsf{SPGGen}, \Psi^{\mathsf{g\text{-}indc}}, \mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{NR}', \mathsf{SPGGen}, \mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{crh}}_{\mathsf{H}_{\mathsf{cr}}, \mathcal{B}'}(\lambda)$. ■

Now, Theorem 3.9 follows by combining Theorem 3.2 and Lemma 3.10.

4 CPRF for $\mathbf{NC}^1$ Circuits

In this section, we first show a construction of a CPRF for $\mathbf{NC}^1$ circuits with no-evaluation security, where an adversary is not allowed to make evaluation queries (Section 4.1). We then show that by combining the scheme with our CIH in Section 3, we can upgrade the security to the selective single-key security, where the adversary is allowed to make evaluation queries unbounded times after it is given the secret key (Section 4.2). We also show that the adaptive security can be achieved in the random oracle model (Section 4.3).

4.1 Our Basic Constrained PRF

Here, we give a construction of a CPRF for $\mathbf{NC}^1$ with no-evaluation security. We then prove that the scheme has additional properties that we call semi-evaluability and universality. These properties will be used in Section 4.2 and Section 4.3.

Notations.

In the following, we will sometimes abuse notation and evaluate a boolean circuit $C(\cdot) : \{0,1\}^\ell \to \{0,1\}$ on input $y \in \mathbb{R}^\ell$ for some ring $\mathbb{R}$. The evaluation is done by regarding $C(\cdot)$ as the arithmetic circuit whose AND gates $(y_1, y_2) \mapsto y_1 \wedge y_2$ being changed to the multiplication gates $(y_1, y_2) \mapsto y_1 y_2$, NOT gates $y \mapsto \neg y$ changed to the gates $y \mapsto 1 - y$, and the OR gates $(y_1, y_2) \mapsto y_1 \vee y_2$ changed to the gates $(y_1, y_2) \mapsto y_1 + y_2 - y_1 y_2$. It is easy to observe that if the input is confined within $\{0,1\}^\ell \subseteq \mathbb{R}$, the evaluation of the arithmetized version of $C(\cdot)$ equals to that of the binary version. (Here, we identify ring elements $0, 1 \in \mathbb{R}$ with the binary bit.) In that way, we can regard $C(\cdot)$ as an ℓ-variate polynomial over $\mathbb{R}$. The degree of $C(\cdot)$ is defined as the maximum of the total degree of all the polynomials that appear during the computation.

Class of Functions.

Let $n = \mathrm{poly}(\lambda)$, $z(n) = \mathrm{poly}(n)$, and $d(n) = O(\log n)$ be parameters. The function class that will be dealt with by the scheme is denoted by $\mathcal{F}^{\mathbf{NC}^1} = \{\mathcal{F}^{\mathbf{NC}^1}_{\lambda,n(\lambda)}\}_{\lambda\in\mathbb{N}}$, where $\mathcal{F}^{\mathbf{NC}^1}_{\lambda,n}$ consists of (Boolean) circuits f whose input size is $n(\lambda)$, the description size is $z(n)$, and the depth is $d(n)$. We can set the parameters arbitrarily large as long as they do not violate the asymptotic bounds above, and thus the function class corresponds to $\mathbf{NC}^1$ circuits with bounded size. The following lemma will be helpful when describing our scheme.

Lemma 4.1. *Let $n = \mathrm{poly}(\lambda)$. There exists a family of universal circuit $\{U_n\}_{n\in\mathbb{N}}$ of degree $D(\lambda) = \mathrm{poly}(\lambda)$ such that $U_n(f, x) = f(x)$ for any $f \in \mathcal{F}^{\mathbf{NC}^1}_{\lambda,n(\lambda)}$ and $x \in \{0,1\}^n$.*

Proof. Due to the result by Cook and Hoover [CH85], there exists a universal circuit $U_n(\cdot)$ of depth $O(d) = O(\log n)$ and size $\mathrm{poly}(n, z, d) = \mathrm{poly}(\lambda)$. Furthermore, the degree of $U_n(\cdot)$ is bounded by $2^{O(d)} = \mathrm{poly}(n) = \mathrm{poly}(\lambda)$. ■

Construction.

Let $\mathcal{F}^{\mathbf{NC}^1} = \{\mathcal{F}^{\mathbf{NC}^1}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$ be the family of the circuit defined as above and $\{U_n\}_{n\in\mathbb{N}}$ be the family of the universal circuit defined in Lemma 4.1. Let the parameter $D(\lambda)$ be the degree of the universal circuit (chosen as specified in Lemma 4.1). Since we will fix n in the construction, we drop the subscripts and just denote $\mathcal{F}^{\mathbf{NC}^1}$ and U in the following. We also let HGen be any group generator. The description of our CPRF $\mathsf{CPRF}_{\mathsf{NE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ is given below.

$\mathsf{Setup}(1^\lambda)$**:** It obtains the group description $\mathcal{H} = (\mathbb{H}, p)$ by running $\mathcal{H} \xleftarrow{\mathsf{R}} \mathsf{HGen}(1^\lambda)$. It then outputs the public parameter $\mathsf{pp} := \mathcal{H}$.[20]

$\mathsf{KeyGen}(\mathsf{pp})$**:** It chooses $(b_1, ..., b_z) \xleftarrow{\mathsf{R}} \mathbb{Z}_p^z$, $\alpha \xleftarrow{\mathsf{R}} \mathbb{Z}_p^*$, and $g, h_1, \ldots, h_n \xleftarrow{\mathsf{R}} \mathbb{H}$. Then it outputs $\mathsf{msk} := (b_1, \ldots, b_z, \alpha, g, h_1, \ldots, h_n)$.

$\mathsf{Eval}(\mathsf{msk}, x)$**:** Given input $x \in \{0,1\}^n$, it computes and outputs

$$X := g^{U((b_1,\ldots,b_z),(x_1,\ldots,x_n))/\alpha} \cdot \prod_{i\in[n]} h_i^{x_i}.$$

$\mathsf{Constrain}(\mathsf{msk}, f)$**:** It first parses $(b_1, ..., b_z, \alpha, g, h_1, \ldots, h_n) \leftarrow \mathsf{msk}$. Then it sets

$$b'_i := (b_i - f_i)\alpha^{-1} \mod p \quad \text{for } i \in [z]$$

[20] Here, we intentionally use the symbol $\mathbb{H}$ and HGen instead of $\mathbb{G}$ and GGen. Looking ahead, in Section 4.2, the latter symbols will be used to represent yet another group of order q and corresponding group generator. There, we should require $\mathbb{H}$ to be $\mathbb{QR}_q$.

where f_i is the i-th bit of the binary representation of f. It then outputs

$$\mathsf{sk}_f := (f, b'_1, \ldots, b'_z, g, g^{\alpha}, \ldots, g^{\alpha^{D-1}}, h_1, \ldots, h_n).$$

$\mathsf{CEval}(\mathsf{sk}_f, x)$: It parses $(f, b'_1, \ldots, b'_z, g, g^{\alpha}, \ldots, g^{\alpha^{D-1}}, h_1, \ldots, h_n) \leftarrow \mathsf{sk}_f$. As proved in Lemma 4.2 below, it is possible to efficiently compute $\{c_i\}_{i\in[D]}$ that satisfies

$$U((b_1, \ldots, b_z), (x_1, \ldots, x_n)) = f(x) + \sum_{i=1}^{D} c_i \alpha^i \tag{4}$$

from sk_f and x. If $f(x) = 0$, it computes $X := \prod_{i=1}^{D} (g^{\alpha^{i-1}})^{c_i} \cdot \prod_{j=1}^{n} h_j^{x_j}$ and outputs X. Otherwise it outputs $\perp$.

Correctness and semi-evaluability.

In order to prove the correctness, it suffices to show the following lemma.

Lemma 4.2. *Given* sk_f, x, *one can efficiently compute* $\{c_i\}_{i\in[D]}$ *satisfying Eq.(4).*

Proof. The algorithm evaluates the circuit $U(\cdot)$ on input $(b'_1\mathsf{Z} + f_1, \ldots, b'_z\mathsf{Z} + f_z, x_1, \ldots, x_n)$ to obtain $\{c_i\}_{i\in\{0,1,\ldots,D\}}$ such that

$$U(b'_1\mathsf{Z} + f_1, \ldots, b'_z\mathsf{Z} + f_z, x_1, \ldots, x_n) = c_0 + \sum_{i\in[D]} c_i \mathsf{Z}^i \tag{5}$$

where Z denotes the indeterminant of the polynomial ring $\mathbb{Z}_p[\mathsf{Z}]$. Note that the computation is done over the ring $\mathbb{Z}_p[\mathsf{Z}]$ and can be efficiently performed, since we have $D = \mathrm{poly}(\lambda)$. We prove that $\{c_i\}_{i\in[D]}$ actually satisfies Equation (4). To see this, we first observe that by setting $\mathsf{Z} = 0$ in Equation (5), we obtain $c_0 = U(f_1, \ldots, f_z, x_1 \ldots, x_n) = f(x)$. To conclude, we further observe that by setting $\mathsf{Z} = \alpha$ in Equation (5), we recover Equation (4), since we have $b_j = b'_j\alpha + f_j$ by the definition of b'_j. This completes the proof of the lemma. ■

The lemma implies an additional property of the CPRF that we call *semi-evaluability*, which will be useful in our security proof. In more detail, we have the following lemma:

Lemma 4.3. *We say that a CPRF scheme has semi-evaluability if there exist deterministic and efficient algorithms* SEval *and* Aux *satisfying the following property. For all* $f \in \mathcal{F}^{\boldsymbol{NC}^1}$ *and* x *such that* $f(x) = 1$ *and for all possible* $\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp}), \mathsf{sk}_f \xleftarrow{\mathsf{R}} \mathsf{Constrain}(\mathsf{msk}, f)$, *we have*

$$\mathsf{SEval}(\mathsf{sk}_f, x) \cdot \mathsf{Aux}(\mathsf{msk}) = \mathsf{Eval}(\mathsf{msk}, x).$$

In the above, "$\cdot$" indicates the group operation on $\mathbb{H}$.

Proof. We define SEval and Aux as follows.

$\mathsf{SEval}(\mathsf{sk}_f, x)$: It first parses $(f, b'_1, \ldots, b'_z, g, g^{\alpha}, \ldots, g^{\alpha^{D-1}}, h_1, \ldots, h_n) \leftarrow \mathsf{sk}_f$. It then compute $\{c_j\}_{j\in[D]}$ that satisfies Equation (4). It finally computes $X' := \prod_{i=1}^{D} (g^{\alpha^{i-1}})^{c_i} \cdot \prod_{j\in[n]} h_j^{x_j}$ and outputs X'.

$\mathsf{Aux}(\mathsf{msk})$: It parses $(b_1, \ldots, b_z, \alpha, g, h_1, \ldots, h_n) \leftarrow \mathsf{msk}$ and outputs $g^{1/\alpha}$.

The lemma readily follows from Equation (4) and $f(x) = 1$. ■

Universality.

The following lemma indicates that the above scheme can be seen as a universal hashing. The only reason why we need $h_1, \dots, h_n$ in pp is to ensure this property. Formally, we have the following lemma. The lemma will be used later in this section.

Lemma 4.4. *For all $x, x' \in \{0,1\}^n$ with $x \neq x'$ and pp output by $\mathsf{Setup}(1^\lambda)$, we have*

$$\Pr[\, \mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp}) \;:\; \mathsf{Eval}(\mathsf{msk}, x) = \mathsf{Eval}(\mathsf{msk}, x') \,] = \frac{1}{p}.$$

Proof. Since $x \neq x'$, there exists an index i such that $x_i \neq x'_i$. Let us fix msk except for h_i. Then, we can see that there exists a unique h_i such that $\mathsf{Eval}(\mathsf{msk}, x) = \mathsf{Eval}(\mathsf{msk}, x')$ holds. Since h_i is chosen uniformly at random from $\mathbb{H}$, the lemma follows. ■

No-evaluation security.

Theorem 4.5. *If the $(D-1)$-DDHI assumption holds with respect to HGen, then $\mathsf{CPRF}_{\mathsf{NE}}$ defined above satisfies no-evaluation security as a CPRF for the circuit class $\mathcal{F}^{\mathbf{NC}^1}$.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any no-evaluation adversary that attacks the no-evaluation security of CPRF. We prove the above theorem by considering the following sequence of games.

Game 0**:** This is the real single-key security experiment $\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF}_{\mathsf{NE}}, \mathcal{F}^{\mathbf{NC}^1}, \mathcal{A}}(\lambda)$ against the no-evaluation adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. Namely,

```
$\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$
$\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda)$
$\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp})$
$X^* \xleftarrow{\mathsf{R}} \mathbb{H}$
$(f, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}_1(\mathsf{pp})$
$\mathsf{sk}_f \xleftarrow{\mathsf{R}} \mathsf{Constrain}(\mathsf{msk}, f)$
$\widehat{\mathsf{coin}} \xleftarrow{\mathsf{R}} \mathcal{A}_2^{\mathcal{O}_{\mathsf{Chal}}(\cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$
Return $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$
```

where the challenge oracle $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ is described below.

$\mathcal{O}_{\mathsf{Chal}}(x^*)$**:** Given $x^* \in \{0,1\}^n$ as input, it returns $\mathsf{Eval}(\mathsf{msk}, x^*)$ if $\mathsf{coin} = 1$ and X^* if $\mathsf{coin} = 0$.

We recall that $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ is queried at most once during the game.

Game 1**:** In this game, we change the way sk_f is sampled. In particular, we change the way of choosing $\{b_i\}_{i \in [z]}$ and $\{b'_i\}_{i \in [z]}$. Namely, given the constraining query f from $\mathcal{A}_1$, the game picks $(b'_1, \dots, b'_z) \xleftarrow{\mathsf{R}} \mathbb{Z}_p^z$, $\alpha \xleftarrow{\mathsf{R}} \mathbb{Z}_p^*$, and sets $b_i := b'_i \alpha + f_i \mod p$ for $i \in [z]$.

Game 2 In this game, we change the challenge oracle $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ as follows:

$\mathcal{O}_{\mathsf{Chal}}(x^*)$**:** Given $x^* \in \{0,1\}^n$ as input, it returns $\mathsf{SEval}(\mathsf{sk}_f, x^*) \cdot \mathsf{Aux}(\mathsf{msk})$ if $\mathsf{coin} = 1$ and X^* if $\mathsf{coin} = 0$.

Game 3**:** In this game, we further change the challenge oracle as follows:

$\mathcal{O}_{\mathsf{Chal}}(x^*)$**:** Given $x^* \in \{0,1\}^n$ as input, it first picks $\psi \xleftarrow{\mathsf{R}} \mathbb{H}$ and returns $\mathsf{SEval}(\mathsf{sk}_f, x) \cdot \psi$ if $\mathsf{coin} = 1$ and X^* if $\mathsf{coin} = 0$.

Game 4 In this game, the oracle is changed as follows.

$\mathcal{O}_{\mathsf{Chal}}(x^*)$**:** Given $x^* \in \{0,1\}^n$ as input, it returns X^* regardless of the value of coin.

Let T_i be the event that Game i returns 1.

Lemma 4.6. $\Pr[\mathsf{T}_1] = \Pr[\mathsf{T}_0]$

Proof. It can be seen that the distributions of sk_f are exactly the same in these games. Since the change is only conceptual, the lemma follows. ■

Lemma 4.7. $\Pr[\mathsf{T}_2] = \Pr[\mathsf{T}_1]$

Proof. The change is only conceptual due to the semi-evaluability and thus the lemma follows. ■

Lemma 4.8. *If the* $(D-1)$*-DDH assumption holds, then* $|\Pr[\mathsf{T}_3] - \Pr[\mathsf{T}_2]| = \mathrm{negl}(\lambda)$.

Proof. For the sake of the contradiction, let us assume that $|\Pr[\mathsf{T}_3] - \Pr[\mathsf{T}_2]|$ is non-negligible. We then construct an adversary $\mathcal{B}$ that breaks the L-DDHI assumption using $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

$\mathcal{B}(\mathcal{H}, g, g^{\alpha}, g^{\alpha^2}, ..., g^{\alpha^{D-1}}, \psi)$**:** Given the problem instance, $\mathcal{B}$ first gives the group description $\mathsf{pp} := \mathcal{H}$ to $\mathcal{A}_1$. Then, $\mathcal{A}_1$ outputs a constraining query f along with its state $\mathsf{st}_{\mathcal{A}}$. Then, $\mathcal{B}$ picks $\mathsf{coin} \overset{\mathsf{R}}{\leftarrow} \{0,1\}$, $(b'_1, ..., b'_z) \leftarrow \mathbb{Z}_p^z$, $h_1, \ldots, h_n, X^* \overset{\mathsf{R}}{\leftarrow} \mathbb{H}$ and gives $\mathsf{sk}_f = (f, b'_1, \ldots, b'_z, g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{D-1}}, h_1, \ldots, h_n)$ and the state $\mathsf{st}_{\mathcal{A}}$ to $\mathcal{A}_2$. When $\mathcal{A}_2$ makes a challenge query x^* for $\mathcal{O}_{\mathsf{Chal}}(\cdot)$, $\mathcal{B}$ returns $\psi \cdot \mathsf{SEval}(\mathsf{sk}_f, x^*)$ if $\mathsf{coin} = 1$ and X^* if $\mathsf{coin} = 0$ to $\mathcal{A}_2$. Finally, $\mathcal{A}_2$ outputs its guess $\widehat{\mathsf{coin}}$. $\mathcal{B}$ then outputs $(\mathsf{coin} \overset{?}{=} \widehat{\mathsf{coin}})$ as its guess.

It can easily be seen that $\mathcal{B}$ simulates Game_2 if $\psi = g^{1/\alpha} = \mathsf{Aux}(\mathsf{msk})$ and Game_3 if $\psi \overset{\mathsf{R}}{\leftarrow} \mathbb{H}$. The lemma readily follows. ■

Lemma 4.9. $\Pr[\mathsf{T}_3] = \Pr[\mathsf{T}_4]$

Proof. In Game 3, the response to the challenge query is a random group element of $\mathbb{H}$ regardless of the value of coin. Therefore, the change is only conceptual. ■

Lemma 4.10. *We have* $|\Pr[\mathsf{T}_4] - 1/2| = 0$.

Proof. In Game 4 everything $\mathcal{A}$ sees is independent from coin, and thus there is no way to guess it with non-zero advantage. ■

Therefore, the advantage of $\mathcal{A}$ is $\mathsf{Adv}^{\mathsf{cprf}}_{\mathsf{CPRF}_{\mathsf{NE}}, \mathcal{F}^{\mathbf{NC}^1}, \mathcal{A}}(\lambda) = 2 \cdot |\Pr[\mathsf{T}_0] - 1/2| = \mathrm{negl}(\lambda)$. This completes the proof of the theorem. ■

4.2 Selectively-secure CPRF in the Standard Model

Here, we give our CPRF for $\mathbf{NC}^1$ with selectively single-key security in the standard model. The scheme is obtained by combining our CPRF $\mathsf{CPRF}_{\mathsf{NE}} = (\mathsf{Setup}_{\mathsf{NE}}, \mathsf{KeyGen}_{\mathsf{NE}}, \mathsf{Eval}_{\mathsf{NE}}, \mathsf{Constrain}_{\mathsf{NE}}, \mathsf{CEval}_{\mathsf{NE}})$ for the function class $\mathcal{F}^{\mathbf{NC}^1}$ in Section 4.1 with our CIH $\mathsf{CIH}_{\widetilde{\mathsf{BC}}} = (\mathsf{PrmGen}_{\widetilde{\mathsf{BC}}}, \mathsf{Eval}_{\widetilde{\mathsf{BC}}})$ constructed in Section 3. For the simplicity of the notation, we will denote $\mathsf{Eval}_{\widetilde{\mathsf{BC}}}(\mathsf{pp}_{\mathsf{CIH}}, \cdot)$ by $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}(\cdot)$ when $\mathsf{pp}_{\mathsf{CIH}}$ is clear. Let SPGGen denote the group generator defined in Section 3. The construction of our scheme $\mathsf{CPRF}_{\mathbf{NC}^1\text{-}\mathsf{Sel}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ is as follows:

$\mathsf{Setup}(1^{\lambda})$**:** It first runs $\mathcal{G}_0 \overset{\mathsf{R}}{\leftarrow} \mathsf{SPGGen}(1^{\lambda})$ to obtain the group description $\mathcal{G}_0 := (\mathbb{G}, q)$. Recall that $\mathcal{G}_0$ also defines the description of the group $\mathbb{QR}_q \subset \mathbb{Z}_q^*$ of prime order $p = (q-1)/2$. We denote the description of the group by $\mathcal{G}_1 := (\mathbb{QR}_q, p)$. It then samples $\mathsf{pp}_{\mathsf{CIH}} \overset{\mathsf{R}}{\leftarrow} \mathsf{PrmGen}_{\widetilde{\mathsf{BC}}}(\mathcal{G}_0)$. Let $\mathsf{pp}_{\mathsf{NE}} := \mathcal{G}_1$. It outputs $\mathsf{pp} := (\mathsf{pp}_{\mathsf{CIH}}, \mathsf{pp}_{\mathsf{NE}})$.

$\mathsf{KeyGen}(\mathsf{pp})$**:** It first parses $(\mathsf{pp}_{\mathsf{CIH}}, \mathsf{pp}_{\mathsf{NE}}) \leftarrow \mathsf{pp}$ and runs $\mathsf{msk}_i \overset{\mathsf{R}}{\leftarrow} \mathsf{KeyGen}_{\mathsf{NE}}(\mathsf{pp}_{\mathsf{NE}})$ for $i \in [m]$. It then outputs $\mathsf{msk} := (\mathsf{msk}_1, ..., \mathsf{msk}_m)$.

$\mathsf{Eval}(\mathsf{msk}, x)$**:** It first parses $(\mathsf{msk}_1, ..., \mathsf{msk}_m) \leftarrow \mathsf{msk}$ and outputs

$$y := \mathsf{CIH}_{\widetilde{\mathsf{BC}}}\Big(\mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_1, x), ..., \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_m, x)\Big).$$

where we recall that we have $\mathsf{CIH}_{\widetilde{\mathsf{BC}}} : (\mathbb{QR}_q)^m \to \mathbb{G}$ and $\mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, \cdot) : \{0,1\}^n \to \mathbb{QR}_q$ for $i \in [m]$ (for simplicity, we omit writing $\mathsf{pp}_{\mathsf{CIH}}$ and $\mathsf{pp}_{\mathsf{NE}}$ here).

$\mathsf{Constrain}(\mathsf{msk}, f)$**:** It first parses $(\mathsf{msk}_1, ..., \mathsf{msk}_m) \leftarrow \mathsf{msk}$. It then computes $\mathsf{sk}_{f,i} \xleftarrow{\mathsf{R}} \mathsf{Constrain}_{\mathsf{NE}}(\mathsf{msk}_i, f)$ for $i \in [m]$ and outputs $\mathsf{sk}_f := (\mathsf{sk}_{f,1}, ..., \mathsf{sk}_{f,m})$.

$\mathsf{CEval}(\mathsf{sk}_f, x)$**:** It first parses $(\mathsf{sk}_{f,1}, ..., \mathsf{sk}_{f,m}) \leftarrow \mathsf{sk}_f$. It then computes $X_i := \mathsf{Eval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x)$ for $i \in [m]$ and outputs $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}(X_1, ..., X_m)$.

Remark 4.11. In the above, we need m instances of $\mathsf{CPRF}_{\mathsf{NE}}$, which may seem redundant. This is necessary because the domain of the CIH constructed in Section 3 is $\mathbb{QR}^m$ for $m = poly(\lambda)$, and thus input of the CIH must be an m-dimensional vector. If we had a CIH for group-induced function on $\mathbb{QR}$, then the m times blowup could be avoided.

Remark 4.12. The algorithm Setup implicitly uses the group generator SPGGen' that first runs SPGGen to obtain $\mathcal{G} = (\mathbb{G}, q)$ and then outputs the group description $(\mathbb{QR}_q, p)$. Here, from the technical reason, we assume that the description of $\mathbb{QR}_q$ implicitly contains that of $\mathbb{G}$ as well. While our construction in Section 4.1 can be instantiated with any prime-order group generator HGen, our scheme above requires to instantiate the scheme with the specific group generator SPGGen'.

It is easy to observe that the correctness of the above scheme follows from that of the underlying schemes. The following theorem addresses the security of the scheme.

Theorem 4.13. *The above construction* $\mathsf{CPRF}_{\mathbf{NC}^1\text{-}\mathsf{Sel}}$ *is a selective single-key secure CPRF for the function class* $\mathcal{F}^{\mathbf{NC}^1}$ *if the DDHI assumption holds with respect to* SPGGen' *(see Remark 4.12) and the DDH assumption holds with respect to* SPGGen.

Proof. The security of the scheme will be proven by the no-evaluation security, semi-evaluability, and universality of $\mathsf{CPRF}_{\mathsf{NE}}$ as well as correlated-input security of $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$ for (component-wise) group-induced functions. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any selectively admissible adversary that attacks the selective single-key security of CPRF. For simplicity, we assume that $\mathcal{A}_2$ never makes the same query twice, makes a challenge query only once (see Remark 2.6), and all evaluation queries x made by $\mathcal{A}_2$ satisfy $f(x) = 1$. In the following, Q denotes the upper bound on the number of the access to the Evaluation oracle $\mathsf{Eval}(\mathsf{msk}, \cdot)$ made by $\mathcal{A}_2$. We prove the theorem by considering the following sequence of games.

Game 0**:** This is the actual single-key security experiment $\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF}_{\mathbf{NC}^1\text{-}\mathsf{Sel}}, \mathcal{F}^{\mathbf{NC}^1}, \mathcal{A}}(\lambda)$ against the selective adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where the coin of the game is fixed to $\mathsf{coin} = 1$. Namely,

```
$\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda)$
$\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp})$
$(f, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}_1(\mathsf{pp})$
$\mathsf{sk}_f \xleftarrow{\mathsf{R}} \mathsf{Constrain}(\mathsf{msk}, f)$
$\widehat{\mathsf{coin}} \xleftarrow{\mathsf{R}} \mathcal{A}_2^{\mathcal{O}_{\mathsf{Chal}}(\cdot), \mathsf{Eval}(\mathsf{msk}, \cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$
Return $(\widehat{\mathsf{coin}} \overset{?}{=} 1)$
```

where we describe $\mathsf{Eval}(\mathsf{msk}, \cdot)$ and $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ below.

$\mathsf{Eval}(\mathsf{msk}, \cdot)$**:** Given $x \in \{0,1\}^n$ as input, it returns $\mathsf{Eval}(\mathsf{msk}, x)$.

$\mathcal{O}_{\mathsf{Chal}}(\cdot)$**:** Given $x^* \in \{0,1\}^n$ as input, it returns $y^* = \mathsf{Eval}(\mathsf{msk}, x^*)$. (Recall that we set $\mathsf{coin} = 1$ in this game.)

Game 1**:** In this game, we do not differentiate the challenge oracle $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ from $\mathsf{Eval}(\mathsf{msk}, \cdot)$ and identify them. Namely, $\mathcal{A}_2$ is equipped with the following oracle $\mathcal{O}_{\mathsf{Merge}}(\cdot)$ instead of $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ and $\mathsf{Eval}(\mathsf{msk}, \cdot)$.

$\mathcal{O}_{\mathsf{Merge}}(\cdot)$**:** Given the j-th query $x^{(j)} \in \{0,1\}^n$ from the adversary, it first computes $X_i^{(j)} := \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, x^{(j)})$ for $i \in [m]$ and then returns $y^{(j)} := \mathsf{CIH}_{\widetilde{\mathsf{BC}}}(X_1^{(j)}, \ldots, X_m^{(j)})$.

We note that $\mathcal{O}_{\mathsf{Merge}}(\cdot)$ simply returns $\mathsf{Eval}(\mathsf{msk}, x)$ given x. Here, we elaborate on the description of the oracle in order to fix notations that will be used in the following. We also note that we do not differentiate the challenge query from the evaluation query in this game, and we have $x^* = x^{(j^*)}$ for some $j^* \in [Q+1]$.

$\mathsf{Game}\ 2$**:** Let Col be the event that there exist $j_1 \neq j_2 \in [Q+1]$ such that $(X_1^{(j_1)}, \ldots, X_m^{(j_1)}) = (X_1^{(j_2)}, \ldots, X_m^{(j_2)})$. If Col occurs, the game immediately aborts and outputs a uniformly random bit. The rest is the same as the previous game.

$\mathsf{Game}\ 3$ In this game, we change the way $\{X_i^{(j)}\}_{i\in[m], j\in[Q+1]}$ is created. In particular, $\mathcal{O}_{\mathsf{Merge}}(\cdot)$ works as follows:

$\mathcal{O}_{\mathsf{Merge}}(\cdot)$**:** Given the j-th query $x^{(j)} \in \{0,1\}^n$ from the adversary, it proceeds as follows. There are two cases to consider:

1. For the first query $x^{(1)}$, the oracle computes
$$X_i^{(1)} := \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, x^{(1)}) \quad \text{for } i \in [m], \quad y^{(1)} := \mathsf{CIH}_{\widetilde{\mathsf{BC}}}(X_1^{(1)}, \ldots, X_m^{(1)}),$$
and returns $y^{(1)}$.
2. To answer evaluation queries $x^{(j)}$ with $j > 1$, it first computes
$$X_i^{(j)} := X_i^{(1)} \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(1)})^{-1} \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j)}) \tag{6}$$
for $i \in [m]$. Then it computes and returns $y^{(j)} = \mathsf{CIH}_{\widetilde{\mathsf{BC}}}(X_1^{(j)}, \ldots, X_m^{(j)})$.

Note that during the above phase, as soon as the game finds $j_1 \neq j_2 \in [Q+1]$ such that $(X_1^{(j_1)}, \ldots, X_m^{(j_1)}) = (X_1^{(j_2)}, \ldots, X_m^{(j_2)})$, the game aborts and outputs a random bit (as specified in $\mathsf{Game}\ 2$).

$\mathsf{Game}\ 4$ We define Col' as the event that there exist $j_1 \neq j_2 \in [Q+1]$ such that
$$\mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j_1)}) = \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j_2)}) \quad \forall i \in [m].$$
In this game, the game aborts when Col' occurs instead of Col.

$\mathsf{Game}\ 5$**:** In this game, we change the way $X_i^{(1)}$ is chosen. In particular, the first item of the description of the oracle $\mathcal{O}_{\mathsf{Merge}}(\cdot)$ in $\mathsf{Game}\ 3$ is changed as follows:

1. For the first query $x^{(1)}$, the oracle sets
$$X_i^{(1)} \overset{\mathsf{R}}{\leftarrow} \mathbb{QR}_q \quad \text{for } i \in [m], \quad y^{(1)} := \mathsf{CIH}_{\widetilde{\mathsf{BC}}}(X_1^{(1)}, \ldots, X_m^{(1)}),$$
and returns $y^{(1)}$.

$\mathsf{Game}\ 6$ In this game, we further change the oracle $\mathcal{O}_{\mathsf{Merge}}(\cdot)$ as follows:

$\mathcal{O}_{\mathsf{Merge}}(\cdot)$**:** Given the j-th query from the adversary $x^{(j)} \in \{0,1\}^n$, it picks $y^{(j)} \overset{\mathsf{R}}{\leftarrow} \mathbb{G}$ and returns it.

Game 7 This is the real game with the coin being fixed to coin $= 0$. Namely, $\mathcal{A}_2$ is equipped with the oracles $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ and $\mathsf{Eval}(\mathsf{msk},\cdot)$ that works as follows. (We do not consider $\mathcal{O}_{\mathsf{Merge}}(\cdot)$ any more.)

$\mathsf{Eval}(\mathsf{msk},\cdot)$: Given $x \in \{0,1\}^n$ as input, it returns $\mathsf{Eval}(\mathsf{msk},x)$.

$\mathcal{O}_{\mathsf{Chal}}(\cdot)$**:** Given $x^* \in \{0,1\}^n$ as input, it picks $y^* \xleftarrow{\mathsf{R}} \mathbb{G}$ and returns it. (Recall that we set coin $= 0$ in this game.)

Let T_i be the event that Game i returns 1.

Lemma 4.14. $\Pr[\mathsf{T}_1] = \Pr[\mathsf{T}_0]$.

Proof. Since coin $= 1$ in Game 0, we have $\mathcal{O}_{\mathsf{Chal}}(\cdot) = \mathsf{Eval}(\mathsf{msk},\cdot)$. Therefore, this is only the conceptual change. ■

Lemma 4.15. *If* $m \geq n$, $|\Pr[\mathsf{T}_2] - \Pr[\mathsf{T}_1]| = \mathrm{negl}(\lambda)$.

Proof. It is easy to see that we have $|\Pr[\mathsf{T}_2] - \Pr[\mathsf{T}_1]| \leq \Pr[\mathsf{Col}]$. We will show that Col occurs only with negligible probability. We observe that

$$\Pr[\mathsf{Col}] \quad \leq \quad \Pr\left[\begin{array}{c} \mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda), \quad \mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp}) : \\ \exists x, x' \in \{0,1\}^n \quad \text{s.t.} \quad x \neq x' \wedge (X_1,\ldots,X_m) = (X'_1,\ldots,X'_m) \end{array}\right]$$

where $X_i = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, x)$ and $X'_i = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, x')$ in the above. Therefore, it suffices to show that for any pp output by $\mathsf{Setup}(1^\lambda)$,

$$\Pr\left[\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp}) \ : \ \exists x, x' \in \{0,1\}^n \quad \text{s.t.} \quad x \neq x' \wedge (X_1,\ldots,X_m) = (X'_1,\ldots,X'_m)\right]$$

is negligible, We can bound the term by

$$\begin{aligned} &\leq \sum_{x,x' \in \{0,1\}^n, x \neq x'} \Pr\left[\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp}) \ : \ (X_1,\ldots,X_m) = (X'_1,\ldots,X'_m)\right] \\ &= \sum_{x,x' \in \{0,1\}^n, x \neq x'} \left(\prod_{i \in [m]} \Pr\left[\mathsf{msk}_i \xleftarrow{\mathsf{R}} \mathsf{KeyGen}_{\mathsf{NE}}(\mathsf{pp}_{\mathsf{NE}}) \ : \ \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, x) = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, x')\right]\right) \\ &\leq \frac{2^{2n}}{p^m} = \frac{4^n}{p^m}, \end{aligned}$$

where we used the union bound in the first inequality and the universality of $\mathsf{CPRF}_{\mathsf{NE}}$ (Lemma 4.4) in the last inequality. The quantity is negligible when $m \geq n$ as desired. ■

Lemma 4.16. $\Pr[\mathsf{T}_3] = \Pr[\mathsf{T}_2]$.

Proof. We prove that the change is only conceptual. The difference between the games is that $X_i^{(j)}$ is computed as $\mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, x^{(j)})$ in Game 2, whereas it is computed as the right-hand side of Equation (6) in Game 3. We show here that they are actually equivalent. The right-hand side of Equation (6) equals to

$$\begin{aligned} & X_i^{(1)} \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(1)})^{-1} \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j)}) \\ = \ & \mathsf{Aux}_{\mathsf{NE}}(\mathsf{msk}_i) \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(1)}) \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(1)})^{-1} \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j)}) \\ = \ & \mathsf{Aux}_{\mathsf{NE}}(\mathsf{msk}_i) \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j)}) \\ = \ & \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, x^{(j)}) \end{aligned}$$

where we used our simplification assumption that $f(x^{(1)}) = f(x^{(j)}) = 1$ and semi-evaluability (Lemma 4.3) in the first and the last equations above. ■

Lemma 4.17. $\Pr[\mathsf{T}_4] = \Pr[\mathsf{T}_3]$.

Proof. It suffices to show that the abort conditions Col and Col' are equivalent. We have

$$\begin{aligned}
&\mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j_1)}) = \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j_2)}) \quad \forall i \in [m] \\
&\Leftrightarrow \mathsf{Aux}_{\mathsf{NE}}(\mathsf{msk}_i) \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j_1)}) = \mathsf{Aux}_{\mathsf{NE}}(\mathsf{msk}_i) \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j_2)}) \quad \forall i \in [m] \\
&\Leftrightarrow X_i^{(j_1)} = X_i^{(j_2)} \quad \forall i \in [m].
\end{aligned}$$

Hence, the change is only conceptual. The lemma readily follows. ■

Lemma 4.18. *If* $\mathsf{CPRF}_{\mathsf{NE}}$ *satisfies the no-evaluation security when instantiated by the group generator* $\mathsf{HGen} := \mathsf{SPGGen}'$, *we have* $|\Pr[\mathsf{T}_5] - \Pr[\mathsf{T}_4]| = \mathrm{negl}(\lambda)$.

Proof. For the sake of the contradiction, let us assume $|\Pr[\mathsf{T}_5] - \Pr[\mathsf{T}_4]|$ is non-negligible for the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. We consider the following hybrid games for $k \in \{0, 1, \dots, m\}$:

Game $4.k$: This is the same as Game 4 with the following difference. In this game, $X_i^{(1)}$ is set as $X_i^{(1)} = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}_i, x^{(1)})$ when $i > k$ and $\tilde{X}_i \xleftarrow{\mathsf{R}} \mathbb{QR}_q$ when $i \le k$.

By the definition, we have Game 4.0 (resp. Game $4.m$) is equivalent to Game 4 (resp. Game 5). Therefore, we have

$$|\Pr[\mathsf{T}_5] - \Pr[\mathsf{T}_4]| = \Pr[\mathsf{T}_{4.m}] - \Pr[\mathsf{T}_{4.0}]| \ge \sum_{k\in[m]} |\Pr[\mathsf{T}_{4.k}] - \Pr[\mathsf{T}_{4.k-1}]|$$

where $\Pr[\mathsf{T}_i]$ denotes the probability that Game $4.k$ outputs 1. By the above inequality, we have that there exists an index k^* such that $|\Pr[\mathsf{T}_{4.k^*}] - \Pr[\mathsf{T}_{4.k^*-1}]|$ is non-negligible. We then construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the no-evaluation security of the underlying scheme $\mathsf{CPRF}_{\mathsf{NE}}$. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}_1(\mathsf{pp}_{\mathsf{NE}})$**:** Given the group description $\mathsf{pp}_{\mathsf{NE}} = (\mathbb{QR}_q, p)$, $\mathcal{B}_1$ first recovers the group description $\mathcal{G}_0 = (\mathbb{G}, q)$ from $(\mathbb{QR}_q, p)$ (See remark Remark 4.12). $\mathcal{B}_1$ then samples $\mathsf{pp}_{\mathsf{CIH}} \xleftarrow{\mathsf{R}} \mathsf{PrmGen}_{\widetilde{\mathsf{BC}}}(\mathcal{G}_0)$ and sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{CIH}}, \mathsf{pp}_{\mathsf{NE}})$. It then runs $(f, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}_1(\mathsf{pp})$ and outputs $(f, \mathsf{st}_{\mathcal{B}} := \mathsf{st}_{\mathcal{A}})$.

$\mathcal{B}_2^{\mathcal{O}_{\mathsf{Chal}}(\cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{B}})$**:** Here, we denote the master secret key of the no-evaluation security game (played for $\mathcal{B}$) by msk'. The task of $\mathcal{B}_2$ is to distinguish whether $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ corresponds to $\mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}', \cdot)$ or $\mathsf{RF}(\cdot)$. First, $\mathcal{B}_2$ picks $\mathsf{msk}_i \xleftarrow{\mathsf{R}} \mathsf{KeyGen}_{\mathsf{NE}}(\mathsf{pp}_{\mathsf{NE}})$ for $i \in \{k^* + 1, \dots, m\}$. $\mathcal{B}_2$ then runs $\mathcal{A}_2(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$ and simulates $\mathcal{O}_{\mathsf{Merge}}(\cdot)$ for $\mathcal{A}_2$ as follows:

- To answer the first query $x^{(1)}$ from $\mathcal{A}_2$, $\mathcal{B}_2$ submits the same $x^{(1)}$ to its challenge oracle $\mathcal{O}_{\mathsf{Chal}}(\cdot)$. Then, $\mathcal{B}_2$ is given R. Then, $\mathcal{B}_2$ sets $X_i^{(1)} = \mathsf{SEval}_{\mathsf{NE}}(\mathsf{msk}_i, x^{(1)})$ for $i \ge k^* + 1$, $X_{k^*}^{(1)} = R$, and samples $X_i^{(1)} \xleftarrow{\mathsf{R}} \mathbb{QR}_q$ for $i \le k^* - 1$. Finally, $\mathcal{B}_2$ returns $y^{(1)} = \mathsf{CIH}_{\widetilde{\mathsf{BC}}}(X_1^{(1)}, \dots, X_m^{(1)})$ to $\mathcal{A}_2$.
- To answer the query $x^{(j)}$ with $j > 1$ from $\mathcal{A}_2$, $\mathcal{B}_2$ first parses $\mathsf{sk}_f \to (\mathsf{sk}_{f,1}, \dots, \mathsf{sk}_{f,m})$ and computes $X_i^{(j)} := X_i^{(1)} \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(1)})^{-1} \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j)})$ for $i \in [m]$. It then returns $y^{(j)} = \mathsf{CIH}_{\widetilde{\mathsf{BC}}}(X_1^{(j)}, \dots, X_m^{(j)})$ to $\mathcal{A}_2$.

Note that during the above phase, as soon as $\mathcal{B}_2$ finds $j_1 \ne j_2 \in [Q]$ such that $(X_1^{(j_1)}, \dots, X_m^{(j_1)}) = (X_1^{(j_2)}, \dots, X_m^{(j_2)})$, $\mathcal{B}_2$ aborts and outputs a random bit. When $\mathcal{A}_2$ terminates with output $\widehat{\mathsf{coin}}$, $\mathcal{B}_2$ outputs $\widehat{\mathsf{coin}}$ as its guess and terminates.

The above completes the description of $\mathcal{B}$. It is straightforward to see that $\mathcal{B}$ makes only single challenge query. It is also easy to see that $\mathcal{B}$ simulates Game $4.(k^* - 1)$ for $\mathcal{A}$ when $\mathcal{B}$'s challenge oracle is $\mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}', \cdot)$ and Game $4.k^*$ when $\mathcal{B}$'s challenge oracle is $\mathsf{RF}(\cdot)$. Note that in the former case, $\mathcal{B}$ implicitly sets $\mathsf{msk}_{k^*} := \mathsf{msk}'$. Since $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ outputs 1, we have that $\mathcal{B}$'s advantage is $|\Pr[\mathsf{T}_{4.k^*-1}] - \Pr[\mathsf{T}_{4.k^*}]|$, which is non-negligible. This completes the proof of the lemma. ■

Lemma 4.19. *If* $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$ *is* $\Psi^{\text{g-indc}}$*-CIH with respect to* SPGGen*, then we have* $|\Pr[\mathsf{T}_6]-\Pr[\mathsf{T}_5]| = \mathrm{negl}(\lambda)$.

Proof. For the sake of the contradiction, let us assume that $|\Pr[\mathsf{T}_6] - \Pr[\mathsf{T}_5]|$ is non-negligible for the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. We then construct an adversary $\mathcal{B}$ that breaks the security of $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}$ as follows.

$\mathcal{B}^{\mathcal{O}(\cdot)}(\mathsf{pp}_{\mathsf{CIH}})$**:** At the beginning of the game, $\mathcal{B}$ is given the public parameter $\mathsf{pp}_{\mathsf{CIH}}$ of the CIH. Then it parses the group description $(\mathbb{G}, q)$ from $\mathsf{pp}_{\mathsf{CIH}}$ and obtains the description of another group $\mathsf{pp}_{\mathsf{NE}} := (\mathbb{QR}_q, p)$. It then sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{CIH}}, \mathsf{pp}_{\mathsf{NE}})$ and runs $(f, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}_1(\mathsf{pp})$. It further samples $\mathsf{msk}_i \xleftarrow{\mathsf{R}} \mathsf{KeyGen}_{\mathsf{NE}}(\mathsf{pp}_{\mathsf{NE}})$ and $\mathsf{sk}_{f,i} \xleftarrow{\mathsf{R}} \mathsf{Constrain}_{\mathsf{NE}}(\mathsf{msk}_i, f)$ for $i \in [m]$. It then gives the input $\mathsf{sk}_f := (\mathsf{sk}_{f,1}, \ldots, \mathsf{sk}_{f,m})$ and $\mathsf{st}_{\mathcal{A}}$ to $\mathcal{A}_2$ and simulates $\mathcal{O}_{\mathsf{Merge}}(\cdot)$ for $\mathcal{A}_2$ as follows:

- To answer the first query $x^{(1)}$ from $\mathcal{A}_2$, $\mathcal{B}$ queries its oracle on input $\vec{\phi}^{(1)} := (1, \ldots, 1) \in \mathbb{QR}_q^m$ to obtain $y^{(1)}$. It then passes $y^{(1)}$ to $\mathcal{A}_2$.
- To answer the query $x^{(j)}$ with $j > 1$ from $\mathcal{A}_2$, $\mathcal{B}$ first parses $\mathsf{sk}_f \to (\mathsf{sk}_{f,1}, \ldots, \mathsf{sk}_{f,m})$ and computes $\phi_i^{(j)} := \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(1)})^{-1} \cdot \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j)})$ for $i \in [m]$. $\mathcal{B}$ then sets $\vec{\phi}^{(j)} = (\phi_1^{(j)}, \ldots, \phi_m^{(j)})$ and queries $\vec{\phi}^{(j)}$ to its oracle. Given the response $y^{(j)}$ from the oracle, $\mathcal{B}_2$ relays the same value to $\mathcal{A}_2$.

Note that during the above phase, as soon as $\mathcal{B}$ finds $j_1 \neq j_2 \in [Q]$ such that $\mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j_1)}) = \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j_2)})$ for all $i \in [m]$, it aborts and outputs a random bit. When $\mathcal{A}_2$ terminates with output $\widehat{\mathsf{coin}}$, $\mathcal{B}$ outputs the same $\widehat{\mathsf{coin}}$ and terminates.

The above completes the description of $\mathcal{B}$. Here, we prove that $\mathcal{B}$ simulates Game 5 when $\mathcal{B}$'s challenge coin coin' is 1 and Game 6 when $\mathsf{coin}' = 0$.

We start by proving the former statement. When $\mathsf{coin}' = 1$, the CIH security experiment chooses randomness $\vec{R} := (R_1, \ldots, R_m) \xleftarrow{\mathsf{R}} \mathbb{QR}_q^m$ during the game and the oracle $\mathcal{O}(\cdot)$ returns $\mathsf{CIH}_{\widetilde{\mathsf{BC}}}(\vec{R} \star \vec{\phi})$ on input $\mathcal{B}$'s query $\vec{\phi} = (\phi_1, \ldots, \phi_m) \in \mathbb{QR}_q^m$. The view of $\mathcal{A}_2$ corresponds to Game 5, with $X_i^{(1)}$ being implicitly set as $X_i^{(1)} := R_i$ for $i \in [m]$.

We next show the latter statement. When $\mathsf{coin}' = 0$, the CIH security experiment chooses randomness $\vec{R} := (R_1, \ldots, R_m) \xleftarrow{\mathsf{R}} \mathbb{QR}_q^m$ during the game and the oracle $\mathcal{O}(\cdot)$ returns $\mathsf{RF}(\vec{R} \star \vec{\phi})$ on input $\mathcal{B}$'s query $\vec{\phi} = (\phi_1, \ldots, \phi_m)$ where $\mathsf{RF}(\cdot)$ is a random function. In order to prove that $\mathcal{B}$ simulates Game 6, it suffices to show that all the queries made by $\mathcal{B}$ are distinct. We have

$$\phi_i^{(j_1)} = \phi_i^{(j_2)} \quad \iff \quad \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f_i}, x^{(j_1)}) = \mathsf{SEval}_{\mathsf{NE}}(\mathsf{sk}_{f,i}, x^{(j_2)})$$

by the definition. Since $\mathcal{B}$ aborts whenever Col' occurs, this implies that $\mathcal{B}$ does not make the same oracle query twice. ■

Lemma 4.20. *We have* $|\Pr[T_7] - \Pr[T_6]| = \mathrm{negl}(\lambda)$.

Proof. This can be proven by applying the same game changes as that from Game 0 to Game 6 in a reverse order, with the only difference that the challenge query x^* is always returned by a uniformly random group element $y^* \xleftarrow{\mathsf{R}} \mathbb{G}$. ■

We have

$$\begin{aligned}
\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},n,\mathcal{A}}(\lambda) &= 2 \cdot \left|\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - \frac{1}{2}\right| \\
&= |\Pr[\mathcal{A} \text{ outputs } 1 \mid \mathsf{coin} = 1] - \Pr[\mathcal{A} \text{ outputs } 1 \mid \mathsf{coin} = 0]| \\
&= |\Pr[\mathsf{T}_7] - \Pr[\mathsf{T}_0]| \\
&\leq \sum_{i=1}^{7} |\Pr[\mathsf{T}_i] - \Pr[\mathsf{T}_{i-1}]| \\
&= \mathrm{negl}(\lambda).
\end{aligned}$$

This completes the proof of the theorem. ■

4.3 Adaptively-secure CPRF in the Random Oracle Model

Here, we construct an adaptively single-key secure CPRF for $\mathbf{NC}^1$ in the random oracle model. As a building block, we use our no-evaluation secure CPRF $\mathsf{CPRF}_{\mathsf{NE}}$ in Section 4.1. We first show that $\mathsf{CPRF}_{\mathsf{NE}}$ satisfies the property that we call statistical collision resistance, which will be defined below.

Statistical collision resistance. We say that a $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ is statistically collision resistant if

$$\Pr[\exists x \neq x' \text{ s.t. } \mathsf{Eval}(\mathsf{msk}, x) = \mathsf{Eval}(\mathsf{msk}, x')] = \mathrm{negl}(\lambda)$$

where $\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda)$, $\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp})$.

We remark that $\mathsf{CPRF}_{\mathsf{NE}}$ constructed in Section 4.1 satisfies statistical collision resistance if the underlying group $\mathcal{H} = (\mathbb{H}, p)$ is chosen so that $p \geq 2^{2n+\lambda}$. This can be seen by

$$\begin{aligned}
&\Pr[\exists x \neq x' \text{ s.t. } \mathsf{Eval}(\mathsf{msk}, x) = \mathsf{Eval}(\mathsf{msk}, x')] \\
&\leq 2^{2n} \max_{x \neq x'} \Pr[\mathsf{Eval}(\mathsf{msk}, x) = \mathsf{Eval}(\mathsf{msk}, x')] && \text{(from union bound)} \\
&= 2^{2n}/p && \text{(from the universality of } \mathsf{CPRF}_{\mathsf{NE}}) \\
&\leq 2^{-\lambda}.
\end{aligned}$$

Construction. Here, we give a construction of an adaptively secure CPRF $\mathsf{CPRF}_{\mathsf{ro}} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ in the random oracle model. Let $\mathsf{CPRF}_{\mathsf{NE}} = (\mathsf{Setup}_{\mathsf{NE}}, \mathsf{KeyGen}_{\mathsf{NE}}, \mathsf{Eval}_{\mathsf{NE}}, \mathsf{Constrain}_{\mathsf{NE}}, \mathsf{CEval}_{\mathsf{NE}})$ be our no-evaluation secure CPRF for a function class $\mathcal{F}^{\mathbf{NC}^1}$ in Section 4.1. Let H be a function from $\{0,1\}^*$ to $\mathcal{R}$, which will be modeled as a random oracle in the security proof.

$\mathsf{Setup}(1^\lambda)$**:** It runs $\mathsf{pp}_{\mathsf{NE}} \xleftarrow{\mathsf{R}} \mathsf{Setup}_{\mathsf{NE}}(1^\lambda)$, sets $\mathsf{pp} := \mathsf{pp}_{\mathsf{NE}}$, and outputs pp.

$\mathsf{KeyGen}(\mathsf{pp})$**:** It runs $\mathsf{msk}_{\mathsf{NE}} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}_{\mathsf{NE}}(\mathsf{pp})$, sets $\mathsf{msk} := \mathsf{msk}_{\mathsf{NE}}$, and outputs msk.

$\mathsf{Eval}(\mathsf{msk}, x)$**:** It computes $X := \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x)$ and $y := H(X)$ and outputs y.

$\mathsf{Constrain}(\mathsf{msk}, f)$**:** It runs $\mathsf{sk}^{\mathsf{NE}}_f := \mathsf{Constrain}_{\mathsf{NE}}(\mathsf{msk}, f)$, sets $\mathsf{sk}_f := \mathsf{sk}^{\mathsf{NE}}_f$, and outputs sk_f.

$\mathsf{CEval}(\mathsf{sk}_f, x)$**:** It computes $X := \mathsf{Eval}_{\mathsf{NE}}(\mathsf{sk}_f, x)$ and $y := H(X)$, and outputs y.

Theorem 4.21. *Our scheme* $\mathsf{CPRF}_{\mathsf{ro}}$ *defined above is adaptively single-key secure CPRF for the function class* $\mathcal{F}^{\mathbf{NC}^1}$ *in the random oracle model, if the DDHI assumption holds with respect to the group generator* HGen *(which is internally used by* $\mathsf{CPRF}_{\mathsf{NE}}$.*)*

Proof. The security of the scheme will be reduced to that of $\mathsf{CPRF}_{\mathsf{NE}}$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any admissible adversary that attacks the adaptive single-key security of $\mathsf{CPRF}_{\mathsf{ro}}$. For simplicity, we assume that $\mathcal{A}$ makes a challenge query only once (see Remark 2.6). We also assume that $\mathcal{A}$ never makes the same query twice and all evaluation queries x made by $\mathcal{A}_2$ satisfy $f(x) = 1$.

Game 0**:** This game is $\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF}_{\mathsf{ro}},\mathcal{BF},\mathcal{A}}(\lambda)$. Namely,

$\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$
$\mathsf{pp} := \mathsf{pp}_{\mathsf{NE}} \xleftarrow{\mathsf{R}} \mathsf{Setup}_{\mathsf{NE}}(1^\lambda)$
$\mathsf{msk} := \mathsf{msk}_{\mathsf{NE}} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}_{\mathsf{NE}}(\mathsf{pp})$
$y^* \xleftarrow{\mathsf{R}} \mathcal{R}$
$(f, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}_1^{\mathsf{RO}(\cdot),\mathcal{O}_{\mathsf{Eval}}(\cdot),\mathcal{O}_{\mathsf{Chal}}(\cdot)}(\mathsf{pp})$
$\mathsf{sk}_f := \mathsf{sk}_f^{\mathsf{NE}} \xleftarrow{\mathsf{R}} \mathsf{Constrain}_{\mathsf{NE}}(\mathsf{msk}, f)$
$\widehat{\mathsf{coin}} \xleftarrow{\mathsf{R}} \mathcal{A}_2^{\mathsf{RO}(\cdot),\mathcal{O}_{\mathsf{Eval}}(\cdot),\mathcal{O}_{\mathsf{Chal}}(\cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$
Return $(\widehat{\mathsf{coin}} \stackrel{?}{=} \mathsf{coin})$
where $\mathsf{RO}(\cdot)$, $\mathcal{O}_{\mathsf{Eval}}(\cdot)$ and $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ are oracles described below.

$\mathsf{RO}(\cdot)$**:** Given $X \in \{0,1\}^m$ as input, it returns $H(X)$.

$\mathcal{O}_{\mathsf{Eval}}(\cdot)$**:** Given $x \in \{0,1\}^n$ as input, it computes $X := \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x)$ and $y := H(X)$ and returns y.

$\mathcal{O}_{\mathsf{Chal}}(\cdot)$**:** Given $x^* \in \{0,1\}^n$ as input, if $\mathsf{coin} = 1$, then it works similarly to $\mathcal{O}_{\mathsf{Eval}}$. Otherwise it returns y^*.

We remark that we simplify the experiment compared to the definition given in Section 2.3 by using the assumption that $\mathcal{A}$ makes a challenge query at most once.

Game 1**:** In this game, the random oracle is sampled lazily. Namely, oracles RO, $\mathcal{O}_{\mathsf{Eval}}$ and $\mathcal{O}_{\mathsf{Chal}}$ are modified as follows. These oracles share a list HList, which is initialized to be empty at the beginning of the game.

$\mathsf{RO}(\cdot)$**:** Given the input X, it returns y if there exists $y \in \mathcal{R}$ such that $(X, y) \in \mathsf{HList}$. Otherwise it picks $y \xleftarrow{\mathsf{R}} \mathcal{R}$, adds (X, y) to HList, and returns y.

$\mathcal{O}_{\mathsf{Eval}}(\cdot)$**:** Given the input x, it first computes $X := \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x)$. If there exists $y \in \mathcal{R}$ such that $(X, y) \in \mathsf{HList}$, then it returns y. Otherwise it picks $y \xleftarrow{\mathsf{R}} \mathcal{R}$, adds (X, y) to HList, and returns y.

$\mathcal{O}_{\mathsf{Chal}}(\cdot)$**:** If $\mathsf{coin} = 1$, then it works similarly to $\mathcal{O}_{\mathsf{Eval}}$. Otherwise it returns y^* given the input x^*.

Game 2**:** In this game, the evaluation and the challenge oracles do not refer to HList at all, and updates of HList by these oracles are delayed until $\mathcal{A}_1$ declares its constrain query. Namely, $\mathcal{O}_{\mathsf{Eval}}$ and $\mathcal{O}_{\mathsf{Chal}}$ are modified as follows, and the procedure $\mathsf{HashSet}$ defined below runs immediately after $\mathcal{A}_1$ outputs $(f, \mathsf{st}_{\mathcal{A}})$. $\mathcal{O}_{\mathsf{Eval}}$ and $\mathcal{O}_{\mathsf{Chal}}$ maintain a list EList, which is initialized to be empty at the beginning of the game, instead of HList. Note that RO still maintains and refers HList as in the previous game.

$\mathcal{O}_{\mathsf{Eval}}(\cdot)$**:** Given the input x, it returns y if there exists $y \in \mathcal{R}$ such that $(x, y) \in \mathsf{EList}$. Otherwise it picks $y \xleftarrow{\mathsf{R}} \mathcal{R}$, adds (x, y) to EList, and returns y.

$\mathcal{O}_{\mathsf{Chal}}(\cdot)$**:** If $\mathsf{coin} = 1$, then it works similarly to $\mathcal{O}_{\mathsf{Eval}}$. Otherwise it returns y^* given the input x^*.

$\mathsf{HashSet}$**:** For all x such that $f(x) = 0$ and there exists $y \in \mathcal{R}$ such that$(x, y) \in \mathsf{EList}$, it computes $X := \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x)$ and adds (X, y) to HList.

Game 3**:** In this game, the challenge oracle always returns y^* regardless of coin. Namely, $\mathcal{O}_{\mathsf{Chal}}$ is modified as follows.

$\mathcal{O}_{\mathsf{Chal}}(x)$**:** Given the input x^*, it returns y^*.

This completes the description of games. Let T_i be the event that Game i returns 1. Then we have to prove that $|\Pr[\mathsf{T}_0] - 1/2|$ is negligible.

Lemma 4.22. *We have* $\Pr[\mathsf{T}_1] = \Pr[\mathsf{T}_0]$

Proof. The modification from Game 0 to Game 1 is just conceptual. ■

Lemma 4.23. *If* $\mathsf{CPRF}_{\mathsf{NE}}$ *is no-evaluation secure, then we have* $|\Pr[\mathsf{T}_2] - \Pr[\mathsf{T}_1]| = \mathrm{negl}(\lambda)$.

Proof. In the following, HList_1 denotes the set of X such that there exists y such that $(X, y) \in \mathsf{HList}$ and EList_1 denotes the set of x such that there exists y satisfying $(x, y) \in \mathsf{EList}$. Game 2 differs from Game 1 only when either of the following events occurs.

1. $\mathcal{A}_1$ makes a query X to RO such that there exists $x \in \mathsf{EList}_1$ satisfying $X = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x)$.

2. $\mathcal{A}_2$ makes a query X to RO such that there exists $x \in \mathsf{EList}_1$ satisfying $X = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x)$ and $f(x) = 1$.

3. $\mathcal{A}_1$ or $\mathcal{A}_2$ makes a query x to $\mathcal{O}_{\mathsf{Eval}}$ or $\mathcal{O}_{\mathsf{Chal}}$ satisfying $X \in \mathsf{HList}_1$ for $X := \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x)$.

4. $\mathcal{A}_1$ or $\mathcal{A}_2$ makes a distinct queries x and x' to $\mathcal{O}_{\mathsf{Eval}}$ or $\mathcal{O}_{\mathsf{Chal}}$ such that $\mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x) = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x')$.

If one of the above events occurs, then one of the events Bad_1, Bad_2 or Col defined below occurs . (If Event 1 occurs, then Bad_1 occurs, if Event 2 occurs, then Bad_2 occurs, if Event 3 occurs, then Bad_1 or Bad_2 occurs, and if Event 4 occurs, then Col occurs.)

Bad_1**:** At the point just after $\mathcal{A}_1$ halts (before $\mathsf{HashSet}$ runs) in Game 2, there exist $x \in \mathsf{EList}_1$ and $X \in \mathsf{HList}_1$ such that $X = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x)$.

Bad_2**:** At the end of Game 2, there exist $x \in \mathsf{EList}_1$ and $X \in \mathsf{HList}_1$ such that $X = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x)$ and $f(x) = 1$ hold.

Col**:** $\mathcal{A}_1$ or $\mathcal{A}_2$ makes a distinct queries x and x' to $\mathcal{O}_{\mathsf{Eval}}$ or $\mathcal{O}_{\mathsf{Chal}}$ such that $\mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x) = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, x')$.

Therefore we have $|\Pr[\mathsf{T}_2] - \Pr[\mathsf{T}_1]| \leq \Pr[\mathsf{Bad}_1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Col}]$. First, we prove that $\Pr[\mathsf{Bad}_1]$ is negligible if $\mathsf{CPRF}_{\mathsf{NE}}$ is no-evaluation secure. We assume that $\Pr[\mathsf{Bad}_1]$ is non-negligible and we construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the no-evaluation security of $\mathsf{CPRF}_{\mathsf{NE}}$ as follows.

$\mathcal{B}_1(\mathsf{pp})$**:** It sets $\mathsf{st}_{\mathcal{B}} := \mathsf{pp}$ and outputs $(f_{\mathsf{one}}, \mathsf{st}_{\mathcal{B}})$, where f_{one} is a function that outputs 1 for all inputs.

$\mathcal{B}_2^{\mathcal{O}(\cdot)}(\mathsf{sk}_{f_{\mathsf{one}}}, \mathsf{st}_{\mathcal{B}} = \mathsf{pp})$**:** (where $\mathcal{O}(\cdot)$ is either $\mathsf{Eval}(\mathsf{msk}, \cdot)$ or $\mathsf{RF}(\cdot) \xleftarrow{\mathsf{R}} \mathsf{Func}(\{0,1\}^n, \mathbb{H})$) It initializes HList and EList to be empty, picks $\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$ and $y^* \xleftarrow{\mathsf{R}} \mathcal{R}$, and runs $\mathcal{A}_1^{\mathsf{RO}(\cdot), \mathcal{O}_{\mathsf{Eval}}(\cdot), \mathcal{O}_{\mathsf{Chal}}(\cdot)}(\mathsf{pp})$. (We remark that it can simulate oracles since they do not need msk.) It maintains HList and EList as in Game 2. When $\mathcal{A}_1$ halts, $\mathcal{B}_2$ randomly picks $\tilde{x}^* \xleftarrow{\mathsf{R}} \mathsf{EList}_1$ and queries $\tilde{x}^*$ to its own challenge oracle to obtain $\tilde{X}^*$. If $\tilde{X}^* \in \mathsf{HList}_1$ holds, then it outputs 1 and otherwise outputs 0.

This completes the description of $\mathcal{B}$. First, we remark that $\mathcal{B}$ is an admissible adversary since $f_{\mathsf{one}}(x) = 1$ for all $x \in \{0,1\}^n$. We prove that $\mathcal{B}$ distinguishes whether $\mathcal{O}(\cdot) = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, \cdot)$ where $\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}_{\mathsf{NE}}(\mathsf{pp})$ or $\mathcal{O}(\cdot) = \mathsf{RF}(\cdot)$ where $\mathsf{RF}(\cdot) \xleftarrow{\mathsf{R}} \mathsf{Func}(\{0,1\}^n, \mathbb{H})$ with a non-negligible advantage. (Here, we recall that the range of the function $\mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, \cdot)$ is $\mathbb{H}$, which is a prime-order group defined by $\mathsf{pp} = \mathsf{pp}_{\mathsf{NE}}$.) When $\mathcal{O}(\cdot) = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, \cdot)$, if Bad_1 occurs, then $\tilde{X}^* \in \mathsf{HList}_1$ holds with probability at least $1/|\mathsf{EList}|$. Therefore $\mathcal{B}$ outputs 1 with probability at least $\Pr[\mathsf{Bad}_1]/|\mathsf{EList}|$, which is non-negligible. On the other hand, if $\mathcal{O}(\cdot) = \mathsf{RF}(\cdot)$, then $\tilde{X}^*$ is a truly random group element of $\mathbb{H}$, and thus $\tilde{X}^* \in \mathsf{HList}_1$ holds with probability at most $|\mathsf{HList}|/|\mathbb{H}|$, which is negligible. Therefore $\mathcal{B}$ distinguishes these two cases with non-negligible advantage, and this contradicts the security of $\mathsf{CPRF}_{\mathsf{NE}}$. Hence $\Pr[\mathsf{Bad}_1]$ is negligible.

Next, we prove that $\Pr[\mathsf{Bad}_2]$ is negligible if $\mathsf{CPRF}_{\mathsf{NE}}$ is no-evaluation secure. We assume that $\Pr[\mathsf{Bad}_2]$ is non-negligible and we construct an adversary $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ that breaks the no-evaluation security of $\mathsf{CPRF}_{\mathsf{NE}}$ as follows.

$\mathcal{C}_1(\mathsf{pp})$**:** It initializes HList and EList to be empty, picks $\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$ and $y^* \xleftarrow{\mathsf{R}} \mathcal{R}$, and runs $\mathcal{A}_1^{\mathsf{RO}(\cdot), \mathcal{O}_{\mathsf{Eval}}(\cdot), \mathcal{O}_{\mathsf{Chal}}(\cdot)}(\mathsf{pp})$. (We remark that it can simulate oracles since they do not need msk.) It takes over HList and EList from $\mathsf{st}_{\mathcal{C}}$ and maintains them as in Game 2. Let $(f, \mathsf{st}_{\mathcal{A}})$ be an output by $\mathcal{A}_1$. Then it sets $\mathsf{st}_{\mathcal{C}} := (\mathsf{st}_{\mathcal{A}}, \mathsf{HList}, \mathsf{EList})$ and outputs $(f, \mathsf{st}_{\mathcal{C}})$.

$\mathcal{C}_2^{\mathcal{O}(\cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{C}})$**:** (where $\mathcal{O}(\cdot)$ is either $\mathsf{Eval}(\mathsf{msk}, \cdot)$ or $\mathsf{RF}(\cdot) \xleftarrow{\mathsf{R}} \mathsf{Func}(\{0,1\}^n, \mathbb{H})$) It parses $(\mathsf{st}_{\mathcal{A}}, \mathsf{HList}, \mathsf{EList}) \leftarrow \mathsf{st}_{\mathcal{C}}$ and runs $\mathcal{A}_2^{\mathsf{RO}(\cdot), \mathcal{O}_{\mathsf{Eval}}(\cdot), \mathcal{O}_{\mathsf{Chal}}(\cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$. (We remark again that it can simulate oracles since they do not need msk.) It maintains HList and EList as in Game 2. When $\mathcal{A}_2$ halts, $\mathcal{B}_2$ randomly picks $\tilde{x}^*$ such that $f(\tilde{x}^*) = 1$ from EList_1 and queries $\tilde{x}^*$ to its own oracle to obtain $\tilde{X}^*$. If $\tilde{X}^* \in \mathsf{HList}_1$ holds, then it outputs 1 and otherwise outputs 0.

This completes the description of $\mathcal{C}$. First, we remark that $\mathcal{C}$ is an admissible adversary since we have $f(\tilde{x}^*) = 1$. We prove that $\mathcal{C}$ distinguishes whether $\mathcal{O}(\cdot) = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, \cdot)$ where $\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}_{\mathsf{NE}}(\mathsf{pp})$ or $\mathcal{O}(\cdot) = \mathsf{RF}(\cdot)$ where $\mathsf{RF}(\cdot) \xleftarrow{\mathsf{R}} \mathsf{Func}(\{0,1\}^n, \mathbb{H})$ with a non-negligible advantage. When $\mathcal{O}(\cdot) = \mathsf{Eval}_{\mathsf{NE}}(\mathsf{msk}, \cdot)$, if Bad_2 occurs, then $\tilde{X}^* \in \mathsf{HList}_1$ holds with probability at least $1/|\mathsf{EList}|$. Therefore $\mathcal{B}$ outputs 1 with probability at least $\Pr[\mathsf{Bad}_2]/|\mathsf{EList}|$, which is non-negligible. On the other hand, if $\mathcal{O}(\cdot) = \mathsf{RF}(\cdot)$, then $\tilde{X}^*$ is a truly random group element of $\mathbb{H}$, and thus $\tilde{X}^* \in \mathsf{HList}_1$ holds with probability at most $|\mathsf{HList}|/|\mathbb{H}|$, which is negligible. Therefore $\mathcal{C}$ distinguishes these two cases with non-negligible advantage, and this contradicts the security of $\mathsf{CPRF}_{\mathsf{NE}}$. Hence $\Pr[\mathsf{Bad}_2]$ is negligible.

Finally, it is clear that $\Pr[\mathsf{Col}]$ is negligible due to the statistical collision resistance of $\mathsf{CPRF}_{\mathsf{NE}}$. By combining the above, we can conclude that $|\Pr[\mathsf{T}_2] - \Pr[\mathsf{T}_1]|$ is negligible, and the lemma is proven. ■

Lemma 4.24. *We have* $\Pr[\mathsf{T}_3] = \Pr[\mathsf{T}_2]$.

Proof. Let x^* be the $\mathcal{A}$'s challenge query and $\widehat{y}^*$ be the random value that is picked by $\mathcal{O}_{\mathsf{Chal}}$ to answer the query when $\mathsf{coin} = 1$. (We remark that $\mathcal{O}_{\mathsf{Chal}}$ must pick a fresh random value $\widehat{y}^*$ since a challenge query x^* is different from all evaluation queries and thus $x^* \notin \mathsf{EList}_1$.) We claim that no information of $\widehat{y}^*$ is revealed to $\mathcal{A}$ except that from $\mathcal{O}_{\mathsf{Chal}}$. First, since we have $x \neq x^*$ for all evaluation queries x, $\widehat{y}^*$ cannot be revealed through evaluation queries. Second, since we have $f(x^*) = 1$, no information of $\widehat{y}^*$ is used to create HList, and thus no information of $\widehat{y}^*$ is revealed through hash queries. In summary, $\mathcal{A}$ cannot obtain any information on $\widehat{y}^*$, and thus $\mathcal{A}$ cannot notice any difference if that is replaced by a fresh random element. ■

Lemma 4.25. *We have* $|\Pr[\mathsf{T}_3] - 1/2| = 0$.

Proof. Game 3 uses no information on coin, and thus $\mathcal{A}$ cannot distinguish cases of $\mathsf{coin} = 0$ and $\mathsf{coin} = 1$ with a positive advantage. ■

This completes proof of the adaptive single-key security. ■

5 Private Constrained PRF for Bit-fixing

In this section, we construct a single-key private CPRF for bit-fixing. Our scheme is selectively secure under the DDH assumption. We also construct an adaptively secure single-key private CPRF for bit-fixing in the ROM in Section 5.2.

Bit-fixing functions. First, we define a function class of bit-fixing functions formally. The class $\mathcal{BF} = \{\mathcal{BF}_n\}_{n\in\mathbb{N}}$ of bit-fixing functions is defined as follows [21]. $\mathcal{BF}_n$ is defined to be a set $\{\mathsf{BF}_c\}_{c\in\{0,1,*\}^n}$ where $\mathsf{BF}_c(x) := \begin{cases} 0 & \text{if for all } i,\ c_i = * \text{ or } x_i = c_i \\ 1 & \text{otherwise} \end{cases}$. By an abuse of notation, we often write c to mean BF_c when that is given as an input to an algorithm.

CIH for affine functions. We introduce the notion of affine functions for CIH since it is used in our private CPRF for bit-fixing. *The class of affine functions with respect to a group generator* GGen, denoted by $\Phi^{\mathsf{aff}} = \{\Phi^{\mathsf{aff}}_{\lambda,z}\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$, is a function class satisfying the following property for every $(\lambda, z) \in \mathbb{N}\times\{0,1\}^*$: If z can be parsed as a tuple $(\mathcal{G}, m, z')$ so that $\mathcal{G} = (\mathbb{G}, p)$ is a group description output by $\mathsf{GGen}(1^\lambda)$, $m \in \mathbb{N}$, and $z' \in \{0,1\}^*$, then we have $\Phi^{\mathsf{aff}}_{\lambda,z} = \{\phi_{\vec{u},\vec{v}} : \mathbb{Z}_p^m \to \mathbb{Z}_p^m \mid \vec{u} \in (\mathbb{Z}_p^*)^m, \vec{v} \in \mathbb{Z}_p^m\}$, where for each $\vec{u}, \vec{v}$, $\phi_{\vec{u},\vec{v}}(\vec{x}) := \vec{u} \odot \vec{x} + \vec{v} \in \mathbb{Z}_p^m$ and $\odot$ denotes the component-wise multiplication in $\mathbb{Z}_p$.

We will use the following theorem that is implicit in [ABPP14] (see also Remark 2.12).

Theorem 5.1. *(implicit in [ABPP14, Theorem 7]) Let* GGen *be a group generator. If the DDH assumption holds with respect to* GGen*, then for any polynomial $m = m(\lambda) \in \Omega(\lambda)$, there exists a Φ^{aff}-CIH* $\mathsf{CIH}_{\mathsf{aff}} = (\mathsf{PrmGen}_{\mathsf{aff}}, \mathsf{Eval}_{\mathsf{aff}})$ *with respect to* GGen*, with the following property: For all $\lambda \in \mathbb{N}$, if $\mathcal{G} = (\mathbb{G}, p) \xleftarrow{\mathsf{R}} \mathsf{GGen}(1^\lambda)$ and $\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{PrmGen}_{\mathsf{aff}}(\mathcal{G})$, then* pp *can be parsed as $(\mathcal{G}, m, z')$ for some $z' \in \{0,1\}^*$, and furthermore $\mathsf{Eval}_{\mathsf{aff}}(\mathsf{pp}, \cdot)$ is a function with domain $\mathbb{Z}_p^m$ and range $\mathbb{G}$.*

This theorem is derived from the following facts. (1) Abdalla et al. [ABPP14] constructed RKA-PRF for affine functions based on the DDH assumption. (2) Bellare and Cash [BC10b] show that RKA-PRF for a function class implies RKA-PRG for the same function class. (3) Our definition of CIH is the same as that of RKA-PRG (See Remark 2.12).

5.1 Construction in the Standard Model

Construction.

Here, we give a construction of a selectively secure private CPRF for bit-fixing. Our CPRF is built on a Φ^{aff}-CIH, which is known to exist under the DDH assumption [ABPP14]. Let GGen be a group generator that given 1^λ, generates a description of group of an ℓ_p-bit prime order, and $\mathsf{CIH}_{\mathsf{aff}} = (\mathsf{PrmGen}_{\mathsf{aff}}, \mathsf{Eval}_{\mathsf{aff}})$ be a Φ^{aff}-CIH. For simplicity, we denote $\mathsf{Eval}_{\mathsf{CIH}}(\mathsf{pp}_{\mathsf{CIH}}, \cdot)$ by $\mathsf{CIH}_{\mathsf{aff}}(\cdot)$ when $\mathsf{pp}_{\mathsf{CIH}}$ is clear. Our scheme $\mathsf{CPRF}_{\mathsf{priv,std}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ is described as follows. Let $n(\lambda)$ (often denoted as n for short) be an integer, which is used as an input length of $\mathsf{CPRF}_{\mathsf{priv,std}}$.

$\mathsf{Setup}(1^\lambda)$: It generates $\mathcal{G} \xleftarrow{\mathsf{R}} \mathsf{GGen}(1^\lambda)$ to obtain the group description $\mathcal{G} := (\mathbb{G}, p)$, and runs $\mathsf{pp}_{\mathsf{CIH}} \xleftarrow{\mathsf{R}} \mathsf{PrmGen}_{\mathsf{aff}}(\mathcal{G})$ to obtain $\mathsf{pp}_{\mathsf{CIH}} := (\mathcal{G}, m, z')$. Recall that $\mathsf{pp}_{\mathsf{CIH}}$ specifies a domain $\mathbb{Z}_p^m$ and a range $\mathcal{R}$ of $\mathsf{CIH}_{\mathsf{aff}}$. It outputs $\mathsf{pp} := (\mathsf{pp}_{\mathsf{CIH}}, 1^n)$.

$\mathsf{KeyGen}(\mathsf{pp})$: It chooses $s_{i,b,j} \xleftarrow{\mathsf{R}} \mathbb{Z}_p$ for $i \in [n]$, $b \in \{0,1\}$ and $j \in [m]$, and outputs $\mathsf{msk} := \{s_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]}$.

[21] According to the definition given in Section 2.4, we should give $\mathcal{BF}_{\lambda,n}$ for all $\lambda \in \mathbb{N}$ and $n \in \mathbb{N}$. However, since $\mathcal{BF}_{\lambda,n}$ is the same for all λ if n is fixed in the case of the bit-fixing, we use this simpler notation.

$\mathsf{Eval}(\mathsf{msk}, x)$: It parses $\{s_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]} \leftarrow \mathsf{msk}$. It computes $X_j := \sum_{i=1}^{n} s_{i,x_i,j}$ for $j \in [m]$. Then it computes $y := \mathsf{CIH}_{\mathsf{aff}}(X_1, ..., X_m)$ and outputs it.

$\mathsf{Constrain}(\mathsf{msk}, c \in \{0,1,*\}^n)$**:** It parses $\{s_{i,b}\}_{i\in[n],b\in\{0,1\}} \leftarrow \mathsf{msk}$, picks $\alpha_j \overset{\mathsf{R}}{\leftarrow} \mathbb{Z}_p$ for $j \in [m]$. Then it defines $\{t_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]}$ as follows. For all $i \in [n]$, $b \in \{0,1\}$ and $j \in [m]$, it sets

$$t_{i,b,j} := \begin{cases} s_{i,b,j} & \text{If } c_i = * \text{ or } b = c_i \\ s_{i,b,j} - \alpha_j & \text{If } c_i \neq * \text{ and } b = 1 - c_i \end{cases} .$$

Then it outputs $\mathsf{sk}_c := \{t_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]}$.

$\mathsf{CEval}(\mathsf{sk}_c, x)$**:** It parses $\{t_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]} \leftarrow \mathsf{sk}_c$, computes $X_j := \sum_{i=1}^{n} t_{i,x_i,j}$ for $j \in [m]$ and $y := \mathsf{CIH}_{\mathsf{aff}}(X_1, ..., X_m)$, and outputs y.

Correctness.

For any $\lambda \in \mathbb{N}$ and $c \in \{0,1,*\}^n$, we let $\mathsf{pp} \overset{\mathsf{R}}{\leftarrow} \mathsf{Setup}(1^\lambda)$, $\{s_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]} = \mathsf{msk} \overset{\mathsf{R}}{\leftarrow} \mathsf{KeyGen}(\mathsf{pp}, 1^n)$ and $\{t_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]} = \mathsf{sk}_c \overset{\mathsf{R}}{\leftarrow} \mathsf{Constrain}(\mathsf{msk}, c)$. For any $x \in \{0,1\}^n$ such that $\mathsf{BF}_c(x) = 0$ holds, we have $t_{i,x_i,j} = s_{i,x_i,j}$ for all $i \in [n]$ and $j \in [m]$. Therefore we have $\mathsf{CEval}(\mathsf{sk}_c, x) = \mathsf{CIH}_{\mathsf{aff}}(\sum_{i=1}^{n} t_{i,x_i,1}, ..., \sum_{i=1}^{n} t_{i,x_i,m}) = \mathsf{CIH}_{\mathsf{aff}}(\sum_{i=1}^{n} s_{i,x_i,1}, ..., \sum_{i=1}^{n} s_{i,x_i,m}) = \mathsf{Eval}(\mathsf{msk}, x)$.

Security.

Theorem 5.2. *If* CIH *is* Φ^{aff}*-CIH and* $2^{2n-m\ell_p}$ *is negligible, then the above scheme is a selectively single-key secure CPRF for* $\mathcal{BF}$ *with a selective single-key privacy.*

Proof of Theorem 5.2. Due to Lemma 2.10, we only have to prove that the above scheme is selectively single-key simulation-secure that is defined in Section 2.4. We consider a simulator $\mathcal{S}$ described below.

$\mathcal{S}(1^\lambda)$: This algorithm chooses $t_{i,b,j} \overset{\mathsf{R}}{\leftarrow} \mathbb{Z}_p$ for $i \in [n]$, $b \in \{0,1\}$ and $j \in [m]$ and outputs $\mathsf{sk} = \{t_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]}$.

What we have to prove is that for any admissible adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$|\Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF}_{\mathsf{priv,std}},\mathcal{BF},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF}_{\mathsf{priv,std}},\mathcal{BF},\mathcal{S},\mathcal{A}}(\lambda) = 1]|$$

is negligible. (Recall that an admissible adversary only makes evaluation queries x such that $f(x) = 1$ where f is constraint specified by $\mathcal{A}_1$ and does not make the same query twice.)

To prove that, we consider the following sequence of games. Let Q be the maximum number of $\mathcal{A}$'s evaluation queries.

$\mathsf{Game}\ 0$**:** This game is $\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF}_{\mathsf{priv,std}},\mathcal{BF},\mathcal{A}}(\lambda)$ itself. Namely,

```
$(\mathbb{G}, p) := \mathcal{G} \overset{\mathsf{R}}{\leftarrow} \mathsf{GGen}(1^\lambda)$ and $\mathsf{pp} := \mathsf{pp}_{\mathsf{CIH}} \overset{\mathsf{R}}{\leftarrow} \mathsf{PrmGen}_{\mathsf{aff}}(\mathcal{G})$
$\mathsf{msk} := \{s_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]} \overset{\mathsf{R}}{\leftarrow} \mathsf{KeyGen}(\mathsf{pp})$
$(c, \mathsf{st}_{\mathcal{A}}) \overset{\mathsf{R}}{\leftarrow} \mathcal{A}_1(\mathsf{pp})$
$\mathsf{sk}_c := \{t_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]} \overset{\mathsf{R}}{\leftarrow} \mathsf{Constrain}(\mathsf{msk}, f)$
$\widehat{\mathsf{coin}} \overset{\mathsf{R}}{\leftarrow} \mathcal{A}_2^{\mathsf{Eval}(\mathsf{msk},\cdot)}(\mathsf{sk}_c, \mathsf{st}_{\mathcal{A}})$
Return $\widehat{\mathsf{coin}}$
```

where we describe $\mathsf{Eval}(\mathsf{msk}, \cdot)$ below.

$\mathsf{Eval}(\mathsf{msk}, \cdot)$**:** Given $x \in \{0,1\}^n$ as input, it returns $\mathsf{Eval}(\mathsf{msk}, x)$.

Game 1**:** In this game, we modify how $\{X_j\}_{j\in[m]}$ is generated by the evaluation oracle. Namely, the oracle $\mathsf{Eval}(\mathsf{msk},\cdot)$ is modified as follows.

$\mathsf{Eval}(\mathsf{msk},\cdot)$**:** Given the k-th query $x^{(k)}$ from $\mathcal{A}$, for $j\in[m]$, it computes

$$u^{(k)} := |\{i\in[n] : c_i \neq * \wedge x_i^{(k)} = 1-c_i\}|,$$
$$v_j^{(k)} := \sum_{i=1}^{n} t_{i,x_i^{(k)},j}.$$

Then it computes $X_j^{(k)} := u^{(k)}\alpha_j + v_j^{(k)}$ for $j\in[m]$ and $y^{(k)} := \mathsf{CIH}_{\mathsf{aff}}(X_1^{(k)},...,X_m^{(k)})$, and returns $y^{(k)}$.

The rest remains unchanged from the previous game.

Game 2**:** In this game, we modify how to generate $\{t_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]}$. Namely, for all $i\in[n]$, $b\in\{0,1\}$, and $j\in[m]$, the game generates $t_{i,b,j}\overset{\mathsf{R}}{\leftarrow}\mathbb{Z}_p$. We note that α_j is still generated as $\alpha_j\overset{\mathsf{R}}{\leftarrow}\mathbb{Z}_p$ for $j\in[m]$ similarly to the previous game. The rest remains unchanged from the previous game.

Game 3**:** In this game, we modify how the output y is computed by the evaluation query. Namely, the oracle $\mathsf{Eval}(\mathsf{msk},\cdot)$ is modified as follows.

$\mathsf{Eval}(\mathsf{msk},\cdot)$**:** Given the k-th query $x^{(k)}$ from $\mathcal{A}$, it picks a random function $\mathsf{RF}\overset{\mathsf{R}}{\leftarrow}\mathtt{Func}(\mathbb{Z}_p^m,\mathcal{Y})$. For $j\in[m]$, it computes

$$u^{(k)} := |\{i\in[n] : c_i \neq * \wedge x_i^{(k)} = 1-c_i\}|,$$
$$v_j^{(k)} := \sum_{i=1}^{n} t_{i,x_i^{(k)},j}.$$

Then it computes $X_j^{(k)} := u^{(k)}\alpha_j + v_j^{(k)}$ for $j\in[m]$ and $y^{(k)} := \mathsf{RF}(X_1^{(k)},...,X_m^{(k)})$, and returns $y^{(k)}$.

Game 4**:** In this game, we modify how the output y is computed by the evaluation query. Namely, the oracle $\mathsf{Eval}(\mathsf{msk},\cdot)$ is modified as follows.

$\mathsf{Eval}(\mathsf{msk},\cdot)$**:** Given the k-th query $x^{(k)}$ from $\mathcal{A}$, it picks $y^{(k)}\overset{\mathsf{R}}{\leftarrow}\mathcal{Y}$ and returns $y^{(k)}$.

It is easy to see that this game is identical to $\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF}_{\mathsf{priv,std}},\mathcal{BF},n,\mathcal{A}}(\lambda)$.

Let T_i be the event that $\mathcal{A}$ outputs 1 in Game i. Then we have $\Pr[\mathsf{T}_0] = \Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}real}}_{\mathsf{CPRF}_{\mathsf{priv,std}},\mathcal{BF},\mathcal{S},\mathcal{A}}(\lambda) = 1]$ and $\Pr[\mathsf{T}_4] = \Pr[\mathsf{Expt}^{\mathsf{cprf\text{-}sim\text{-}ideal}}_{\mathsf{CPRF}_{\mathsf{priv,std}},\mathcal{BF},\mathcal{S},\mathcal{A}}(\lambda) = 1]|$. Thus what we have to prove is that $|\Pr[\mathsf{T}_4]-\Pr[\mathsf{T}_0]|$ is negligible. We prove it by the following lemmas.

Lemma 5.3. *We have* $\Pr[\mathsf{T}_1] = \Pr[\mathsf{T}_0]$.

Proof. The difference between these games is that for $i\in[n]$, $j\in[m]$ and $k\in[Q]$, $X_j^{(k)}$ is generated as

$$X_j^{(k)} := \sum_{i=1}^{n} s_{i,x_i^{(k)},j}$$

in Game 0 whereas it is generated as

$$X_j^{(k)} := u^{(k)}\alpha_j + v_j^{(k)}$$

in Game 1. For $i \in [n]$, $j \in [m]$ and $k \in [q]$, we have

$$s_{i,x_i^{(k)},j} = \begin{cases} t_{i,x_i^{(k)},j} & \text{If } c_i = * \text{ or } x_i^{(k)} = c_i \\ t_{i,x_i^{(k)},j} + \alpha_j & \text{If } c_i \neq * \text{ and } x_i^{(k)} = 1 - c_i \end{cases}$$

We also have

$$u^{(k)} := |\{i \in [n] : c_i \neq * \wedge x_i^{(k)} = 1 - c_i\}|,$$
$$v_j^{(k)} := \sum_{i=1}^{n} t_{i,x_i^{(k)},j}.$$

by the definition. By using the above equations, it is easy to see that we have

$$u^{(k)}\alpha_j + v_j^{(k)} = \sum_{i=1}^{n} s_{i,x_i^{(k)},j}$$

for all $i \in [n]$, $j \in [m]$ and $k \in [Q]$. Therefore these games are identical from the view of $\mathcal{A}$. ■

Lemma 5.4. *We have* $\Pr[\mathsf{T}_2] = \Pr[\mathsf{T}_1]$.

Proof. It is easy to see that the joint distributions of $\{t_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]}$ and $\{\alpha_j\}_{j\in[m]}$ in Game 1 and Game 2 are completely identical. Therefore these games are identical from the view of $\mathcal{A}$. ■

Lemma 5.5. *If* CIH *is* Φ^{aff}*-CIH, then* $|\Pr[\mathsf{T}_3] - \Pr[\mathsf{T}_2]|$ *is negligible.*

Proof. We construct an adversary $\mathcal{B}$ against CIH as follows.

$\mathcal{B}^{\mathcal{O}_{\mathsf{CIH}}(\cdot)}(\mathsf{pp}_{\mathsf{CIH}})$**:** It sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{CIH}}, 1^n)$ and runs $\mathcal{A}_1(\mathsf{pp}_{\mathsf{CIH}})$, which outputs $(c, \mathsf{st}_{\mathcal{A}})$. Then it generates $t_{i,b,j} \xleftarrow{\mathsf{R}} \mathbb{Z}_p$ for $i \in [n]$, $b \in \{0,1\}$ and $j \in [m]$ and runs $\mathcal{A}_2^{\mathsf{Eval}(\mathsf{msk},\cdot)}(\mathsf{sk}_c, \mathsf{st}_{\mathcal{A}})$. It simulates the oracle $\mathsf{Eval}(\mathsf{msk},\cdot)$ as follows. When $\mathcal{A}_2$ makes its k-th evaluation query $x^{(k)}$, it computes

$$u^{(k)} := |\{i \in [n] : c_i \neq * \wedge x_i^{(k)} = 1 - c_i\}|,$$
$$v_j^{(k)} := \sum_{i=1}^{n} t_{i,x_i^{(k)},j}.$$

for $j \in [m]$. Then it queries $(\phi_1^{(k)}, ..., \phi_m^{(k)})$ to its own oracle where $\phi_j^{(k)}$ is an affine function defined by $\phi_j^{(k)}(Z) := u^{(k)}Z + v_j^{(k)}$ for $j \in [m]$. Let $y^{(k)}$ be the response by the oracle. Then $\mathcal{B}$ gives $y^{(k)}$ to $\mathcal{A}_2$ as the response to the query. Finally, $\mathcal{A}_2$ outputs a bit $\widehat{\mathsf{coin}}$. Then, $\mathcal{B}$ outputs the same bit $\widehat{\mathsf{coin}}$ as its guess.

The above completes the description of $\mathcal{B}$. We prove that $\mathcal{B}$ distinguishes whether the challenge coin for $\mathcal{B}$ is $\mathsf{coin} = 1$ (i.e., $R_1, \ldots, R_m \xleftarrow{\mathsf{R}} \mathbb{Z}_p$ are chosen during the game and $\mathcal{O}_{\mathsf{CIH}}$ returns $\mathsf{CIH}_{\mathsf{aff}}(\phi_1(R_1), \ldots, \phi_m(R_m))$ on input $\mathcal{B}$'s query $(\phi_1, \ldots, \phi_m)$) or $\mathsf{coin} = 0$ (i.e., $\mathcal{O}_{\mathsf{CIH}}$ returns $\mathsf{RF}(\phi_1(R_1), \ldots, \phi_m(R_m))$ on input $\mathcal{B}$'s query $(\phi_1, \ldots, \phi_m)$, where $\mathsf{RF} \xleftarrow{\mathsf{R}} \mathsf{Func}(\mathbb{Z}_p^m, \mathcal{R})$).

First, we prove that $\mathcal{B}$ is an admissible adversary Φ^{aff}-CIH. Since $\mathcal{A}$ is an admissible adversary against the simulation-based security of the private CPRF, all evaluation queries $x^{(k)}$ made by $\mathcal{A}$ satisfy

$\mathsf{BF}_c(x^{(k)}) = 1$. Therefore there exists $i^{(k)} \in [n]$ that satisfies $c_{i^{(k)}} \neq *$ and $x^{(k)}_{i^{(k)}} = 1 - c_{i^{(k)}}$. Since we have $u^{(k)} \leq n < p$, we have $u^{(k)} \neq 0 \mod p$ for all $k \in [Q]$, which implies that $\mathcal{B}$ is an admissible adversary.

We then observe that if we have $\mathsf{coin} = 1$, then $\mathcal{B}$ perfectly simulates the environment of Game 2 (where α_j chosen in the CIH experiment is implicitly set as $\alpha_j := R_j$ for $j \in [m]$). On the other hand, if $\mathsf{coin} = 0$, then $\mathcal{B}$ perfectly simulates the environment of Game 3 (again α_j is set as $\alpha_j := R_j$). Therefore we have $|\Pr[\mathsf{T}_3] - \Pr[\mathsf{T}_2]| = \mathsf{Adv}^{\mathsf{cih}}_{\mathsf{CIH}_{\mathsf{aff}},\Phi^{\mathsf{aff}},\mathcal{B}}(\lambda)$.

■

Lemma 5.6. $\Pr[\mathsf{T}_4] - \Pr[\mathsf{T}_3] \leq 2^{2n - m\ell_p}$

Proof. Let Col denotes the event that there exists $k_1 \neq k_2$ such that

$$(X_1^{(k_1)}, ..., X_m^{(k_1)}) = (X_1^{(k_2)}, ..., X_m^{(k_2)})$$

in Game 3. It is easy to see that we have

$$|\Pr[\mathsf{T}_4] - \Pr[\mathsf{T}_3]| \leq \Pr[\mathsf{Col}]$$

since unless Col occurs, $\mathsf{RF}(X_1^{(k)}, ..., X_m^{(k)})$ for each $k \in [Q]$ is independently and uniformly distributed on $\mathcal{Y}$. In the following, we give an upper bound for $\Pr[\mathsf{Col}]$. We fix c and $\{\alpha_j\}_{j\in[m]}$, and define a keyed function $F_t : \{0,1\}^n \to \mathbb{Z}_p^m$ by

$$F_{\vec{t}}(x) := (u\alpha_m + v_1, ..., u\alpha_m + v_m)$$

where $\vec{t}$ denotes $\{t_{i,b,j}\}_{i\in[n],b\in\{0,1\},j\in[m]} \in \mathbb{Z}_p^{2nm}$, and for $j \in [m]$, we define

$$u := |\{i \in [n] : c_i \neq * \wedge x_i = 1 - c_i\}|,$$
$$v_j := \sum_{i=1}^{n} t_{i,x_i,j}.$$

Then it is easy to see that for $k \in [q]$, we have

$$(X_1^{(k)}, ..., X_m^{(k)}) = F_{\vec{t}}(x^{(k)}).$$

Since we assume that $x^{(k_1)} \neq x^{(k_2)}$ for all $k_1 \neq k_2$, we have

$$\Pr[\mathsf{Col}] \leq \Pr_{\vec{t} \xleftarrow{\mathsf{R}} \mathbb{Z}_p^{2nm}} [\exists (x, x') \in (\{0,1\}^n)^2 \text{ s.t. } x \neq x' \wedge F_t(x) = F_t(x')].$$

In the following, we give an upper bound for the right hand term. It is easy to see that $F_{\vec{t}}$ is pairwise independent, and especially for any fixed $x \neq x'$, we have

$$\Pr_{\vec{t} \xleftarrow{\mathsf{R}} \mathbb{Z}_p^{2nm}} [F_t(x) = F_t(x')] \leq 1/p^m \leq 2^{-m\ell_p}.$$

Therefore by the union bound, we have

$$\Pr_{\vec{t} \xleftarrow{\mathsf{R}} \mathbb{Z}_p^{2nm}} [\exists (x, x') \in (\{0,1\}^n)^2 \text{ s.t. } x \neq x' \wedge F_t(x) = F_t(x')] \leq 2^{2n - m\ell_p}.$$

Combining the above equations, the lemma is proven. ■

By combining the above lemmas, Theorem 5.2 is proven. ■

5.2 Construction in the Random Oracle Model

We give a construction of an adaptively secure private constrained PRF for bit-fixing in the random oracle model. Our scheme $\mathsf{CPRF}_{\mathsf{priv,ro}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ is described as follows. Let $n(\lambda)$ (often denoted as n for short) be an integer, which is used as an input length of $\mathsf{CPRF}_{\mathsf{priv,std}}$ and H be a hash function from $\{0,1\}^{n\lambda}$ to $\mathcal{R}$.

$\mathsf{Setup}(1^\lambda)$**:** It sets $\mathsf{pp} := (1^\lambda, 1^n)$ and outputs pp.

$\mathsf{KeyGen}(\mathsf{pp})$**:** It chooses $s_{i,b} \xleftarrow{\mathsf{R}} \{0,1\}^\lambda$ for $i \in [n]$ and $b \in \{0,1\}$. It outputs $\mathsf{msk} := \{s_{i,b}\}_{i\in[n],b\in\{0,1\}}$.

$\mathsf{Eval}(\mathsf{msk}, x)$**:** It computes and outputs $y := H(s_{1,x_1}\|...\|s_{n,x_n})$.

$\mathsf{Constrain}(\mathsf{msk}, c \in \{0,1,*\}^n)$**:** It parses $\{s_{i,b}\}_{i\in[n],b\in\{0,1\}} \leftarrow \mathsf{msk}$. Then, it defines $\{t_{i,b}\}_{i\in[n],b\in\{0,1\}}$ as follows. For all $i \in [n]$ and $b \in \{0,1\}$, it sets

$$t_{i,b} \begin{cases} := s_{i,b} & \text{If } c_i = * \text{ or } b = c_i \\ \xleftarrow{\mathsf{R}} \mathbb{Z}_p & \text{If } c_i \neq * \text{ and } b = 1 - c_i \end{cases}.$$

Then it outputs $\mathsf{sk}_c := \{t_{i,b}\}_{i\in[n],b\in\{0,1\}}$.

$\mathsf{CEval}(\mathsf{sk}_c, x)$**:** It parses $\{t_{i,b}\}_{i\in[n],b\in\{0,1\}} \leftarrow \mathsf{sk}_c$. It computes $H(t_{1,x_1}\|...\|t_{n,x_n})$ and outputs it.

Correctness.

For any $\lambda \in \mathbb{N}$ and $c \in \{0,1,*\}^n$, we let $\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda)$, $\{s_{i,b}\}_{i\in[n],b\in\{0,1\}} = \mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{pp})$ and $\{t_{i,b}\}_{i\in[n],b\in\{0,1\}} = \mathsf{sk}_c \xleftarrow{\mathsf{R}} \mathsf{Constrain}(\mathsf{msk}, c)$. For any $x \in \{0,1\}^n$ such that $\mathsf{BF}_c(x) = 0$ holds, we have $t_{i,x_i} = s_{i,x_i}$ for all $i \in [n]$. Therefore, we have $\mathsf{CEval}(\mathsf{sk}_c, x) = H(t_{1,x_1}\|...\|t_{n,x_n}) = H(s_{1,x_1}\|...\|s_{n,x_n}) = \mathsf{Eval}(\mathsf{msk}, x)$.

Security.

Theorem 5.7. *The above scheme is an adaptively single-key secure and private CPRF for $\mathcal{BF}$ in the random oracle model where H is modeled as a random oracle.*

Proof.

CPRF security. We first prove the above scheme satisfies the security as an ordinary CPRF. Actually, this can be proven very similarly to Theorem 4.21, and many parts of proofs are virtually identical.

For an admissible adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we consider the following sequence of games. For simplicity, we assume that $\mathcal{A}$ makes a challenge query only once (see Remark 2.6). We also assume that $\mathcal{A}$ never makes the same query twice and all evaluation queries x made by $\mathcal{A}_2$ satisfy $\mathsf{BF}_c(x) = 1$.

$\mathsf{Game}\ 0$**:** This game is $\mathsf{Expt}^{\mathsf{cprf}}_{\mathsf{CPRF}_{\mathsf{priv,ro}},\mathcal{BF},\mathcal{A}}(\lambda)$. Namely,

$\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$
$s_{i,b} \xleftarrow{\mathsf{R}} \{0,1\}^\lambda$ for $i \in [n]$ and $b \in \{0,1\}$
$\mathsf{msk} := \{s_{i,b}\}_{i\in[n],b\in\{0,1\}}$
$y^* \xleftarrow{\mathsf{R}} \mathcal{R}$
$(c, \mathsf{st}_\mathcal{A}) \xleftarrow{\mathsf{R}} \mathcal{A}_1^{\mathsf{RO}(\cdot),\mathcal{O}_{\mathsf{Eval}}(\cdot),\mathcal{O}_{\mathsf{Chal}}(\cdot)}(\mathsf{pp})$
$$t_{i,b} \begin{cases} := s_{i,b} & \text{If } c_i = * \text{ or } b = c_i \\ \xleftarrow{\mathsf{R}} \mathbb{Z}_p & \text{If } c_i \neq * \text{ and } b = 1 - c_i \end{cases}$$

$\mathsf{sk}_c := \{t_{i,b}\}_{i\in[n],b\in\{0,1\}}$
$\widehat{\mathsf{coin}} \xleftarrow{\mathsf{R}} \mathcal{A}_2^{\mathsf{RO}(\cdot),\mathcal{O}_{\mathsf{Eval}}(\cdot),\mathcal{O}_{\mathsf{Chal}}(\cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$
Return $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$
where $\mathsf{RO}(\cdot)$, $\mathcal{O}_{\mathsf{Eval}}(\cdot)$ and $\mathcal{O}_{\mathsf{Chal}}(\cdot)$ are oracles described below.

$\mathsf{RO}(\cdot)$**:** Given $X \in \{0,1\}^{n\lambda}$ as input, it returns $H(X)$.

$\mathcal{O}_{\mathsf{Eval}}(\cdot)$**:** Given $x \in \{0,1\}^n$ as input, it returns $H(s_{1,x_1}\|...\|s_{n,x_n})$.

$\mathcal{O}_{\mathsf{Chal}}(\cdot)$**:** Given $x^* \in \{0,1\}^n$ as input, it returns $H(s_{1,x_1^*}\|...\|s_{n,x_n^*})$ if $\mathsf{coin} = 1$. Otherwise, it returns y^*.

We remark that we simplify the experiment compared to the definition given in Section 2.3 by using the assumption that $\mathcal{A}$ makes the challenge query at most once.

Game 1**:** In this game, the random oracle is sampled lazily. Namely, oracles RO, $\mathcal{O}_{\mathsf{Chal}}$ and $\mathcal{O}_{\mathsf{Eval}}$ are modified as follows. These oracles shares a list HList, which is initialized to be empty at the beginning of the game.

$\mathsf{RO}(\cdot)$**:** Given the input X, if there exists $y \in \mathcal{R}$ such that $(X, y) \in \mathsf{HList}$, then it returns y. Otherwise it picks $y \xleftarrow{\mathsf{R}} \mathcal{R}$, adds (X, y) to HList and returns y.

$\mathcal{O}_{\mathsf{Eval}}(\cdot)$**:** Given the input x, it computes $X := s_{1,x_1}\|...\|s_{n,x_n}$. If there exists $y \in \mathcal{R}$ such that $(X, y) \in \mathsf{HList}$, then it returns y. Otherwise it picks $y \xleftarrow{\mathsf{R}} \mathcal{R}$, adds (X, y) to HList and returns y.

$\mathcal{O}_{\mathsf{Chal}}(\cdot)$**:** Given the input x^*, if $\mathsf{coin} = 1$, then it works similarly to $\mathcal{O}_{\mathsf{Eval}}$. Otherwise it returns y^*.

Game 2**:** In this game, evaluation and challenge oracles do not refer to HList at all, and updates of HList by these oracles are delayed until $\mathcal{A}_1$ declares its constrain query c. Namely, $\mathcal{O}_{\mathsf{Chal}}$ and $\mathcal{O}_{\mathsf{Eval}}$ are modified as follows, and a procedure $\mathsf{HashSet}$ defined below runs immediately after $\mathcal{A}_1$ outputs $(c, \mathsf{st}_{\mathcal{A}})$. $\mathcal{O}_{\mathsf{Eval}}$ and $\mathcal{O}_{\mathsf{Chal}}$ maintain a list EList, which is initialized to be empty at the beginning of the game, instead of HList. Note that RO still maintains and refers HList as in the previous game.

$\mathcal{O}_{\mathsf{Eval}}(\cdot)$**:** Given the input x, if there exists $y \in \mathcal{R}$ such that $(x, y) \in \mathsf{EList}$, then it returns y. Otherwise it picks $y \xleftarrow{\mathsf{R}} \mathcal{R}$, adds (x, y) to EList and returns y.

$\mathcal{O}_{\mathsf{Chal}}(\cdot)$**:** Given the input x^*, if $\mathsf{coin} = 1$, then it works similarly to $\mathcal{O}_{\mathsf{Eval}}$. Otherwise it returns y^*.

$\mathsf{HashSet}$**:** For all x such that there exists $y \in \mathcal{R}$ such that$(x, y) \in \mathsf{EList}$ and $\mathsf{BF}_c(x) = 0$, it computes $X := s_{1,x_1}\|...\|s_{n,x_n}$ and adds (X, y) to HList.

Game 3**:** In this game, a challenge oracle always returns y^* regardless of coin. Namely, $\mathcal{O}_{\mathsf{Chal}}$ is modified as follows.

$\mathcal{O}_{\mathsf{Chal}}(\cdot)$**:** Given the input x^*, it returns y^*.

This completes the description of games. Let T_i be the event that Game i returns 1. Then we have to prove that $|\Pr[\mathsf{T}_0] - 1/2|$ is negligible.

Lemma 5.8. *We have* $\Pr[\mathsf{T}_1] = \Pr[\mathsf{T}_0]$

Proof. The modification from Game 0 to Game 1 is just conceptual. ■

Lemma 5.9. *We have* $|\Pr[\mathsf{T}_2] - \Pr[\mathsf{T}_1]| \leq Q_H(1 + Q_E)(2^{-\lambda} + 2^{-n\lambda}) + n \cdot 2^{-\lambda}$ *where* Q_H *and* Q_E *denote the numbers of* $\mathcal{A}$*'s hash queries and evaluation queries, respectively.*

Proof. In the following, HList_1 denotes the set of X such that there exists y satisfying $(X, y) \in \mathsf{HList}$ and EList_1 denotes the set of x such that there exists y satisfying $(x, y) \in \mathsf{EList}$. Game 2 differs from Game 1 only when either of the following events occurs.

1. $\mathcal{A}_1$ makes a query X to RO such that there exists $x \in \mathsf{EList}_1$ satisfying $X = s_{1,x_1}\|...\|s_{n,x_n}$ holds.
2. $\mathcal{A}_2$ makes a query X to RO such that there exists $x \in \mathsf{EList}_1$ satisfying $X = s_{1,x_1}\|...\|s_{n,x_n}$ and $\mathsf{BF}_c(x) = 1$ hold.
3. $\mathcal{A}_1$ or $\mathcal{A}_2$ makes a query x to $\mathcal{O}_{\mathsf{Eval}}$ or $\mathcal{O}_{\mathsf{Chal}}$ such that $X \in \mathsf{HList}_1$ holds where $X := s_{1,x_1}\|...\|s_{n,x_n}$.
4. $\mathcal{A}_1$ or $\mathcal{A}_2$ makes a distinct queries x and x' to $\mathcal{O}_{\mathsf{Eval}}$ or $\mathcal{O}_{\mathsf{Chal}}$ such that $s_{1,x_1}\|...\|s_{n,x_n} = s_{1,x'_1}\|...\|s_{n,x'_n}$.

If one of the above events occurs, then one of the events Bad_1, Bad_2 or Col defined below occurs . (If Event 1 occurs, then Bad_1 occurs, if Event 2 occurs, then Bad_2 occurs, if Event 3 occurs, then Bad_1 or Bad_2 occurs, and if Event 4 occurs, then Col occurs.)

Bad_1: At the point just after $\mathcal{A}_1$ halts (before HashSet runs) in Game 2, there exist $x \in \mathsf{EList}_1$ and $X \in \mathsf{HList}_1$ such that $X = s_{1,x_1}\|...\|s_{n,x_n}$ holds.

Bad_2: At the end of Game 2, there exist $x \in \mathsf{EList}_1$ and $X \in \mathsf{HList}_1$ such that $X = s_{1,x_1}\|...\|s_{n,x_n}$ and $f(x) = 1$ hold.

Col: $\mathcal{A}_1$ or $\mathcal{A}_2$ makes a distinct queries x and x' to $\mathcal{O}_{\mathsf{Eval}}$ or $\mathcal{O}_{\mathsf{Chal}}$ such that $s_{1,x_1}\|...\|s_{n,x_n} = s_{1,x'_1}\|...\|s_{n,x'_n}$.

Therefore we have $|\Pr[\mathsf{T}_2] - \Pr[\mathsf{T}_1]| \leq \Pr[\mathsf{Bad}_1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Col}]$.

Since $\mathcal{A}_1$ is given no information of $\{s_{i,b}\}_{i\in[n],b\in\{0,1\}}$, for all $x \in \mathsf{EList}_1$, we have $\Pr[s_{1,x_1}\|...\|s_{n,x_n} = X'] = 2^{-n\lambda}$ for any $X' \in \{0,1\}^{n\lambda}$. Since we have $|\mathsf{EList}_1| \leq (1 + Q_E)$[22] and $|\mathsf{HList}| \leq Q_H$, we have $\Pr[\mathsf{Bad}_1] \leq Q_H(1+Q_E)2^{-n\lambda}$. Similarly, $\mathcal{A}_2$ is given no information of $s_{i,b}$ such that $c_i \neq *$ and $b = 1-c_i$. Therefore for any x such that $\mathsf{BF}_c(x) = 1$, there exists i such that s_{i,x_i} is completely hidden from $\mathcal{A}_2$. Hence for all $x \in \mathsf{EList}$ such that $\mathsf{BF}_c(x) = 1$, we have $\Pr[s_{1,x_1}\|...\|s_{n,x_n} = X'] = 2^{-\lambda}$ for any $X' \in \{0,1\}^{n\lambda}$. Since we have $|\mathsf{EList}_1| \leq (1 + Q_E)$ and $|\mathsf{HList}| \leq Q_H$, we have $\Pr[\mathsf{Bad}_2] \leq Q_H(1 + Q_E)2^{-\lambda}$. Finally, we have $Pr[\mathsf{Col}] \leq \Pr[\exists i \in [n] \text{ s.t. } s_{i,0} = s_{i,1}] \leq n \cdot 2^{-\lambda}$ due to the union bound. Hence the lemma is proven. ∎

Lemma 5.10. *We have* $\Pr[\mathsf{T}_3] = \Pr[\mathsf{T}_2]$.

Proof. Let x^* be the $\mathcal{A}$'s challenge query and $\widehat{y}^*$ be the random value that is picked by $\mathcal{O}_{\mathsf{Chal}}$ for replying the challenge query when $\mathsf{coin} = 1$. (We remark that $\mathcal{O}_{\mathsf{Chal}}$ must pick a fresh random value $\widehat{y}^*$ since the challenge query x^* is different from all evaluation queries and thus $x^* \notin \mathsf{EList}_1$.) We claim that no information of $\widehat{y}^*$ is revealed to $\mathcal{A}$ except that from $\mathcal{O}_{\mathsf{Chal}}$. First, since we have $x \neq x^*$ for all evaluation queries x, $\widehat{y}^*$ cannot be revealed through evaluation queries. Second, since we have $\mathsf{BF}_c(x^*) = 1$, no information of $\widehat{y}^*$ is used to create HList, and thus no information of $\widehat{y}^*$ is revealed through hash queries. In summary, $\mathcal{A}$ cannot obtain any information on $\widehat{y}^*$, and thus $\mathcal{A}$ cannot notice any difference if that is replaced by a fresh random element. ∎

Lemma 5.11. *We have* $|\Pr[\mathsf{T}_3] - 1/2| = 0$.

Proof. Game 3 uses no information on coin, and thus $\mathcal{A}$ cannot distinguish cases of $\mathsf{coin} = 0$ and $\mathsf{coin} = 1$ with a positive advantage. ∎

This completes the proof of CPRF security.

[22] This should be $Q_E + 1$ rather than Q_E since $\mathcal{O}_{\mathsf{Chal}}$ also accesses EList.

Privacy. Next, we prove that the above scheme satisfies the privacy. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we consider the following sequence of games. We assume that $\mathcal{A}$ never makes the same query twice and all evaluation queries x made by $\mathcal{A}_2$ satisfy $\mathsf{BF}_{c_0}(x) = \mathsf{BF}_{c_1}(x) = 1$.

Game 0**:** This game is $\mathsf{Expt}^{\mathsf{cprf\text{-}priv}}_{\mathsf{CPRF}_{\mathsf{priv,ro}},\mathcal{BF},\mathcal{A}}(\lambda)$. Namely,

$\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$
$s_{i,b} \xleftarrow{\mathsf{R}} \{0,1\}^\lambda$ for $i \in [n]$ and $b \in \{0,1\}$
$\mathsf{msk} := \{s_{i,b}\}_{i\in[n],b\in\{0,1\}}$
$(c_0, c_1, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}_1^{\mathsf{RO}(\cdot),\mathcal{O}_{\mathsf{Eval}}(\cdot)}(\mathsf{pp})$
$t_{i,b} \begin{cases} := s_{i,b} & \text{If } c_{\mathsf{coin},i} = * \text{ or } b = c_{\mathsf{coin},i} \\ \xleftarrow{\mathsf{R}} \mathbb{Z}_p & \text{If } c_{\mathsf{coin},i} \neq * \text{ and } b = 1 - c_{\mathsf{coin},i} \end{cases}$
$\mathsf{sk}_{c_{\mathsf{coin}}} := \{t_{i,b}\}_{i\in[n],b\in\{0,1\}}$
$\widehat{\mathsf{coin}} \xleftarrow{\mathsf{R}} \mathcal{A}_2^{\mathsf{RO}(\cdot),\mathcal{O}_{\mathsf{Eval}}(\cdot)}(\mathsf{sk}_{c_{\mathsf{coin}}}, \mathsf{st}_{\mathcal{A}})$
Return $(\widehat{\mathsf{coin}} \stackrel{?}{=} \mathsf{coin})$. where $\mathsf{RO}(\cdot)$ and $\mathcal{O}_{\mathsf{Eval}}(\cdot)$ are oracles described below.

$\mathsf{RO}(\cdot)$**:** Given $X \in \{0,1\}^{n\lambda}$ as input, this algorithm returns $H(X)$.

$\mathcal{O}_{\mathsf{Eval}}(\cdot)$**:** Given $x \in \{0,1\}^n$ as input, this algorithm returns $H(s_{1,x_1}\|...\|s_{n,x_n})$.

Game 1**:** In this game, the random oracle is sampled lazily. Namely, oracles RO and $\mathcal{O}_{\mathsf{Eval}}$ are modified as follows. These oracles share a list HList, which is initialized to be empty at the beginning of the game.

$\mathsf{RO}(\cdot)$**:** Given the input X, if there exists $y \in \mathcal{R}$ such that $(X, y) \in \mathsf{HList}$, then it returns y. Otherwise it picks $y \xleftarrow{\mathsf{R}} \mathcal{R}$, adds (X, y) to HList and returns y.

$\mathcal{O}_{\mathsf{Eval}}(\cdot)$**:** Given the input x, it computes $X := s_{1,x_1}\|...\|s_{n,x_n}$. If there exists $y \in \mathcal{R}$ such that $(X, y) \in \mathsf{HList}$, then it returns y. Otherwise it picks $y \xleftarrow{\mathsf{R}} \mathcal{R}$, adds (X, y) to HList and returns y.

Game 2**:** In this game, an evaluation oracle does not refer to HList at all, and updates of HList by the oracle are delayed until $\mathcal{A}_1$ declares its constrain query. Namely, $\mathcal{O}_{\mathsf{Eval}}$ is modified as follows, and a procedure $\mathsf{HashSet}$ defined below runs immediately after $\mathcal{A}_1$ outputs $(c_0, c_1, \mathsf{st}_{\mathcal{A}})$. $\mathcal{O}_{\mathsf{Eval}}$ maintains a list EList, which is initialized to be empty at the beginning of the game, instead of HList. Note that RO still maintains and refers HList as in the previous game.

$\mathcal{O}_{\mathsf{Eval}}(\cdot)$**:** Given the input x, if there exists $y \in \mathcal{R}$ such that $(x, y) \in \mathsf{EList}$, then it returns y. Otherwise it picks $y \xleftarrow{\mathsf{R}} \mathcal{R}$, adds (x, y) to EList and returns y.

$\mathsf{HashSet}$**:** For all x such that there exists $y \in \mathcal{R}$ such that$(x, y) \in \mathsf{EList}$ and $\mathsf{BF}_{c_0}(x) = \mathsf{BF}_{c_1}(x) = 0$, it computes $X := s_{1,x_1}\|...\|s_{n,x_n}$ and adds (X, y) to HList.

Game 3**:** This game is the same as the previous game except that $t_{i,b}$ is independently and uniformly sampled from $\mathbb{Z}_p$ for $i \in [n]$ and $b \in \{0,1\}$.

Let T_i be the event that Game i returns 1. Then we have to prove that $|\Pr[\mathsf{T}_0] - 1/2|$ is negligible.

Lemma 5.12. *We have* $|\Pr[\mathsf{T}_2] - \Pr[\mathsf{T}_0]| \leq Q_H(1 + Q_E)(2^{-\lambda} + 2^{-n\lambda})$ *where* Q_H *and* Q_E *denote the numbers of* $\mathcal{A}$*'s hash queries and evaluation queries, respectively.*

Proof. This can be proven similarly to Lemma 5.8 and Lemma 5.9. ■

Lemma 5.13. *We have* $\Pr[\mathsf{T}_3] = \Pr[\mathsf{T}_2]$.

$\mathsf{Expt}^{\mathsf{cpa}}_{\mathsf{SKE},\mathcal{A}}(\lambda)$:	$\mathcal{O}_{\mathsf{Enc}}(\mathsf{msg}_0, \mathsf{msg}_1)$:
$\mathsf{coin} \xleftarrow{R} \{0,1\}$	Return $\mathsf{ct} \xleftarrow{R} \mathsf{Enc}(\mathsf{k}, \mathsf{msg}_{\mathsf{coin}})$.
$\mathsf{k} \xleftarrow{R} \{0,1\}^{\ell_{\mathsf{k}}}$	
$\widehat{\mathsf{coin}} \xleftarrow{R} \mathcal{A}^{\mathcal{O}_{\mathsf{Enc}}(\cdot,\cdot)}(1^\lambda)$	
Return $(\widehat{\mathsf{coin}} \stackrel{?}{=} \mathsf{coin})$.	

Figure 4: **Left:** The security experiment for SKE. **Right:** The definition of the oracle $\mathcal{O}_{\mathsf{Enc}}$ in the experiment.

Proof. If $\{s_{i,b}\}_{i\in[n],b\in\{0,1\}}$ is not given, then the distribution of $\{t_{i,b}\}_{i\in[n],b\in\{0,1\}}$ in Game 2 is uniformly and independently random. We remark that $\{s_{i,b}\}_{i\in[n],b\in\{0,1\}}$ is not used at all in Game 2 and Game 3. Therefore these games are identical from the view of $\mathcal{A}$. ■

Lemma 5.14. *We have* $|\Pr[\mathsf{T}_3] - 1/2| = 0$.

Proof. In Game 3, no information of coin is used. Therefore $\mathcal{A}$ cannot distinguish the cases of $\mathsf{coin} = 0$ and $\mathsf{coin} = 1$ with a positive advantage. ■

This completes the proof of privacy.

Combining the above lemmas, the theorem is proven. ■

6 Application to Secret-Key ABE

In this section, we give a construction of single-key secret key attribute-based encryption (SK-ABE) scheme based on a single-key secure CPRF. Our SK-ABE scheme achieves optimal ciphertext overhead. Namely, a ciphertext of our SK-ABE scheme can be as compact as any CPA secure symmetric key encryption (SKE) scheme. By instantiating the construction based on CPRFs given in Section 4, we obtain a single-key secure SK-ABE scheme for $\mathbf{NC}^1$ with optimal ciphertext overhead. To the best of our knowledge, this is the first construction of such a primitive based on traditional groups.

6.1 Definitions

Symmetric Key Encryption. We recall the definition of SKE. An SKE scheme with key size ℓ_{k}, a message space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$ consists of two PPT algorithms $(\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$.

$\mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg})$**:** This is the encryption algorithm that takes a key $\mathsf{k} \in \{0,1\}^{\ell_{\mathsf{k}}}$ and a message $\mathsf{msg} \in \mathcal{M}$ as input, and outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.

$\mathsf{SKE.Dec}(\mathsf{k}, \mathsf{ct})$**:** This is the decryption algorithm that takes a key $\mathsf{k} \in \{0,1\}^{\ell_{\mathsf{k}}}$ and a ciphertext $\mathsf{ct} \in \mathcal{C}$ as input, and outputs a message $\mathsf{msg} \in \mathcal{M}$.

For correctness of a SKE scheme, we require that for all $\lambda \in \mathbb{N}$, $\mathsf{k} \in \{0,1\}^{\ell_{\mathsf{k}}}$, $\mathsf{msg} \in \mathcal{M}$, we have $\mathsf{SKE.Dec}(\mathsf{k}, \mathsf{Enc}(\mathsf{k}, \mathsf{msg})) = \mathsf{msg}$.

The definition of the CPA security of SKE is given below.

Definition 6.1. *We say that an SKE scheme* $\mathsf{SKE} = (\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ *is CPA secure if for all PPT adversary $\mathcal{A}$, the advantage* $\mathsf{Adv}^{\mathsf{cpa}}_{\mathsf{SKE},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{cpa}}_{\mathsf{SKE},\mathcal{A}}(\lambda) = 1] - 1/2|$ *is negligible where* $\mathsf{Adv}^{\mathsf{cpa}}_{\mathsf{SKE},\mathcal{A}}(\lambda)$ *is an experiment defined in Figure 4.*

It is well known that we can construct a SKE scheme based on any PRF. The construction is roughly described as follows.

```
$\mathsf{Expt}^{\mathsf{single\text{-}key}}_{\mathsf{ABE},\mathcal{A}}(\lambda)$ :
    $\mathsf{coin} \overset{\mathsf{R}}{\leftarrow} \{0,1\}$
    $(\mathsf{pp}, \mathsf{msk}) \overset{\mathsf{R}}{\leftarrow} \mathsf{ABE.Setup}(1^\lambda)$
    $(f, \mathsf{st}_\mathcal{A}) \overset{\mathsf{R}}{\leftarrow} \mathcal{A}_1^{\mathcal{O}_{\mathsf{Enc}}(\cdot,(\cdot,\cdot))}(\mathsf{pp})$
    $\mathsf{sk}_f \overset{\mathsf{R}}{\leftarrow} \mathsf{Constrain}(\mathsf{msk}, f)$
    $\widehat{\mathsf{coin}} \overset{\mathsf{R}}{\leftarrow} \mathcal{A}_2^{\mathcal{O}_{\mathsf{Enc}}(\cdot,(\cdot,\cdot))}(\mathsf{sk}_f, \mathsf{st}_\mathcal{A})$
    Return $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$.
```

```
$\mathcal{O}_{\mathsf{Enc}}(x, (\mathsf{msg}_0, \mathsf{msg}_1))$ :
    Return $\mathsf{ct} \overset{\mathsf{R}}{\leftarrow} \mathsf{Enc}(\mathsf{msk}, x, \mathsf{msg}_{\mathsf{coin}})$.
```

Figure 5: **Left:** The security experiment for SK-ABE. **Right:** The definition of the oracle $\mathcal{O}_{\mathsf{Enc}}$ in the experiment.

$\mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg})$**:** It picks $r \overset{\mathsf{R}}{\leftarrow} \{0,1\}^\ell$ and outputs a ciphertext $\mathsf{ct} := (r, \mathsf{PRF}(\mathsf{k}, r) \oplus \mathsf{msg})$.

$\mathsf{SKE.Dec}(\mathsf{k}, \mathsf{ct})$**:** It parses $(r, \mathsf{ct}') \leftarrow \mathsf{ct}$ and outputs $\mathsf{PRF}(\mathsf{k}, r) \oplus \mathsf{ct}'$ and a ciphertext $\mathsf{ct} \in \mathcal{C}$ as input, and outputs a message $\mathsf{msg} \in \mathcal{M}$.

This scheme is CPA secure if $\ell = \omega(\log(\lambda))$. Intuitively, this can be seen by the fact that the probability that the same r is reused is negligible when we have $\ell = \omega(\log(\lambda))$ and encryption is done polynomial times.

Secret-key Attribute-based Encryption. Here, we recall the definition of SK-ABE. An SK-ABE scheme for a function class $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$ with an attribute space $\mathcal{X} \subseteq \{0,1\}^n$, a message space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$ consists of four PPT algorithms $(\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$.

$\mathsf{ABE.Setup}(1^\lambda) \overset{\mathsf{R}}{\to} (\mathsf{pp}, \mathsf{msk})$: This is the setup algorithm that takes a security parameter 1^λ as input, and outputs a public parameter pp and a master secret key msk. We assume that pp is given as input to all other algorithms without explicitly denoting it.[23]

$\mathsf{ABE.KeyGen}(\mathsf{msk}, f) \overset{\mathsf{R}}{\to} \mathsf{sk}_f$: This is the key generation algorithm that takes a master secret key msk and a function $f \in \mathcal{F}$ as input, and outputs a secret key sk_f.

$\mathsf{ABE.Enc}(\mathsf{msk}, x, \mathsf{msg}) \overset{\mathsf{R}}{\to} \mathsf{ct}$: This is the encryption algorithm that takes a master secret key msk, an attribute $x \in \mathcal{X}$ and a message $\mathsf{msg} \in \mathcal{M}$ as input, and outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.

$\mathsf{ABE.Dec}(\mathsf{sk}_f, x, \mathsf{ct}') \overset{\mathsf{R}}{\to} \mathsf{msg}$: This is the decryption algorithm that takes a secret key sk_f, an attribute $x \in \mathcal{X}$, a ciphertext $\mathsf{ct} \in \mathcal{C}$ as input, and outputs a message $\mathsf{msg} \in \mathcal{M}$.

For correctness of an SK-ABE scheme for a function class $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$, we require that for all $\lambda \in \mathbb{N}$, $\mathsf{msk} \overset{\mathsf{R}}{\leftarrow} \mathsf{ABE.KeyGen}(1^\lambda)$, $f \in \mathcal{F}_{\lambda,n}$, $x \in \mathcal{X}$ satisfying $f(x) = 0$, and $\mathsf{msg} \in \mathcal{M}$, we have

$$\mathsf{ABE.Dec}(\mathsf{ABE.KeyGen}(\mathsf{msk}, f), x, \mathsf{Enc}(\mathsf{msk}, x, \mathsf{msg})) = \mathsf{msg}.$$

Remark 6.2. We note that in our definition, the decryptable condition is "reversed" from a commonly used definition of (SK-)ABE, in the sense that correctness is required if $f(x) = 0$ instead of $f(x) = 1$. This is for compatibility to our definition of CPRF (See also Remark 2.4).

Then we define a security notion for AB-SKE. In this paper, we only consider single-key security where an adversary obtains at most one decryption key.

We say that an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the experiment $\mathsf{Expt}^{\mathsf{single\text{-}key}}_{\mathsf{ABE},\mathcal{A}}(\lambda)$ is *admissible* if $\mathcal{A}_1$ and $\mathcal{A}_2$ are PPT and respect the following restrictions:

[23]Since we consider the symmetric key setting, we can drop pp by letting msk include pp. We define pp for compatibility to our definition of CPRF.

- $f \in \mathcal{F}_{\lambda,n}$ holds for f output by $\mathcal{A}_1$.
- All queries $(x, (\mathsf{msg}_0, \mathsf{msg}_1))$ made by $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $f(x) = 1$.

Furthermore, we say that $\mathcal{A}$ is *key-selectively admissible* if, in addition to the above restrictions, $\mathcal{A}_1$ makes no query. That is, $\mathcal{A}$ sends no encryption query before it sends a key query.

Definition 6.3. *We say that an SK-ABE scheme* $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ *is adaptively single-key secure if for all admissible adversary* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathsf{single\text{-}key}}_{\mathsf{ABE},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{single\text{-}key}}_{\mathsf{ABE},\mathcal{A}}(\lambda) = 1] - 1/2|$ *is negligible where* $\mathsf{Adv}^{\mathsf{single\text{-}key}}_{\mathsf{ABE},\mathcal{A}}(\lambda)$ *is an experiment defined in Figure 5.*

We define key-selective single-key security *of* ABE *analogously, by replacing the phrase "all admissible adversaries* $\mathcal{A}$*" in the above definition with "all key-selectively admissible adversaries* $\mathcal{A}$*".*

Remark 6.4. In an ABE setting, the term "selective" usually means that $\mathcal{A}$ has to make a challenge query at the beginning of the security experiment. On the other hand, "key-selective" as defined above means that $\mathcal{A}$ has to make a key query at the beginning of the security experiment, and can make a challenge query any time.

6.2 Construction

Here, we construct a single-key secure SK-ABE scheme with optimal ciphertext overhead. Let $\mathsf{CPRF} = (\mathsf{CPRF.Setup}, \mathsf{CPRF.KeyGen}, \mathsf{CPRF.Eval}, \mathsf{CPRF.Constrain}, \mathsf{CPRF.CEval})$ be a CPRF for a function class $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$ with an input length n and output space is $\{0,1\}^{\ell_{\mathsf{k}}}$, and $\mathsf{SKE} = (\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be an SKE scheme with a key size ℓ_{k}, a message space $\mathcal{M}$ and a ciphertext $\mathcal{C}$. We construct an SK-ABE scheme $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ for a function class $\mathcal{F} = \{\mathcal{F}_{\lambda,k}\}_{\lambda,k\in\mathbb{N}}$ with an attribute space $\{0,1\}^n$, a message space $\mathcal{M}$ and a ciphertext $\mathcal{C}$ as follows.

$\mathsf{ABE.Setup}(1^\lambda)$: It computes $\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{CPRF.Setup}(1^\lambda)$ and $\mathsf{msk} \leftarrow \mathsf{CPRF.KeyGen}(\mathsf{pp})$, and outputs a public parameter pp and a master secret key msk.

$\mathsf{ABE.KeyGen}(\mathsf{msk}, f)$: It computes $\mathsf{sk}_f \leftarrow \mathsf{CPRF.Constrain}(\mathsf{msk}, f)$, and outputs sk_f.

$\mathsf{ABE.Enc}(\mathsf{msk}, x, \mathsf{msg})$: It computes $\mathsf{k} \leftarrow \mathsf{CPRF.Eval}(\mathsf{msk}, x)$ and $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg})$, and outputs ct.

$\mathsf{ABE.Dec}(\mathsf{sk}_f, \mathsf{ct})$: It computes $\mathsf{k} \leftarrow \mathsf{CPRF.CEval}(\mathsf{sk}_f, x)$, and outputs $\mathsf{SKE.Dec}(\mathsf{k}, \mathsf{ct})$.

The correctness of the scheme is easy to see from the correctness of CPRF and SKE.
The security of the above construction is stated as follows.

Theorem 6.5. *If* CPRF *is selectively (resp. adaptively) single-key secure and* SKE *is CPA secure, then* ABE *is key-selectively (resp. adaptively) single-key secure.*

Proof. We prove the theorem only for the selective case here because the proof can be easily extended to the adaptive case. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any adversary that attacks the key-selective single-key security of ABE. Let Q be the maximum number of $\mathcal{A}$'s query. We prove the theorem by considering the following sequence of games.

Game 0.1**:** This is the actual single-key security experiment $\mathsf{Expt}^{\mathsf{single\text{-}key}}_{\mathsf{ABE},\mathcal{A}}(\lambda)$ against the key-selective adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. Namely,

```
coin $\xleftarrow{R} \{0,1\}$
pp $\xleftarrow{R}$ CPRF.Setup$(1^\lambda)$
msk $\xleftarrow{R}$ CPRF.KeyGen(pp)
$(f, \mathsf{st}_\mathcal{A}) \xleftarrow{R} \mathcal{A}_1(\mathsf{pp})$
$\mathsf{sk}_f \xleftarrow{R} \mathsf{Constrain}(\mathsf{msk}, f)$
$\widehat{\mathsf{coin}} \xleftarrow{R} \mathcal{A}_2^{\mathcal{O}_{\mathsf{Enc}}(\cdot,(\cdot,\cdot))}(\mathsf{sk}_f, \mathsf{st}_\mathcal{A})$
Return $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$
```

where we describe $\mathcal{O}_{\mathsf{Enc}}(\cdot,(\cdot,\cdot))$ below.

$\mathcal{O}_{\mathsf{Enc}}(\cdot,(\cdot,\cdot))$**:** Given $(x, (\mathsf{msg}_0, \mathsf{msg}_1))$ as input, it computes $\mathsf{k} := \mathsf{CPRF.Eval}(\mathsf{msk}, x)$ and returns $\mathsf{ct} \xleftarrow{R} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_{\mathsf{coin}})$.

Game $i.0$**:** For $i \in [Q]$, Game $i.0$ is defined as follows. The difference from Game 0.1 is that a list L is maintained in a game, and $\mathcal{O}_{\mathsf{Enc}}$ works in a different way. Intuitively, $\mathcal{O}_{\mathsf{Enc}}$ encrypts msg_0 under an independently random key regardless of coin for the first $i-1$ distinct attributes, encrypts $\mathsf{msg}_{\mathsf{coin}}$ under a randomly generated key for the i-th distinct attribute, and encrypts $\mathsf{msg}_{\mathsf{coin}}$ under a key generated by CPRF as in Game 0.1 for the rest of attributes. Namely, a list L is initialized to be an empty set at the beginning of the game, and $\mathcal{O}_{\mathsf{Enc}}$ works as follows.

$\mathcal{O}_{\mathsf{Enc}}(\cdot,(\cdot,\cdot))$**:** Given $(x, (\mathsf{msg}_0, \mathsf{msg}_1))$ as input, if there exists k such that $(x, \mathsf{k}) \in L$ (i.e., x has already appeared in $\mathcal{A}_2$'s query), then it returns $\mathsf{ct} \xleftarrow{R} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_0)$. Otherwise it sets $N := |L| + 1$. (N is defined so that x is the N-th distinct attribute queried by $\mathcal{A}_2$.)

- If we have $N < i$, then it picks $\mathsf{k} \xleftarrow{R} \{0,1\}^{\ell_\mathsf{k}}$, updates $L \leftarrow L \cup \{(x, \mathsf{k})\}$ and returns $\mathsf{ct} \xleftarrow{R} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_0)$.
- If we have $N = i$, then it picks $\mathsf{k} \xleftarrow{R} \{0,1\}^{\ell_\mathsf{k}}$, updates $L \leftarrow L \cup \{(x, \mathsf{k})\}$ and returns $\mathsf{ct} \xleftarrow{R} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_{\mathsf{coin}})$.
- If we have $N > i$, then it computes $\mathsf{k} := \mathsf{CPRF.Eval}(\mathsf{msk}, x)$ and returns $\mathsf{ct} \xleftarrow{R} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_{\mathsf{coin}})$.

Game $i.1$**:** For $i \in [Q]$, Game $i.1$ is defined as follows. The difference from Game $i.0$ is that $\mathcal{O}_{\mathsf{Enc}}$ encrypts msg_0 instead of $\mathsf{msg}_{\mathsf{coin}}$ for the i-th distinct attribute. Namely, $\mathcal{O}_{\mathsf{Enc}}$ in this game works as follows.

$\mathcal{O}_{\mathsf{Enc}}(\cdot,(\cdot,\cdot))$**:** Given $(x, (\mathsf{msg}_0, \mathsf{msg}_1))$ as input, if there exists k such that $(x, \mathsf{k}) \in L$ (i.e., x has already appeared in $\mathcal{A}_2$'s query), then it returns $\mathsf{ct} \xleftarrow{R} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_0)$. Otherwise it sets $N := |L| + 1$. (N is defined so that x is the N-th distinct attribute queried by $\mathcal{A}_2$.)

- If we have $N \le i$, then it picks $\mathsf{k} \xleftarrow{R} \{0,1\}^{\ell_\mathsf{k}}$, updates $L \leftarrow L \cup \{(x, \mathsf{k})\}$ and returns $\mathsf{ct} \xleftarrow{R} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_0)$.
- If we have $N > i$, then it computes $\mathsf{k} := \mathsf{CPRF.Eval}(\mathsf{msk}, x)$ and returns $\mathsf{ct} \xleftarrow{R} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_{\mathsf{coin}})$.

Let $\mathsf{T}_{i.b}$ be the event that Game $i.b$ returns 1 for $i = 0, 1, ..., Q$ and $b = 0, 1$. We have $\mathsf{Adv}^{\mathsf{single\text{-}key}}_{\mathsf{ABE},\mathcal{A}}(\lambda) = 2 \cdot |\Pr[\mathsf{T}_{0.1}] - 1/2|$. We prove the following lemmas.

Lemma 6.6. *If* CPRF *is selectively single-key secure, then for all* $i \in [Q]$*, we have* $|\Pr[\mathsf{T}_{(i-1).1}] - \Pr[\mathsf{T}_{i.0}]| = \mathrm{negl}(\lambda)$.

Proof. We construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the selective single-key security of CPRF. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}_1(\mathsf{pp})$: Given a public parameter pp, it runs $(f, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}_1(\mathsf{pp})$ and outputs $(f, \mathsf{st}_{\mathcal{A}})$.

$\mathcal{B}_2^{\mathcal{O}_{\mathsf{Chal}}(\cdot),\mathsf{CPRF.Eval}(\mathsf{msk},\cdot)}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$: Given $(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$, it picks $\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$, runs $\widehat{\mathsf{coin}} \xleftarrow{\mathsf{R}} \mathcal{A}_2^{\mathcal{O}_{\mathsf{Enc}}(\cdot,(\cdot,\cdot))}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$ and outputs $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$. Here, $\mathcal{B}_2$ simulates $\mathcal{O}_{\mathsf{Enc}}$ as follows. First, $\mathcal{B}_2$ initialize a list L to be an empty set. For $\mathcal{A}_2$'s query $(x, (\mathsf{msg}_0, \mathsf{msg}_1))$, if there exists k such that $(x, \mathsf{k}) \in L$, then it returns $\mathsf{ct} \xleftarrow{\mathsf{R}} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_0)$. Otherwise it sets $N := |L| + 1$.

- If we have $N < i$, then it picks $\mathsf{k} \xleftarrow{\mathsf{R}} \{0,1\}^{\ell_{\mathsf{k}}}$, updates $L \leftarrow L \cup \{(x, \mathsf{k})\}$ and returns $\mathsf{ct} \xleftarrow{\mathsf{R}} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_0)$.
- If we have $N = i$, then it queries x to $\mathcal{O}_{\mathsf{Chal}}$ to obtain k, updates $L \leftarrow L \cup \{(x, \mathsf{k})\}$ and returns $\mathsf{ct} \xleftarrow{\mathsf{R}} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_{\mathsf{coin}})$.
- If we have $N > i$, then it queries x to $\mathsf{CPRF.Eval}(\mathsf{msk}, \cdot)$ to obtain k and returns $\mathsf{ct} \xleftarrow{\mathsf{R}} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_{\mathsf{coin}})$.

This completes the description of $\mathcal{B}$. First, we check that $\mathcal{B}$ is selectively admissible adversary against CPRF.

- We have $f \in \mathcal{F}_{\lambda,n}$ because this is required for a key-selectively admissible adversary against SK-ABE.
- $\mathcal{B}_2$ never make the same query twice because if $\mathcal{A}_2$'s query use an attribute x that has already appeared in a former query, then $\mathcal{B}_2$ refers a list L to obtain a key k, and does not query x twice to $\mathsf{CPRF.Eval}(\mathsf{msk}, \cdot)$.
- The query x^* to $\mathcal{O}_{\mathsf{Chal}}$ made by $\mathcal{B}_2$ satisfies $f(x^*) = 1$ because the key-selective admissibility against SK-ABE requires that $f(x) = 1$ holds for all x that appears in $\mathcal{A}$'s query.

Therefore $\mathcal{B}$ is selectively admissible. If the coin of the CPRF experiment in which $\mathcal{B}$ is involved is equal to 1, $\mathcal{O}_{\mathsf{Chal}}$ responds similarly to $\mathsf{CPRF.Eval}(\mathsf{msk}, \cdot)$, and thus $\mathcal{B}$ perfectly simulates Game $(i-1).1$ to $\mathcal{A}$. On the other hand, if the coin is equal to 0, then $\mathcal{O}_{\mathsf{Chal}}$ returns a uniformly random string, and thus $\mathcal{B}$ perfectly simulates Game $(i-1).1$ to $\mathcal{A}$. Therefore we have $|\Pr[\mathsf{T}_{(i-1).1}] - \Pr[\mathsf{T}_{i.0}]| = \mathsf{Adv}^{\mathsf{cprf}}_{\mathsf{CPRF},\mathcal{F},\mathcal{B}}(\lambda)$. Since we assume that CPRF is selectively single-key secure, this is negligible. ■

Lemma 6.7. *If* SKE *is CPA secure, then for all $i \in [Q]$, we have* $|\Pr[\mathsf{T}_{i.0}] - \Pr[\mathsf{T}_{i.1}]| = \mathrm{negl}(\lambda)$.

Proof. We construct an adversary $\mathcal{B}$ that breaks the CPA security of SKE. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}^{\mathcal{O}_{\mathsf{SKE.Enc}}(\cdot)}(1^\lambda)$: It picks $\mathsf{coin} \xleftarrow{\mathsf{R}} \{0,1\}$, computes $\mathsf{pp} \xleftarrow{\mathsf{R}} \mathsf{CPRF.Setup}(1^\lambda)$, $\mathsf{msk} \xleftarrow{\mathsf{R}} \mathsf{CPRF.KeyGen}(\mathsf{pp})$, $(f, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}_1(\mathsf{pp})$, $\mathsf{sk}_f \xleftarrow{\mathsf{R}} \mathsf{Constrain}(\mathsf{msk}, f)$, and $\widehat{\mathsf{coin}} \xleftarrow{\mathsf{R}} \mathcal{A}_2^{\mathcal{O}_{\mathsf{Enc}}(\cdot,(\cdot,\cdot))}(\mathsf{sk}_f, \mathsf{st}_{\mathcal{A}})$, and outputs $(\widehat{\mathsf{coin}} \overset{?}{=} \mathsf{coin})$ where it simulates $\mathcal{O}_{\mathsf{Enc}}$ as follows. First, it initializes a list L to be an empty set. For $\mathcal{A}_2$'s query $(x, (\mathsf{msg}_0, \mathsf{msg}_1))$, if $(x, \mathsf{Chal}) \in L$, then it queries $(\mathsf{msg}_0, \mathsf{msg}_{\mathsf{coin}})$ to $\mathcal{O}_{\mathsf{SKE.Enc}}$ to obtain ct and returns ct. Else if there exists k such that $(x, \mathsf{k}) \in L$, then it returns $\mathsf{ct} \xleftarrow{\mathsf{R}} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_0)$. Otherwise it sets $N := |L| + 1$.

- If we have $N < i$, then it picks $\mathsf{k} \xleftarrow{\mathsf{R}} \{0,1\}^{\ell_{\mathsf{k}}}$, updates $L \leftarrow L \cup \{(x, \mathsf{k})\}$ and returns $\mathsf{ct} \xleftarrow{\mathsf{R}} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_0)$.
- If we have $N = i$, then it queries $(\mathsf{msg}_0, \mathsf{msg}_{\mathsf{coin}})$ to $\mathcal{O}_{\mathsf{SKE.Enc}}$ to obtain ct, updates $L \leftarrow L \cup \{(x, \mathsf{Chal})\}$ and returns ct.
- If we have $N > i$, then it computes $\mathsf{k} := \mathsf{CPRF.Eval}(\mathsf{msk}, x)$ and returns $\mathsf{ct} \xleftarrow{\mathsf{R}} \mathsf{SKE.Enc}(\mathsf{k}, \mathsf{msg}_{\mathsf{coin}})$.

This completes the description of $\mathcal{B}$. If the coin of the SKE experiment in which $\mathcal{B}$ is involved is equal to 1, $\mathcal{O}_{\mathsf{SKE.Enc}}$ given $(\mathsf{msg}_0, \mathsf{msg}_{\mathsf{coin}})$ encrypts $\mathsf{msg}_{\mathsf{coin}}$ and thus $\mathcal{B}$ perfectly simulates Game $i.0$ to $\mathcal{A}$. On the other hand, if the coin is equal to 0, $\mathcal{O}_{\mathsf{SKE.Enc}}$ given $(\mathsf{msg}_0, \mathsf{msg}_{\mathsf{coin}})$ encrypts msg_0 and thus $\mathcal{B}$ perfectly simulates Game $i.1$ to $\mathcal{A}$. Therefore we have $|\Pr[\mathsf{T}_{i.0}] - \Pr[\mathsf{T}_{i.1}]| = \mathsf{Adv}^{\mathsf{cpa}}_{\mathsf{SKE},\mathcal{B}}(\lambda)$. Since we assume that SKE is CPA secure, this is negligible. ■

Lemma 6.8. *We have* $\Pr[\mathsf{T}_{Q,1}] = 1/2$.

Proof. In Game $Q.1$, coin is used only when $N > Q$ where N is as defined in the description of $\mathcal{O}_{\mathsf{Enc}}$ in Game $Q.1$. However, since $\mathcal{A}$ makes at most Q queries and N is incremented by at most 1 by each query, N cannot exceed Q. Therefore coin is not used at all in Game $Q.1$, and thus it is information theoretically impossible for $\mathcal{A}$ to guess coin with a non-zero advantage. ■

By combining the above lemmas, we have $\mathsf{Adv}^{\mathsf{single\text{-}key}}_{\mathsf{ABE},\mathcal{A}}(\lambda) = 2 \cdot |\Pr[\mathsf{T}_{0.1}] - 1/2| = \mathrm{negl}(\lambda)$ and the theorem is proven. ■

Discussion. Our SK-ABE scheme can be instantiated based on a CPRFs constructed in Section 4. If we instantiate the scheme based on a CPRF given in Section 4.2, then we obtain a key-selectively single-key secure SK-ABE scheme for $\mathbf{NC}^1$ based on the L-DDHI assumption on $\mathbb{QR}_q$ and the DDH assumption on a group $\mathbb{G}$ of an order q in the standard model [24]. If we instantiate the scheme based on a CPRF given in Section 4.3, then we obtain an adaptively single-key secure SK-ABE scheme for $\mathbf{NC}^1$ based on the L-DDHI assumption on any prime order cyclic group in the random oracle model. We note that if we instantiate the scheme based on the LWE-based CPRF given by Brakerski and Vaikuntanathan [BTVW17, PS18], we obtain a key-selectively secure single-key SK-ABE scheme for circuits based on the LWE assumption.

A ciphertext overhead (e.g., ciphertext length minus message length) of our scheme is optimal. Namely, a ciphertext of our scheme consists of only one ciphertext of an underlying CPA secure SKE scheme. Especially, we can make a ciphertext overhead any function $\ell(\lambda)$ as long as $\ell(\lambda) = \omega(\log \lambda)$.

To the best of our knowledge, the only known construction of an ABE scheme with such a compact ciphertext without obfuscation is the one given by Zhandry [Zha16] even in secret key and single key setting. [25] Though their scheme achieves much stronger functionality and security than ours (i.e., their scheme is public key ABE scheme for circuits, and achieves multi-key security), their scheme is based on a multilinear map, which is rather a strong primitive. Our construction illustrates that it is possible to construct an ABE scheme with an optimal ciphertext overhead based on a traditional group if we relax the functionality to be SK-ABE for $\mathbf{NC}^1$ and security to be the single-key security.

Acknowledgments

We thank Keita Xagawa for letting us know the relation between CIH and RKA-PRG.

References

[ABPP14] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. Related-key security for pseudorandom functions beyond the linear barrier. In Juan A. Garay and

[24]For instantiating our scheme, an output of an underlying CPRF should be a bit string whereas that is a group element in our actual construction given in Section 4.2. However, that can be converted to a bit string by applying an appropriate key derivation function.

[25]Actually, a ciphertext overhead of the scheme given in [Zha16] is $O(\lambda)$. We can make it $\ell(\lambda)$ for any $\ell(\lambda) = \omega(\log(\lambda))$ if we assume an existence of an exponentially secure one-way function.

Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 77–94. Springer, Heidelberg, August 2014. (Cited on page 16, 37.)

[AFP16] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Constrained PRFs for unbounded inputs. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 413–428. Springer, Heidelberg, February / March 2016. (Cited on page 2, 8.)

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004. (Cited on page 3.)

[BC10a] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. *IACR Cryptology ePrint Archive*, 2010:397, 2010. Version 20150729:233210. Preliminary version appeared in CRYPTO 2010. (Cited on page 5, 6, 16, 22.)

[BC10b] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, Heidelberg, August 2010. (Cited on page 37.)

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. (Cited on page 2.)

[BFP$^+$15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 31–60. Springer, Heidelberg, March 2015. (Cited on page 2, 4, 8.)

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

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014. (Cited on page 1, 4.)

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016. (Cited on page 2.)

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 567–594. Springer, Heidelberg, November 2017. (Cited on page 2, 4.)

[BKM17] Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable PRFs from standard lattice assumptions. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 415–445. Springer, Heidelberg, May 2017. (Cited on page 2, 4, 8.)

[BLW17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 494–524. Springer, Heidelberg, March 2017. (Cited on page 1, 2, 8, 12.)

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 264–302. Springer, Heidelberg, November 2017. (Cited on page 2, 4, 51.)

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, March 2015. (Cited on page 1, 2, 4, 11.)

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013. (Cited on page 1, 2, 4, 8, 10, 11.)

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 446–476. Springer, Heidelberg, May 2017. (Cited on page 2, 4, 8.)

[CGV15] Aloni Cohen, Shafi Goldwasser, and Vinod Vaikuntanathan. Aggregate pseudorandom functions and connections to learning. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 61–89. Springer, Heidelberg, March 2015. (Cited on page 8.)

[CH85] Stephen A. Cook and H. James Hoover. A depth-universal circuit. *SIAM J. Comput.*, 14(4):833–839, 1985. (Cited on page 4, 24.)

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Heidelberg, May 2005. (Cited on page 3.)

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017. (Cited on page 2.)

[DKW16] Apoorvaa Deshpande, Venkata Koppula, and Brent Waters. Constrained pseudorandom functions for unconstrained inputs. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 124–153. Springer, Heidelberg, May 2016. (Cited on page 2, 8, 11.)

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, December 2014. (Cited on page 11.)

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013. (Cited on page 2.)

[GGH$^+$16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016. (Cited on page 2.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986. (Cited on page 1, 4, 20.)

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 537–566. Springer, Heidelberg, November 2017. (Cited on page 2, 4.)

[GL10] David Goldenberg and Moses Liskov. On related-secret pseudorandomness. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 255–272. Springer, Heidelberg, February 2010. (Cited on page 5.)

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. *IACR Cryptology ePrint Archive*, 2011:233, 2011. Version 20110517:062434. Preliminary version appeared in TCC 2011. (Cited on page 5, 6, 15, 16.)

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012. (Cited on page 4.)

[HKKW14] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. Cryptology ePrint Archive, Report 2014/720, 2014. `http://eprint.iacr.org/2014/720`. (Cited on page 2, 8.)

[HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 79–102. Springer, Heidelberg, November / December 2015. (Cited on page 2, 8.)

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003. (Cited on page 5.)

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, November 2013. (Cited on page 1, 4.)

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51(2):231–262, 2004. (Cited on page 17, 20.)

[PS18] Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In *PKC 2018 (to appear)*, 2018. IACR Cryptology ePrint Archive 2017/1094. (Cited on page 2, 4, 51.)

[Yam17] Shota Yamada. Asymptotically compact adaptively secure lattice IBEs and verifiable random functions via generalized partitioning techniques. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 161–193. Springer, Heidelberg, August 2017. (Cited on page 5.)

[Zha16] Mark Zhandry. How to avoid obfuscation using witness PRFs. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 421–448. Springer, Heidelberg, January 2016. (Cited on page 3, 51.)