A Bit-fixing PRF with O(1) Collusion-Resistance from LWE

Alex Davidson[1,⋆] and Ryo Nishimaki[2]

[1] ISG, Royal Holloway University of London, UK
alex.davidson.2014@rhul.ac.uk
[2] NTT Secure Platform Laboratories, Tokyo, Japan
nishimaki.ryo@lab.ntt.co.jp

Abstract. Constrained pseudorandom functions (CPRFs) allow learning modified PRF keys that can evaluate the PRF on a subset of the input space, or based on some sort of predicate. First introduced by Boneh and Waters [Asiacrypt 2013], they have been shown to be a useful cryptographic primitive with many applications. The full security definition of CPRFs requires the adversary to learn multiple constrained keys, a requirement for all of these applications. Unfortunately, existing constructions of CPRFs satisfying this security notion are only known from exceptionally strong cryptographic assumptions, such as indistinguishability obfuscation and the existence of multilinear maps, even for very weak predicates. CPRFs from more standard assumptions only satisfy security when one key is learnt.
In this work, we give the first construction of a CPRF that can issue a constant number of constrained keys for bit-fixing predicates, from learning with errors (LWE). Our construction immediately leads to the first constructions of broadcast encryption with optimal ciphertext size and non-interactive ID-based key exchange from LWE, for constant-sized user groups. It also satisfies 1-key privacy (otherwise known as constraint-hiding). Finally, our construction achieves fully adaptive security with polynomial security loss; the only construction to achieve such security under a standard assumption.
Our technique represents a noted departure existing for CPRF constructions. We hope that it may lead to future constructions that can expose a greater number of keys, or consider more expressive predicates (such as circuit-based constraints).

1 Introduction

Historically, pseudorandom functions (PRFs) provide the basis of a huge swathe of cryptography. Intuitively, such a function takes a uniform key and some binary string x as input, and outputs (deterministically) some value y. The pseudorandomness of the function dictates that y is indistinguishable from some output obtained from a uniformly sampled function operating solely on x. Importantly, PRFs can provide useful sources of randomness in constructions that take adversarially-chosen inputs.

Simple constructions of PRFs exist based on well-known standard assumptions: Naor and Reingold [NR97] give a simple and elegant construction from number-theoretic assumptions related to the discrete log assumption; Goldreich, Goldwasser and Micali give a construction based on the existence of pseudorandom generators [GGM84].

There have been numerous expansions of the definitional framework surrounding PRFs. In this work we focus on a strand of PRFs that are known as *constrained* PRFs or CPRFs. CPRFs were first introduced by Boneh and Waters [BW13] (Kiayias, Papadopoulos, Triandopoulos, and Zacharias [KPTZ13] and Boyle, Goldwasser, and Ivan [BGI14] also proposed the same notion in their concurrent and independent works) and depart from standard PRFs in that the adversary receives more power to evaluate the function. In particular, the adversary can

⋆ This work was completed while the author undertook a research internship at NTT. The author was also supported by the EPSRC and the UK Government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/K035584/1).

receive *constrained keys* that allow evaluating the function on a subset of the input space. Security now dictates that the CPRF remains pseudorandom on the points that lie outside of the subsets of the constrained keys that are learnt by the adversary.

Predicates. While constrained keys can be defined with respect to subsets, a more natural definition defines functionality with respect to predicates. That is, the constrained key allows evaluation of the function on x, if and only if the associated predicate is equal to 1 on x. We denote such a predicate by P and thus $P(x) = 1$ indicates that the input satisfies the constraint.

Using this formulation, Boneh and Waters developed three independent constructions of constrained PRFs that allow predicates of different expressibility. The most structured predicate is the *left-right* (LR) predicate that requires the inputs be taken from $\{0,1\}^{2\ell}$. Constrained keys in this setting can be learnt for strings of length $\{0,1\}^{\ell}$ corresponding to the first half (left) or the second half (right) of the input. That is, a *left* constrained key for some string $v \in \{0,1\}^{\ell}$ can evaluate $x \in \{0,1\}^{2\ell}$ if $v_i = x_i$, for $i \in \{1,\ldots,\ell\}$. A right key is defined in the same way except that it evaluates those inputs x satisfying $v_i = x_i$, for $i \in \{\ell+1,\ldots,2\ell\}$ instead.

A more expressive predicate is the bit-fixing (BF) predicate. Such a predicate takes some string $v \in \{0,1,*\}^{\ell}$ as input; where $v_i = *$ indicates a *wildcard* entry. Denote the predicate by $\mathsf{P}_v(x)$ for some valid input $x \in \{0,1\}^{\ell}$. Then we say that $\mathsf{P}_v(x) = 1$ iff $(x_i = v_i) \vee (v_i = *)$ for all $i \in [\ell]$ — that is, the string x is equal to v on all positions that are not wildcards. Notice that bit-fixing predicates completely subsume left-right predicates by treating the right/left side as a sequence of ℓ wildcards.

Finally, the last predicate considered by [BW13] is that of circuits. Where $\mathsf{P}_C(x) = 1$ iff $C(x) = 1$ for some circuit C, and CK_C can evaluate the CPRF on x if this predicate is satisfied in this way. Achieving constrained pseudorandom functions for circuits in P/poly represents the most expressive predicate that we could hope to achieve for a CPRF.

***m*-key privacy.** An additional security requirement that was introduced by Boneh et al. [BLW17] is that the constrained keys do not reveal the constraint that is encoded in them. In other words, given a constrained key for one of two adversarially-chosen constraints, the same adversary is unable to distinguish which constraint is encoded with anything other than negligible advantage. The definition is made stronger by requiring that the adversary is given m keys for $m \geq 1$. A CPRF satisfying this definition of security is known as a private CPRF or PCPRF.[3]

It was shown by Canetti and Chen [CC17] that a CPRF satisfying privacy for more than one key implies the existence of IO. In [CC17], the definition is also given in a simulation-based setting, rather than the indistinguishability-based framework of [BLW17]. It is shown in the former that security in the simulation model implies security in the indistinguishability model, but the reverse does not hold.

1.1 Existing constructions

Since the original work of [BW13], numerous constructions of CPRFs have been given, relying on different primitives and providing a range of functionality. The work of [BW13] gave constructions of CPRFs for LR, BF and NC^1 circuit predicates. The LR predicate CPRF was derived from the bilinear decisional diffie-hellman (BDDH) assumption and the existence of random oracles. The other constructions were derived from multilinear maps that satisfy the multilinear DDH (MDDH) assumption. These original constructions satisfy collusion-resistance for any polynomial number of constrained keys. That is, the adversary in the CPRF security game can learn polynomially-many constrained keys.

[3] They are also known as ‘constraint-hiding’ CPRFs.

Hofheinz et al. [HKKW14] develop a construction with stronger security guarantees in that all queries can be answered adaptively, this is the only construction that satisfies adaptive security via a polynomial-time reduction. The adaptive security model allows the adversary to specify all CPRF queries (input, constrained key, challenge) at any point through the security experiment. Unfortunately their construction is based on very strong assumptions — namely a combination of indistinguishability obfuscation (IO) and random oracles. All other works require sub-exponential time reductions to achieve adaptive security.

More recent constructions have constructed CPRFs from much weaker assumptions, at the expense of providing weaker guarantees. The works of [BV15, CC17, PS18, BTVW17, CVW18] construct CPRFs from learning with errors (LWE), and other lattice-based assumptions. Unfortunately, none of these constructions satisfy collusion-resistance or adaptive security (via polynomial-time reductions). However, each of these constructions can create a constrained key for circuits taken from the class NC^1. The works of [BTVW17, PS18, CVW18, BV15] actually provide constrained keys for P/poly. The work of [AMN$^+$18] provides a construction of CPRFs, from assumptions in traditional groups, for circuit predicates in NC^1 — again security only holds for one constrained key query.

Achieving private constraints. The constructions of [BLW17] provides poly-many privately constrained keys for circuit predicates, under the existence of IO. The PCPRFs of [CC17, BTVW17, PS18, CVW18] also satisfy the privacy guarantee for circuit predicates, but for only one constrained key query. This is unsurprising given that there are no known instantiations of IO from standard cryptographic assumptions. Such a result would imply the existence of IO from LWE.

Applications. In the original work of [BW13], a number of applications were given that highlighted the utility of CPRFs. They give a cryptographic primitive that can be instantiated by CPRFs for left-right, BF and NC^1 predicates, respectively:

- LR CPRF $\implies$ identity-based non-interactive key exchange (ID-NIKE);
- BF CPRF $\implies$ broadcast encryption with optimal ciphertext size;[4]
- $(\mathrm{NC}^1 \vee \mathsf{P}/\mathrm{poly})$ CPRF $\implies$ policy-based key distribution.

Since this initial study, there have been no alternative constructions of these primitives from methods that do not utilise CPRFs. A key property of each of the applications is that they require a CPRF that remains secure even when multiple constrained keys have been learnt. The initial construction of [BW13] satisfies this property but constructions relying on standard assumptions (i.e. not obfuscation-based) cannot instantiate these applications meaningfully, since they only permit one constrained key to be learnt.

1.2 Our contribution

In this work we develop a new CPRF construction for the bit-fixing predicate from LWE that satisfies collusion-resistance for $r = O(1)$ constrained keys. Furthermore, our security proof holds in the adaptive security setting with only a polynomial loss of security. Our construction is the first to satisfy either of these requirements from any standard assumptions. Finally, our construction satisfies 1-key privacy by the definition of [BLW17]. We summarise our contribution alongside the existing state of the art in Table 1.

By the work of [BW13], we immediately achieve the first constructions of ID-based non-interactive key exchange (ID-NIKE) and broadcast encryption with optimal ciphertext size from LWE. These are the only known constructions of such schemes from a standard assumption. For instance, ID-NIKE was only previously known for 3 users from bilinear maps. For $N > 3$, the only constructions were from multilinear maps or IO.

[4] For the broadcast encryption scheme, by optimal we mean that the length of the *header*, a traditional metric by which the efficiency of broadcast encryption schemes are measured, is 0.

Table 1. List of existing constructions of CPRFs along with their functionality and the assumptions required. The adaptive column only refers to works that achieve adaptive security in polynomial-time. We say that privacy is satisfied even if it only holds for one constrained key.

	Collusion-resistance	Privacy	Adaptive	Predicate	Assumption
[BW13]	$\mathsf{poly}(\lambda)$	0	◊	LR	BDDH & ROM
	$\mathsf{poly}(\lambda)$	0	◊	BF	MDDH
	$\mathsf{poly}(\lambda)$	0	◊	NC^1	MDDH
[HKKW14]	$\mathsf{poly}(\lambda)$	0	◆	$\mathsf{P/poly}$	IO & ROM
[BV15]	1	0	◊	$\mathsf{P/poly}$	LWE
[BLW17]	$\mathsf{poly}(\lambda)$	1	◊	Puncturing	MDDH
	$\mathsf{poly}(\lambda)$	1	◊	BF	MDDH
	$\mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda)$	◊	$\mathsf{P/poly}$	IO
[CC17]	1	1	◊	BF	LWE
	1	1	◊	NC^1	LWE
[BTVW17]	1	1	◊	$\mathsf{P/poly}$	LWE
[PS18]	1	1	◊	$\mathsf{P/poly}$	LWE
[CVW18]	1	1	◊	$\mathsf{P/poly}$	LWE
[AMN$^+$18]	1	0	◊	NC^1	L-DDHI
	1	1	◊	BF	DDH
	1	1	◊	NC^1	ROM
This work	$O(1)$	1	◆	BF	LWE

It should be noted that since our construction only allows a constant number of constrained keys, this limitation also arises in these applications. That is, the broadcast encryption scheme only permits the adversary to learn the secret keys of constant-sized user groups; and similarly for the ID-NIKE scheme. Therefore, we give the first fully secure constructions of both of these primitives for groups of users of size $r = O(1)$.

Roadmap In Section 2 we give a technical overview of our contribution. In Section 3 we cover preliminary definitions. In Section 4 we give our construction.

2 Technical overview

2.1 Lattice-based constructions

The idea for this work originates from the lattice-based CPRF for bit-fixing constraints of Canetti and Chen [CC17], though the techniques are very similar in other lattice-based constructions [BV15, PS18]. In these works the adversary is allowed to query for one constrained key that is chosen selectively (rather than adaptively). The PRF is defined over an input $x \in \{0,1\}^\ell$ and the master secret key is a set of Gaussian-distributed matrices $\{\boldsymbol{D}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$. These matrices are thought of as representatives of LWE secrets, the underlying technique is borrowed from the PRFs of [BPR12, BLMR13]. The constrained key is then some $v \in \{0,1,*\}^\ell$, where $\boldsymbol{D}_{i,v_i}$ is revealed if $v_i \in \{0,1\}$, and both $\{\boldsymbol{D}_i\}_{b\in b\in\{0,1\}}$ are revealed if $v_i = *$, for each $i \in [\ell]$. Finally, newly sampled $\overline{\boldsymbol{D}_{i,1-v_i}} \leftarrow_{\$} D_{\mathbb{Z}^{m\times m},\sigma}$ replace the matrices that are not learnt.[5] In the public parameters, there is a matrix $\boldsymbol{A}$, and for an input x, the PRF evaluation is $\boldsymbol{A} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,x_i}$.

The key observation of [CC17] is that pseudorandomness only has to hold for some challenge $x^\dagger$ where $(x_j \neq v_j) \wedge (v_j \neq *)$. Then when the PRF is evaluated at $x^\dagger$, the output includes the matrix $\boldsymbol{D}_{j,x_j}$ that is not revealed in the constrained key (and thus to the adversary). As a result, their security proof relies on an LWE security reduction, where $\boldsymbol{D}_{j,x_j}$ ultimately acts as an unknown LWE secret. It is also noted by [CC17] that a very similar

[5] This is not required for standard CPRF security, but only for the extra privacy property.

argument can be used to show that the [BLMR13] PRF is also a PCPRF for bit-fixing constraints. For circuit-based constraints, this proof technique does not apply since the matrices are no longer tied explicitly to one bit of the constraint query. In these cases, a more careful LWE argument is used in relation to the secret distribution that is considered.

Unfortunately, the analysis for bit-fixing does not follow for more than one key. It is entirely possible to choose two constrained keys that would reveal the entire set $\{\boldsymbol{D}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$, without compromising all evaluation points.[6] Therefore, the LWE argument cannot be used since all the secrets are effectively public. The main issue of this technique is that one bit of the PRF input is tied concretely to one bit of the master secret key. Consequently, when valid constraints reveal both components of the master secret key for each bit, security is effectively lost.

2.2 Our scheme

To improve on the functionality of previous schemes, we design a CPRF construction that analyses r input bits at a time, for $r \geq 1$. It may not be clear how just yet, but this allows us to provably hide matrices for up to r constrained key queries. Unfortunately, we require that $r = O(1)$ as the size of the public parameters and the master secret key depends exponentially on the size of r.

Key generation. To be more precise, let $\rho = \sum_{\iota=1}^{r} \binom{\ell}{\iota}$; then our CPRF consists of public parameters of the form:

$$\{\boldsymbol{A}_i\}_{i\in[\ell]} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}, \{\boldsymbol{D}_{j,b}^{(i)}\}_{j\in[\ell],b\in\{0,1\}} \leftarrow_{\$} D_{\mathbb{Z}^{m\times m},\sigma}, \boldsymbol{D}^{\mathsf{end}} \leftarrow_{\$} D_{\mathbb{Z}^{\rho m\times m},\sigma};$$

for the Gaussian distribution $D_{\mathbb{Z},\sigma}$.

The master secret key consists of $\sum_{i=1}^{r} 2^i \cdot \binom{\ell}{i}$ different matrices. In other words, for each ordered set $S \subset [\ell]$ where $|S| \in [r]$ there are $2^{|S|}$ possible matrices, corresponding to the possible bits of $x|_S$ (where $x_i \in x|_S$ if and only if $i \in S$). Let the vectors $\boldsymbol{t} \in [\ell]^z$ correspond to all possible ordered sets S, for $z \in [r]$. The master secret key is then defined to be the set:

$$\{\boldsymbol{D}_{\boldsymbol{t},\boldsymbol{b}}^{G}\}_{\boldsymbol{t}\in[\ell]^z,\boldsymbol{b}\in\{0,1\}^z,z\in[r]} \leftarrow_{\$} D_{\mathbb{Z}_q^{zm\times m},\sigma}.$$

Evaluation. For each input to the CPRF, $x \in \{0,1\}^\ell$, we iterate through each of the possible vectors $\boldsymbol{t} \in [\ell]^z$. We write $x_{\boldsymbol{t}} \leftarrow \mathsf{reindex}(x, \boldsymbol{t})$ to denote the bits x_{t_i} for $t_i = \boldsymbol{t}[i]$, abusing notation and letting $\boldsymbol{t}$ denote a set. For each $\boldsymbol{t}$, there are matrices $\boldsymbol{D}_{\boldsymbol{t},\boldsymbol{b}}^{G}$ for all $\boldsymbol{b} \in \{0,1\}^z$. Then, to evaluate the PRF output, we first choose the set of matrices $\{\boldsymbol{D}_{\boldsymbol{t},x_{\boldsymbol{t}}}^{G}\}_{\boldsymbol{t}\in[\ell]^z,z\in[r]}$.

Let

$$\boldsymbol{Y}_i^x = \boldsymbol{A}_i \cdot \prod_{j=1}^{\ell} D_{j,x_j}^{(i)}.$$

Then, for each $\boldsymbol{t}$, we compute $\boldsymbol{Y}_{\boldsymbol{t}}^x = [\boldsymbol{Y}_{t_1} \| \dots \| \boldsymbol{Y}_{t_z}]$, where t_i is the ith component of $\boldsymbol{t}$, and:

$$\boldsymbol{Z}_{\boldsymbol{t}}^x = \boldsymbol{Y}_{\boldsymbol{t}}^x \boldsymbol{D}_{\boldsymbol{t},x_{\boldsymbol{t}}}^{G}.$$

We set $\boldsymbol{Z}_{\mathbb{T}}^x = [\{\boldsymbol{Z}_{\boldsymbol{t}}^x\}]_{\boldsymbol{t}\in[\ell]^z,z\in[r]}$, where $[\{\boldsymbol{Z}_{\boldsymbol{t}}^x\}]_{\boldsymbol{t}\in[\ell]^z,z\in[r]}$ is the matrix that concatenates each $\boldsymbol{Z}_{\boldsymbol{t}}^x$ according to the lexicographic ordering (from least to most) inferred by the (1) dimension of $\boldsymbol{t}$; (2) the value of $\sum_{l=1}^{z} t_l$.

Finally, the PRF output is computed as:

$$\lfloor \boldsymbol{Z}_{\mathbb{T}}^x \boldsymbol{D}^{\mathsf{end}} \rceil_p$$

[6] For example, choosing the constraints $v_1 = 1 * * * 1$ and $v_2 = 0 * * * 0$; where $x = 1 * * * 0$ is still a constrained point.

for some $p > 0$, using a similar choice to previous lattice-based PRFs [BPR12, BLMR13, BP14, CC17]. In contrast, our scheme makes a noted departure from previous designs in the sense that we use a concatenated matrix $\boldsymbol{Y}^x$, and then perform multiplications with each of the chosen $\boldsymbol{D}^G$ matrices. In the previous schemes, a uniform matrix $\boldsymbol{A}$ is used, and then the output is computed sequentially, by computing $\boldsymbol{A}\boldsymbol{D}_{x_1} \dots \boldsymbol{D}_{x_\ell}$.

Constraint queries. Let $\in \{0, 1, *\}^\ell$ be some bit-fixing constraint query, where $*$ is the designated wildcard character. We answer a constraint query by computing $v_{\boldsymbol{t}} \leftarrow \mathsf{reindex}(v, \boldsymbol{t})$ for each $\boldsymbol{t} \in [\ell]^z$ and then returning all matrices $\boldsymbol{D}^G_{\boldsymbol{t},\boldsymbol{b}}$ such that $1 \leftarrow \mathsf{P}_v(\boldsymbol{b})$ (i.e. $(v_{\boldsymbol{t},i} = b_i) \vee (v_{\boldsymbol{t},i} = *)$). As an example, if $v = 01 * * * 1$ and $\boldsymbol{t} = (1, 3)$, then we would return the matrices $\boldsymbol{D}^G_{\boldsymbol{t},(0,0)}$ and $\boldsymbol{D}^G_{\boldsymbol{t},(0,1)}$.

To satisfy 1-key privacy, we also sample a dummy master secret key consisting of matrices $\overline{\boldsymbol{D}^G_{\boldsymbol{t},\boldsymbol{b}}}$. For $\boldsymbol{b}$ such that $0 \leftarrow \mathsf{P}_v(\boldsymbol{b})$, we return $\overline{\boldsymbol{D}^G_{\boldsymbol{t},\boldsymbol{b}}}$. Therefore, in the example above, the matrices $\overline{\boldsymbol{D}^G_{\boldsymbol{t},(1,0)}}$ and $\overline{\boldsymbol{D}^G_{\boldsymbol{t},(1,1)}}$ would also be returned. The matrices are identically distributed and so they hide the constraint if only one key is learnt. If more keys are learnt then it would be simple to match up those matrices corresponding to the real master secret key, and vice-versa.

Proof overview. The proof is similar in spirit to the [CC17] proof argument. Consider any set $(v^{(1)}, \dots, v^{(r)})$ of bit-fixing constraints, i.e. $v^{(i)} \in \{0, 1, *\}^\ell$ where $*$ is a wildcard character. Then, each $v^{(i)}$ must have at least one point t_i where $t_i \neq *$. Let us assume that $v^{(i)}$ are ordered, without loss of generality, so that $t_i \leq t_{i+1}$.

Then consider the vector $\boldsymbol{t}$ that contains all the unique indices $t_i \in \ell$, where $t_i > t_{i-1}$ or $i = 1$. Then $\boldsymbol{t} \in [\ell]^z$, where $z \leq r$ is the number of such unique indices. Now, consider the vector $\boldsymbol{b} = (1 - v^{(1)}_{t_1}, \dots, 1 - v^{(r)}_{t_r})$ corresponding to the inverse bits in the constraint. Then we know that $\boldsymbol{D}^G_{\boldsymbol{t},\boldsymbol{b}}$ is never revealed to the adversary, since all of the constraint queries satisfy $0 \leftarrow \mathsf{P}_{v^{(i)}}(\boldsymbol{b})$.

For security to hold, we need to show that the CPRF output remains pseudorandom on some point $x^\dagger$ that remains constrained, relative to the constraint queries that the adversary makes. As a consequence, there must be some vector $\boldsymbol{t}^\dagger$ of the form above, since there must be an index in $t_i \in [\ell]$ for each $v^{(i)}$ such that $(x^\dagger_i \neq v^{(i)}_{t_i}) \wedge (v^{(i)}_{t_i} \neq *)$. Therefore, we know that the matrix $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$ is never revealed by the adversary.

In essence, the output of the CPRF on $x^\dagger$ can now be written as:

$$\left\lfloor \boldsymbol{Z}^{x^\dagger}_{\mathbb{T}} \boldsymbol{D}^{\mathsf{end}} \right\rceil_p = \left\lfloor \boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger} \boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} + \sum_{\boldsymbol{t} \neq \boldsymbol{t}^\dagger} \boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}} \boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}} \right\rceil_p$$

where $\boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} \in \mathbb{Z}^{m \times m}_q$ is the vertical component of $\boldsymbol{D}^{\mathsf{end}}$ corresponding to the ordering of $\boldsymbol{t}^\dagger$ in $[\ell]^r$. Moreover, $\boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger} = \boldsymbol{Y}^{x^\dagger}_{\boldsymbol{t}^\dagger} \boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$, where $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$ is never learnt by the adversary.

Firstly, we are able to replace $\boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger}$ with the RHS of a Leftover Hash Lemma (Lemma 3.3) sample $(\boldsymbol{A}, \boldsymbol{B})$, where $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{R}$ or $\boldsymbol{B} \leftarrow_\$ \mathbb{Z}^{n \times zm}_q$. We set $\boldsymbol{Y}^{x^\dagger}_{\boldsymbol{t}^\dagger} = \boldsymbol{A}$, and implicitly set $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}} = \boldsymbol{R}$ and the lemma follows as long as $m = \Omega(n \log q)$. Secondly, we can introduce an extra error matrix that is summed with $\boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger} \boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger}$, since the rounding $\lfloor \cdot \rceil_p$ ensures that the output will be the same with high probability. Finally, we can argue that $\boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger} \boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} + \boldsymbol{E}$ is indistinguishable from uniform, using the non-uniform LWE argument of [BLMR13], where $\boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger}$ is a public low-norm matrix and $\boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger}$ is the uniform secret. The output is now a sum of a uniform matrix (that is completely dependent on $x^\dagger$) with other matrices corresponding to $\boldsymbol{t} \neq \boldsymbol{t}^\dagger$, and so the output is uniform.

For further details of the proof see Theorem 4.3.

Proof subtleties. Of course, the argument that we have given above is heavily simplified. An astute reader may have noticed that it is impossible to simulate input queries that make use of the matrix $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$, but that are also not the challenge point. To get around this, we assume that the number of input queries $Q = \mathsf{poly}(\lambda)$ is known by the proof reduction beforehand. We then introduce a trapdoor approach that allows us to sample independent uniform matrices $\boldsymbol{Y}^x_i$ for each $i \in [\ell]$. Using the trapdoors we essentially set

$$\boldsymbol{Y}^x_i = \boldsymbol{A}_i \prod_{j=1}^{\ell} \boldsymbol{D}^{(i)}_{j,x_j},$$

and this allows us to render the input queries and the challenge query completely independent evaluations. See the proof of Lemma 4.4 for more details.

Adaptive security. Essentially, our construction arrives at adaptive security *for free*. Previous constructions incur sub-exponential security losses during the reduction from selective to adaptive security, by essentially attempting to guess the challenge point $x^\dagger$ that a selective adversary would use. We are able to achieve adaptive security with a polynomial security loss (e.g. $1/\mathsf{poly}(\lambda)$): by simply guessing the matrix $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$ that is implicitly used by the adversary. If this matrix is not eventually used by the challenge ciphertext, or it is revealed via a constrained key query, then the reduction aborts. This is because the entire proof hinges on the choice of this matrix, rather than the input itself. Since there are polynomially many matrices (for $r = O(1)$), we can achieve adaptive security with only a $1/\mathsf{poly}(\lambda)$ probability of aborting. See Lemma 4.5 for more details.

3 Preliminaries

We provide the notational and definitional framework for the construction that we give.

3.1 Notation

For some space D, we write $x \leftarrow\!\!\$\, D$ to indicate that x has been sampled from D using the uniform distribution. For $n \in \mathbb{N}$, we write $[n]$ to represent the set $\{1, \ldots, n\}$. We write $[n_1, n_2]$ to denote the set $\{n_1, \ldots, n_2\}$ for $n_1 < n_2$ and $n_1, n_2 \in \mathbb{N}$. For a string $x = x_1 \ldots x_\ell \in \{0,1\}^\ell$, we let $x|_t = x_t \ldots x_\ell$, $x|^t = x_1 \ldots x_t$, and $x|^{t_2}_{t_1} = x_{t_1} \ldots x_{t_2}$, for $t_2 \geq t_1$. Alternatively, let $T \subset [\ell]$ be some subset of indices, we write $x|_T$ to denote the bits $x_i \in \{0,1\}$ such that $i \in T$.

Bit-fixing predicates. We write 'bit-fixing' constraints as $v \in \{0,1,*\}^\ell$, denoting a bit-fixing predicate for v by $\mathsf{P}_v : \mathcal{X} \mapsto \{0,1\}$. The predicate satisfies $\mathsf{P}_v(x) = 1$ for $x \in \{0,1\}^\ell$ iff $((x_i = v_i) \vee (v_i = *))$ for each $i \in \ell$.

Vectors and matrices. Vectors are written as lower-cased bold-face (i.e. $\boldsymbol{v}$) and are assumed to be in horizontal notation by default. We write $\boldsymbol{v}^T$ to denote the vector in column-form. Denote by v_i, the i^{th} entry of $\boldsymbol{v}$.

We denote matrices by capitalized bold-face (i.e. $\boldsymbol{A}$). We write $[\boldsymbol{A}_1 \| \boldsymbol{A}_2]$ to denote the horizontal concatenation of matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. Let $[\boldsymbol{A}_1 \|_\uparrow \boldsymbol{A}_2]$ denote the vertical concatenation of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. We define an 'empty' matrix by setting $\boldsymbol{A}_1 = []$, in this case we define $[\boldsymbol{A}_1 \| \boldsymbol{A}_2] = \boldsymbol{A}_2$.

Let $\mathbb{T} = \{(t_1, \ldots, t_r) \in [\ell]^r \| t_1 \leq t_2 \leq \ldots \leq t_r\}$. We write $[\boldsymbol{B}_\mathbb{T}]$ to denote the matrix

$$[\boldsymbol{B}_1 \| \boldsymbol{B}_2 \| \ldots \| \boldsymbol{B}_\ell \| \boldsymbol{B}_{1,\ell} \| \ldots \| \boldsymbol{B}_{\ell-1,\ell} \| \ldots \| \boldsymbol{B}_{1,2,\ldots,\ell-1,\ell}]$$

that concatenates the matrices with respect to all unique entries of $\boldsymbol{t} = (t_1, \ldots, t_r) \leftarrow \mathbb{T}$. The lexicographic ordering of $\boldsymbol{t}$ in $\mathbb{T}$ is from lowest to highest after removing non-unique entries, using the criteria: (1) the dimension of $\boldsymbol{t} \in [\ell]^z$; (2) the value of $\sum_{i=1}^{z} t_i$.

Let $\boldsymbol{v}$ be an ordered vector $(v_1 \leq \ldots \leq v_\ell)$. Then we write $z \leftarrow \mathsf{unique}(\boldsymbol{v})$ to denote the number of positions $i \in [\ell]$ s.t. $v_{i-1} < v_i$. Let $\mathbb{B}_\ell = \{(b_1, \ldots, b_\ell) \in \{0,1\}^\ell\}$, and let $\boldsymbol{b} \leftarrow \mathbb{B}_\ell$. We write $\boldsymbol{b}_{\boldsymbol{v}} \leftarrow \mathsf{reindex}(\boldsymbol{b}, \boldsymbol{v})$ to denote the reindexing $\boldsymbol{b}_{\boldsymbol{v}}$ of $\boldsymbol{b}$ with respect to unique entries v_i of $\boldsymbol{v}$ (i.e. where $v_{i-1} < v_i$). That is, if the unique entries of $\boldsymbol{v}$ are v_2, v_5, v_7, v_9, then $\boldsymbol{b}_{\boldsymbol{t}} = (b_{v_2}, b_{v_5}, b_{v_7}, b_{v_9})$.

Rounding. For some $c \in \mathbb{Q}$, we write $\lfloor c \rceil = w$ where $|w - c| \leq |w' - c|$ for all $w' \in \mathbb{Z}$; rounding towards 0 in case of a tie. For $c \in \mathbb{Z}_q$, we denote the act of rounding into $\mathbb{Z}_p$ (for some $p \in \mathbb{Z}$) by $\lfloor c \rceil_p = \lfloor (p/q)c \rceil$.

Security experiments. We use the abbreviation PPT to refer to probabilistic polynomial time. Let λ be a security parameter and let $\mathsf{exp}_{b,\mathcal{A}}(1^\lambda)$, for $b \in \{0,1\}$, be a pair of *decisional* experiments along with a PPT algorithm $\mathcal{A}$ (known as the *adversary*) that attempts to distinguish the cases where $b = 0$ and $b = 1$. We define decisional experiments such that $b_\mathcal{A} \leftarrow \mathsf{exp}_{b,\mathcal{A}}(1^\lambda)$ is the final output, where $b_\mathcal{A}$ is the 'guess' of $\mathcal{A}$. This guess is made in the final step of the game. Let

$$\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{A}}(1^\lambda)) = \left|\Pr\left[0 \leftarrow \mathsf{exp}_{0,\mathcal{A}}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathsf{exp}_{1,\mathcal{A}}(1^\lambda)\right]\right|$$

denote the *advantage* of the adversary $\mathcal{A}$ in distinguishing the two experiments. We say that $\mathsf{exp}_{0,\mathcal{A}}(1^\lambda)$ and $\mathsf{exp}_{1,\mathcal{A}}(1^\lambda)$ are *computationally indistinguishable* (or $\mathsf{exp}_{0,\mathcal{A}}(1^\lambda) \overset{c}{=} \mathsf{exp}_{1,\mathcal{A}}(1^\lambda)$) if

$$\max_{\mathcal{A}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{A}}(1^\lambda))) < \mathsf{negl}(\lambda)$$

for some negligible function negl, where the maximum is taken over all PPT algorithms $\mathcal{A}$.[7] We say that they are statistically indistinguishable (or replace $\overset{c}{=}$ with $\overset{s}{=}$) if they are negligibly close for adversaries of unbounded running time.

Hybrid games. Let H_i and H_{i+1} denote consecutive games within a hybrid argument. Define $\mathsf{exp}_{b,\mathcal{D}}^{\mathsf{H}_i;\mathsf{H}_{i+1}}(1^\lambda)$ to be a decisional experiment, where the PPT algorithm $\mathcal{D}$ attempts to distinguish between H_i and H_{i+1}. In this situation, $b_\mathcal{D} \leftarrow \mathcal{D}$ is such that $b_\mathcal{D} = 0$ if $\mathcal{D}$ guesses H_i, and $b_\mathcal{D} = 1$ if $\mathcal{D}$ guesses H_{i+1}.

Oracles. We write $\mathcal{A}^{\mathcal{O}_\mathcal{Y}(f(\cdot))}$ in a security game to indicate that a PPT algorithm $\mathcal{A}$ has 'oracle access' to the function f with domain $\mathcal{Y}$. During this access, $\mathcal{A}$ submits queries $y \in \mathcal{Y}$ to a challenger who returns $f(y)$ and can choose these queries adaptively. If the challenger keeps track of the queries to the oracle using a set $\mathcal{Q}$, then we write $\mathcal{A}^{\mathcal{O}_\mathcal{Y}(f(\cdot),\mathcal{Q})}$ where for every query $y \in \mathcal{Y}$, then $y \to \mathcal{Q}$. If the adversary is only permitted to make $m \in \mathbb{Z}$ calls to the oracle, we will write $\mathcal{A}^{\mathcal{O}_\mathcal{Y}(f(\cdot);[m])}$.

3.2 Lattice preliminaries

An n-dimensional lattice Λ is a discrete, additive subgroup of $\mathbb{R}^n$. Given n linearly independent basis vectors $\boldsymbol{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\} \in \mathbb{R}^{n \times n}$, the lattice generated by $\boldsymbol{B}$ is $\Lambda(\boldsymbol{B}) = \{\boldsymbol{v}_i \mid \boldsymbol{v}_i = \sum_{i=1}^n x_i \boldsymbol{b}_i;\ x_i \in \mathbb{Z}\}$. Let $\Lambda + \boldsymbol{c} = \{\boldsymbol{v} + \boldsymbol{c} \mid \boldsymbol{v} \in \Lambda\}$ denote the $\boldsymbol{c}$ coset of Λ. The *rank* of the lattice is defined to be the rank of the matrix $\boldsymbol{B}$. We will only concern ourselves with lattices Λ s.t. $q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m$.

The i^{th} *successive minima* of a lattice, denoted by $\lambda_i(\Lambda)$, is the radius of the smallest ball (centred at the origin) that contains i linearly independent vectors $\boldsymbol{v}_i \in \Lambda$. As such, $\lambda_1(\Lambda)$ is the length of the shortest vectors in Λ. This notation should not be confused with the security parameter λ that we use throughout this paper.

Let $\gamma > 1$ be an approximation factor. The GapSVP$_\gamma$ problem is a widely-known hard problem that is used for characterising the hardness of cryptographic assumptions relating to lattices.

[7] We sometimes omit explicit mention of the security parameter if the context is obvious.

Definition 3.1. (γ-Gap Shortest Vector Problem (GapSVP$_\gamma$)) *Given a basis $\boldsymbol{B}$ of a lattice $\Lambda = \Lambda(\boldsymbol{B})$ and a real number $d > 0$, output* 1 *if $\lambda_1(\Lambda) \le d$ and* 0 *if $\lambda_1(\Lambda) > \gamma \cdot d$. There are no requirements if the value is between d and $\gamma \cdot d$.*

Gaussian distributions. For any $s > 0$, define the *Gaussian function* on $\mathbb{R}^n$ centred at $\boldsymbol{c} \in \mathbb{R}^n$ with parameter s to be:

$$\forall\, \boldsymbol{x} \in \mathbb{R}^n,\ \rho_{s,\boldsymbol{c}}(\boldsymbol{x}) = e^{-\pi \|\boldsymbol{x}-\boldsymbol{c}\|^2/s^2}.$$

Likewise, for any $s, \boldsymbol{c}$ as above and n-dimensional lattice Λ, define the *discrete Gaussian distribution* over Λ as:

$$\forall\, \boldsymbol{x} \in \Lambda,\ D_{\Lambda+\boldsymbol{c},s}(\boldsymbol{x}) = \frac{\rho_{s,\boldsymbol{c}}(\boldsymbol{x})}{\rho_{s,\boldsymbol{c}}(\Lambda)}.$$

The parameter s is referred to as the width of the distribution. Sometimes we use σ instead if we want to refer to the width using the standard deviation of the distribution explicitly.

We will state a number of well-known lemmas, the proofs are omitted from this paper and we urge the readers to refer to the citations for the full proofs.

In this work, we will make use of *error distributions*, samples from this distribution should have norms bounded below some known value with high probability. We can show such a result for the Gaussian distribution $D_{\Lambda,s}$ over the lattice Λ, with parameter $s > 0$.

Lemma 3.2. ([MR04, PR06]) *Let B be a basis of an n-dimensional lattice Λ and let $\tilde{B}$ denote the Gram-Schmidt orthogonalisation of B. Let $s \ge \|\tilde{B}\| \cdot \omega(\log \lambda)$ and $\mathbf{x} \leftarrow_{\$} D_{\Lambda,s}$, then:*

$$\Pr\left[(\|\mathbf{x}\| \ge s\sqrt{n}) \vee (\boldsymbol{x} = \boldsymbol{0})\right] < \mathsf{negl}(\lambda)\,.$$

The lemma below states that, for $\boldsymbol{A} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$, where $m = \Omega(n\log(q))$, then $\boldsymbol{A}\boldsymbol{r} \overset{s}{=} \boldsymbol{u}$; where $\boldsymbol{r} \leftarrow_{\$} D_{\mathbb{Z}^m,\sigma}$ and $\boldsymbol{u} \leftarrow_{\$} \mathbb{Z}_q^m$. It is sometimes known as the leftover-hash lemma for lattices.

Lemma 3.3. ([GPV08]) *Let $q > 0$ be a prime, let n, m be positive integers such that $m \ge 2n\log(q))$, let $\sigma \ge \omega(\sqrt{\log n})$. Then for $\boldsymbol{A} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{r} \leftarrow_{\$} D_{\mathbb{Z}^m,\sigma}$, the distribution $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{r})$ is statistically indistinguishable from the distribution $(\boldsymbol{A}, \boldsymbol{u})$, for $\boldsymbol{u} \leftarrow_{\$} \mathbb{Z}_q^n$.*

We now state the following corollary that we eventually use in the security proof.

Corollary 3.4. *Let $(\boldsymbol{A}^{(u)}, \boldsymbol{A}^{(u)}\boldsymbol{r})_{u\in[\ell]\cup\{0\}} \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$ for $\ell = \mathsf{poly}(\lambda)$. Then this distribution is statistically indistinguishable from the distribution $(\boldsymbol{A}^{(0)}, \boldsymbol{c}), (\boldsymbol{A}^{(u)}, \boldsymbol{A}^{(u)}\boldsymbol{r})_{u\in[\ell]}$ where $\boldsymbol{c} \leftarrow_{\$} \mathbb{Z}_q^n$.*

Proof. We give a sketch of the hybrid argument that is required. Firstly, we invoke $\ell + 1$ independent distributions from Lemma 3.3 to switch all pairs to be of the form:

$$(\boldsymbol{A}^{(u)}, \boldsymbol{c}^{(u)})_{u\in[\ell]\cup\{0\}}$$

for $\boldsymbol{c}^{(u)} \leftarrow_{\$} \mathbb{Z}_q^n$. This is possible, because the samples are distributed independently of each other, by the independent choice of $\boldsymbol{A}^{(u)}$ for each $u \in [\ell] \cup \{0\}$. Secondly, we invoke the reverse transformation for $u \in [\ell]$ so that we acquire a distribution of the form:

$$(\boldsymbol{A}^{(0)}, \boldsymbol{c}), (\boldsymbol{A}^{(u)}, \boldsymbol{A}^{(u)}\boldsymbol{r})_{u\in[\ell]}$$

by invoking Lemma 3.3 in reverse, ℓ times. This is identical to the second distribution and so the distinguishing adversary now has no advantage. □

3.3 Trapdoor matrices.

In the following lemmas, we will consider matrices taken from the rings $\mathbb{Z}_q^{n\times m}$ and $\mathbb{Z}_q^{m\times m}$ for parameters $n = \mathsf{poly}(\lambda)$ and $m = \Omega(n\log(q))$; modulus $q \geq 2$; and where $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ generically denotes quotient ring with respect to parameter q.

Lemma 3.5. (Trapdoor sampling [Ajt99, GPV08, MP12]) *There is a PPT algorithm denoted by* $\mathsf{TrapSamp}(1^\lambda, 1^n, 1^m, q)$ *that outputs a pair* $(\boldsymbol{A}, \boldsymbol{T}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{m\times m}$*, where the distribution of* $\boldsymbol{A}$ *is* $\mathsf{negl}(\lambda)$ *statistical distance away from uniform.*

Lemma 3.6. (Preimage sampling [Ajt99, GPV08, MP12]) *There is a PPT algorithm denoted by* $\mathsf{PreImgSamp}(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{Y}, \sigma)$ *that, with overwhelming probability over* $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow_{\$} \mathsf{TrapSamp}(1^n, 1^m, q)$*, for sufficiently large* $\sigma = \Omega(\sqrt{n\log(q)})$*, satisfies:*

$$\{(\boldsymbol{A}, \boldsymbol{D}, \boldsymbol{Y}) \mid \boldsymbol{D} \leftarrow_{\$} \mathsf{PreImgSamp}(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{Y}, \sigma)\} \overset{c}{=} \{(\boldsymbol{A}, \boldsymbol{D}, \boldsymbol{Y}) \mid \boldsymbol{D} \leftarrow_{\$} D_{\mathbb{Z},\sigma}^{m\times m}, \boldsymbol{Y} = \boldsymbol{A}\boldsymbol{D}\}$$

for all PPT distinguishing algorithms.

Finally, we prove a corollary of Lemma 3.6 that we use in the proof of Theorem 4.3.

Corollary 3.7. *With overwhelming probability over* $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow_{\$} \mathsf{TrapSamp}(1^\lambda, 1^n, 1^m, q)$*, and for sufficiently large* $\sigma = \Omega(\sqrt{n\log(q)})$*, a PPT distinguishing algorithm cannot distinguish samples* $(\boldsymbol{A}_1, \dots, \boldsymbol{A}_\ell, \boldsymbol{D}_1, \dots, \boldsymbol{D}_\ell, \boldsymbol{Y})$*, where:*

$$\boldsymbol{A}_L = \boldsymbol{A}_{L-1}\boldsymbol{D}_{L-1} \text{ for } L \in [\ell];\ \boldsymbol{Y} \leftarrow \boldsymbol{A}_\ell\boldsymbol{D}_\ell; \tag{1}$$

or:

$$\boldsymbol{A}_{\ell+1} \leftarrow_{\$} \mathbb{Z}_q^{n\times m};\ \{\boldsymbol{A}_i, \boldsymbol{T}_i\}_{i\in[\ell]} \leftarrow_{\$} \mathsf{TrapSamp}(1^\lambda, 1^n, 1^m, q);\ \{\boldsymbol{D}_i \leftarrow_{\$} \mathsf{PreImgSamp}(\boldsymbol{A}_i, \boldsymbol{T}_i, \boldsymbol{A}_{i+1})\}_{i\in[\ell]}; \tag{2}$$

and $\boldsymbol{Y} = \boldsymbol{A}_{\ell+1}$.

Proof. We prove this corollary by showing that we can switch to a game where the first method of sampling is statistically close the second method. This provides a distinguishing game that a PPT adversary has no advantage in.

- H_0: This is the same as Equation (1).
- H_i ($i \in [\ell]$): Same as H_{i-1}, except sample:

$$\boldsymbol{A}_i, \boldsymbol{T}_i \leftarrow_{\$} \mathsf{TrapSamp}(1^\lambda, 1^n, 1^m, q);\ \boldsymbol{D}_i \leftarrow_{\$} \mathsf{PreImgSamp}(\boldsymbol{A}_i, \boldsymbol{T}_i, \boldsymbol{A}_{i+1});$$

  where $\boldsymbol{A}_{\ell+1} = \boldsymbol{Y}$.

Claim 3.7.1. $\max_{\mathcal{D}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{D}}^{\mathsf{H}_{i-1},\mathsf{H}_i}(1^\lambda))) < \mathsf{negl}(\lambda)$ *by Lemmas 3.5 and 3.6.*

Proof. By Lemma 3.5, distinguishing $\boldsymbol{A}_i \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{A}_i \leftarrow_{\$} \mathsf{TrapSamp}(1^\lambda, 1^n, 1^m, q)$ is statistically indistinguishable. Since $\boldsymbol{A}_i$ is statistically close to being uniformly distributed, we can argue by Lemma 3.6 that the method used for sampling $\boldsymbol{D}_i$ is indistinguishable in both hybrids.

The rest of the matrix sampling can be done trivially so the proof of Claim 3.7.1 is finished. □

The proof of Corollary 3.7 is inferred directly by repeated application of Claim 3.7.1 for each $i \in [\ell]$. □

3.4 Learning with errors

Throughout this work we rely on the hardness of the learning with errors (LWE) problem, first introduced by Regev in 2005 [Reg05].

Definition 3.8. (LWE [Reg05]) *Let $q, n, m = \mathsf{poly}(\lambda)$ be parameters and let χ be an error distribution. The* learning with errors problem *($\mathsf{LWE}_{q,n,m,\chi}$) is to distinguish between:*

$$(\boldsymbol{A}, \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m \tag{3}$$

and

$$(\boldsymbol{A}, \boldsymbol{U}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m \tag{4}$$

for $\boldsymbol{A}, \boldsymbol{U} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$; $\boldsymbol{s} \leftarrow_{\$} \chi^{n\times n}$; and $\boldsymbol{e} \leftarrow_{\$} \chi^m$.

Regev [Reg05] gave a quantum reduction showing that $\mathsf{LWE}_{q,n,m,\chi}$ is at least as hard as solving GapSVP$_\gamma$ to $\gamma = \tilde{O}(n/\alpha)$ approximation factors, when $\chi = D_{\mathbb{Z},\sigma}$ for $\sigma = \alpha \cdot q$ and $\alpha > 0$. Peikert [Pei09] gave a classical reduction for the same problem. We give specific parameters later when discussing the hardness of our schemes from the $\mathsf{LWE}_{q,n,m,\chi}$ problem.

Let $\mathsf{exp}^{\mathsf{lwe}}_{b,\mathcal{A}}(1^\lambda, q, n, m, \chi)$ denote the experiment where a PPT adversary $\mathcal{A}$ attempts to distinguish samples from the LWE problem. We let $b = 0$ denote the case where $\mathcal{A}$ receives samples as in Equation (3), and $b = 1$ denote the case where $\mathcal{A}$ receives samples as in Equation (4). Let $\mathsf{Adv}(\max_{\mathcal{A}}(\mathsf{exp}^{\mathsf{lwe}}_{b,\mathcal{A}}(1^\lambda, q, n, m, \chi)))$ denote the advantage of $\mathcal{A}$. We may omit the additional parameters q, n, m, χ if they are obvious from context.

Matrix LWE. We will require the usage of a form of learning with errors where the secret $\boldsymbol{s}$ in Definition 3.8 is replaced with a matrix $\boldsymbol{S} \in \mathbb{Z}^{n\times n}$. Note that the hardness of this form of the problem can be trivially bounded by $n \cdot \max_{\mathcal{A}}(\mathsf{Adv}(\mathsf{exp}^{\mathsf{lwe}}_{b,\mathcal{A}}(1^\lambda, q, n, m, \chi)))$. Therefore, we will use this form of LWE in the following sections, without distinguishing from the explicit format given in Definition 3.8. This also applies to the binary LWE problem.

3.5 Non-uniform learning with errors

Boneh et al. [BLMR13] introduced the notion of 'Non-uniform learning with errors' (NULWE) where the distribution of the public element, $\boldsymbol{A}$, in an LWE sample is not necessarily uniform. They show that a reduction from LWE to NULWE exists in the case where $\boldsymbol{A} \leftarrow_{\$} \gamma^{n\times m}$, where γ is a 'coset-samplable' distribution and $\boldsymbol{S} \leftarrow_{\$} \mathbb{Z}_q^{n\times n}$. Formal definitions of NULWE and coset-samplable distributions are given below.

Definition 3.9. (Non-uniform LWE [BLMR13]) *Let q, n, m, χ be as in Definition 3.8, let $k = \mathsf{poly}(\lambda)$, and let γ be a distribution over $\mathbb{Z}_q$. The* non-uniform learning with errors problem *($\mathsf{NULWE}_{q,n,k,m,\chi,\gamma}$) is to distinguish between:*

$$(\boldsymbol{D}, \boldsymbol{R}\boldsymbol{D} + \boldsymbol{E}) \in \mathbb{Z}_q^{k\times m} \times \mathbb{Z}_q^{n\times m}$$

and

$$(\boldsymbol{D}, \boldsymbol{U}) \in \mathbb{Z}_q^{k\times m} \times \mathbb{Z}_q^{n\times m}$$

for $\boldsymbol{D} \leftarrow_{\$} \gamma^{k\times m}$; $\boldsymbol{U} \leftarrow_{\$} \mathbb{Z}_q^{k\times m}$; $\boldsymbol{R} \leftarrow_{\$} \mathbb{Z}_q^{n\times k}$; and $\boldsymbol{E} \leftarrow_{\$} \chi^{k\times m}$.

Let $\max_{\mathcal{A}}(\mathsf{Adv}(\mathsf{exp}^{\mathsf{nulwe}}_{b,\mathcal{A}}(1^\lambda, q, n, k, m, \chi, \gamma)))$ denote the advantage of all PPT adversaries $\mathcal{A}$ in distinguishing the samples in the $\mathsf{NULWE}_{q,n,k,m,\chi,\gamma}$ problem (which we categorise as the experiments $\mathsf{exp}^{\mathsf{nulwe}}_{b,\mathcal{A}}(1^\lambda, q, n, k, m, \chi, \gamma)$).

Definition 3.10. (n-coset samplable distributions [BLMR13]) *For parameters $q, n, m, k = \mathsf{poly}(\lambda)$, we say that a distribution $\gamma = \gamma(\lambda)$ over $\mathbb{Z}_q$ is* n-coset samplable *if there are two PPT algorithms* ($\mathsf{MatSamp}()$, $\mathsf{PreImgSamp}()$) *such that:*

- $\mathsf{MatSamp}(q, 1^n, 1^k, 1^m)$: *outputs a matrix* $\boldsymbol{M} \in \mathbb{Z}_q^{n\times k}$ *and auxiliary data* $\boldsymbol{T}$*;*
- $\mathsf{PreImgSamp}(\boldsymbol{Y} \in \mathbb{Z}_q^{n\times m}, \boldsymbol{T})$: *outputs* $\boldsymbol{D} \in \mathbb{Z}^{k\times m}$ *satisfying* $\boldsymbol{MD} = \boldsymbol{Y}$ *where if* $\boldsymbol{Y} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$ *then* $\boldsymbol{D}$ *is distributed statistically close to* $\gamma^{k\times m}$*.*

It was shown in [BLMR13] that a reduction from $\mathsf{LWE}_{q,n,m,\chi}$ to $\mathsf{NULWE}_{q,n,k,m,\chi,\gamma}$ in the case where γ is an n-coset samplable distribution where $k \geq n$. We briefly summarise the results of [BLMR13] in Lemma 3.12.

Remark 3.11. *For now, we abuse notation and use the same notation* $\mathsf{PreImgSamp}()$ *for this algorithm as was used in Lemma 3.6. However, we show later that the distribution* γ *specifies that* $\mathsf{PreImgSamp}()$ *is the same algorithm in both cases.*

Lemma 3.12. ([BLMR13, Lemma 4.3]) *Let* $\gamma = \gamma(\lambda)$ *be an* n*-coset samplable distribution and let* $\epsilon = \epsilon(\lambda)$*. If there is a PPT algorithm* $\mathcal{A}$ *satisfying* $\max_{\mathcal{A}}(\mathsf{Adv}(\mathsf{exp}^{\mathsf{nulwe}}_{b,\mathcal{A}}(1^\lambda, q, n, k, m, \chi, \gamma))) = \epsilon$*, then there is a PPT algorithm* $\mathcal{B}$ *satisfying* $\max_{arg}(\mathsf{Adv}(\mathsf{exp}^{\mathsf{lwe}}_{c,\mathcal{B}}(1^\lambda, q, n, m, \chi))) = \epsilon$*.*

Finally, it was shown in [BLMR13] that γ can be instantiated with the following distributions:

- $\gamma_{\{0,1\}}$: the uniform distribution on $\{0,1\}^{k\times m}$ for sufficiently large k;
- γ_V: a uniform distribution over a sufficiently large linear subspace V of $\mathbb{Z}_q^{k\times m}$;
- γ_σ: a discrete Gaussian, $D_{\mathbb{Z},\sigma}$, on $\mathbb{Z}^{k\times m}$ with sufficiently large k and standard deviation σ.

That γ_σ is n-coset samplable follows directly from Lemma 3.5 and Lemma 3.6 by instantiating $\mathsf{MatSamp}()$ with $\mathsf{TrapSamp}()$ and likewise using $\mathsf{PreImgSamp}()$ as it is defined. We state this formally along with parameter settings below.

Corollary 3.1. *Let* $q, n = \mathsf{poly}(\lambda)$ *be defined as previously, let* $k \geq 6n\log q$*,* $\sigma = \Omega(\sqrt{n\log q})$*, and* $\gamma_\sigma = D_{\mathbb{Z},\sigma}$*. Then* $\mathsf{NULWE}_{q,n,k,m,\chi,\gamma_\sigma}$ *is at least as hard as* $\mathsf{LWE}_{q,n,m,\chi}$*.*

Proof. We simply show that γ_σ is n-coset samplable as in Lemma 3.12.

- $\mathsf{MatSamp}(q, 1^n, 1^k, 1^m)$ runs $\mathsf{TrapSamp}(1^\lambda, 1^n, 1^k, q)$ and outputs $(\boldsymbol{M}, \boldsymbol{T})$.
- $\mathsf{PreImgSamp}(\boldsymbol{M}, \boldsymbol{T}, \boldsymbol{Y}, \sigma)$ simply outputs $\boldsymbol{D} \in \mathbb{Z}^{k\times m}$ exactly as defined in Lemma 3.6.

$\square$

Notice, that we can replace k with m providing that $m = \Omega(n\log q)$, as defined in Lemma 3.6. Similar proofs can be made for $\gamma_{\{0,1\}}$ and γ_V. We refer the reader to [BLMR13] for the explicit arguments.

3.6 Pseudorandom functions

A pseudorandom function (PRF) is a tuple $\mathsf{PRF} = (\mathsf{Setup}, \mathsf{Eval})$. The security requirement is that, for $K \leftarrow_{\$} \mathcal{K}$, then the outputs of the function $\mathsf{PRF.Eval} : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$ on K and adversarial $x \in \mathcal{X}$ are computationally indistinguishable from the evaluations of a random function $f : \mathcal{X} \mapsto \mathcal{Y}$ on the same x.

More formally, let $\mathcal{F} = \{f : f : \mathcal{X} \mapsto \mathcal{Y}\}$, then the PRF indistinguishability game asks an adversary to distinguish the two experiments in Figure 1. All oracle queries are handled by the real evaluation function $\mathsf{PRF.Eval}$, this oracle is denoted by $\mathcal{O}_{\mathcal{X}}(\mathsf{PRF.Eval}(\mathsf{msk}, x))$ where x is the input query — recalling that $\boldsymbol{\varphi}$ collates the input queries that have been asked by $\mathcal{A}$. At the challenge point, $x^\dagger$, the output is taken from either: the PRF in $\mathsf{exp}^{\mathsf{prf}}_{0,\mathcal{A}}(1^\lambda)$; or a uniform function in $\mathsf{exp}^{\mathsf{prf}}_{1,\mathcal{A}}(1^\lambda)$. We give a formalisation of the security requirement in Definition 3.1.

Definition 3.1. (Pseudorandom function) *Let* $\mathsf{PRF} = (\mathsf{Setup}, \mathsf{Eval})$ *be a tuple of algorithms and let* λ *be the security parameter. Let* $\mathcal{X}$ *be the* input space*, and let* $\mathcal{Y}$ *be the* output space*; and define the algorithms in the following way.*

```
$\mathsf{exp}^{\mathsf{prf}}_{0,\mathcal{A}}(1^\lambda)$
1 : $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{PRF.Setup}(1^\lambda)$;
2 : $b_{\mathcal{A}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{X}}(\mathsf{PRF.Eval}(\mathsf{msk},\cdot))}(1^\lambda, \mathsf{pp})$;
3 : return $b_{\mathcal{A}}$;
```

```
$\mathsf{exp}^{\mathsf{prf}}_{1,\mathcal{A}}(1^\lambda)$
1 : $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{PRF.Setup}(1^\lambda); f \leftarrow_{\$} \mathsf{pp}.\mathcal{F}$;
2 : $b_{\mathcal{A}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{X}}(f(\cdot))}(1^\lambda, \mathsf{pp})$;
3 : return $b_{\mathcal{A}}$;
```

Fig. 1. Standard PRF indistinguishability game.

- $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{PRF.Setup}(1^\lambda)$*: On input the security parameter, outputs a pair* $(\mathsf{pp}, \mathsf{msk})$ *consisting of public parameters and a master secret key, respectively.*
- $y \leftarrow \mathsf{PRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)$*: On input* $(\mathsf{pp}, \mathsf{msk})$ *and* $x \in \mathcal{X}$*; outputs a value* $y \in \mathcal{Y}$*.*

We say that PRF *is a* pseudorandom function*, or a PRF, if*

$$\max_{\mathcal{A}}(\mathsf{Adv}(\mathsf{exp}^{\mathsf{prf}}_{b,\mathcal{A}}(1^\lambda))) < \mathsf{negl}(\lambda)$$

holds, where $\mathcal{A}$ *is any PPT algorithm and* $\mathsf{exp}^{\mathsf{prf}}_{b,\mathcal{A}}(1^\lambda)$ *is defined as in Figure 1 for* $b \leftarrow_{\$} \{0, 1\}$*. We may equivalently say that* PRF *satisfies* pseudorandomness*.*

3.7 Constrained PRFs

Definition 3.2. *A constrained PRF is a tuple* CPRF *consisting of four algorithms:*

$$(\mathsf{Setup}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval}),$$

and satisfying the following functionality:

- $\mathsf{CPRF.Setup}(1^\lambda, 1^r, \mathcal{C})$*: On input the security parameter* λ*, a parameter* $r > 0$*, and a class of predicates* $\mathcal{C}$*: outputs public parameters* pp *and master secret key* msk*;*
- $\mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x \in \mathcal{X})$*: On input* $x \in \mathcal{X}$*, outputs some value* $y \in \mathcal{Y}$*.*
- $\mathsf{CPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, C \in \mathcal{C})$*: On input* $C \in \mathcal{C}$*, outputs a* constrained *key* CK_C*.*
- $\mathsf{CPRF.CEval}(\mathsf{pp}, \mathsf{CK}_C, x)$*: On input a constrained key* CK_C *for* $C \subseteq \mathcal{C}$*, if* $1 \leftarrow \mathsf{P}_C(x)$*: then outputs* $y \in \mathcal{Y}$*, else: outputs* $\perp$*.*

We may sometimes omit the class $\mathcal{C}$ *from the inputs to* CPRF.Setup*, if it is obvious from context.*

It is clear that, in comparison with a standard PRF, a CPRF is augmented with the additional functionality of Constrain and CEval.

A constrained key CK obtained from $\mathsf{CPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, C)$ can evaluate the original pseudorandom function at inputs $x \in \mathcal{X}$ satisfying the predicate $\mathsf{P}_C(x)$, using $\mathsf{CPRF.CEval}(\mathsf{CK}_C, x)$. Such inputs x are termed *unconstrained*, inputs that cannot be evaluated (i.e. $0 \leftarrow \mathsf{P}_C(x'))$ are termed *constrained.*

The parameter r that is input to the setup algorithm is used as a bound on the number of queries that can be made. If this parameter is omitted, we assume that the number of constrained keys that can be learnt is unbounded. We may include additional setup parameters if they are required from a specific scheme.

Correctness. Let $C \in \mathcal{C}$ and let:

$$P = \Pr\left[\mathsf{CPRF.CEval}(\mathsf{pp}, \mathsf{CK}_C, x) \neq \mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x) \middle| \begin{array}{c} (\mathsf{pp},\mathsf{msk}) \leftarrow \mathsf{CPRF.Setup}(1^\lambda, 1^r, \mathcal{C}) \\ \mathsf{CK}_C \leftarrow \mathsf{CPRF.Constrain}(\mathsf{msk}, \mathsf{pp}, C) \\ x \in \mathcal{X};\ 1 \leftarrow \mathsf{P}_C(x) \end{array}\right],$$

Then CPRF is *correct* if we have that 1. $P < \mathsf{negl}(\lambda)$; and 2. each algorithm in the tuple CPRF runs in time $\mathsf{poly}(\lambda)$. We say that it is *perfectly correct* if $P = 0$.

Security. For the constrained PRF indistinguishability security game [BW13], we modify the adversary $\mathcal{A}$ so that it also has access to an oracle

$$\mathcal{O}_{\mathcal{C}}^{\varphi}(\cdot) = \mathcal{O}_{\mathcal{C}}(\mathsf{CPRF.Constrain}(\mathsf{msk}, \mathsf{pp}, \cdot), \varphi)$$

for learning constrained keys, and additionally an oracle $\mathcal{O}_{\mathcal{X}}^{\varphi}(\cdot) = \mathcal{O}_{\mathcal{X}}(\mathsf{CPRF.Eval}(\mathsf{msk}, \mathsf{pp}, \cdot), \varphi)$ for learning PRF evaluations in $\mathsf{exp}_{b,\mathcal{A}}^{\mathsf{cprf}}(1^\lambda, 1^r)$. The set φ is used to keep track of the points that $\mathcal{A}$ can currently evaluate, using constrained keys $\mathsf{CK}_C \leftarrow \mathcal{O}_{\mathcal{C}}(\mathsf{CPRF.Constrain}(\mathsf{msk}, \mathsf{pp}, C), \varphi)$, and of points x where $\mathcal{A}$ has queried $y \leftarrow \mathcal{O}_{\mathcal{X}}(\mathsf{CPRF.Eval}(\mathsf{msk}, \mathsf{pp}, x), \varphi)$.

Additionally, we specify a bound r on the number of constraint queries that can be ran respectively. These are included as extra inputs to the oracle $\mathcal{O}_{\mathcal{C}}^{\varphi}(\cdot)$. That is, there is an internal state in this oracle that monitors the number of queries that have been asked by $\mathcal{A}$. If r is exceeded then the oracle simply outputs $\perp$. Recall from Section 3.1 that we write $\mathcal{O}_{\mathcal{C}}^{\varphi}(\cdot\ ; [r])$ to indicate that the oracles are bounded in such a way. If r is removed from the inputs then we say that the number of allowed constraint queries is unbounded.

The entire (adaptive) security game is given in Figure 2, and formal specification of security is given in Definition 3.3.

```
$\mathsf{exp}_{0,\mathcal{A}}^{\mathsf{cprf}}(1^\lambda, 1^r, \mathcal{C})$
1 : $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{CPRF.Setup}(1^\lambda, 1^r, \mathcal{C})$;
2 : $x^\dagger \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{X}}^{\varphi}(\cdot), \mathcal{O}_{\mathcal{C}}^{\varphi}(\cdot\ ;[r])}(1^\lambda, 1^r, \mathsf{pp})$;
3 : if $x^\dagger \in \varphi$ :
4 :     return $\perp$;
5 : $y^\dagger \leftarrow \mathsf{CPRF.Eval}(\mathsf{msk}, x^\dagger)$;
6 : $b_{\mathcal{A}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{X}}^{\varphi}(\cdot), \mathcal{O}_{\mathcal{C}}^{\varphi}(\cdot\ ;[r])}(1^\lambda, 1^r, \mathsf{pp}, y^\dagger)$;
7 : if $x^\dagger \in \varphi$ :
8 :     return $\perp$;
9 : return $b_{\mathcal{A}}$;
```

```
$\mathsf{exp}_{1,\mathcal{A}}^{\mathsf{cprf}}(1^\lambda, 1^r, \mathcal{C})$
1 : $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{CPRF.Setup}(1^\lambda, 1^r, \mathcal{C})$; $f \leftarrow_{\$} \mathsf{pp}.\mathcal{F}$;
2 : $x^\dagger \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{X}}^{\varphi}(\cdot), \mathcal{O}_{\mathcal{C}}^{\varphi}(\cdot\ ;[r])}(1^\lambda, 1^r, \mathsf{pp})$;
3 : if $x^\dagger \in \varphi$ :
4 :     return $\perp$;
5 : $y^\dagger \leftarrow f(x^\dagger)$;
6 : $b_{\mathcal{A}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{X}}^{\varphi}(\cdot), \mathcal{O}_{\mathcal{C}}^{\varphi}(\cdot\ ;[r])}(1^\lambda, 1^r, \mathsf{pp}, y^\dagger)$;
7 : if $x^\dagger \in \varphi$ :
8 :     return $\perp$;
9 : return $b_{\mathcal{A}}$;
```

Fig. 2. CPRF indistinguishability game (adaptive).

Definition 3.3. (CPRF security) *Let* $\mathsf{exp}_{b,\mathcal{A}}^{\mathsf{cprf}}(1^\lambda, 1^r)$ *be the experiments defined as in Figure 2. We say that* CPRF *is an r-key secure,* constrained *pseudorandom function (or a CPRF) if*

$$\max_{\mathcal{A}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{A}}^{\mathsf{cprf}}(1^\lambda, 1^r))) < \mathsf{negl}(\lambda)\,,$$

holds for all PPT adversaries $\mathcal{A}$.

The game ultimately requires $\mathcal{A}$ to distinguish a PRF evaluation on a constrained input $x^\dagger$ (chosen by the adversary) from a uniformly distributed output. It concludes when the adversary submits a bit $b_{\mathcal{A}} \in \{0, 1\}$ indicating its decision. The formulation in Figure 2 targets adaptive security, since all queries are made adaptively. We can modify to target selective security by specifying that a subset of the queries are specified by the adversary before step one is ran.

1-key privacy. As an additional requirement, we can specify that a CPRF is a *private* CPRF (or PCPRF) if the constrained keys for two different constraints are indistinguishable. We can define the security game as in Figure 3 based on the indistinguishability model given in [BLW17]. The explicit formalisation is given in Definition 3.4.

```
$\mathsf{exp}^{\mathsf{pcprf}}_{b,\mathcal{A}}(1^\lambda, \mathcal{C})$
1 : $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{CPRF.Setup}(1^\lambda, \mathcal{C})$;
2 : $C_0, C_1 \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$;
3 : **if** $(C_0 \notin \mathcal{C}) \vee (C_1 \notin \mathcal{C})$ :
4 :     **return** $\perp$;
5 : $\mathsf{CK}_b \leftarrow \mathsf{CPRF.Constrain}(\mathsf{msk}, \mathsf{pp}, C_b)$;
6 : $b_{\mathcal{A}} \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}, \mathsf{CK}_b)$;
```

Fig. 3. Privately constrained property in indistinguishability framework.

Definition 3.4. (1-key privacy) *Let* $\mathsf{exp}^{\mathsf{pcprf}}_{b,\mathcal{A}}(1^\lambda)$ *denote the experiments from Figure 3. We say that* CPRF *is a* private *constrained pseudorandom function (or PCPRF) if*

$$\max_{\mathcal{A}}(\mathsf{Adv}(\mathsf{exp}^{\mathsf{pcprf}}_{b,\mathcal{A}}(1^\lambda))) < \mathsf{negl}(\lambda)$$

holds for all PPT adversaries $\mathcal{A}$.

We can expand the definition to include m-key privacy, for $m > 1$, by allowing the adversary in $\mathsf{exp}^{\mathsf{pcprf}}_{b,\mathcal{A}}(1^\lambda)$ to submit 2 vectors of length m of viable constraint circuits. We do not extend the definition in this work, as we can only achieve security in the $m = 1$ setting.

Remark 3.13. *This definition of key privacy that we use corresponds to a weaker definition that was given by [BLW17]. This is because the adversary does not get access to the CPRF evaluation oracle during* $\mathsf{exp}^{\mathsf{pcprf}}_{b,\mathcal{A}}(1^\lambda)$. *The work of [BLW17] also considered this stronger format in a separate definition.*

In the simulation-based framework of [CC17], the simulator has no access to the constraint when answering queries using CPRF.Constrain. We are unable to prove our construction secure in this setting, and so we use the weaker indistinguishability framework above.

4 Construction

Before describing our construction, we recall some notation that we defined in Section 3.1. let $\mathbb{T}, \mathbb{B}_r$ be sets such that

$$\mathbb{T} = \{(t_1, \ldots, t_r) \in [\ell]^r \mid t_1 \le t_2 \le \ldots \le t_r\};$$

and

$$\mathbb{B}_\ell = \left\{(b_1, \ldots, b_\ell) \in \{0,1\}^\ell\right\}.$$

For $\boldsymbol{b} \leftarrow \mathbb{B}_\ell$, we will write $\boldsymbol{b_t} \leftarrow \mathsf{reindex}(\boldsymbol{b}, \boldsymbol{t})$ to denote the vector that is reindexed with respect to the unique entries t_i in $\boldsymbol{t} \in \mathbb{T}$ (entries where $(i = 1) \vee (t_{i-1} < t_i)$). Let z be the number of such unique indices; for shorthand, we write $z \leftarrow \mathsf{unique}(\boldsymbol{t})$. Then $\boldsymbol{b_t} \in \{0,1\}^z$, including only those components $b_{t_i} \in \boldsymbol{b_t}$ for each $t_i \in [\ell]$ once. We may abuse notation and write $x_{\boldsymbol{t}} \leftarrow \mathsf{reindex}(x, \boldsymbol{t})$ similarly, where $x \in \{0,1\}^\ell$ is explicitly said to be a bitstring.

Let $\boldsymbol{t} \leftarrow \mathbb{T}$. In Figure 5, we define a function $\mathsf{ComputeSet}(\cdot)$ that takes a set of matrices, ordered with respect to $\boldsymbol{t}$, as input; and outputs the concatenation of said matrices with unique indices. As an example, for $r = 6$, if we have indices $\boldsymbol{t} = (t_1, \ldots, t_6) \leftarrow \mathbb{T}$; where $t_1 = t_2$, $t_2 < t_3$, $t_3 < t_4$, $t_4 = t_5 = t_6$, then $z = 3 \leftarrow \mathsf{unique}(\boldsymbol{t})$. Now, let $\boldsymbol{A}_{t_i} \in \mathbb{Z}_q^{n \times m}$ for $i \in [6]$. Then running

$$\boldsymbol{A_t} \leftarrow \mathsf{ComputeSet}(\{\boldsymbol{A}_{t_i}\}_{i \in [6]})$$

gives

$$\boldsymbol{A_t} = [\boldsymbol{A}_{t_1} \| \boldsymbol{A}_{t_3} \| \boldsymbol{A}_{t_4}] \in \mathbb{Z}_q^{n \times 3m}.$$

In addition, for such a $\boldsymbol{t}$ and $\boldsymbol{b} \in \mathbb{B}_\ell$, then we would have that $\boldsymbol{b_t} = (b_{t_1}, b_{t_3}, b_{t_4}) \leftarrow (\boldsymbol{b}, \boldsymbol{t})$. Furthermore, $z \leq r = O(1)$ by definition.

Finally, we will let $\mathit{\Gamma}_v$, denote the set

$$\mathit{\Gamma}_v = \left\{ (\boldsymbol{t}, \boldsymbol{b_t}) \;\middle|\; \begin{array}{l} (\boldsymbol{t},\boldsymbol{b}) \leftarrow \mathbb{T} \times \mathbb{B}_\ell, \\ \boldsymbol{b_t} \leftarrow \mathsf{reindex}(\boldsymbol{b},\boldsymbol{t}), \\ (v_{t_i} = b_{t_i}) \vee (v_{t_i} = *) \end{array} \right\}$$

for some $v \in \{0, 1, *\}$. Alternatively, we can infer that there exists some $j \in [r]$ such that $(v_{t_j} \neq b_{t_j}) \wedge (v_{t_j} \neq *)$. In particular, for the set of r constraint queries $\{v^{(l)}\}_{l \in [r]}$ made by the distinguishing adversary in $\mathsf{exp}^{\mathsf{cprf}}_{b,\mathcal{D}}(1^\lambda, 1^r)$, then $\exists$ a pair such that $(\boldsymbol{t}, \boldsymbol{b_t}) \notin \mathit{\Gamma}_{v^{(l)}}$, for each $l \in [r]$.

Construction 4.1. We provide our construction of an r-key secure CPRF for bit-fixing constraints, where $r = O(1)$. We also prove that our construction satisfies 1-key privacy. Our construction is presented in Figures 4, 5, 6, and 7. Note that constrained evaluation, on unconstrained inputs, is identical to the real evaluation algorithm using the master secret key.[8] For this reason, we will simply write $\mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{CK}_v, x)$ for $\mathsf{CK}_v \leftarrow \mathsf{CPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, v)$ in the sequel. We prove this explicitly in Theorem 4.2.

To run in time $\mathsf{poly}(\lambda)$, we require that $r = O(1)$. For instance, the master secret key msk contains $\sum_{k=1}^{r} 2^k \cdot \binom{\ell}{k}$ matrices, and so $r = \mathit{\Omega}(1)$ would result in super-polynomial key size.

<ins>$\mathsf{CPRF.Setup}(1^\lambda, 1^r, 1^n, 1^m, 1^\ell, \chi)$:</ins>

- For $i \in [\ell]$, sample the following:

$$\{\boldsymbol{A}_i\}_{i \in [\ell]} \leftarrow_\$ \mathbb{Z}_q^{n \times m};$$

$$\{\boldsymbol{D}^{(i)}_{j,b_j}\}_{\substack{j \in [\ell] \\ b_j \in \{0,1\}}} \leftarrow_\$ \chi^{m \times m};$$

- For each $\boldsymbol{t} \leftarrow \mathbb{T}$:
  - Let $z \leftarrow \mathsf{unique}(\boldsymbol{t})$
  - For each $\boldsymbol{b} \in \mathbb{B}_z$:
    - Sample a matrix:

$$\boldsymbol{D}^G_{\boldsymbol{t},\boldsymbol{b}} \leftarrow_\$ \chi^{zm \times m};$$

- Let $\rho = \sum_{k=1}^{r} \binom{\ell}{k}$ and sample:

$$\boldsymbol{D}^{\mathsf{end}} \leftarrow_\$ \chi^{\rho m \times m};$$

- Output

$$\mathsf{pp} = \left(\{\boldsymbol{A}_{i,b}\}_{\substack{i \in [\ell] \\ b \in \{0,1\}}}, \{\boldsymbol{D}^{(i)}_{j,b_j}\}_{\substack{j \in [\ell] \\ b_j \in \{0,1\}}}, \boldsymbol{D}^{\mathsf{end}} \right);$$

$$\mathsf{msk} = \left(\left\{ \boldsymbol{D}^G_{\boldsymbol{t},\boldsymbol{b}} \right\}_{\substack{\boldsymbol{t} \in \mathbb{T}, \\ z \leftarrow \mathsf{unique}(\boldsymbol{t}), \\ \boldsymbol{b} \in \mathbb{B}_z}} \right).$$

Fig. 4. Setup algorithm for CPRF. Note that we use the set $\mathbb{B}_z$ rather than $\mathbb{B}_\ell$ in this definition. This is effectively so that we can iterate over all possible $\boldsymbol{b_t} \leftarrow \mathsf{reindex}(\boldsymbol{b}, \boldsymbol{t}) \in \{0,1\}^z$, for $\boldsymbol{b} \in \mathbb{B}_\ell$.

Parameter settings. Before we discuss the correctness and security of our construction, we give a brief overview of the parameter settings that we require. We employ similar techniques to [BLMR13, BVWW16, BV15, GGH15, CC17], but we can leverage a slightly smaller q. The reason for this is that we do not need to invoke multiple LWE samples as part of a product

[8] In particular, a constrained key is statistically indistinguishable from the master secret key, we prove this in Lemma 4.6.

```
CPRF.Eval(pp, msk, $x \in \{0,1\}^\ell$):

  – Compute $\boldsymbol{Y}_i^x = \boldsymbol{A}_i \prod_{j=1}^{\ell} \boldsymbol{D}_{j,x_j}^{(i)}$ for $i \in [\ell]$;
  – For each $\boldsymbol{t} \leftarrow \mathbb{T}$:
      • Let $x_{\boldsymbol{t}} \leftarrow \mathsf{reindex}(x, \boldsymbol{t})$;
      • Compute $\boldsymbol{Y}_{\boldsymbol{t}}^x = \mathsf{ComputeSet}(\{\boldsymbol{Y}_{t_l}^x\}_{l \in [r]}) \in \mathbb{Z}_q^{n \times zm}$;
      • Compute:
            $\boldsymbol{Z}_{\boldsymbol{t}}^x = \boldsymbol{Y}_{\boldsymbol{t}}^x \boldsymbol{D}_{\boldsymbol{t},x_{\boldsymbol{t}}}^G \in \mathbb{Z}_q^{n \times m}$;
  – Let $\rho = \sum_{k=1}^{r} \binom{\ell}{k}$, and:
            $\boldsymbol{Z}_{\mathbb{T}}^x = [\{\boldsymbol{Z}_{\boldsymbol{t}}^x\}_{\boldsymbol{t} \in \mathbb{T}}] \in \mathbb{Z}_q^{n \times \rho m}$;
  – Output:
            $\left\lfloor \boldsymbol{Z}_{\mathbb{T}}^x \boldsymbol{D}^{\mathsf{end}} \right\rceil_p \in \mathbb{Z}_p^{n \times m}$;

ComputeSet($\{\boldsymbol{Y}_{t_l}\}_{l \in [r]}$):

  – $\boldsymbol{Y} = []$;
  – for $l \in [r]$:
      • if $(l = 1) \vee (t_{l-1} < t_l)$:
            $\boldsymbol{Y} \leftarrow [\boldsymbol{Y} \| \boldsymbol{Y}_{t_l}]$;
  – Output: $\boldsymbol{Y}$.
```

Fig. 5. Evaluation algorithm for CPRF.

```
CPRF.Constrain(pp, msk, $v \in \{0,1,*\}^\ell$):

  – if msk.st = $\emptyset$:
            $\mathsf{msk.st} = \left\{ \overline{\boldsymbol{D}_{\boldsymbol{t},\boldsymbol{b}}^G} \right\}_{\substack{\boldsymbol{t} \in \mathbb{T}, \\ \boldsymbol{b} \in \mathbb{B}}} \leftarrow_{\$} \chi^{zm \times m}$;
    where $z \leftarrow \mathsf{unique}(\boldsymbol{t})$.
  – For each $\boldsymbol{t} \leftarrow \mathbb{T}$:
      • For each $\boldsymbol{b} \leftarrow \mathbb{B}_z$:
          -- If $(\boldsymbol{t}, \boldsymbol{b}) \in \Gamma_v$:
                $G_{\boldsymbol{t},\boldsymbol{b}}^{(v)} = \boldsymbol{D}_{\boldsymbol{t},\boldsymbol{b}}^G \leftarrow \mathsf{msk}$;
          -- Else if $(\boldsymbol{t}, \boldsymbol{b}) \notin \Gamma_v$:
                $G_{\boldsymbol{t},\boldsymbol{b}}^{(v)} = \overline{\boldsymbol{D}_{\boldsymbol{t},\boldsymbol{b}}^G} \leftarrow \mathsf{msk.st}$;
      • Let $G_{\boldsymbol{t}}^{(v)} = \{G_{\boldsymbol{t},\boldsymbol{b}}^{(v)}\}_{\boldsymbol{b} \in \mathbb{B}_z}$;
  – Let $G^{(v)} = \{G_{\boldsymbol{t}}^{(v)}\}_{\boldsymbol{t} \in \mathbb{T}}$;
  – Output $\mathsf{CK}_v = G^{(v)}$, and msk.
```

Fig. 6. Constraining algorithm for CPRF.

```
CPRF.CEval(pp, $\mathsf{CK}_v$, $x$):

  – Output CPRF.Eval(pp, $\mathsf{CK}_v$, $x$).
```

Fig. 7. Constrained evaluation algorithm for CPRF. Since CK_v and msk are statistically indistinguishable (by Lemma 4.6), we can just use CK_v as input to the CPRF.Eval() algorithm.

during the security proof. We only require the addition of an error term at the end, that is not multiplied with any other matrices. This leads to concrete efficiency benefits and, in turn, a polynomial noise-to-modulus ratio.

Let λ be the security parameter, $\alpha, \sigma > 0$ and $\chi = D_{\mathbb{Z},\sigma}$. We set $m = 6n\log(q)$ (for satisfying Lemma 3.3, Lemma 3.12 and Corollary 3.1); $q/p > n^{3/2}m\sigma 2^{\ell+\lambda}$; $\sigma = \omega(\sqrt{n\log(q)})$, $\alpha = \sigma/q$ and $n\alpha < 2^{\lambda^{1-\epsilon}}$ for $0 < \epsilon < 1$ (for satisfying the reduction from GapSVP$_\gamma$ to $\mathsf{LWE}_{q,n,m,\chi}$ with approximation factors $\tilde{O}(n/\alpha)$).

4.1 Correctness

Theorem 4.2. *Construction 4.1 is perfectly* correct.

Proof. Let CK_v be some constrained key for $v \in \{0,1,*\}^\ell$, and let $x \in \{0,1\}^\ell$ be such that $\mathsf{P}_v(x) = 1$. Additionally let $\boldsymbol{t} \in \mathbb{T}$, $x_{\boldsymbol{t}} \leftarrow \mathsf{reindex}(x, \boldsymbol{t})$ and write

$$\boldsymbol{Y}^x_{\boldsymbol{t}} = \mathsf{ComputeSet}(\{\boldsymbol{Y}^x_{t_l}\}_{l\in[r]}) \in \mathbb{Z}_q^{n\times zm},$$

where $z \leftarrow \mathsf{unique}(\boldsymbol{t})$ and $\boldsymbol{Y}^x_{t_l} = \boldsymbol{A}_{t_l} \cdot \prod_{j=1}^{\ell} \boldsymbol{D}^{(t_l)}_{j,x_j}$. Then:

$$\begin{aligned}\mathsf{CPRF.CEval}(\mathsf{pp}, \mathsf{CK}_v, x) &= \left\lfloor \boldsymbol{Z}^x_{\mathbb{T}} \boldsymbol{D}^{\mathsf{end}} \right\rceil_p = \left\lfloor [\{\boldsymbol{Z}^x_{\boldsymbol{t}}\}_{\boldsymbol{t}\in\mathbb{T}}] \, \boldsymbol{D}^{\mathsf{end}} \right\rceil_p ; \\ &= \left\lfloor \left[\left\{ \boldsymbol{Y}_{\boldsymbol{t}} \boldsymbol{D}^G_{\boldsymbol{t},x_{\boldsymbol{t}}} \right\}_{\boldsymbol{t}\in\mathbb{T}} \right] \boldsymbol{D}^{\mathsf{end}} \right\rceil_p ; \\ &= \mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x).\end{aligned}$$

The final equality follows since x is unconstrained. That is, $(\boldsymbol{t}, x_{\boldsymbol{t}}) \in \Gamma_v$ for all $\boldsymbol{t} \in \mathbb{T}$. Therefore, $\mathsf{CK}_v = G^{(v)}$ only contains matrices taken from msk, and not $\mathsf{msk.st}$ (see Figure 6). ☐

4.2 Security

In this section we prove the main security theorem (Theorem 4.3) for our CPRF. In Lemma 4.4 we show that Construction 4.1 is a CPRF. Secondly, in Lemma 4.5 we show that our security proof holds in the adaptive security model with only polynomial security loss. Lastly, in Lemma 4.6 we show that our construction satisfies 1-key privacy from Definition 3.4.

The main computational assumption that we use is $\mathsf{LWE}_{q,n,m,\chi}$, where $\chi = D_{\mathbb{Z},\sigma}$, for appropriately chosen σ.

Theorem 4.3. *Construction 4.1 is an r-key secure, constraint-hiding CPRF from $\mathsf{LWE}_{n,m,q,\chi}$ (where $r = O(1)$) against adaptively chosen queries.*

Proof. The proof of this theorem follows from the proofs of Lemma 4.4, Lemma 4.5 and Lemma 4.6. Our proof strategy for Lemma 4.4 follows a similar to the strategy used by [CC17] for their bit-fixing CPRF, and is made in the selective query model. Recall that we do not consider the simulation-based security framework, however. Moreover, our scheme does not require the GGH15 [GGH15] trapdoor sampling strategy used by [CC17].[9] We obtain adaptive security via Lemma 4.5 and the proof of Lemma 4.6 follows almost immediately from the fact that our constrained keys retain very little structure.

Lemma 4.4. (Pseudorandomness on constrained points) *Construction 4.1 is an r-key secure CPRF for bit-fixing constraints, against $Q = \mathsf{poly}(\lambda)$ (selective) input queries and $\mathsf{poly}(\lambda)$ (selective) constraint queries; assuming the hardness of $\mathsf{LWE}_{q,n,m,\chi}$, where $r = O(1)$.*

[9] Though we do use the trapdoor sampling strategy during the proof, see Hybrid $\mathsf{H}_{1.(\iota[\mathrm{u}])}$.

Proof. We prove this theorem using the following sequence of hybrid arguments. In each hybrid step $\mathsf{H}_i \to \mathsf{H}_{i+1}$, we show that an adversary attempting to solve an instance of a given hardness assumption can simulate the two distributions. In $\mathsf{H}_{6.(0[\mathrm{Q}])}$, a PPT adversary $\mathcal{A}$ clearly has no advantage in distinguishing between $\mathsf{exp}^{\mathsf{cprf}}_{0,\mathcal{A}}(1^\lambda, 1^r)$ and $\mathsf{exp}^{\mathsf{cprf}}_{1,\mathcal{A}}(1^\lambda, 1^r)$.

Let $v^{(1)}, v^{(2)}, \ldots, v^{(r)} \in \{0,1,*\}^\ell$ denote the selectively chosen constraint queries, and let $x^\dagger \in \{0,1\}^\ell$ denote the selectively chosen challenge query by $\mathcal{A}$. Additionally, let $x^\dagger_{\boldsymbol{t}^\dagger} \leftarrow \mathsf{reindex}(x^\dagger, \boldsymbol{t}^\dagger)$ and $z \leftarrow \mathsf{unique}(\boldsymbol{t}^\dagger)$. Since the challenge query $x^\dagger$ should be a constrained input, then necessarily $\mathsf{P}_{v^{(i)}}(x^\dagger) = 0$ for each $i \in [r]$. We use the set $(x^{(1)}, \ldots, x^{(Q)})$ to denote the set of $Q = \mathsf{poly}(\lambda)$ input queries that we consider.

For each of the hybrid arguments, we consider a specific choice of $\boldsymbol{t}^\dagger \leftarrow \mathbb{T}$, where $(v^{(i)}_{t^\dagger_i} \neq *)$ for $i \in [r]$. There is at least one such $\boldsymbol{t}^\dagger \in \mathbb{T}$ for any r constraint queries corresponding to the challenge input $x^\dagger$. Otherwise $x^\dagger$ would be unconstrained for at least one of $v^{(i)}$. In other words, we can be sure that $(\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}) \notin \varGamma_{v^{(i)}}$ for $i \in [r]$, since $v^{(i)}_{t^\dagger_i} \neq x^\dagger_{t^\dagger_i}$.[10]

- H_0: This is Construction 4.1 in the experiment $\mathsf{exp}^{\mathsf{cprf}}_{0,\mathcal{D}}(1^\lambda, 1^r)$.

- $\mathsf{H}_{1.(\iota[0])}$ $(\iota \in [\ell])$: Same as $\mathsf{H}_{1.(\iota-1)}$, except: (1) sample $\boldsymbol{Y}^{x^\dagger}_\iota \leftarrow_\$ \mathbb{Z}_q^{n\times m}$; (2) compute:

$$\left\{\boldsymbol{A}^{(\iota,\dagger)}_j, \boldsymbol{T}^{(\iota)}_j \leftarrow_\$ \mathsf{TrapSamp}(1^\lambda, 1^n, 1^m, q)\right\}_{j\in[\ell]},$$

  and

$$\left\{\boldsymbol{D}^{(\iota)}_{j,x^\dagger_\iota} \leftarrow_\$ \mathsf{PreImgSamp}(\boldsymbol{A}^{(\iota,\dagger)}_j, \boldsymbol{T}^{(\iota)}_j, \boldsymbol{A}^{(\iota,\dagger)}_{j+1})\right\}_{j\in[\ell]},$$

  where $\boldsymbol{A}^{(\iota,\dagger)}_1 = \boldsymbol{A}_\iota$ and $\boldsymbol{A}^{(\iota,\dagger)}_{\ell+1} = \boldsymbol{Y}^{x^\dagger}_\iota$.

- $\mathsf{H}_{1.(\iota[\mathrm{u}])}$ $(\iota \in [\ell], u \in [Q])$: Same as $\mathsf{H}_{1.(\iota[\mathrm{u}-1])}$, except: (1) sample $\boldsymbol{Y}^u_\iota \leftarrow_\$ \mathbb{Z}_q^{n\times m}$. (2) Let ν be the bit where $x^{(u)}$ deviates from all other inputs $(x^\dagger, x^{(1)}, \ldots, x^{(u-1)})$. Compute:

$$\left\{\boldsymbol{A}^{(\iota,u)}_j, \boldsymbol{T}^{(\iota)}_j \leftarrow_\$ \mathsf{TrapSamp}(1^\lambda, 1^n, 1^m, q)\right\}_{j\in[\nu+1,\ell]},$$

  and

$$\left\{\boldsymbol{D}^{(\iota)}_{j,x^\dagger_\iota} \leftarrow_\$ \mathsf{PreImgSamp}(\boldsymbol{A}^{(\iota,u)}_j, \boldsymbol{T}^{(\iota)}_j, \boldsymbol{A}^{(\iota,u)}_{j+1})\right\}_{j\in[\nu,\ell]},$$

  where $\boldsymbol{A}^{(\iota,u)}_{\ell+1} = \boldsymbol{Y}^u_\iota$.

- H_2: Set $\boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger} = \boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger} \leftarrow_\$ \mathbb{Z}_q^{n\times m}$.

- H_3: Let $\boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} \in \mathbb{Z}^{m\times m}$ be the square block matrix that is multiplied with $\boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger}$ when computing $\boldsymbol{Z}^{x^\dagger}_{\mathbb{T}} \boldsymbol{D}^{\mathsf{end}} \in \mathbb{Z}_q^{n\times m}$. Replace the matrix product $\boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger} \boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger}$ with $\boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger} \boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} + \boldsymbol{E}^{\mathsf{end}}_{\boldsymbol{t}^\dagger}$ for $\boldsymbol{E}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} \leftarrow_\$ \chi^{n\times m}$.

- H_4: Replace

$$\boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger} \boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} + \boldsymbol{E}^{\mathsf{end}}_{\boldsymbol{t}^\dagger}$$

  with $\widehat{\boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger}} \leftarrow_\$ \mathbb{Z}_q^{n\times m}$.

- H_5: Replace the output of $\mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x^\dagger)$ with $\widehat{\boldsymbol{U}^{x^\dagger}} \leftarrow_\$ \mathbb{Z}_p^{n\times m}$.

[10] Note that we reorder the queries, without loss of generality, so that $t^\dagger_1 \le t^\dagger_2 \le \ldots \le t^\dagger_r$.

- $\mathsf{H}_{6.(\iota[\mathrm{u}])}$: Undo the step $\mathsf{H}_{1.(\iota[\mathrm{u}])}$, for decreasing $\iota \in [\ell]$ and $u \in [Q]$. In other words, for $x^{(u)}$, sample:
$$\left\{ \boldsymbol{D}^{(\iota)}_{j,x^{\dagger}_{\iota}} \leftarrow_{\$} \chi^{m\times m} \right\}_{j\in[\nu+1,\ell]} ;$$
where ν is the bit where $x^{(u)}$ deviates from $(x^{\dagger}, x^{(1)}, \dots, x^{(Q)})$. When $u = 0$, sampling for $\iota \in [\ell]$ should be of the form:
$$\boldsymbol{A}_{\iota} \leftarrow_{\$} \mathbb{Z}_q^{n\times m} \text{ and } \left\{ \boldsymbol{D}^{(\iota)}_{j,b} \right\}_{j\in[\ell], b\in\{0,1\}} \leftarrow_{\$} \chi^{m\times m}.$$

Claim 4.4.1. $\max_{\mathcal{D}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{D}}^{\mathsf{H}_{1.(\iota-1[\mathrm{Q}])},\mathsf{H}_{1.(\iota[0])}}(1^{\lambda}))) < \mathsf{negl}(\lambda)$ *by Corollary 3.7.*

Proof. Let $\mathcal{A}$ be an adversary who sees the distribution $(\{\boldsymbol{A}_j\}_{j\in[\ell]}, \{\boldsymbol{D}_j\}_{j\in[\ell]}, \boldsymbol{Y})$ in Corollary 3.7. Using the selectively chosen input query, $x^{\dagger}$, $\mathcal{A}$ sets $\boldsymbol{A}_j^{(\iota,\dagger)} = \boldsymbol{A}_j$ for $j \in [\ell]$ and sets $\boldsymbol{Y}_{\iota}^{x^{\dagger}} = \boldsymbol{Y}$. Finally, $\mathcal{A}$ sets $\boldsymbol{D}^{(\iota)}_{j,x^{\dagger}_j} = \boldsymbol{D}_j$.

Sample the rest of the matrices obliviously: i.e. $\boldsymbol{D}^{(\iota)}_{j,1-x^{\dagger}_j} \leftarrow_{\$} \chi^{m\times m}$.

For $\iota' < \iota$, sample the paths indexed by $(\iota', x^{\dagger}_{\iota'})$ as in $\mathsf{H}_{1.(\iota')}$. For $\iota'' > \iota$, sample the paths indexed by $(\iota'', x^{\dagger}_{\iota''})$ by sampling each matrix obliviously. All queries can be handled in exactly the same way in both hybrids, since the only difference is in the way that the matrices are sampled.

In the case of Equation (1), then $\mathcal{A}$ simulates $\mathsf{H}_{1.(\iota-1)}$ for $\mathcal{D}$. In the case of Equation (2), $\mathcal{A}$ simulates $\mathsf{H}_{1.(\iota)}$ for $\mathcal{D}$. Therefore, we can infer that if $\mathcal{D}$ had advantage ϵ in distinguishing the two hybrids, then $\mathcal{A}$ would succeed with the same advantage. Since Corollary 3.7 shows that $\mathcal{A}$ has negligible advantage, therefore we must have that $\epsilon < \mathsf{negl}(\lambda)$.

Since H_0 is equivalent to $\mathsf{H}_{1.(0[\mathrm{Q}])}$, this completes the proof of Claim 4.4.1. □

Claim 4.4.2. $\max_{\mathcal{D}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{D}}^{\mathsf{H}_{1.(\iota[\mathrm{u}-1])},\mathsf{H}_{1.(\iota[\mathrm{u}])}}(1^{\lambda}))) < \mathsf{negl}(\lambda)$ *by Corollary 3.7.*

Proof. Let $x^{(u)}$ be an input query from the set $(x^{(1)}, \dots, x^{(Q)})$, for $u \in [Q]$. Let ν be the first bit such that $x^{(u)}|^{\nu} \notin \{x^{\dagger}|^{\nu}, x_1|^{\nu}, \dots, x_{u-1}|^{\nu}$. In other words, the prefix of length ν of $x^{(u)}$ is distinct from all previous queries. Note that $\nu \in \ell$ because $x^{(u)}$ is not permitted to be the same as any other query.

Then, let $(\{\boldsymbol{A}_l\}_{l\in[\nu,\ell]}, \{\boldsymbol{D}_l\}_{l\in[\nu,\ell]}, \boldsymbol{Y}^u)$ be the distribution seen by adversary $\mathcal{B}$, taken from Corollary 3.7 (for the smaller range $[\nu+1, \ell]$). Let $\boldsymbol{A}_{\nu}$ be sampled as in $\mathsf{H}_{1.(\iota[\mathrm{u}-1])}$. Employing a similar argument to the previous claim, $\mathcal{B}$ sets $\boldsymbol{A}_l^{(\iota,u)} = \boldsymbol{A}_l$ for $l \in [\ell]$ and sets $\boldsymbol{Y}_{\iota}^{u} = \boldsymbol{Y}$. Finally, $\mathcal{B}$ sets $\boldsymbol{D}^{(\iota)}_{l,x^{(u)}_l} = \boldsymbol{D}_l$. All other matrices are sampled obliviously as in Claim 4.4.1.

By a similar argument to the proof of Claim 4.4.1, when the distribution from Corollary 3.7 is as in Equation (1), then the sampling is as in $\mathsf{H}_{1.(\iota[\mathrm{u}-1])}$, and otherwise it as in $\mathsf{H}_{1.(\iota[\mathrm{u}])}$. Therefore, $\mathcal{B}$ simulates the two hybrids for $\mathcal{D}$ within $\epsilon < \mathsf{negl}(\lambda)$ statistical distance of each other, by Corollary 3.7. □

Claim 4.4.3. $\max_{\mathcal{D}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{D}}^{\mathsf{H}_{1.(\ell[\mathrm{Q}])},\mathsf{H}_2}(1^{\lambda}))) < \mathsf{negl}(\lambda)$ *by Lemma 3.3 and our parameter choices.*

Proof. Let $\mathcal{A}$ be an adversary receives $(Q+1)m$ samples of the form $(\{(\boldsymbol{Y}^u, \boldsymbol{c}_i^{(u)})_{i\in[m]}\}_{u\in[Q]\cup\{0\}})$ given in Corollary 3.4. Specifically, $(\boldsymbol{Y}^u, \boldsymbol{c}_i) \in \mathbb{Z}_q^{n\times zm} \times \mathbb{Z}_q^{n}$ for each $i \in [m]$ and $u \in [Q] \cup \{0\}$. We require that, either $\boldsymbol{c}_i^{(u)} = \boldsymbol{Y}^u \boldsymbol{r}_i$ for $\boldsymbol{r}_i \leftarrow_{\$} \chi^{zm}$; or $\boldsymbol{c}_i^{(u)} \leftarrow_{\$} \mathbb{Z}_q^{m}$. Since $z \geq 1$, this is equivalent to receiving $(Q+1)m$ independent samples from Lemma 3.3; using the same set of m secret vectors $\boldsymbol{r}_i$.

Let $(\boldsymbol{Y}^u, \boldsymbol{C}^u) \in \mathbb{Z}_q^{n\times zm} \times \mathcal{Z}_q^{n\times m}$ refer to the concatenation of these samples, where the i^{th} column of $\boldsymbol{C}^u$ is set to be $\boldsymbol{c}_i^{(u)}$. Furthermore, write $\boldsymbol{Y}^u = [\boldsymbol{Y}^u[1] \| \dots \| \boldsymbol{Y}^u[z]]$ to denote the individual concatenated matrix components, where $\boldsymbol{Y}^u[l] \in \mathbb{Z}_q^{n\times m}$ for $l \in [z]$.

The adversary, $\mathcal{A}$ runs

$$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{CPRF.Setup}(1^\lambda, 1^r, 1^n, 1^m, 1^\ell, \chi)$$

as in $\mathsf{H}_{1.(\ell[\mathrm{Q}])}$, except for the challenge query $x^\dagger$, it sets $\boldsymbol{Y}^{x^\dagger}_{t^\dagger_l} = \boldsymbol{Y}^0[l] \in \mathbb{Z}_q^{n\times m}$, and $\boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger} = \boldsymbol{C}^0$.

For input queries $x^{(u)}$ (for $u \in [Q]$):

– If $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^{(u)}_{\boldsymbol{t}^\dagger}} \neq \boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$: then answer the query by simply running the evaluation algorithm using the simulated msk. This is the situation, when ${x^{(u)}}_{\boldsymbol{t}^\dagger} \neq {x^\dagger}_{\boldsymbol{t}^\dagger}$.
– If $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^{(u)}_{\boldsymbol{t}^\dagger}} = \boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$: then answer the query in the same way as the challenge query, except use $\boldsymbol{Y}^u_{t^\dagger_l} = \boldsymbol{Y}^u[l]$ for $i \in [\ell]$; and set $\boldsymbol{Z}^u_{\boldsymbol{t}^\dagger} = C^u$.

Constraint queries are answered as normal and recall that $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$ is never revealed to the adversary during these queries. When $\mathcal{A}$ makes constraint queries, the fact that $x^\dagger$ is constrained means that only $\overline{\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}} \leftarrow_{\$} \chi^{zm\times m}$ is revealed in constrained keys; and this is sampled independently. Thus, $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$ is never sampled explicitly by $\mathcal{A}$.

In the case where $\boldsymbol{C}^0 = \boldsymbol{Y}^0\boldsymbol{R}$, for $\boldsymbol{R} \in \mathbb{Z}_q^{zm\times m}$ the matrix where the i^{th} column is set to $\boldsymbol{r}_i$, then $\mathcal{A}$ simulates $\mathsf{H}_{1.(\ell[\mathrm{Q}])}$ with $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}} = \boldsymbol{R}$ inferred, implicitly. The distribution of $\boldsymbol{R}$ is identical to the distribution of $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$ and so the simulation is admissible.

If $\boldsymbol{C}^0 = \boldsymbol{U} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$, then this argument simulates H_2. By Corollary 3.3 of Lemma 3.3, we know that the two distributions:

$$((\boldsymbol{Y}^0, \boldsymbol{Y}^0\boldsymbol{R}), (\boldsymbol{Y}^u, \boldsymbol{C}^u)_{u\in[\ell]}) \text{ and } ((\boldsymbol{Y}^0, \boldsymbol{U}), (\boldsymbol{Y}^u, \boldsymbol{C}^u)_{u\in[\ell]})$$

are statistically close, providing that $m \geq 2n\log(q)$. In our parameter settings, we indeed choose $m = 6n\log(q)$. Thus, we can bound the advantage of any adversary $\mathcal{D}$ against $\mathsf{exp}^{\mathsf{cprf}}_{b,\mathcal{D}}(1^\lambda, 1^r)$ by the advantage of $\mathcal{A}$ distinguishing the distributions in Lemma 3.3 (or more accurately Corollary 3.4). Since these distributions are statistically close, then the advantage of $\mathcal{D}$ must be bounded by a negligible function. $\square$

Claim 4.4.4. $\max_{\mathcal{D}}(\mathsf{Adv}(\mathsf{exp}^{\mathsf{H}_2,\mathsf{H}_3}_{b,\mathcal{D}}(1^\lambda))) < \mathsf{negl}(\lambda)$ *by Lemma 3.2 and our choice of parameters.*

Proof. We know that $\boldsymbol{Z}^{x^\dagger}_{\mathbb{T}} = \left[\left\{\boldsymbol{Z}^{x^\dagger_{\boldsymbol{t}}}_{\boldsymbol{t}}\right\}_{\boldsymbol{t}\in\mathbb{T}}\right]$, and from H_2 then we have $\boldsymbol{Z}^{x^\dagger}_{\boldsymbol{t}^\dagger} = \boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$.

In the product $\boldsymbol{U}^{x^\dagger}_{\mathbb{T}}\boldsymbol{D}^{\mathsf{end}}$, let $\boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} \in \mathbb{Z}_q^{m\times m}$ be the matrix that is multiplied directly with $\boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger}$. Then the two differing distributions of H_2 and H_3 can be written as:

$$\left\lfloor \boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger}\boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} + \sum_{\boldsymbol{t}\neq\boldsymbol{t}^\dagger} \boldsymbol{Z}^{x^\dagger_{\boldsymbol{t}}}_{\boldsymbol{t}}\boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}} \right\rceil_p \text{ and } \left\lfloor \boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger}\boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}^\dagger} + \boldsymbol{E}_{\boldsymbol{t}^\dagger} + \sum_{\boldsymbol{t}\neq\boldsymbol{t}^\dagger} \boldsymbol{Z}^{x^\dagger_{\boldsymbol{t}}}_{\boldsymbol{t}}\boldsymbol{D}^{\mathsf{end}}_{\boldsymbol{t}} \right\rceil_p ;$$

for some matrix $\boldsymbol{E}_{\boldsymbol{t}^\dagger} \leftarrow_{\$} \chi^{m\times m}$. In other words, the only difference is the addition of this error matrix. Therefore, we can only distribute the two hybrid games, if adding $\boldsymbol{E}_{\boldsymbol{t}^\dagger}$ causes the output of the evaluation to change. That is, all queries are answered in the same manner as the previous hybrid, apart from this small change.

By the choice of $\chi = D_{\mathbb{Z},\sigma}$ and by Lemma 3.2, we have that

$$\|\boldsymbol{E}_{\boldsymbol{t}^\dagger}\|_\infty = B \leq \sigma\sqrt{n}$$

with overwhelming probability — i.e. $\boldsymbol{E}_{\boldsymbol{t}^\dagger}$ has small-norm relative to q. Then, the probability that such an event occurs for any given coordinate is $(2B+1)p/q$. Applying a union bound for all nm coordinates gives a total probability of $(2B+1)nmp/q$. By our choice of q, this probability is necessarily negligible.

Let $\Pr[\mathsf{BAD}_x] = (2B+1)nmp/q$ denote the probability of this occurring for some input x. Then $\Pr[\mathsf{BAD}] \leq 2^\ell \cdot \Pr[\mathsf{BAD}_x]$ is the probability that this event occurs for any given $x \in \{0,1\}^\ell$. Again, this probability remains statistically negligible and, consequently, H_2 and H_3 are statistically indistinguishable. □

Claim 4.4.5. $\max_{\mathcal{D}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{D}}^{\mathsf{H}_3,\mathsf{H}_4}(1^\lambda))) < \mathsf{negl}(\lambda)$ *by* $\mathsf{LWE}_{q,n,m,\chi}$.

Proof. Let $(\boldsymbol{D},\boldsymbol{B}) \in \chi^{k\times m} \times \mathbb{Z}_q^{n\times m}$ be a NULWE sample. We set $k=m$ and thus we use the $\mathsf{NULWE}_{q,n,m,m,\chi}$ assumption, which is implied by $\mathsf{LWE}_{q,n,m,\chi}$ by the results of Corollary 3.1; since clearly $k = \Omega(n\log(q))$ by the fact that $m = \Omega(n\log(q))$.

Let $\mathcal{A}$ be a distinguishing adversary against $\mathsf{NULWE}_{q,n,m,m,\chi}$, that attempts to simulate the two hybrids for the CPRF adversary $\mathcal{D}$. Then, $\mathcal{A}$ sets $\boldsymbol{D}_{\boldsymbol{t}^\dagger}^{\mathsf{end}} = \boldsymbol{D}$ and the rest of $\boldsymbol{D}^{\mathsf{end}}$ can be sampled obliviously from $\chi^{m\times m}$ for each $m\times m$ block corresponding to the pairs $(\boldsymbol{t}, x_{\boldsymbol{t}}^\dagger)$ where $\boldsymbol{t} \neq \boldsymbol{t}^\dagger$. The rest of $\mathsf{CPRF.Setup}(1^\lambda, 1^r, 1^n, 1^m, 1^\ell, \chi)$ can be sampled as normal according to the procedure in H_3.

For the challenge input query, compute the output as:

$$\left\lfloor \boldsymbol{B} + \sum_{\substack{\boldsymbol{t}\in\mathbb{T},\\ \boldsymbol{t}\neq\boldsymbol{t}^\dagger}} \boldsymbol{Z}_{\boldsymbol{t}}^{x^\dagger} \boldsymbol{D}_{\boldsymbol{t}}^{\mathsf{end}} \right\rceil_p .$$

For all input queries $x^{(u)}$, $u \in [Q]$; compute the output as:

$$\left\lfloor \sum_{\boldsymbol{t}\in\mathbb{T}} \boldsymbol{Z}_{\boldsymbol{t}}^{u} \boldsymbol{D}_{\boldsymbol{t}}^{\mathsf{end}} \right\rceil_p .$$

where $\boldsymbol{Z}_{\boldsymbol{t}}^u$ is as described in H_2. Constraint queries from $\mathcal{D}$ are handled normally (as in the proof of Claim 4.4.4, the matrix $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$ is never revealed).

Now, if $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{D} + \boldsymbol{E}$, then the output on $x^\dagger$ is effectively computed as:

$$\left\lfloor \boldsymbol{U}_{\boldsymbol{t}^\dagger}^{x^\dagger} \boldsymbol{D}_{\boldsymbol{t}^\dagger}^{\mathsf{end}} + \boldsymbol{E}_{\boldsymbol{t}^\dagger}^{\mathsf{end}} + \sum_{\substack{\boldsymbol{t}\in\mathbb{T},\\ \boldsymbol{t}\neq\boldsymbol{t}^\dagger}} \boldsymbol{Z}_{\boldsymbol{t}}^{x^\dagger} \boldsymbol{D}_{\boldsymbol{t}}^{\mathsf{end}} \right\rceil_p ,$$

for $\boldsymbol{U}_{\boldsymbol{t}^\dagger}^{x^\dagger} = \boldsymbol{U} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$, and $\boldsymbol{E}_{\boldsymbol{t}^\dagger}^{\mathsf{end}} = \boldsymbol{E}$. This is equivalent to how the output is constructed in H_3. Otherwise, if $\boldsymbol{B} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$, then $\mathcal{A}$ has simulated H_4, where $\widehat{\boldsymbol{U}_{\boldsymbol{t}^\dagger}^{x^\dagger}} = \boldsymbol{B}$.

As a consequence, this implies that we can bound the advantage of $\mathcal{D}$ by the advantage of $\mathcal{A}$ against $\mathsf{NULWE}_{q,n,m,m,\chi}$. Furthermore, by the reduction in Lemma 3.12 we can further bound this advantage by $\mathsf{LWE}_{q,n,m,\chi}$ and the proof of the claim is complete. □

Claim 4.4.6. $\max_{\mathcal{D}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{D}}^{\mathsf{H}_4,\mathsf{H}_5}(1^\lambda))) = 0$.

Proof. In H_4, the output of $\mathcal{O}_{\mathcal{X}}(\mathsf{CPRF.Eval}(\mathsf{msk}, \mathsf{pp}, x^\dagger),)$ takes the form:

$$\left\lfloor \widehat{\boldsymbol{U}_{\boldsymbol{t}^\dagger}^{x^\dagger}} + \left(\sum_{\substack{\boldsymbol{t}\in\mathbb{T}\\ \boldsymbol{t}\neq\boldsymbol{t}^\dagger}} \boldsymbol{Z}_{\boldsymbol{t}}^{x^\dagger} \boldsymbol{D}_{\boldsymbol{t}}^{\mathsf{end}} \right) \right\rceil_p ,$$

where $\widehat{\boldsymbol{U}^{x^\dagger}_{\boldsymbol{t}^\dagger}} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$ is unknown to the adversary. Therefore this sum is distributed identically to an output of the form:

$$\left\lfloor \widehat{\boldsymbol{U}^{x^\dagger}} \right\rceil_p ,$$

for $\widehat{\boldsymbol{U}^{x^\dagger}} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$. It is important to note that $\widehat{\boldsymbol{U}^{x^\dagger}}$ is sampled only for the challenge input $x^\dagger$, and thus is uniformly distributed with respect to other evaluation queries. This means that all other queries can be simulated independently of $x^\dagger$. Thus H_4 and H_5 are perfectly indistinguishable. □

Claim 4.4.7. $\max_{\mathcal{D}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{D}}^{\mathsf{H}_{6.(\iota[\mathrm{u}])},\mathsf{H}_{6.(\iota[\mathrm{u}-1])}}(1^\lambda))) < \mathsf{negl}(\lambda)$

Proof. The proof of this claim is essentially the reverse statement from Claim 4.4.2. Let ν be the first bit where $x^{(u)}$ deviates from the inputs $(x^\dagger, x^{(1)}, \dots, x^{(u-1)})$.

Let $\mathcal{A}$ be an adversary attempting to distinguish the samples in Corollary 3.7 for the interval $[\nu+1, \ell]$. $\mathcal{A}$ constructs the public parameters using the distribution that it receives, in the same way as Claim 4.4.2. Notice that the output for the PRF on $x^\dagger$ can be sampled uniformly by the previous hybrid. The rest of the simulation (for input and constraint queries) are carried out just using the simulated public parameters.

Then if $\mathcal{A}$ has access to the distribution of Equation (2), where trapdoor sampling is used, this corresponds to $\mathsf{H}_{6.(\iota[\mathrm{u}])}$. If it receives Equation (1), then it has simulated $\mathsf{H}_{6.(\iota[\mathrm{u}-1])}$. Since $\mathcal{A}$ has negligible statistical advantage in Corollary 3.7, then $\mathcal{D}$ must also have advantage bounded by the same amount. □

Claim 4.4.8. $\max_{\mathcal{D}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{D}}^{\mathsf{H}_{6.(\iota[0])},\mathsf{H}_{6.(\iota-1[\mathrm{Q}])}}(1^\lambda))) < \mathsf{negl}(\lambda)$

Proof. This argument follows the same structure as Claim 4.4.7, except focusing only on the input $x^\dagger$. The simulation is identical and so we point the reader to the previous proof for the details. □

In $\mathsf{H}_{6.(0[Q])}$ we have a CPRF scheme that outputs uniform values $\lfloor \widehat{\boldsymbol{U}^{x^\dagger}} \rceil_p$ on the challenge input $x^\dagger$, but where the rest of the simulation is identical to the actual construction. This is due to the fact that all trapdoors have been removed from the public parameters, and so all constraint and input queries are answered using the real construction. This is identical to the situation that $\mathcal{D}$ witnesses in $\mathsf{exp}_{1,\mathcal{D}}^{\mathsf{cprf}}(1^\lambda, 1^r)$.

Therefore, $\mathsf{exp}_{0,\mathcal{D}}^{\mathsf{cprf}}(1^\lambda, 1^r)$ and $\mathsf{exp}_{1,\mathcal{D}}^{\mathsf{cprf}}(1^\lambda, 1^r)$ are computationally indistinguishable, under the above claims. We can conclude that CPRF is an r-key secure CPRF against selective queries and the proof of Lemma 4.4 is complete. □

Assuming that $\max_{\mathcal{A}}(\mathsf{Adv}(\mathsf{exp}_{b,\mathcal{A}}^{\mathsf{cprf}}(1^\lambda, 1^r))) = \epsilon$ for all PPT adversaries $\mathcal{A}$ in the selective security model, then there is an adversary $\mathcal{B}$ that has advantage $(1/\mathsf{poly}(\lambda))\epsilon$ in succeeding in $\mathsf{exp}_{c,\mathcal{B}}^{\mathsf{cprf}}(1^\lambda, 1^r)$ using adaptive queries.

Lemma 4.5. (Adaptive security) *Let $\mathcal{B}$ be a PPT adversary attempting to distinguish* $\mathsf{exp}_{c,\mathcal{B}}^{\mathsf{cprf}}(1^\lambda, 1^r)$ *for $c \in \{0,1\}$ in the adaptive security model. Then, if $\mathcal{A}$ is an adversary that distinguishes* $\mathsf{exp}_{b,\mathcal{A}}^{\mathsf{cprf}}(1^\lambda, 1^r)$ *with advantage ϵ in the selective security model, then $\mathcal{B}$ can succeed with advantage* $(1/\mathsf{poly}(\lambda))\epsilon$.

Proof. The proof of Lemma 4.4 hinges on a specific choice of matrix $\boldsymbol{D}^G_{\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}}$, that is inferred by the pair $(\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger})$, for $x^\dagger_{\boldsymbol{t}^\dagger} \leftarrow \mathsf{reindex}(x_{\boldsymbol{t}^\dagger})$. We show that $\mathcal{B}$ can run $\mathcal{A}$ as a subroutine via a polynomial-time reduction to obtain advantage $(1/\mathsf{poly}(\lambda))\epsilon$ advantage in the adaptive query model.

At first $\mathcal{B}$ obtains the output of $\mathsf{CPRF.Setup}$ and guesses a pair $(\boldsymbol{t}^\dagger, \boldsymbol{b}^\dagger)$ for $\boldsymbol{b}^\dagger \in \{0,1\}^z$, where $z \leftarrow \mathsf{unique}(\boldsymbol{t}^\dagger)$. Then, $\mathcal{B}$ receives the selective queries of $\mathcal{A}$: constraint queries $(v^{(1)}, \dots, v^{(r)})$ for $r = O(1)$, and input queries $(x^{(1)}, \dots, x^{(Q)})$ for $Q = \mathsf{poly}(\lambda)$.

– If $\mathcal{A}$ asks a query $v^{(i)} \in \{0,1,*\}^\ell$ whereby $(\boldsymbol{t}^\dagger, \boldsymbol{b}^\dagger) \in \Gamma_{v^{(i)}}$, then $\mathcal{B}$ aborts the reduction.
– For the challenge query $x^\dagger$, if $(\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}) \neq (\boldsymbol{t}^\dagger, \boldsymbol{b}^\dagger)$, then $\mathcal{B}$ aborts the security game.
– Otherwise, answer all the queries by sending them to the challenger in $\mathsf{exp}^{\mathsf{cprf}}_{c,\mathcal{B}}(1^\lambda, 1^r)$ and returning the output to $\mathcal{A}$.

If the game is not aborted, then this is identical to the game $\mathsf{exp}^{\mathsf{cprf}}_{b,\mathcal{A}}(1^\lambda, 1^r)$ witnessed by $\mathcal{A}$. To ensure that the reduction incurs only a polynomial security loss, we have to ensure that the probability of aborting is $1/\mathsf{poly}(\lambda)$.

There are $\sum_{j=1}^{r} 2^j \binom{\ell}{j}$ possible pairs $(\boldsymbol{t}, \boldsymbol{b})$. Since $(\boldsymbol{t}^\dagger, \boldsymbol{b}^\dagger)$ is chosen uniformly by $\mathcal{B}$, then the minimum probability that $(\boldsymbol{t}^\dagger, \boldsymbol{b}^\dagger) \notin \Gamma_{v^{(i)}}$, for each $i \in [r]$, is $1/(\sum_{j=1}^{r} 2^j \binom{\ell}{j})$ which is $1/\mathsf{poly}(\lambda)$. This follows by the fact that $r = O(1)$, and so the denominator is $\mathsf{poly}(\lambda)$ by the fact that $\ell = \mathsf{poly}(\lambda)$.

If the above is satisfied, then there exists a pair satisfying $(\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}) \notin \Gamma_{v^{(i)}}$ for each $i \in [r]$. Note that the set of viable pairs is polynomial in size and bounded above by $\sum_{j=1}^{r} 2^j \binom{\ell}{j}$. Therefore, the probability of $(\boldsymbol{t}^\dagger, x^\dagger_{\boldsymbol{t}^\dagger}) = (\boldsymbol{t}^\dagger, \boldsymbol{b}^\dagger)$ occurring, for the originally chosen pair $(\boldsymbol{t}^\dagger, \boldsymbol{b}^\dagger)$ is $\geq 1/(\sum_{j=1}^{r} 2^j \binom{\ell}{j})$. Which is identical to the above.

Therefore, the probability that $\mathcal{B}$ does *not* abort is $\geq (1/(\sum_{j=1}^{r} 2^j \binom{\ell}{j}))^2 = 1/\mathsf{poly}(\lambda)$. Therefore, the probability that $\mathcal{B}$ succeeds is identical to $(1/\mathsf{poly}(\lambda)) \cdot \mathsf{Adv}(\mathsf{exp}^{\mathsf{cprf}}_{b,\mathcal{A}}(1^\lambda, 1^r)) = (1/\mathsf{poly}(\lambda))\epsilon$ by the statement of the claim, and we are done. □

Lemma 4.6. (1-key privacy) *Construction 4.1 is a 1-key private CPRF.*

Proof. The proof of this theorem follows from the fact that a constrained key $G^{v^{(c)}}$ contains only $(2\ell)^r$ matrices sampled from $\chi^{zm \times m}$. For any two constraints $v^{(0)}, v^{(1)} \leftarrow \mathcal{A}(1^\lambda, 1^\ell)$ where $v^{(c)} \in \{0,1,*\}^\ell$, then the challenger returns $\mathsf{CK}_{v^{(c)}} = G^{v^{(c)}}$. The keys $\mathsf{CK}_{v^{(c)}}$ are distributed identically for $c \in \{0,1\}$ and thus $\mathcal{A}$ cannot distinguish which constrained key has been returned. □

By the results of Lemma 4.4 and Lemma 4.6, the statement of Theorem 4.3 follows immediately. □

References

Ajt99. Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 99*, volume 1644 of *LNCS*, pages 1–9. Springer, Heidelberg, July 1999.

AMN+18. Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained PRFs for NC^1 in traditional groups. In Shacham and Boldyreva [SB18], pages 543–574.

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.

BLMR13. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013.

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.

BP14. Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 353–370. Springer, Heidelberg, August 2014.

BPR12. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In Pointcheval and Johansson [PJ12], pages 719–737.

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.

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 Dodis and Nielsen [DN15], pages 1–30.

BVWW16. Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring LWE. In Madhu Sudan, editor, *ITCS 2016*, pages 147–156. ACM, January 2016.

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.

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, April / May 2017.

CVW18. Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Shacham and Boldyreva [SB18], pages 577–607.

DN15. Yevgeniy Dodis and Jesper Buus Nielsen, editors. *TCC 2015, Part II*, volume 9015 of *LNCS*. Springer, Heidelberg, March 2015.

GGH15. Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Dodis and Nielsen [DN15], pages 498–527.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

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`.

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.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval and Johansson [PJ12], pages 700–718.

MR04. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

NR97. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudorandom functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.

Pei09. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009.

PJ12. David Pointcheval and Thomas Johansson, editors. *EUROCRYPT 2012*, volume 7237 of *LNCS*. Springer, Heidelberg, April 2012.

PR06. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 145–166. Springer, Heidelberg, March 2006.

PS18. Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 675–701. Springer, Heidelberg, March 2018.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

SB18. Hovav Shacham and Alexandra Boldyreva, editors. *CRYPTO 2018, Part II*, volume 10992 of *LNCS*. Springer, Heidelberg, August 2018.