Efficient Range-Trapdoor Functions and Applications: Rate-1 OT and More*

Sanjam Garg† Mohammad Hajiabadi‡ Rafail Ostrovsky§

Abstract

Substantial work on trapdoor functions (TDFs) has led to many powerful notions and applications. However, despite tremendous work and progress, all known constructions have prohibitively large public keys.

In this work, we introduce new techniques for realizing so-called range-trapdoor hash functions with short public keys. This notion, introduced by Döttling et al. [Crypto 2019], allows for encoding a range of indices into a public key in a way that the public key leaks no information about the range, yet an associated trapdoor enables recovery of the corresponding input part.

We give constructions of range-trapdoor hash functions, where for a given range I the public key consists of $O(n)$ group elements, improving upon $O(n|I|)$ achieved by Döttling et al. Moreover, by designing our evaluation algorithm in a special way involving Toeplitz matrix multiplication and by showing how to perform fast-Fourier transforms in the exponent, we arrive at $O(n \log n)$ group operations for evaluation, improving upon $O(n^2)$, required of previous constructions. Our constructions rely on power-DDH assumptions in pairing-free groups.

As applications of our results we obtain

1. The first construction of (rate-1) lossy TDFs with public keys consisting of a linear number of group elements (without pairings).
2. Rate-1 string OT with receiver communication complexity of $O(n)$ group elements, where n is the sender's message size, improving upon $O(n^2)$ [Crypto 2019]. This leads to a similar result in the context of private-information retrieval (PIR).
3. Semi-compact homomorphic encryption for branching programs: A construction of homomorphic encryption for branching programs, with ciphertexts consisting of $O(\lambda nd)$ group elements, improving upon $O(\lambda^2 nd)$. Here λ denotes the security parameter, n the input size and d the depth of the program.

1 Introduction

Trapdoor cryptosystems are at the heart of modern cryptography. What is common among all these cryptosystems is the notion of a trapdoor key, which allows a certain computation to be inverted. The exact formulation of what inversion means specifies the strength of the notion.

*Research supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, AFOSR Award FA9550-19-1-0200, AFOSR YIP Award, NSF CNS Award 1936826, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies

†University of California, Berkeley

‡University of California, Berkeley

§University of California, Los Angeles

For example, trapdoor functions (TDFs) extend the functionality of public-key encryption (PKE) by allowing the inversion algorithm to recover the entire input. This seemingly slight extension makes the notion relatively versatile, enabling applications (from variants of TDFs) including CCA2-secure PKE, selective-opening security and designated-verifier non-interactive (NIZK) [PW08, BFOR08, BHY09, LQR+19], which are currently out of reach of the basic PKE primitives.

Perhaps not surprisingly, trapdoor systems that demand a richer functionality are harder to realize, and in cases this is possible, the resulting realizations come with poor efficiency. For instance, while for PKE we have a plethora of instantiations with close to optimal public-key, secret-key and ciphertext sizes, the situation for TDFs is much different. Concretely, the public keys of all DDH-based TDFs consist of $O(n^2)$ group elements, where n is the input size, lagging behind their PKE counterparts, which consist of a constant number of group elements. Although recent works [GH18, GGH19, DGI+19] showed how to make the image size of TDFs almost the same as the input size, they too are stuck with the $O(n^2)$ group elements overhead for the public key. As we will see later, this is due to a lack of *batching techniques* for TDF keys. Our goal, in this work, is to develop techniques that help us mitigate this issue. We will do this in a way general enough to be applicable not just to TDFs, but also to more advanced primitives, such as lossy TDFs [PW08] and trapdoor hash functions [DGI+19].

Trapdoor Hash (TDH) functions. Recently, Döttling, Garg, Ishai, Malavolta, Mour and Ostrovsky introduced a primitive, called trapdoor hash functions [DGI+19], and showed strong applications of this notion, including lossy TDFs, rate-1 oblivious transfer (OT), private information retrieval (PIR) with low communication complexity and more. In its simplest form, a TDH scheme comes with a length-compressing hash function $\mathsf{H}_{\mathsf{hk}}\colon \{0,1\}^n \to \{0,1\}^\lambda$ and an evaluation algorithm E. The scheme allows one to generate an evaluation/trapdoor key $(\mathsf{ik}_i, \mathsf{tk}_i)$ for any particular index $i \in [n]$ in such a way that (1) the output of $\mathsf{E}(\mathsf{ik}_i, \mathsf{x})$ is a single bit, (2) using tk_i, one may retrieve the value of x_i from $\mathsf{H}(\mathsf{hk}, \mathsf{x}) \in \{0,1\}^\lambda$ and $\mathsf{E}(\mathsf{ik}_i, \mathsf{x}) \in \{0,1\}$ and (3) ik_i hides the index i.

Usefulness of trapdoor hash. To show the utility of this notion, let us sketch a construction of lossy TDFs using this primitive, given by [DGI+19]. Consider a sequence of TDH-evaluation keys $\mathsf{ik}_1, \ldots, \mathsf{ik}_{n/2}$ generated for the range of indices $[1, n/2]$ and suppose we additionally include a message $\mathsf{x}^* \xleftarrow{\$} \{0,1\}^{n/2}$ as part of the public key. Assume the input x to the lossy TDF has $n/2$ bits. To evaluate x, form a bigger string $\mathsf{x}' := (\mathsf{x}||\mathsf{x}^*) \in \{0,1\}^n$ and return $(\mathsf{H}(\mathsf{hk}, \mathsf{x}'), \mathsf{E}(\mathsf{ik}_1, \mathsf{x}'), \ldots, \mathsf{E}(\mathsf{ik}_{n/2}, \mathsf{x}'))$.[1] Using the trapdoor keys of $\mathsf{ik}_1, \ldots, \mathsf{ik}_{n/2}$, we may recover x. Now if we switch the evaluation keys to $\mathsf{ik}_{n/2+1}, \ldots, \mathsf{ik}_n$ corresponding to the second-half range of indices, then we will statistically lose information about x. The reason is that $n/2 - \lambda$ bits of information are lost about x.

Rate-1 two-round oblivious transfer (OT): Another important application of trapdoor hash is in realizing rate-1 two-round OT protocols [DGI+19]. We say that an OT protocol achieves rate-1 if the ratio $|\mathsf{m}_0|/|\mathsf{ots}|$ asymptotically approaches one, where ots is the sender's protocol message on a pair of inputs $(\mathsf{m}_0 \in \{0,1\}^n, \mathsf{m}_1 \in \{0,1\}^n)$ and on the corresponding message otr of the receiver. As shown by Ishai and Paskin [IP07], rate-1 OT leads to constructions of semi-compact homomorphic

[1] Here for simplicity we assume that E is deterministic and that each trapdoor enables perfect recovery of the underlying indexed bit. Under the actual definition, the function E should be randomized, so as to provide the desired privacy guarantees, needed by OT, etc applications. This issue can be handled by using a fixed randomness for the sketched construction.

encryption for branching programs (where the ciphertext size grows only with the depth as opposed to the size of the program) as well as communication-efficient private-information retrieval (PIR) protocols. All these applications rely on the rate-1 property of the OT in a crucial way, allowing one to sequentially pass ots as an input to a new OT-sender's message and pass the resulting ots to the next sender's message and so on. Trapdoor hash schemes provide an elegant way for realizing rate-1 OT [DGI+19]. Specifically, if the size of each message of the sender is n, the receiver on an input bit b sends n evaluations key $\mathsf{ek}_1, \ldots, \mathsf{ek}_n$ corresponding to either indices in $[1, n]$ or $[n+1, \ldots, 2n]$. The sender then returns $(\mathsf{H}(\mathsf{hk}, \mathsf{m}_0||\mathsf{m}_1)), \mathsf{E}(\mathsf{ek}_1, \mathsf{m}_0||\mathsf{m}_1), \ldots, \mathsf{E}(\mathsf{ek}_n, \mathsf{m}_0||\mathsf{m}_1)$. The receiver may then use his trapdoors to recover the corresponding message.[2] We have $|\mathsf{ots}| = n + \mathsf{poly}(\lambda)$, where poly is a fixed function, and hence the protocol has rate-1 (asymptotically). Döttling et al. [DGI+19] used the above protocol to get the first constructions of rate-1 OT from DDH, OR and LWE.

Lack of batching techniques for evaluation keys. In the applications above, the public key of the lossy TDF or the receiver's message in the OT protocol each consists of $O(n)$ TDH-evaluation keys. Under DDH instantiations of TDH [DGI+19], an evaluation key for any given index has $O(n)$ group elements, resulting in $O(n^2)$ group elements for the whole range, an overhead alluded to earlier. Moreover, lack of batching methods affects similarly the other applications: the ciphertext size in the case of homomorphic encryption for branching programs, and the client's message size in the case of PIR. While bilinear maps may open up venues for batching-style tricks [BW10, DGI+19], it is not clear how to do so without pairings. (See Section 1.3 for more details.)

Obtaining linear-sized public keys asymptotically. We note that if one's goal is solely to obtain TDFs with public-key size linear in input size, that is easy to do by making the input larger; e.g., $\mathsf{TDF}(\mathsf{ik}, \mathsf{x}_1||\ldots||\mathsf{x}_n) = \mathsf{TDF}(\mathsf{ik}, \mathsf{x}_1)||\cdots||\mathsf{TDF}(\mathsf{ik}_n, \mathsf{x}_n)$. Similarly, one may make the size of the receiver's message otr in an OT protocol almost the same as that of the sender's input, by making each of the sender's input consist of (sufficiently) many blocks of messages and re-using otr across each opposite pair of them. These results are only for the asymptotic case, falling short in concrete cases. For example, increasing the size of the sender's input messages — in order to make the size of otr close to that of the sender's message — translates into larger homomorphically-evaluated ciphertexts in the case of homomorphic encryption for branching programs.

1.1 Our Results

In this work, we will mitigate the above-mentioned issue, through efficient realizations of a new notion of range-trapdoor hash, which we introduce next.

Range-Trapdoor Hash. We introduce a notion called range-trapdoor hash functions, which is an immediate generalization of TDH schemes for index functions. In particular, under range-trapdoor hash, one would issue evaluation keys ek_I (and some public parameter s) for a range of indices $I = [i+1, ..., i+s]$, in such a way that given ek_I's trapdoor key, one can recover $\mathsf{x}[I] := (\mathsf{x}_{i+1}, \ldots, \mathsf{x}_{i+s})$ from $\mathsf{H}(\mathsf{hk}, \mathsf{x})$ and $\mathsf{E}(\mathsf{ek}_I, \mathsf{x})$. We require that ek_I should hide I (except for $|I|$) and that $|\mathsf{E}(\mathsf{ek}_I, \mathsf{x})| = |I|$. Under Diffie-Hellman type assumptions, we seek realizations where ek_I consist only of $O(n)$ group elements, as opposed to $O(n|I|)$.

[2] Again, we are giving an over-simplified construction, by assuming that decryption has perfect correctness. Moreover, in the actual construction, the function H should be randomized, so to provide sender privacy.

Our construction. We give constructions of range-trapdoor hash schemes, where on inputs of length n, an encoding key for a given range $I \subseteq [n]$ consists of $O(n)$ group elements. Our construction relies on the $2n$-power DDH assumption — namely, that the distribution $(g, g^a, g^{a^2}, \ldots, g^{a^{2n}})$ should be pseudorandom, where g is a random generator of the group and a is a uniformly-random exponent. This notion has been used in some previous works, e.g., [CNs07, AHI11, BMZ19], but for different purposes.

In addition to obtaining a smaller ek_I, we obtain efficiency improvements in the computation time of the evaluation algorithm. Specifically, while the evaluation algorithm of [DGI+19] requires $O(n|I|)$ group operations (among some other private-key operations), the number of public-key operations in our construction is only $O(n \log |I|)$. At a high level, we achieve this by designing our range-trapdoor hash scheme in a structured way, so that the evaluation involves multiplying a Toeplitz matrix (given in the exponent) with an input vector x^T. Since Toeplitz matrices are closely related to circulant matrices which are amenable to the fast-Fourier transform, we show how to do this matrix multiplication in a fast way using (inverse) discrete Fourier transform (IDFT/DFT) modulo $\mathbb{Z}_p$ in the exponent.

Applications: Rate-1 Two-Message String OT and More. Our techniques yield a construction of string OT with rate-1 from the power-DDH assumption with improved communication and computation. Specifically, in our two round protocol the communication from receiver to sender consists of a linear (in sender's message size) number of group elements. The previous work of [DGI+19] required a quadratic number of group elements by relying on DDH. Additionally, our construction also improves the computational cost of the sender — namely, our construction improves the computational effort of the sender from quadratic to quasi-linear. This allows us to obtain the following new results:

1. *Lossy Trapdoor Functions:* We obtain the first construction of lossy trapdoor functions [PW08], where on inputs of size n, the public key consists of $O(n)$ group elements. All previous (even non-lossy) TDF constructions from pairing-free groups had public keys with $O(n^2)$ group elements.

2. *Semi-Compact Homomorphic Encryption for Branching Programs:* A construction of public-key homomorphic encryption for branching programs, with ciphertexts consisting of $O(\lambda nd)$ group elements, improving upon $O(\lambda^2 nd)$ [DGI+19], where d denotes the depth of the program. We achieve this by plugging our rate-1 OT scheme into the homomorphic encryption construction of [IP07].

3. *Private Information Retrieval*: For a database of N bits, we get a two-message PIR protocol with total communication complexity that grows only polylogarithmically with the database size, and with a client's message consisting of $O(\lambda \log N)$ group elements, improving upon $O(\lambda^2 \log N)$, given by [DGI+19].

1.2 Related Work and Open Problems

As mentioned above, Döttling et al. [DGI+19] introduced the notion of trapdoor hash, and used it to build several new primitives. Among others, they obtained the first DDH-based and QR-based constructions of PIR for one-bit records with a total communication complexity that grows polylogarithmically with the database size; i.e., it is $\mathsf{p}(\lambda)\mathsf{polylog}(N))$ for a fixed function p, where

N is the database size and λ is the security parameter. Previously, such protocols were only known under DCR, LWE and Φ-hiding assumptions [CMS99, Cha04, Lip05, OS07].

The notion of trapdoor hash builds on tools that were developed in the context of trapdoor function constructions [GH18, GGH19], as well as those developed in the context of identity-based encryption (IBE) [DG17b, DG17a, BLSV18, DGHM18].

Variants of TDFs are typically used as CCA-enhancing tools [PW08, RS09, GH18, GGH19]. Koppula and Waters [KW19] showed that for CCA applications, full randomness recovery, a feature provided by TDF-based tools, is not necessary. They gave a generic transformation from CPA to CCA for PKE and attribute-based encryption (ABE) using hinting pseudorandom generators (PRGs). The notion of hinting PRGs was later used in subsequent works in contexts such as designated-verifier NIZK [LQR+19] and CCA key-dependent-message (KDM) security [KMT19]. Boyen and Waters show that in the bilinear setting one may shorten the public key the Peikert-Waters lossy-TDF construction from a quadratic number of group elements to linear [BW10].

Open problems. The main open problem is to achieve the same results from DDH.

1.3 Technical Overview

It will be instructive to give an overview of our results in the context of lossy TDFs and then to adapt them to the trapdoor-hash setting. Let us review an optimized version of the DDH-based lossy TDF of [PW08], given by [FGK+10]. Recall that in a group with a generator g, if we have an encoding $[\mathbf{M}] = g^{\mathbf{M}}$ of an invertible matrix $\mathbf{M}$ of exponents, we may then encode any column vector X of bits by computing $\mathbf{M} \cdot \mathsf{X}$ in the exponent. One may invert using $\mathbf{M}^{-1}$. We may argue lossiness by making the matrix $\mathbf{M}$ rank one. The downside of this scheme is that a public key and an image point consist of, respectively, n^2 and n group elements, which is rather large. Recent works [GH18, GGH19], which in turn inspired the notion of TDH, showed how to make the image size linear in input size, but they still leave us with public keys of $O(n^2)$ group elements.

Parallels from Ideal Lattices? To make the public keys smaller, one may be tempted to draw inspirations from ideal lattices [LPR10, LPR13], and especially the way ring-LWE is used to shorten public keys. Sample a vector $\mathbf{v} := (g_1, \ldots, g_n)$ and expand $\mathbf{v}$ into a "circulant-like" matrix

$$\mathbf{M} := \begin{pmatrix} g_1 & g_2 & \cdots & g_{n-1} & g_n \\ g_2 & g_3 & \cdots & g_n & g_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_n & g_1 & \cdots & g_{n-2} & g_{n-1} \end{pmatrix}, \tag{1}$$

and use $\mathbf{M}$ as the public key of the TDF given above. The problem with this approach is that we do not know how to prove one-wayness. Even if there is a clever way to prove one-wayness, this approach does not appear to scale to give us more advanced schemes such as lossy TDFs, (range) trapdoor hash schemes, or TDFs with linear-sized outputs.

Circulant structure using power DDH. We show how to work out the above intuition by relying on the power DDH assumption. Specifically, we give a way of expanding two vectors $(\mathbf{v} \in \mathbb{G}^n, \mathbf{w} \in \mathbb{G}^{2n-1})$ into an $(n+1) \times n$ matrix, and two indistinguishable distributions on $(\mathbf{v}, \mathbf{w})$, where under one distribution we can invert, while under the other, we will lose information.

Given two vectors $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{G}^n$ and $\mathbf{w} = (w_1, \ldots, w_{2n-1}) \in \mathbb{G}^{2n-1}$, we expand them into an $(n+1) \times n$ matrix $\mathbf{M} = \mathsf{Expand}(\mathbf{v}, \mathbf{w})$ as follows:

$$\mathbf{M} := \begin{pmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_{n+1} \end{pmatrix} := \begin{pmatrix} v_1 & v_2 & \ldots & v_n \\ w_1 & w_2 & \ldots & w_n \\ w_2 & w_3 & \ldots & w_{n+1} \\ \vdots & \vdots & \ldots & \vdots \\ w_n & w_{n+1} & \ldots & w_{2n-1} \end{pmatrix} \in \mathbb{G}^{(n+1)\times n} \tag{2}$$

To evaluate an input $\mathsf{x} \in \{0,1\}^n$ using $\mathbf{M}$, return $(\mathsf{x} \cdot \mathbf{m}_1, \ldots, \mathsf{x} \cdot \mathbf{m}_{n+1})$, where $\mathsf{x} \cdot \mathbf{v} := \prod_{i=1}^{n} v_i^{\mathsf{x}_i}$. Define the lossy distribution lossy as

$$\begin{aligned} \mathsf{lossy} \quad \mathbf{v} &:= (g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}) \\ \mathbf{w} &:= (g^{r\alpha}, g^{r\alpha^2}, \ldots, g^{r\alpha^{2n-1}}). \end{aligned}$$

If $(\mathbf{v}, \mathbf{w}) \xleftarrow{\$} \mathsf{lossy}$, then $\mathbf{M} := \mathsf{Expand}(\mathbf{v}, \mathbf{w})$ will be of rank one, statistically losing information about x. We set the real (i.e., injective) distribution by putting a *bump* g on the nth element of $\mathbf{w}$:

$$\mathsf{real} \quad \mathbf{v} := (g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}) \tag{3}$$

$$\mathbf{w} := (g^{r\alpha}, g^{r\alpha^2}, \ldots, g^{r\alpha^{n-1}}, \mathbf{g}g^{r\alpha^n}, g^{r\alpha^{n+1}}, \ldots, g^{r\alpha^{2n-1}}). \tag{4}$$

To see how to invert in injective mode, notice that the matrix $\mathbf{M} := \mathsf{Expand}(\mathbf{v}, \mathbf{w})$ is

$$\mathbf{M} := \begin{pmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_{n+1} \end{pmatrix} = \begin{pmatrix} g^{\alpha} & g^{\alpha^2} & \ldots & g^{\alpha^{n-1}} & g^{\alpha^n} \\ g^{r\alpha} & g^{r\alpha^2} & \ldots & g^{r\alpha^{n-1}} & \mathbf{g}g^{r\alpha^n} \\ g^{r\alpha^2} & g^{r\alpha^3} & \ldots & \mathbf{g}g^{r\alpha^n} & g^{r\alpha^{n+1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g^{r\alpha^{n-1}} & \mathbf{g}g^{r\alpha^n} & \ldots & g^{r\alpha^{2n-3}} & g^{r\alpha^{2n-2}} \\ \mathbf{g}g^{r\alpha^n} & g^{r\alpha^{n+1}} & \ldots & g^{r\alpha^{2n-2}} & g^{r\alpha^{2n-1}} \end{pmatrix} \in \mathbb{G}^{(n+1)\times n}, \tag{5}$$

where the bump $\mathbf{g}$ propagates as indicated. Using the trapdoor values α and r, we show how to recover the ith bit of x from the image $\mathsf{u} := (g_h, g_1, \ldots, g_n) := (\mathsf{x} \cdot \mathbf{m}_1, \ldots, \mathsf{x} \cdot \mathbf{m}_{n+1})$. To do this, notice that the bump that affects the ith bit of x occurs in row $(n-i+2)$ of matrix $\mathbf{M}$, which is off the first row by an exponent $r\alpha^{n-i}$ (excluding the bump). Moreover, the group element from the image u that carries information about x_i is g_{n-i+1}. Thus, we may compute $g^{\mathsf{x}_i} \in \{g^0, g^1\}$ as

$$g^{\mathsf{x}_i} = \frac{g_{n-i+1}}{g_h^{r\alpha^{n-i}}} \in \{g^0, g^1\}. \tag{6}$$

Finally, the indistinguishability between lossy and real follows from $(2n-1)$-power DDH, which implies that the distribution $((g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{2n-1}}), (g^{r\alpha}, g^{r\alpha^2}, \ldots, g^{r\alpha^{2n-1}}))$ is pseudorandom: the pseudorandom of the first vector comes from $2n-1$-power DDH and the pseudorandomness of the second one is implied by the fact that t-power (for $t \geq 3$) implies DDH (Lemma 2.4).

Source of computational efficiency. Excluding the fist row of matrix $\mathbf{M}$, the rest of the matrix is Toeplitz-like, which, if given in the clear as opposed to in the exponent, can be multiplied with any given vector in time $O(n \log n)$ using discrete FFT techniques. We observe that this computation may in fact be carried out in the exponent, providing us with a relatively fast way of $O(n \log n)$ group exponentiations for evaluating an input. See Section 4 for mode details.

Making the image shorter. The public key of the above lossy TDF has $O(n)$ group elements, a goal we had set before. The image, however, is quite large, consisting of $n+1$ group elements. We now show how to use image-shrinking techniques of Garg, Gay and Hajiabadi [GGH19] (later improved by Döttling et al. [DGI+19]) in order to make the image size linear in input size. Looking ahead, this will allow us to make $|\mathsf{E}(\mathsf{ek}_I, \mathsf{x})| = |I|$, where ek_I is the TDH-evaluation key for a range I. For concreteness, let us focus on how to recover the last bit x_n from a succinct output. If the corresponding (long) image of x is $\mathsf{u} := (g_h, g_1, \ldots, g_n)$, then for recovering x_n we have to look at g_h and g_1: we either have $g_1 = g_h^r$, in which case $\mathsf{x}_n = 0$, or $g_1 = g g_h^r$, in which case $\mathsf{x}_n = 1$ (or informally, x_n has hit the bump). Now instead of outputting one whole group element g_1, we output a single bit, corresponding to the output of a hint function $\Phi_k : \mathbb{G} \to \{0,1\}$ on g_1. This function guarantees that for any $g^* \in \mathbb{G}$, the probability that $\Phi_k(g^*) = \Phi_k(g^* g)$ (a.k.a., the *hung* probability) is very small, where k is chosen at random (and included in the public key). The inverter will then match $\Phi_k(g_1)$, comes as part of the image, against $\Phi(g_h^r)$ and $\Phi(g_h^r g)$. Garg, Gay and Hajiabadi [GGH19] gave a function Φ which outputs a constant c number of bits (instead of a single bit) with hung probability being at most $\frac{1}{2^c}$. Later, Döttling et al. [DGI+19] substantially improved this by making Φ output a single bit with hung probability being at most $\frac{1}{n^c}$, for any desired constant c. They achieved this by using a PRF-based distance-function technique from [BGI16]. Finally, since the inversion algorithm may fail (i.e., be hung) for some indices, we pre-process the TDF input using erasure-correcting codes, making the task of decoding easier.

Adaptation to the trapdoor hash setting. The lossy TDF sketched above (without erasure-correcting codes) lends itself naturally into the range TDH setting. Recall that for range trapdoor hash, we encode an index range $I = [s+1, s+t]$ into an encoding key ek in such a way that (1) ek only reveals $|I|$ and (2) Using the associated trapdoors, one can recover each bit of $\mathsf{x}[I]$ with high probability from $\mathsf{H}(\mathsf{hk}, \mathsf{x})$ and $\mathsf{E}(\mathsf{ek}, \mathsf{x}) \in \{0,1\}^{|I|}$. Moreover, ek should only contain $O(n)$ group elements (as opposed to $O(n|I|)$).

We achieve range-trapdoor hash by carefully placing the bump in a coordinate which enables recovery of exactly $\mathsf{x}[I]$, and nothing more. First, let $\mathsf{hk} := \mathbf{v} := (g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^n})$ and define $\mathsf{H}(\mathsf{hk}, \mathsf{x}) = \mathsf{x} \cdot \mathbf{v}$. Assuming $I = [s+1, s+t]$ and noting that $|I| = t$, set $\mathsf{ek} := (\mathbf{w}, t)$, where

$$\mathbf{w} := (g^{r\alpha}, g^{r\alpha^2}, \ldots, g^{r\alpha^{s+t-1}}, g g^{r\alpha^{s+t}}, g^{r\alpha^{s+t+1}}, \ldots, g^{r\alpha^{2n-1}}), \tag{7}$$

obtained from hk by raising every element to the power of r and putting the bump g in the $(s+t)$'th coordinate. Now to evaluate x on $\mathsf{ek} := (\mathbf{w}, t+1)$, return

$$(\mathsf{x} \cdot \mathbf{w}[1, n], \mathsf{x} \cdot \mathbf{w}[2, n+1], \ldots, \mathsf{x} \cdot \mathbf{w}[t, n+t-1]) \in \mathbb{G}^n,$$

where $\mathbf{w}[i, j]$ denotes the elements of $\mathbf{w}$ which are in the range $\{i, i+1, \ldots, j\}$. Given α and r we may recover all the bits $\mathsf{x}[s, s+t]$. The only remaining thing is that the output of E consists of t group elements, as opposed to t bits. We make it consist of t bits by using image-shrinking techniques described above.

2 Preliminaries

Notation. We use λ for the security parameter. We use $\stackrel{c}{\equiv}$ to denote computational indistinguishability and use $\equiv$ to denote two distributions are identical. For a distribution $\mathcal{S}$ we use $x \stackrel{\$}{\leftarrow} \mathcal{S}$

to mean x is sampled according to $\mathcal{S}$ and use $y \in \mathcal{S}$ to mean $y \in \sup(\mathcal{S})$, where sup denotes the support of a distribution. For a set S we overload the notation to use $x \xleftarrow{\$} \mathsf{S}$ to indicate that x is chosen uniformly at random from S. If $\mathsf{A}(x_1, \ldots, x_n)$ is a randomized algorithm, then $\mathsf{A}(a_1, \ldots, a_n)$, for deterministic inputs $a_1, \ldots, a_n$, denotes the random variable obtained by sampling random coins r uniformly at random and returning $\mathsf{A}(a_1, \ldots, a_n; r)$. We use $[n] := \{1, \ldots, n\}$ and $[i, i+s] := \{i, i+1, \ldots, i+s\}$. For a vector $\mathbf{v} = (v_1, \ldots, v_n)$ we define $\mathbf{v}[i, i+s] := (v_i, v_{i+1}, \ldots, v_{i+s})$.

2.1 Standard Definitions and Lemmas

Definition 2.1 (Trapdoor functions (TDFs))**.** *Let $n = n(\lambda)$ be a polynomial. A family of trapdoor functions* TDF *with domain $\{0,1\}^n$ consists of three PPT algorithms* $\mathsf{TDF.KG}$, $\mathsf{TDF.F}$ *and* $\mathsf{TDF.F}^{-1}$ *with the following syntax and security properties.*

- $\mathsf{TDF.KG}(1^\lambda)$*: Takes the security parameter 1^λ and outputs a pair $(\mathsf{ik}, \mathsf{tk})$ of index/trapdoor keys.*
- $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})$*: Takes an index key ik and a domain element $\mathsf{x} \in \{0,1\}^n$ and deterministically outputs an image element u.*
- $\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{u})$*: Takes a trapdoor key tk and an image element u and outputs a value $\mathsf{x} \in \{0,1\}^n \cup \{\perp\}$.*

We require the following properties.

- ***Correctness:***

$$\Pr_{(\mathsf{ik},\mathsf{tk})}[\exists \mathsf{x} \in \{0,1\}^n \text{ s.t. } \mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})) \neq \mathsf{x}] = \mathsf{negl}(\lambda), \tag{8}$$

  where the probability is taken over $(\mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$.

- ***One-wayness:*** *For any PPT adversary $\mathcal{A}$: $\Pr[\mathcal{A}(\mathsf{ik}, \mathsf{u}) = \mathsf{x}] = \mathsf{negl}(\lambda)$, where $(\mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$, $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ and $\mathsf{u} := \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})$.*

Definition 2.2 (Lossy TDFs [PW08, PW11])**.** *An (n,k)-lossy TDF ((n,k)-LTDF) is given by four PPT algorithms* $\mathsf{TDF.KG}$, $\mathsf{TDF.KG}_{\mathsf{ls}}$, $\mathsf{TDF.F}$, $\mathsf{TDF.F}^{-1}$*, where $\mathsf{TDF.KG}_{\mathsf{ls}}(1^\lambda)$ only outputs a single key (as opposed to a pair of keys), and where the following properties hold:*

- ***Correctness in real mode.*** *The TDF $(\mathsf{TDF.KG}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ satisfies correctness in the sense of Definition 2.1.*
- ***k-Lossiness.*** *For all but negligible probability over the choice of $\mathsf{ik}_{\mathsf{ls}} \xleftarrow{\$} \mathsf{TDF.KG}_{\mathsf{ls}}(1^\lambda)$, we have $|\mathsf{TDF.F}(\mathsf{ik}_{\mathsf{ls}}, \{0,1\}^n)| \leq 2^k$, where we use $\mathsf{TDF.F}(\mathsf{ik}_{\mathsf{ls}}, \{0,1\}^n)$ to denote the set of all images of $\mathsf{TDF.F}(\mathsf{ik}_{\mathsf{ls}}, \cdot)$.*
- ***Indistinguishability of real and lossy modes.*** *We have $\mathsf{ik} \stackrel{c}{\equiv} \mathsf{ik}_{\mathsf{ls}}$, where $(\mathsf{ik}, *) \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$ and $\mathsf{ik}_{\mathsf{ls}} \xleftarrow{\$} \mathsf{TDF.KG}_{\mathsf{ls}}(1^\lambda)$.*

Lossiness rate. In the definition above, we refer to the fraction $1 - k/n$ as the *lossiness rate*, describing the fraction of the bits lost. Ideally, we want this fraction to be as close to 1 as possible, e.g., $1 - o(1)$.

Expansion rate. In the definition above, we refer to $n/|u|$ as the expansion rate, and say the scheme has *rate 1* if this fraction approaches one asymptotically.

2.2 Computational Assumptions

We now review the notion of power DDH assumption, which we use in our constructions.

Definition 2.3 (t-power DDH assumption [CNs07, AHI11])**.** *Let* G *be a group-generator scheme, which on input* 1^λ *outputs* $(\mathbb{G}, p, g)$*, where* $\mathbb{G}$ *is the description of a group,* p *is the order of the group which is always a prime number and* g *is a generator for the group. Let* $t := t(\lambda)$*. We say that* G *is* t*-DDH-hard if the distribution* $(g, g^\alpha, \ldots, g^{\alpha^t})$ *is pseudorandom, where* $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{G}(1^\lambda)$ *and* $\alpha \xleftarrow{\$} \mathbb{Z}_p$*.*

The following simple lemma shows a folklore result, that the power DDH assumption implies DDH hardness.

Lemma 2.4. *Let* G *be* t*-power DDH hard. Then* $(g_1, g_1^\alpha, \ldots, g_1^{\alpha^t})$ *is pseudorandom, where* $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{G}(1^\lambda)$*,* $g_1 \xleftarrow{\$} \mathbb{G}$ *and* $\alpha \xleftarrow{\$} \mathbb{Z}_p$*.*[3] *Also, for any* $t \geq 3$*, if a group is* t*-power DDH hard, it is also DDH-hard.*

Proof. The first part of the lemma follows straightforwardly using random self reducibility. For the second part, notice that if a group is $t+1$-power DDH hard, then it is also t-power DDH hard. Thus, it suffices to show that 3-power DDH hardness implies DDH hardness. Suppose for a group $\mathbb{G}$ there is a DDH adversary $\mathcal{A}$ that can distinguish (g, g^a, g^b, g^{ab}) from random. We want to use $\mathcal{A}$ to distinguish $(g, g^\alpha, g^{\alpha^2}, g^{\alpha^3})$ from random, hence breaking 3-power DDH hardness. The problem is that $\mathcal{A}$ is only guaranteed to work as long as the two exponents a and b are chosen uniformly at random — while in the 3-power DDH case the two exponents α and α^2 are correlated.

To fix the above problem, we use the random-self reducibility of DDH [NR97]. That is, letting (g, g_1, g_2, g_3) be the challenge tuple, we sample $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and call $\mathcal{A}$ on $(g, g_1^{r_1}, g_2^{r_2}, g_3^{r_1 r_2})$.

It is easy to see that the above transformation converts a 3-power DDH tuple into a random DDH tuple, and converts a random tuple into another random tuple. □

2.3 Standard Lemmas

Lemma 2.5 (Chernoff inequality)**.** *Let* X *be binomially distributed with parameters* $n \in \mathbb{N}$ *and* $p \in [0, 1]$*. Assuming* $p' > p$*:*

$$\Pr[X > 2p'n] < e^{-p'n/3}.$$

In some of our proofs, we need to use a version of Chernoff bounds involving Bernoulli variables which are not necessarily independent, but where each of them has a bounded probability of success, conditioned on any fixed sequence of outcomes of the others. We give such a version of the Chernoff inequality below, and prove it by relying on Lemma 2.5.

[3] Notice that the only difference between this version and the standard t-power DDH assumption is that the element g_1 is now also chosen uniformly at random — as opposed to it being g, the fixed group generator.

Lemma 2.6 (Chernoff inequality with bounded dependence)**.** *Let $X_1, \ldots, X_n$ be Bernoulli variables (not necessarily independent), where for all i, and for all values $b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n$:*

$$\Pr[X_i = 1 \mid X_1 = b_1, \ldots, X_{i-1} = b_{i-1}, X_{i+1} = b_{i+1}, \ldots, X_n = b_n] \leq p. \tag{9}$$

Assuming $p' > p$:

$$\Pr[\sum_{i \in [n]} X_i > 2p'n] < e^{-p'n/3}.$$

Proof. We will define n random variables $X'_1, \ldots, X'_n$ and also n independent i.i.d. boolean random variables $Y_1, \ldots, Y_n$, where $\Pr[Y_1] = p$, and where

1. $(X'_1, \ldots, X'_n)$ is identically distributed as $(X_1, \ldots, X_n)$; and
2. for all $i \in [n]$, $X'_i \leq Y_i$.

Thus

$$\Pr_{(X_1,\ldots,X_n)}[\sum_{i \in [n]} X_i > 2p'n] = \Pr_{(X'_1,\ldots,X'_n)}[\sum_{i \in [n]} X'_i > 2p'n] \leq \Pr[\sum_{i \in [n]} Y_i > 2p'n] < e^{-p'n/3},$$

where the last inequality comes from Lemma 2.5.

To define Y_i, let U_i for $i \in [n]$ be i.i.d. real-valued random variables, each uniformly distributed over $[0, 1]$. For $i \in [n]$ let Y_i be the Bernouli random variable where $Y_i = 1$ iff $U_i \leq p$.

For $b_1, \ldots, b_{i-1} \in \{0, 1\}$ define $\mathcal{Z} = \Pr[X_1]$ and

$$\mathcal{Z}(b_1, \ldots, b_{i-1}) = \Pr[X_i = 1 | X_1 = b_1, \ldots, X_{i-1} = b_{i-1}].$$

We may now represent the joint distribution $(X_1, \ldots, X_n)$ as

$$(X'_1, \ldots, X'_n) := (U_1 \leq \mathcal{Z}, U_2 \leq \mathcal{Z}(X_1), \ldots, U_n \leq \mathcal{Z}(X_1, \ldots, X_{n-1})), \tag{10}$$

where $A \leq B$ is the Bernoulli random variable which is one if and only if $A \leq B$.

We now show that whenever $U_i \leq \mathcal{Z}(X_1, \ldots, X_{i-1})$, we have $Y_i = 1$, as desired. To see this, recall that by Equation 9 $\mathcal{Z}(X_1, \ldots, X_{i-1}) \leq p$. Thus, whenever $U_i \leq \mathcal{Z}(X_1, \ldots, X_{i-1})$, we have $U_i \leq p$, which means $Y_i = 1$. The proof is now complete. □

2.4 Error Correcting Codes

Definition 2.7 ($(n, m, s)_2$-Codes)**.** *We recall the notion of $(n, m, s)_2$ erasure-correcting codes. Such a code is given by efficiently computable functions $(\mathsf{Encode}, \mathsf{Decode})$, where $\mathsf{Encode} : \{0,1\}^n \to \{0,1\}^m$, and where*

1. ***Minimum distance.*** *For any two distinct $\mathsf{x}_1, \mathsf{x}_2 \in \{0,1\}^n$, $\mathsf{H}_{\mathrm{dst}}(\mathsf{Encode}(\mathsf{x}_1), \mathsf{Encode}(\mathsf{x}_2)) \geq s$, where $\mathsf{H}_{\mathrm{dst}}$ denotes the Hamming distance.*
2. ***Erasure correction.*** *For any $\mathsf{x} \in \{0,1\}^n$, letting $\mathsf{z} := \mathsf{Encode}(\mathsf{x})$, given any string $\mathsf{z}' \in \{0, 1, \perp\}^m$, which has at most $s-1$ $\perp$ symbols, and whose all non-$\perp$ symbols agree with z, we have $\mathsf{Decode}(\mathsf{z}') = \mathsf{x}$.*

We are interested rate-1 codes (that is, n/m approaches 1 asymptotically) with fast encodable and decodable algorithms. If we are willing to settle for a constant rate (as opposed to rate 1), there are binary concatenated codes which are linear time for both encoding and decoding; see, e.g., [GI05], Theorem 6. For rate-1 binary codes, we use the following code from [CDD+16].

Theorem 2.8 ([CDD+16], Theorem 6)**.** *Fix a finite field $\mathbb{F}$ of constant size. There exists a constant $\upsilon > 0$ and a family of $\mathbb{F}$-linear codes $\mathrm{C} = \{\mathrm{C_s}\}_\mathrm{s}$ with codeword length $O(s^2)$, rate $1-\frac{1}{s^\upsilon}$ and minimum distance at least s. Moreover, C admits a linear-time computable encoding algorithm* Encode.

3 Lossy TDFs with Short Public Keys from Power DDH

As a warm-up to our range-trapdoor hash construction, we first give a construction of rate-1 lossy TDFs from $O(n)$-power DDH assumption, where a public key has only $O(n)$ group elements.

For our construction, we need a function $\Phi\colon \mathbb{G} \to \{0,1\}$ which has the property that for any group element h, $\Phi(h) \neq \Phi(hg)$ with high probability. The work of Boyle, Gilboa and Ishai [BGI16] gives such a function. Below we review an adaptation of this function to the binary output space, as done by [DGI+19]. In what follows, we use $\mathsf{LSB}(i)$ to denote the least significant bit of i.

Distance function $\mathsf{Dist}_{\mathbb{G},g}(h,\delta,M,f)$ [BGI16]. a group $\mathbb{G}$ with a generator g, a group element h, a value $0 < \delta < 1$, integer $M \geq 1$ and a function $f : \mathbb{G} \to \{0,1\}^{\log(2M/\delta)}$, we define a function Dist as follows:

1. Let $T := [2M \log_e(2/\delta)]/\delta$ and set $i := 0$.
2. While $i \leq T$:
   (a) if $f(hg^i) = 0^{\log(2M/\delta)}$, then output $\mathsf{LSB}(i)$, otherwise set $i = i + 1$.
3. Output $\mathsf{LSB}(i)$.

T-close/far group elements. For an integer T, we say two group elements g_1 and g_2 are *T-close* with respect to g if $g_2 \in \{g_1, g_1 g, \ldots, g_1 g^T\}$ or $g_1 \in \{g_2, g_2 g, \ldots, g_2 g^T\}$. We say g_1 and g_2 are at least *$(T+1)$-far* with respect to g if g_1 and g_2 are not T-close with respect to g. When g is clear from the context, we simply say g_1 and g_2 are T-far/T-close.

The following lemma is from[BGI16], giving a distance function, defined based on a randomly chosen function f, which serves a hint bit in our construction (i.e., the function Φ described above). We will later replace such a random function with a PRF.

Lemma 3.1 (Proposition 3.2 in [BGI16])**.** *Let $\mathbb{G}$ be a group of prime order p, $g \in \mathbb{G}$, $M \in \mathbb{N}$, $\delta > 0$ and assume $[2M \log_e(2/\delta)]/\delta < p$. Let RF be the set of all functions $f : \mathbb{G} \to \{0,1\}^{\lceil \log(2M/\delta) \rceil}$. Then for any integer $x \leq M$ and $h \in \mathbb{G}$*

$$\Pr_{f \xleftarrow{\$} \mathsf{RF}} [\mathsf{Dist}_{\mathbb{G},g}(h,\delta,M,f) = \mathsf{LSB}(x) - \mathsf{Dist}_{\mathbb{G},g}(hg^x,\delta,M,f)] \geq 1 - \delta. \tag{11}$$

Moreover, for any set of group elements $h_1, \ldots, h_m$ which are mutually at least $(T+2)$-far, the events $\mathsf{Success}_1, \ldots, \mathsf{Success}_m$ are independent, where $\mathsf{Success}_i$ is the event that $\mathsf{Dist}_{\mathbb{G},g}(h_i,\delta,M,f) = 1 - \mathsf{Dist}_{\mathbb{G},g}(h_i g,\delta,M,f)$.

Proof. The first part of the lemma was proved in [BGI16]. The second part follows because (1) the function f is chosen at random and (2) for any group element h, the outputs of $\mathsf{Dist}_{\mathbb{G},g}(h,\delta,M,f)$ and $\mathsf{Dist}_{\mathbb{G},g}(hg,\delta,M,f)$ are only dependent on the outputs of f on group elements $\{h, hg, hg^2, \ldots, hg^{T+1}\}$. □

Notation. We use the following notation below. For a string $\mathsf{s} = \mathsf{s}_1 \ldots \mathsf{s}_m \in \{0,1\}^m$ we define $\mathsf{Reverse}(\mathsf{s}) := s_m \ldots s_1$. For $\mathsf{x} \in \{0,1\}^n$ and $\mathbf{v} := (g_1, \ldots, g_n) \in \mathbb{G}^n$ we define $\mathsf{x} \cdot \mathbf{v} := \prod_{i=1}^n g_i^{\mathsf{x}_i}$.

Construction 3.2 (Doubly-Linear lossy TDF)**.** *Let* G *be a group scheme and let* $(\mathsf{Encode}, \mathsf{Decode})$ *for* $\mathsf{Encode}\colon \{0,1\}^n \to \{0,1\}^m$ *be an ECC code. Let* $\ell := \log(2/\delta)$ *and let* $\mathsf{PRF} : \mathbb{G} \to \{0,1\}^\ell$ *be a PRF with key space* $\{0,1\}^\lambda$*. We will instantiate the values of* δ *later.*

- $\mathsf{TDF.KG}(1^\lambda)$*:*
  1. *Sample* $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{G}(1^\lambda)$*. Sample* $\alpha, r \xleftarrow{\$} \mathbb{Z}_p$ *and set*
  $$\mathbf{v} := (g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^m}) \tag{12}$$
  $$\mathbf{w} := (g^{r\alpha}, g^{r\alpha^2}, \ldots, g^{r\alpha^{m-1}}, gg^{r\alpha^m}, g^{r\alpha^{m+1}}, \ldots, g^{r\alpha^{2m-1}}). \tag{13}$$
  2. *Sample a key* $K \xleftarrow{\$} \{0,1\}^\lambda$ *for* PRF*.*
  3. *Set* $\mathsf{ik} := (K, g, \mathbf{v}, \mathbf{w})$ *and* $\mathsf{tk} := (K, g, \alpha, r)$*. Return* $(\mathsf{ik}, \mathsf{tk})$*.*
- $\mathsf{TDF.KG}_{\mathsf{ls}}(1^\lambda)$*: Return* $\mathsf{ik}_{\mathsf{ls}} := (g, \mathbf{v}, \mathbf{w}')$*, where* g*,* $\mathbf{v}$ *are as above, and*
  $$\mathbf{w}' := (g^{r\alpha}, g^{r\alpha^2}, \ldots, g^{r\alpha^{m-1}}, g^{r\alpha^m}, g^{r\alpha^{m+1}}, \ldots, g^{r\alpha^{2m-1}}). \tag{14}$$
- $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{x} \in \{0,1\}^n)$*: Parse* $\mathsf{ik} := (g, \mathbf{v}, \mathbf{w})$ *and* $\mathsf{z} := \mathsf{Encode}(\mathsf{x})$*. For* $1 \le i \le m$
  1. *let* $\mathbf{w}'_i = \mathbf{w}[i, i+m-1]$*.*
  2. *let* $g_i = \mathsf{z} \cdot \mathbf{w}'_i$*;*
  3. *let* $b_i := \mathsf{Dist}_{\mathbb{G},g}(g_i, \delta, 1, \mathsf{PRF}_K)$*.*

  Let $g_c := \mathsf{z} \cdot \mathbf{v}$ *and return*
  $$\mathsf{u} := (g_c, b_1, \ldots, b_m). \tag{15}$$
- $\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{u})$*: Parse* $\mathsf{u} := (g_c, b_1, \ldots, b_m)$*. Recover* z' *bit-by-bit as follows. For* $i \in [m]$*:*
  1. *Let* $g_{i,0} = g_c^{r\alpha^{i-1}}$ *and* $g_{i,1} = g_{i,0}g$*.*
  2. *If*
     (a) $\mathsf{Dist}_{\mathbb{G},g}(g_{i,0}, \delta, 1, \mathsf{PRF}_K) = \mathsf{Dist}_{\mathbb{G},g}(g_{i,1}, \delta, 1, \mathsf{PRF}_K)$*, then set* $\mathsf{z}_i = \perp$*;*
     (b) *Else, let* b *the bit for which* $\mathsf{Dist}_{\mathbb{G},g}(g_{i,b}, \delta, 1, \mathsf{PRF}_K) = b_i$*, and set* $\mathsf{z}_i = b$*.*

  Return $\mathsf{Decode}(\mathsf{Reverse}(\mathsf{z}))$*.*

We now prove all the required properties of the scheme.

Lemma 3.3 (Mode indistinguishability). *We have* $\mathsf{ik} \stackrel{c}{\equiv} \mathsf{ik}_{\mathsf{ls}}$, *where* $\mathsf{ik} \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$ *and* $\mathsf{ik}_{\mathsf{ls}} \xleftarrow{\$} \mathsf{TDF.KG}_{\mathsf{ls}}$.

Proof. Follows immediately from $(2m-1)$-power DDH (Lemma 2.4). □

Lemma 3.4 (Lossiness). *Assuming p is the oder of the group, for any* $\mathsf{ik}_{\mathsf{ls}} \in \mathsf{TDF.KG}_{\mathsf{ls}}(1^\lambda)$, $|\mathsf{TDF.F}(\mathsf{ik}_{\mathsf{ls}}, \{0,1\}^n)| \leq p$.

Proof. Parse $\mathsf{ik}_{\mathsf{ls}} := (g, \mathbf{v}, \mathbf{w}')$, where $\mathbf{v}$ is sampled as in Equation 12 and $\mathbf{w}'$ is sampled as in Equation 14. We claim the following: for any $\mathsf{x}', \mathsf{x}' \in \{0,1\}^n$, letting $\mathsf{z} := \mathsf{Encode}(\mathsf{x})$ and $\mathsf{z}' := \mathsf{Encode}(\mathsf{x}')$, if $\mathsf{z} \cdot \mathbf{v} = \mathsf{z}' \cdot \mathbf{v}$, then $\mathsf{TDF.F}(\mathsf{ik}_{\mathsf{ls}}, \mathsf{x}) = \mathsf{TDF.F}(\mathsf{ik}_{\mathsf{ls}}, \mathsf{x})$. Assuming the claim holds, the lemma follows immediately. This is because, under the lossy key $\mathsf{ik}_{\mathsf{ls}}$, once the first component g_c of the image $\mathbf{u} := (g_c, \dots)$ is determined, the rest of the output is uniquely determined. To prove the claim, suppose $g_c = \mathsf{z} \cdot \mathbf{v} = \mathsf{z}' \cdot \mathbf{v}$. Notice that the group element g_i computed in Line 2 of $\mathsf{TDF.F}$ is equal to the fixed element $g_c^{r\alpha^{i-1}}$, irrespective of whether the underlying input is x or x'. This follows from the way $\mathbf{w}'$ is formed (Equation 14). The proof is now complete. □

Lemma 3.5 (Correctness). *Let* $(\mathsf{Encode}, \mathsf{Decode})$ *be an* $(n, m, s)_2$ *code, where* $n = \lambda + \omega(\log \lambda)$. *Assuming* $\delta \leq \frac{s-1}{2m}$ *and* $T := \lceil 2\log_e(2/\delta) \rceil / \delta = \mathsf{poly}(\lambda)$, *for any input* x:

$$\beta(\lambda) := \Pr_{(\mathsf{ik},\mathsf{tk})}[\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})) \neq \mathsf{x}] \leq \frac{1}{\mathsf{e}^{\frac{s-1}{6}}} + \mathsf{negl}(\lambda), \tag{16}$$

where the probability is taken over $(\mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$. *In particular, by setting* $n = \lambda + \omega(\log \lambda)$, $s \in \omega(\log \lambda)$ *and* $\delta \leq \frac{s-1}{2m}$, *we will have a negligible inversion error.*

Proof. Fix $\mathsf{x} \in \{0,1\}^n$, let $\mathsf{z} := \mathsf{Encode}(\mathsf{x})$ and $\mathsf{z}' := \mathsf{Reverse}(\mathsf{z})$. Consider a variant of Construction 3.2, in which we replace the PRF PRF_K with a truly random function $f : \mathbb{G} \xleftarrow{\$} \{0,1\}^\ell$. (Recall that $\ell = \log(2/\delta)$.) That is, in this variant, calls of the form $\mathsf{Dist}_{\mathbb{G},g}(g_i, \delta, 1, K)$ are replaced with $\mathsf{Dist}_{\mathbb{G},g}(g_i, \delta, 1, f)$. Let β' be the probability that $\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})) \neq \mathsf{x}$ in this experiment. We will show $\beta' \leq \frac{1}{\mathsf{e}^{\frac{s-1}{6}}} + \mathsf{negl}(\lambda)$. By PRF security we have $\beta \leq \beta' + \mathsf{negl}(\lambda)$, and thus Equation 16 will follow. The reason that we can use PRF security here (despite the fact that K is given in the clear in ik) is that the procedure Dist may efficiently be computed via only blackbox access to PRF_K (resp., f alternatively) and that we evaluate PRF_K on inputs generated independently of K.

For an index $i \in [m]$, let $g_i = \mathsf{z} \cdot \mathbf{w}'_i$ be the group element computed during evaluation for z'_i (i.e., Line 2 of $\mathsf{TDF.F}$), and let $g_{i,0} = g_c^{r\alpha^{i-1}}$ and $g_{i,1} = g_{i,0} g$ be the two corresponding group elements computed during inversion. Notice that $g_i = g_{i,\mathsf{z}'_i}$.

For $i \in [m]$, let the indicator variable

$$\mathsf{Fail}_i = 1 \Leftrightarrow \mathsf{Dist}_{\mathbb{G},g}(g_{i,0}, \delta, 1, f) = \mathsf{Dist}_{\mathbb{G},g}(g_{i,1}, \delta, 1, f).$$

Notice that $\mathsf{Fail}_i = 1$ iff we fail to recover z'_i. For all i, by setting $M = 1$ in Lemma 3.1, $\Pr[\mathsf{Fail}_i] < \delta$, and hence $\Pr[\mathsf{Fail}_i] < p'$, where $p' = \frac{s-1}{2m}$.

Let $\mathsf{Fail} = \sum_{i \in [m]} \mathsf{Fail}_i$. Inversion fails if $\mathsf{Fail} > s-1$. We may now be tempted to use Lemma 2.5 to bound the probability that $\mathsf{Fail} > s-1$. The problem is that the events Fail_i's may not be independent. Thus, we define an event Bad which captures all the dependencies, and then we will argue that conditioned on $\overline{\mathsf{Bad}}$, the events $\{\mathsf{Fail}_i\}_{i \in [m]}$ are independent.

- Bad: there are two distinct indices $i, j \in [m]$ such that $g_{i,0}$ and $g_{j,0}$ are $(T+1)$-close, where $T := [2\log_e(2/\delta)]/\delta$.

By Lemma 3.1 we know that conditioned on $\overline{\mathsf{Bad}}$, the events Fail_i's are independent. Below we will show $\Pr[\mathsf{Bad}] = \mathsf{negl}(\lambda)$, but assuming this for now, we have

$$\Pr[\mathsf{Fail} > s-1] \leq \Pr[\mathsf{Bad}] + \Pr[\mathsf{Fail} > 2p'm \mid \overline{\mathsf{Bad}}] <^* \mathsf{negl}(\lambda) + \frac{1}{\mathsf{e}^{p'm/3}} = \mathsf{negl}(\lambda) + \frac{1}{\mathsf{e}^{\frac{s-1}{6}}},$$

where the inequality marked with * follows from Lemma 2.5, considering the fact that conditioned on $\overline{\mathsf{Bad}}$, the events $\{\mathsf{Fail}_i\}_{i\in[m]}$ are independent.

We are now left to prove $\Pr[\mathsf{Bad}] = \mathsf{negl}(\lambda)$. Recall that $(g_{1,0}, \ldots, g_{m,0}) = (g_c^{r\alpha}, \ldots, g_c^{r\alpha^m})$. Notice that $g_c \neq 1$ except with negligible probability, and thus g_c^r is statistically close to a uniformly random group element. By Lemma 2.4

$$(g_{1,0}, \ldots, g_{m,0}) = (g_c^{r\alpha}, \ldots, g_c^{r\alpha^m}) \stackrel{c}{\equiv} (g'_1, \ldots, g'_m),$$

where g'_i's are random group elements. When replacing $\{g_{i,0}\}_{i\in[m]}$ with $\{g'_i\}_{i\in[m]}$ the probability of the event Bad becomes negligible. (This is because $T = \mathsf{poly}(\lambda)$.) Thus, the event Bad with $g_{i,0}$'s should also be negligible. □

3.1 Running Time of our Lossy TDFs

We count the number of public-key operations (i.e., group operations) involved in the computation of $\mathsf{TDF.F}$. (The other operations involved in $\mathsf{TDF.F}$ are either private-key, i.e., PRF evaluations, or information theoretic; i.e., error correcting codes).[4] For $\mathsf{TDF.F}$, in Line 2, one may compute the group elements $g_i = \mathsf{z} \cdot \mathbf{w}'_i$ one at a time, by using m group multiplications for each of them, hence $O(m^2)$ group multiplications in total. We observe that the computations of all g_i's together may be thought of as multiplying a Toeplitz matrix $g^{\mathbf{M}} \in \mathbb{G}^{m\times m}$, given in the exponent, with a given vector z^T of bits. It is known that one can compute $\mathbf{M} \times \mathsf{z}^T \pmod p$ in $O(m \log m)$ time using (inverse) discrete Fourier transform (IDFT/DFT) modulo p. In the next section we show how to carry out this computation in the exponent, at the cost of $O(m \log m)$ group exponentiations.

Comparison with the trivial approach. As mentioned above, the trivial computation takes $O(m^2)$ group multiplications. Our FFT-based approach takes $O(m \log m)$ group exponentiations, which translate into $O(m\lambda \log m)$ multiplications, assuming $|\mathbb{G}| = 2^\lambda$. So we obtain improvements, when $\lambda \log m \in \omega(m)$. We also note that the reason that the trivial approach takes $O(m \log m)$ multiplications (as opposed to exponentiations) is that we multiply with a bit vector, translating into multiplications. In applications where the entries of the given vector are integers modulo p, the trivial approach will take $O(m^2)$ exponentiations, while our FFT-based approach still takes $O(m \log m)$ exponentiations. This observation may be useful for future work.

[4] We only focus on $\mathsf{TDF.F}$, because $\mathsf{TDF.F}^{-1}$ may be done using n group exponentiations, which seems hard to improve.

4 Fast Fourier Transform in the Exponent

In this section, we show how to perform FFT in the exponent in order to have a fast algorithm for multiplying a circulant or Toeplitz matrix, given in the exponent, with a vector of integers, with the result being computed in the exponent. We begin with some basic background.

For a vector $\mathbf{u}$ of integers and a group element g we use $g^{\mathbf{u}}$ to mean element-wise exponentiation.

Lemma 4.1 (Primitive nth root of unity mod p)**.** *We say $w \in \mathbb{Z}_p$ is a* primitive nth root of unity mod p *if $w^n \equiv 1 \pmod p$ and for all $i \in [n-1]$, $w^i \not\equiv 1 \pmod p$. If p is prime, then $\mathbb{Z}_p$ has a primitive nth root of unity if and only if $p \equiv 1 \pmod n$.*

(Inverse) Discrete Fourier modulo $\mathbb{Z}_p$. Let $w \in \mathbb{Z}_p$ be a primitive nth root of unity modulo p (Lemma 4.1). The discrete fourier transform (DFT) of $(y_0, \ldots, y_{n-1}) \in \mathbb{Z}_p^n$, denoted $\mathsf{DFT}(y_0, \ldots, y_{n-1})$, is $(d_0, \ldots, d_{n-1}) \in \mathbb{Z}_p^n$, where for $k \in \{0\} \cup [n-1]$:

$$d_k = \sum_{j=0}^{n-1} y_j w^{-jk} \pmod p. \tag{17}$$

The inverse discrete Fourier transform (IDFT) inverts the above process. For $(d_0, \ldots, d_{n-1}) \in \mathbb{Z}_p^n$, $\mathsf{IDFT}(d_0, \ldots, d_{n-1})$ is defined to be $(y_0, \ldots, y_{n-1})$, where for $k \in \{0\} \cup [n-1]$

$$y_k = n^{-1} \sum_{j=0}^{n-1} d_j w^{jk} \pmod p. \tag{18}$$

For all $(y_0, \ldots, y_{n-1}) \in \mathbb{Z}_p^n$, $\mathsf{IDFT}(\mathsf{DFT}(y_0, \ldots, y_{n-1})) = (y_0, \ldots, y_{n-1})$.

A major step in performing fast circulant matrix multiplication involves computing DFT and IDFT in a fast way.

Computing (I)DFT in the exponent. For $\mathbf{y} := (y_0, \ldots, y_{n-1}) \in \mathbb{Z}_p^n$, we would like to compute $\mathsf{DFT}(\mathbf{y})$ in the exponent; i.e., to compute $g^{\mathsf{DFT}(\mathbf{Y})}$ from $g^{\mathbf{y}}$. Since $\mathsf{DFT}(\mathbf{y})$ is a linear function in the entries of $\mathbf{y}$ and w is a fixed integer, we may compute each component of $\mathsf{DFT}(\mathbf{y})$ using n exponentiations, resulting in a total of $O(n^2)$ exponentiations. There is, however, a faster, recursive way of doing this using $O(n \log n)$ exponentiations.

Let $f = w^{-1}$, and note that f is also a primitive nth root of unity. Computing $\mathsf{DFT}(\mathbf{y})$ amounts to evaluating a degree $n-1$ polynomial $p(x) = \sum_{j=0}^{n-1} y_j x^j$ at $(p(1), p(f), \ldots, p(f^{n-1}))$. We may now evaluate these n invocations in time $O(n \log n)$ using divide-and-conquer. Specifically, letting $n = 2t$, we can find two degree $t - 1 = n/2 - 1$ polynomials p_{even} and p_{odd} such that

(a) $p(f^{2k}) = p_{\mathsf{even}}(f^{2k})$ for $k \in \{0\} \cup [t-1]$; and

(b) $p(f^{2k+1}) = p_{\mathsf{odd}}(f^{2k})$ for $k \in \{0\} \cup [t-1]$.

Now since f^2 is a primitive t'th root of unity and since the degree of each of p_{even} and p_{odd} is $t-1$, we can recursively continue this process. We now explain how to find p_{even} and p_{odd}.

Specifically, $p_{\mathsf{even}}(x) := \sum_{j=0}^{t-1} \alpha_j x^j$ and $p_{\mathsf{odd}}(x) := \sum_{j=0}^{t-1} \beta_j x^j$, where

$$\alpha_j := y_j + y_{j+t} \qquad \beta_j := (y_j - y_{j+t}) f^j. \tag{19}$$

We now show why p_{even} and p_{odd} satisfy Items (a) and (b) above.

$$p(f^{2k}) = \sum_{j=0}^{t-1} y_j f^{2kj} + \sum_{j=t}^{n-1} y_j f^{2kj} = \sum_{j=0}^{t-1} (y_j f^{2kj} + y_{j+t} f^{2k(j+t)}) = \sum_{j=0}^{t-1} (y_j f^{2kj} + y_{j+t} f^{2kj} f^{kn})$$
$$= \sum_{j=0}^{t-1} (y_j + y_{j+t}) f^{2kj} = p_{\mathsf{even}}(f^{2k}). \quad (20)$$

$$p(f^{2k+1}) = \sum_{j=0}^{t-1} (y_j f^{(2k+1)j} + y_{j+t} f^{(2k+1)(j+t)}) = \sum_{j=0}^{t-1} (y_j f^j) f^{2kj} + (y_{j+t} f^j) f^{2kj} f^{kn+t}$$
$$=^* \sum_{j=0}^{t-1} (y_j f^j) f^{2kj} + (y_{j+t} f^j) f^{2kj} (-1) = \sum_{j=0}^{t-1} (y_j - y_{j+t}) f^{(2k+1)j} = p_{\mathsf{odd}}(f^{2k}), \quad (21)$$

where the equation marked with * follows from the fact that $f^t = f^{n/2} = -1$. Finally, notice that given $\mathbf{y} := (y_0, \ldots, y_{n-1})$ in the exponent (i.e., given $g^{\mathbf{y}}$), the coefficients of p_{even} and p_{odd} (Equation 19) can also be computed in the exponent. Thus we have the following lemma.

Lemma 4.2 (DFT/IDFT in the exponent)**.** *Let n be a power of two, let p be a prime number satisfying $p \equiv 1 \pmod{n}$ and let $\mathbb{G}$ be group of order p with a generator g. Let $w \in \mathbb{Z}_p$ be a primitive nth root of unity modulo p (which exists by Lemma 4.1). For any $\mathbf{y} \in \mathbb{Z}_p^n$ we can compute $g^{\mathsf{DFT}(\mathbf{y})}$ from $g^{\mathbf{y}}$ using $O(n \log n)$ group exponentiations. The same holds for computing $g^{\mathsf{IDFT}(\mathbf{y})}$.*

Circulant matrices. Let $\mathbf{v} = (v_0, \ldots, v_{n-1})$ be a vector of dimension n. The circulant matrix of $\mathbf{v}$, denoted $\mathsf{Rot}(\mathbf{v})$, is

$$\mathsf{Rot}(\mathbf{v}) := \begin{pmatrix} v_0 & v_{n-1} & v_{n-2} & \ldots & v_3 & v_2 & v_1 \\ v_1 & v_0 & v_{n-1} & \ldots & v_4 & v_3 & v_2 \\ v_2 & v_1 & v_0 & \ldots & v_5 & v_4 & v_3 \\ \vdots & \vdots & \vdots & \ldots & \vdots & \vdots & \vdots \\ v_{n-1} & v_{n-2} & v_{n-3} & \ldots & v_0 & v_{n-1} & v_{n-2} \\ v_{n-2} & v_{n-3} & v_{n-4} & \ldots & v_1 & v_0 & v_{n-1} \\ v_{n-1} & v_{n-2} & v_{n-3} & \ldots & v_2 & v_1 & v_0 \end{pmatrix} \quad (22)$$

Lemma 4.3 (Circulant matrix multiplication in the exponent)**.** *Let n, p, $\mathbb{G}$ and w be as in Lemma 4.2. Let $\mathbf{u} := (u_0, \ldots, u_{n-1}) \in \mathbb{Z}_p^n$ and $\mathbf{v} := (v_0, \ldots, v_{n-1}) \in \mathbb{Z}_p^n$ and $\mathbf{M} := \mathsf{Rot}(\mathbf{v})$. Then we can compute $g^{\mathbf{M}\mathbf{u}^\mathsf{T}}$ from $g^{\mathbf{v}}$ and $\mathbf{u}$ via $O(n \log n)$ group exponentiations.*

Proof. Throughout the proof, we may use negative indices, with the understanding the index is taken modulo n. For example, we may write u_{-1} for u_{n-1}. Given $g^{\mathbf{v}}$ and $\mathbf{u}$, for $k \in \{0\} \cup [n-1]$ we need to compute g^{h_k}, where

$$h_k = \sum_{i=0}^{n-1} v_j u_{k-j}. \quad (23)$$

Let $(a_0, \ldots, a_{n-1})$ and $(b_0, \ldots, b_{n-1})$ be the discrete fourier transform of the two sequences $(v_0, \ldots, v_{k-1})$ and $(u_0, \ldots, u_{k-1})$, respectively. That is, for $k \in \{0, \ldots, n-1\}$

$$a_k = \sum_{j=0}^{n-1} v_j w^{-jk} \pmod p \qquad b_k = \sum_{j=0}^{n-1} u_j w^{-jk} \pmod p.$$

It is well-known that the inverse fourier transform of $(a_0 b_0, \ldots, a_{n-1} b_{n-1})$ gives us the values $(h_0, \ldots, h_{n-1})$. That is, for $k \in \{0\} \cup [n-1]$

$$(h_0, \ldots, h_{n-1}) = \mathsf{IDFT}(a_0 b_0, \ldots, a_{n-1} b_{n-1}). \tag{24}$$

By Lemma 4.2 we can perform all the above steps via $O(n \log n)$ exponentiations. □

Fast Toeplitz matrix multiplication. We now show how to perform fast Topelitz matrix multiplication in the exponent, via a well-known conversion to circulant matrices. See [BDD+00] for further conversions. For $\mathbf{x} := (x_1, \ldots, x_{2n-1}) \in \mathbb{Z}_p^{2n-1}$ we define

$$\mathsf{Toep}(\mathbf{x}) := \begin{pmatrix} x_n & x_{n-1} & \ldots & x_1 \\ x_{n+1} & x_n & \ldots & x_2 \\ \vdots & \vdots & \ldots & \vdots \\ x_{2n-1} & x_{2n-2} & \ldots & x_n \end{pmatrix}. \tag{25}$$

Let $\mathbf{M} := \mathsf{Toep}(\mathbf{x})$ and $\mathbf{y} \in \mathbb{Z}_p^n$. We show how to compute $g^{\mathbf{My}}$ from $g^{\mathbf{M}}$ and $\mathbf{y}$. Toward this, define

$$\mathbf{S} := \begin{pmatrix} 0 & x_1 & x_2 & \ldots & x_{n-1} \\ x_{2n-1} & 0 & x_1 & \ldots & x_{n-2} \\ x_{2n-2} & x_{2n-1} & 0 & \ldots & x_{n-3} \\ \vdots & \vdots & \ldots & \vdots & \\ x_{n+1} & x_{n+2} & x_{n+3} & \ldots & 0 \end{pmatrix} \in \mathbb{Z}_p^{n \times n}. \tag{26}$$

Let $\mathbf{T} := \begin{pmatrix} \mathbf{M} & \mathbf{S} \\ \mathbf{S} & \mathbf{M} \end{pmatrix} \in \mathbb{Z}_p^{2n \times 2n}$. Note that $\mathbf{T}$ is a circulant matrix. We have $\mathbf{M} \begin{pmatrix} \mathbf{y} \\ 0_{n \times 1} \end{pmatrix} = \begin{pmatrix} \mathbf{Ty} \\ \mathbf{Sy} \end{pmatrix}$. Thus, we may compute $\mathbf{My}$ in the exponent via $O(n \log n)$ group exponentiations. Thus, we have the following lemma.

Lemma 4.4 (Toeplitz matrix multiplication in the exponent)**.** *Let n, p, $\mathbb{G}$ and w be as in Lemma 4.2. Let $\mathbf{u} := (u_0, \ldots, u_{n-1}) \in \mathbb{Z}_p^n$ and $\mathbf{v} := (v_0, \ldots, v_{n-1}) \in \mathbb{Z}_p^n$ and $\mathbf{M} := \mathsf{Toep}(\mathbf{v})$. Then we can compute $g^{\mathbf{Mu}^\mathsf{T}}$ from $g^{\mathbf{M}}$ and $\mathbf{u}$ using $O(n \log n)$ group exponentiations.*

5 Range-Trapdoor Hash Functions

In this section, we define the notion of range-trapdoor hash functions and give a construction of this notion with short evaluation keys. This notion generalizes the notion of trapdoor hash function [DGI+19] for index keys. We say that an index set I is a *range set* if $I = \{s+1, \ldots, s+t\}$ for some integers s and t. We now give the definition of range-trapdoor hash for the special case where we output a single-bit hint for every index in the range set.

Definition 5.1 (Range Trapdoor Hash)**.** *An n-bit input, range-trapdoor hash is a tuple of PPT algorithms $\mathcal{H} = (\mathsf{S}, \mathsf{KG}, \mathsf{H}, \mathsf{E}, \mathsf{D})$ with the following syntax, correctness and security properties.*

- $\mathsf{S}(1^\lambda, n)$*: Takes the security parameter 1^λ and input length n, and outputs a hashing key hk and a trapdoor key thk.*
- $\mathsf{KG}(\mathsf{hk}, I)$*: Takes hk and a range of indices $I = [s+1, \ldots, s+t] \subseteq [n]$ as input, and outputs an evaluation key ek and a trapdoor key tk. We assume ek contains $|I|$; i.e., $\mathsf{ek} := (|I|, \ldots)$, and also assume $\mathsf{tk} := (I, \ldots)$.*
- $\mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$*: Takes hk, a message $\mathsf{x} \in \{0,1\}^n$ and randomness ρ as input, and outputs a hash value h.*
- $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$*: Takes an evaluation key ek, message x and randomness ρ as input, and outputs a hint value $\mathsf{e} \in \{0,1\}^{|I|}$.*
- $\mathsf{D}(\mathsf{thk}, \mathsf{tk}, \mathsf{h}, \mathsf{e})$*: Takes as input a hash-trapdoor key thk, a trapdoor key $\mathsf{tk} := (I, \ldots)$, a hash value h and a hint value e, and deterministically outputs $|I|$ pairs of $0/1$-encodings $(\mathsf{e}_{i,0}, \mathsf{e}_{i,1}) \in \{0,1\} \times \{0,1\}$, for $i \in [|I|]$.*

We require the following properties.

- ***Correctness****: For $0 \leq \epsilon < 1$ we say $\mathcal{H}$ is $1-\epsilon$ correct (or has ϵ decryption error) if for any n, any range set $I := [s+1, s+t] \subseteq [n]$, both the following conditions hold:*
  1. *For any $i \in [t]$ and for any input $\mathsf{x} \in \{0,1\}^n$, $\Pr[\mathsf{e}_i = \mathsf{e}_{i,\mathsf{x}[s+i]}] = 1$; and*
  2. *For any input $\mathsf{x} \in \{0,1\}^n$, any $i \in [t]$ and any $b_j \in \{0,1\}$ for $j \in [t] \setminus \{i\}$:*

$$\Pr[\mathsf{Fail}_i = 1 \mid \mathsf{Fail}_j = b_j \text{ for } j \in [t]/\{i\}] \leq \epsilon + \mathsf{negl}(\lambda), \tag{27}$$

  where for $i \in [t]$, Fail_i is an indicator variable, defined as $\mathsf{Fail}_i = 1$ if $\mathsf{e}_i = \mathsf{e}_{i,1-\mathsf{x}[s+i]}$,

  where $(\mathsf{hk}, \mathsf{thk}) \xleftarrow{\$} \mathsf{S}(1^\lambda, n)$, $(\mathsf{ek}, \mathsf{tk}) \xleftarrow{\$} \mathsf{KG}(\mathsf{hk}, I)$, $\rho \xleftarrow{\$} \{0,1\}^*$, $\mathsf{h} := \mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$, $\mathsf{e} := \mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$, $(\mathsf{e}_{i,0}, \mathsf{e}_{i,1})_{i \in [t]} := \mathsf{D}(\mathsf{thk}, \mathsf{tk}, \mathsf{h}, \mathsf{e})$.
- ***Range privacy****: For any n and any two range sets $I, I' \subseteq [n]$ satisfying $|I| = |I'|$, $(\mathsf{hk}, \mathsf{ek}) \stackrel{c}{\equiv} (\mathsf{hk}', \mathsf{ek}')$, where $(\mathsf{hk}, *) \xleftarrow{\$} \mathsf{S}(1^\lambda, n)$, $(\mathsf{ek}, *) \xleftarrow{\$} \mathsf{KG}(\mathsf{hk}, I)$ and $(\mathsf{ek}', *) \xleftarrow{\$} \mathsf{KG}(\mathsf{hk}, I')$.*
- ***Input privacy****: Fix polynomial $n := n(\lambda)$. For any two inputs $\mathsf{x}, \mathsf{x}' \in \{0,1\}^n$, $(\mathsf{hk}, \mathsf{h}) \stackrel{c}{\equiv} (\mathsf{hk}, \mathsf{h}')$, where $(\mathsf{hk}, *) \xleftarrow{\$} \mathsf{S}(1^\lambda, n)$, $\mathsf{h} \xleftarrow{\$} \mathsf{H}(\mathsf{hk}, \mathsf{x})$ and $\mathsf{h}' \xleftarrow{\$} \mathsf{H}(\mathsf{hk}, \mathsf{x})$.*
- ***Compactness****: There exists a polynomial $\mathsf{poly}(\lambda)$ such that for all $n := n(\lambda)$, $|\mathsf{H}(\mathsf{hk}, \mathsf{x})| \leq \mathsf{poly}(\lambda)$, where $(\mathsf{hk}, *) \xleftarrow{\$} \mathsf{S}(1^\lambda, n)$ and $\mathsf{x} \in \{0,1\}^n$.*

We note the following remark.

Remark 5.2. *For decryption we also require a trapdoor key thk associated with hk. This will be required in our construction. In contrast, the notion of trapdoor hash as defined in [DGI+19] does not require a trapdoor for the hash function in order to perform decryption. Nonetheless, all applications stated in [DGI+19] still hold with respect to our definition.*

Implicit in the work of [DGI+19] is the following construction of range-trapdoor hash.

Lemma 5.3 (Theorem 4.3 of [DGI+19])**.** *Assuming DDH, there exists a range-trapdoor hash scheme where for inputs of length n, an evaluation key for a range set I consists of $O(n|I|)$ group elements.*

We give the following corollary, which helps one in bounding the number of Fail_i's in situations where, e.g., we need to do error correction, such as the rate-1 OT application. We say $\epsilon > \mathsf{negl}(\lambda)$ if ϵ is not a negligible function.

Lemma 5.4. *Assuming a trapdoor hash scheme $\mathcal{H} = (\mathsf{S}, \mathsf{KG}, \mathsf{H}, \mathsf{E}, \mathsf{D})$ has decryption error ϵ, and that $\epsilon > \mathsf{negl}(\lambda)$, then for any constant $c > 1$:*

$$\Pr[\mathsf{Fail} > 2c\epsilon|I|] < e^{-c\epsilon|I|/3},$$

where $\mathsf{Fail} := \sum_{i=1}^{|I|} \mathsf{Fail}_i$ and Fail_i is defined in the correctness condition of Definition 5.1.

Proof. The proof follows immediately from the bounded-dependence version of the Chernoff bound (Lemma 2.6). □

We now show how to adapt our batching technique from Section 3 to obtain range-trapdoor hash schemes, where the evaluation key consists of $O(n)$ group elements, as opposed to $O(n|I|)$ group elements given by [DGI+19]. As we will see in Section 6, this size reduction results in a shorter receiver's message in rate-1 OT protocols and shorter ciphertexts in homomorphic encryption for branching programs.

5.1 Range-Trapdoor Hash with Linear-Sized Evaluation Keys

Construction 5.5. *Let $\epsilon \in [0, 1)$ be the decryption error we are wiling to tolerate. Let $\ell := \log(2/\epsilon)$, G be a group scheme and $\mathsf{PRF} : \mathbb{G} \to \{0,1\}^\ell$ a PRF with key space $\{0,1\}^\lambda$.*

- $\mathsf{S}(1^\lambda, n)$*: and $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{G}(1^\lambda)$. Sample $\alpha \xleftarrow{\$} \mathbb{Z}_p$, set $\mathsf{thk} := \alpha$ and $\mathsf{hk} := (\mathbb{G}, p, g, \mathbf{v})$, where $\mathbf{v} := (g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^{2n}})$. Return $(\mathsf{hk}, \mathsf{thk})$.*
- $\mathsf{KG}(\mathsf{hk}, I)$*: Sample a key $K \xleftarrow{\$} \{0,1\}^\lambda$ for PRF. Let $I = [s+1, s+t]$. Parse $\mathsf{hk} := (\mathbb{G}, p, g, \mathbf{v})$, where $\mathbf{v} := (g_1, \ldots, g_{2n})$. Sample $r \xleftarrow{\$} \mathbb{Z}_p$ and let*

  $$\mathbf{w} := (g_1^r, g_2^r, \ldots, g_{s+t-1}^r, g g_{s+t}^r, g_{s+t+1}^r, \ldots, g_{2n}^r).$$

  Set $\mathsf{ek} := (t, \mathbf{w}, K)$ and $\mathsf{tk} := (I, r, K)$.
- $\mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$*: Parse $\mathsf{hk} := (\mathbb{G}, p, g, \mathbf{v})$, where $\mathbf{v} := (g_1, \ldots, g_{2n})$. Let $\mathbf{v}' := (g_1, \ldots, g_n)$, and return $(\mathsf{x} \cdot \mathbf{v}')g_1^\rho$.*
- $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$*: Parse $\mathsf{ek} := (t, \mathbf{w}, K)$, where $t \in \mathbb{N}$ and $\mathbf{w} \in \mathbb{G}^{2n}$. Parse $\mathbf{w} := (w_1, \ldots, w_{2n})$. For $i \in [t]$:*
  1. *let $\mathbf{w}'_i = (w_i, \ldots, w_{i+n-1}) \in \mathbb{G}^n$;*
  2. *let $g'_i := (\mathsf{x} \cdot \mathbf{w}'_i)w_i^\rho$;*
  3. *let $b_i := \mathsf{Dist}_{\mathbb{G},g}(g'_i, \epsilon, 1, \mathsf{PRF}_K)$.*

Return $(b_t, \ldots, b_1)$.

- $\mathsf{D}(\mathsf{thk}, \mathsf{tk}, \mathsf{h}, \mathsf{e})$*: Parse* $\mathsf{thk} := \alpha$, $\mathsf{tk} := (I, r, K)$ *and* $I := [s+1, s+t]$. *For* $i \in [t]$, *set* $\mathsf{e}_{i,0} := \mathsf{Dist}_{\mathbb{G},g}(\mathsf{h}^{r\alpha^{i-1}}, \epsilon, 1, \mathsf{PRF}_K)$ *and* $\mathsf{e}_{i,1} := \mathsf{Dist}_{\mathbb{G},g}(g\mathsf{h}^{r\alpha^{i-1}}, \epsilon, 1, \mathsf{PRF}_K)$. *Return* $((\mathsf{e}_{t,0}, \mathsf{e}_{t,1}), \ldots, (\mathsf{e}_{1,0}, \mathsf{e}_{1,1}))$.

The compactness of the scheme is clear. Range privacy follows from $2n$-power DDH. We now prove the input privacy and correctness of the scheme.

Lemma 5.6 (Input privacy)**.** *The scheme provides perfect input privacy: for any two inputs* $\mathsf{x}, \mathsf{x}' \in \{0,1\}^n$, $(\mathsf{hk}, \mathsf{h}) \equiv (\mathsf{hk}, \mathsf{h}')$, *where* $(\mathsf{hk}, *) \xleftarrow{\$} \mathsf{S}(1^\lambda, n)$, $\mathsf{h} \xleftarrow{\$} \mathsf{H}(\mathsf{hk}, \mathsf{x})$ *and* $\mathsf{h}' \xleftarrow{\$} \mathsf{H}(\mathsf{hk}, \mathsf{x})$.

Proof. We need to show $(\mathbf{v}, (\mathsf{x}\cdot\mathbf{v})g^{\alpha\rho})$ is independent of x, where $\mathbf{v} := (g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^{2n}})$ and $\rho \xleftarrow{\$} \mathbb{Z}_p$. This immediately follows from the presence of the masking exponent ρ. □

Lemma 5.7 (Correctness)**.** *Assuming* $T := [2\log_e(2/\epsilon)]/\epsilon = \mathsf{poly}(\lambda)$ *(which is satisfied if* ϵ *is an inverse polynomial), the range TDH scheme provides* $(1-\epsilon)$ *correctness.*

Proof. Fix n, I, $\mathsf{x} \in \{0,1\}^n$ and suppose $I = [s+1, s+t]$. We need to prove Conditions 1 and 2 of the correctness definition. For $i \in [t]$ let g'_i be computed as in E (Line 2 of E's procedure) and let $g_{i,0} = \mathsf{h}^{r\alpha^{i-1}}$ and $g_{i,1} = g\mathsf{h}^{r\alpha^{i-1}}$. Notice that g'_i and $(g_{i,0}, g_{i,1})$ concern the $(s+t-i+1)$'th bit of x. (Recall that at the end of E and D we output the bits in reverse order.)

First, we claim $g'_i = g_{i,\mathsf{x}[s+t-i+1]}$, which proves Condition 1 of the correctness definition. To see why this claim holds, recall that

$$\begin{aligned}
\mathbf{v}' &= (g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^n}) \\
w_i &= g^{r\alpha^i} \\
\mathbf{w}'_i &= (g^{r\alpha^i}, \ldots, g^{r\alpha^{s+t-1}}, \underbrace{g g^{r\alpha^{s+t}}}_{coordinate:s+t-i+1}, g^{r\alpha^{s+t+1}}, \ldots, g^{r\alpha^{i+n-1}}),
\end{aligned}$$

and that $\mathsf{h} = (\mathsf{x}\cdot\mathbf{v}')g^{\alpha\rho}$, $g'_i := (\mathsf{x}\cdot\mathbf{w}'_i)w_i^\rho$. Letting $b = \mathsf{x}[s+t-i+1]$ we have

$$\begin{aligned}
g_{i,\mathsf{x}[s+t-i+1]} = g^b\mathsf{h}^{r\alpha^{i-1}} = g^b((\mathsf{x}\cdot\mathbf{v}')g^{\alpha\rho})^{r\alpha^{i-1}} = g^b(\mathsf{x}\cdot\mathbf{v}')^{r\alpha^{i-1}}(g^{r\alpha^i})^\rho \\
= g^b(\mathsf{x}\cdot\mathbf{v}')^{r\alpha^{i-1}}w_i^\rho = (\mathsf{x}\cdot\mathbf{w}'_i)w_i^\rho = g'_i, \quad (28)
\end{aligned}$$

as desired.

We now prove Condition 2 of the correctness definition. Fix $\mathsf{x} \in \{0,1\}^n$, $i \in [t]$ and $b_j \in \{0,1\}$ for $j \in [t] \setminus \{i\}$, and let

$$\beta := \Pr[\mathsf{Fail}_i = 1 \mid \mathsf{Fail}_j = b_j \text{ for } j \in [t]/\{i\}]. \quad (29)$$

Consider a variant of Construction 5.5, in which we replace the PRF PRF_K with a truly random function $f : \mathbb{G} \xleftarrow{\$} \{0,1\}^\ell$. That is, in this variant, calls of the form $\mathsf{Dist}_{\mathbb{G},g}(g_i, \epsilon, 1, K)$ are replaced with $\mathsf{Dist}_{\mathbb{G},g}(g_i, \epsilon, 1, f)$. Let β' be the probability that

$$\Pr[\mathsf{Fail}_i = 1 \mid \mathsf{Fail}_j = b_j \text{ for } j \in [t]/\{i\}] \quad (30)$$

in the experiment where we replace PRF_K with a random f. We will show $\beta' \le \epsilon + \mathsf{negl}(\lambda)$. By PRF security we have $\beta \le \beta' + \mathsf{negl}(\lambda)$, and thus Equation 29 will follow. The reason that we can

use PRF security here (despite the fact that K is given in the clear in ik) is that the procedure Dist may efficiently be computed via only blackbox access to PRF_K (resp., f alternatively) and that we evaluate PRF_K on inputs generated independently of K.

To bound the probability in Equation 30 we first define an event Bad which captures all the dependencies. Then we will argue that conditioned on $\overline{\mathsf{Bad}}$, the events $\{\mathsf{Fail}_j\}_{j\in[t]}$ are independent. To give some intuition, first notice that Fail_j holds iff

$$\mathsf{Dist}_{\mathbb{G},g}(g_{j,0},\epsilon,1,f) = \mathsf{Dist}_{\mathbb{G},g}(g_{j,0}g,\epsilon,1,f), \tag{31}$$

where recall that $g_{j,0} = \mathsf{h}^{r\alpha^{j-1}}$. Also, by definition of Dist, the outputs of the two distance functions of Equation 31 are only dependent on the outputs of f on group elements $\{g_{j,0}, g_{j,0}g, \ldots, g_{j,0}g^{T+1}\}$, where $T := [2\log_e(2/\epsilon)]/\epsilon$. Since f is chosen at random, we will have dependencies across Fail_j's only when the following event Bad holds:

- Bad: there are two distinct indices $j, h \in [t]$ such that $g_{j,0}$ and $g_{h,0}$ are $(T+1)$-close, where $T := [2\log_e(2/\epsilon)]/\epsilon$.

By Lemma 3.1

$$\Pr[\mathsf{Fail}_i = 1 \mid \overline{\mathsf{Bad}} \wedge \mathsf{Fail}_j = b_j \text{ for } j \in [t]/\{i\}] = \Pr[\mathsf{Fail}_i = 1] \leq \epsilon. \tag{32}$$

Below we will show $\Pr[\mathsf{Bad}] = \mathsf{negl}(\lambda)$, and this will allow us to conclude

$$\begin{aligned}\Pr[\mathsf{Fail}_i = 1 \mid \mathsf{Fail}_j = b_j \text{ for } j \in [t]/\{i\}] &\leq \Pr[\mathsf{Bad}] + \Pr[\mathsf{Fail}_i = 1 \wedge \overline{\mathsf{Bad}} \mid \mathsf{Fail}_j = b_j \text{ for } j \in [t]/\{i\}] \\ &\leq \mathsf{negl}(\lambda) + \Pr[\mathsf{Fail}_i = 1 \mid \overline{\mathsf{Bad}} \wedge \mathsf{Fail}_j = b_j \text{ for } j \in [t]/\{i\}] = \epsilon + \mathsf{negl}(\lambda), \end{aligned} \tag{33}$$

as desired. It only remains to show $\Pr[\mathsf{Bad}] = \mathsf{negl}(\lambda)$. Recall that $(g_{1,0}, g_{2,0}, \ldots, g_{t,0}) = (\mathsf{h}^r, \mathsf{h}^{r\alpha} \ldots, \mathsf{h}^{r\alpha^{t-1}})$. Notice that $\mathsf{h} \neq 1$ except with negligible probability, and thus h^r is statistically close to a uniformly random group element. By Lemma 2.4

$$(g_{1,0}, g_{2,0}, \ldots, g_{t,0}) = (\mathsf{h}^r, \mathsf{h}^{r\alpha} \ldots, \mathsf{h}^{r\alpha^{t-1}}) \stackrel{c}{\equiv} (g'_1, g'_2, \ldots, g'_t),$$

where g'_i's are random group elements. When replacing $\{g_{i,0}\}_{i\in[t]}$ with $\{g'_i\}_{i\in[t]}$, the probability of the event Bad becomes negligible. (This is because $T = \mathsf{poly}(\lambda)$.) Thus, the event Bad with $g_{j,0}$'s should also be negligible. □

Running time: We specify the running time for tolerated error $\epsilon = \frac{1}{n^c}$. For E, we can compute all the values $\mathsf{x} \cdot \mathbf{w'}_i$ altogether with total $O(n \log |I|)$ exponentiations by Lemma 4.4. Also, we spend $|I|$ exponentiations for computing w_i^ρ for $i \in [I]$. Thus, the total number of group operations is $O(n \log |I|)$ exponentiations.

6 Applications of Range-Trapdoor Hash

In this section we review the applications of our range-trapdoor hash scheme.

A two-round OT protocol consists of three PPT algorithms $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$, where $(\mathsf{OT}_1, \mathsf{OT}_3)$ are the two-stage algorithms run by the receiver, and OT_2 is run by the sender. We will be concerned

with honest-but-curious security (for both parties), and the corresponding definitions of security are standard. We use otr and ots to denote the receiver's and sender's message, respectively.

For an OT protocol OT where the size of each message of the sender is n, we call $\frac{|n|}{|\mathsf{ots}|}$ the *download rate* of the protocol. We say OT is *rate-1* if $\frac{|n|}{|\mathsf{ots}|}$ asymptotically approaches one.

As shown in [IP07], a rate-1 OT implies homomorphic encryption for branching programs with *semi-compactness*: the size of ciphertexts only grows with the depth of the program, as opposed to the size.

Let us first present the implication of our results with respect to rate-1 OTs. Implicit in the work of [DGI+19] is a construction of rate-1 OT from range trapdoor-hash schemes; see Constructions 5.1 and 5.2 of [DGI+19]. This result of [DGI+19], combined with Lemma 5.5, gives us the following.

Corollary 6.1 (Rate-1 OT with short receiver's message)**.** *Let G be a group scheme, where the size of a group element is $O(\lambda)$. Fix a message-size function $t(\lambda) \in \omega(\lambda)$. Assuming $2t$-power DDH, there is a rate-1 two-round honest-but-curios OT protocol with sender's input $(\mathsf{m}_0, \mathsf{m}_1) \in (\{0,1\}^t, \{0,1\}^t)$ and receiver's input $b \in \{0,1\}$, where the receiver's message otr consists of $O(t)$ group elements.*

Comparison to [DGI+19]. The work of [DGI+19] gives a DDH-based rate-1 OT, where in the parameter regime of Lemma 6.1, otr consists of $O(t^2)$ group elements. Our efficiency improvement stems from shorter evaluation keys: for a range set I, our scheme's evaluation key contains $O(n)$ group elements, as opposed to $O(n|I|)$ group elements given by [DGI+19]. See Lemma 5.3.

Improving upload rate. As noted in [DGI+19], asymptotically speaking, one may make the length of $|\mathsf{otr}|$ as close as possible to $|\mathsf{m}_0|$ (i.e., achieving *upload rate* 1, defined as $|\mathsf{m}_0|/|\mathsf{otr}|$) by re-using otr and making the input size of the sender larger. For example, assuming $|\mathsf{m}_0| = |\mathsf{m}_1| = O(\lambda^2)$, one may give a two-round OT based on DDH with both download and upload rates being 1. However, in concrete applications (e.g., homomorphic encryption for branching programs), the OT ends up being applied on sender's messages of much smaller asymptotic size, and thus improving the efficiency for this smaller regime leads to efficiency improvements in those applications.

Homomorphic encryption for branching programs with shorter ciphertexts. Ishai and Paskin [IP07] show how to build semi-compact homomorphic encryption for bounded-depth branching programs from rate-1 OT. Semi-compact means that the size of a ciphertexts grows only with the depth and the input size, and is independent of the program size otherwise. For the OT protocol, let $\mathsf{size}_r(\lambda, n)$ denote the size of otr when the length of each of sender's message is n. Assuming the input size is n and the depth of the branching program is at most d, the size of a ciphertext is $nd \times \mathsf{size}_r(\lambda, t)$, where $t \in O(\lambda)$. The result of [DGI+19] gives a DDH-based semi-compact encryption for branching programs with ciphertexts consisting of $O(\lambda^2 nd)$ group elements. Applying Corollary 6.1, our ciphertexts will contain $O(\lambda nd)$ group elements.

Corollary 6.2. *Assuming t-power DDH, there exists a PKE scheme for branching programs of depth d and input size n, where a ciphertext consists of $O(\lambda nd)$ group elements.*

Private information retrieval (PIR) with improved communication. A PIR protocol involves a server, holding $N = 2^d$ blocks $(m_1, \ldots, m_N)$, each of length β, and a client, holding an

index $i \in [N]$. The goal is to allow the client to retrieve m_i while keeping i hidden from the server. We would like to achieve this while minimizing communication complexity. Ishai and Paskin [IP07] gives a two-round block single-server PIR (one message from each side), achieving download rate 1, from rate-1 OT achieving. The download rate of a PIR is defined as the ratio between the server's message and β. The size of the client's message is $O(\mathsf{size}_r(\lambda, \beta) \log N)$, where recall that size_r denotes the size parameter of the receiver's message in the underlying OT protocol. If $\beta = O(\lambda)$, then under DDH, the rate-1 OT of [DGI+19] gives rise to a PIR, where the client's message consists of $O(\lambda^2 \log N)$ group elements. Using Corollary 6.1 and under the power DDH assumption, the client's message will have $O(\lambda \log N)$ group elements.

References

[AHI11] B. Applebaum, D. Harnik, and Y. Ishai. Semantic security under related-key attacks and applications. In *ICS 2011*, pages 45–60, Tsinghua University, Beijing, China, January 7–9, 2011. Tsinghua University Press. 4, 9

[BDD+00] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*. SIAM, 2000. 17

[BFOR08] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *CRYPTO 2008*, *LNCS* 5157, pages 360–378, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 2

[BGI16] E. Boyle, N. Gilboa, and Y. Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO 2016, Part I*, *LNCS* 9814, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 7, 11, 12

[BHY09] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT 2009*, *LNCS* 5479, pages 1–35, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany. 2

[BLSV18] Z. Brakerski, A. Lombardi, G. Segev, and V. Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In *EUROCRYPT 2018, Part I*, *LNCS* 10820, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 5

[BMZ19] J. Bartusek, F. Ma, and M. Zhandry. The distinction between fixed and random generators in group-based assumptions. In *CRYPTO 2019, Part II*, LNCS, pages 801–830, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 4

[BW10] X. Boyen and B. Waters. Shrinking the keys of discrete-log-type lossy trapdoor functions. In *ACNS 10*, *LNCS* 6123, pages 35–52, Beijing, China, June 22–25, 2010. Springer, Heidelberg, Germany. 3, 5

[CDD+16] I. Cascudo, I. Damgård, B. David, N. Döttling, and J. B. Nielsen. Rate-1, linear time and additively homomorphic UC commitments. In *CRYPTO 2016, Part III*, *LNCS* 9816, pages 179–207, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 11

[Cha04] Y.-C. Chang. Single database private information retrieval with logarithmic communication. In *ACISP 04*, *LNCS* 3108, pages 50–61, Sydney, NSW, Australia, July 13–15, 2004. Springer, Heidelberg, Germany. 5

[CMS99] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT'99*, *LNCS* 1592, pages 402–414, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 5

[CNs07] J. Camenisch, G. Neven, and a. shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT 2007*, *LNCS* 4515, pages 573–590, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany. 4, 9

[DG17a] N. Döttling and S. Garg. From selective IBE to full IBE and selective HIBE. In *TCC 2017, Part I*, *LNCS* 10677, pages 372–408, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. 5

[DG17b] N. Döttling and S. Garg. Identity-based encryption from the Diffie-Hellman assumption. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 5

[DGHM18] N. Döttling, S. Garg, M. Hajiabadi, and D. Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In *PKC 2018, Part I*, *LNCS* 10769, pages 3–31, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. 5

[DGI+19] N. Döttling, S. Garg, Y. Ishai, G. Malavolta, T. Mour, and R. Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO 2019, Part III*, LNCS, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 2, 3, 4, 7, 11, 17, 18, 19, 22, 23

[FGK+10] D. M. Freeman, O. Goldreich, E. Kiltz, A. Rosen, and G. Segev. More constructions of lossy and correlation-secure trapdoor functions. In *PKC 2010*, *LNCS* 6056, pages 279–295, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany. 5

[GGH19] S. Garg, R. Gay, and M. Hajiabadi. New techniques for efficient trapdoor functions and applications. In *EUROCRYPT 2019, Part III*, LNCS, pages 33–63, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. 2, 5, 7

[GH18] S. Garg and M. Hajiabadi. Trapdoor functions from the computational Diffie-Hellman assumption. In *CRYPTO 2018, Part II*, *LNCS* 10992, pages 362–391, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 2, 5

[GI05] V. Guruswami and P. Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005. 11

[IP07] Y. Ishai and A. Paskin. Evaluating branching programs on encrypted data. In *TCC 2007*, *LNCS* 4392, pages 575–594, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany. 2, 4, 22, 23

[KMT19] F. Kitagawa, T. Matsuda, and K. Tanaka. CCA security and trapdoor functions via key-dependent-message security. In *CRYPTO 2019, Part III*, LNCS, pages 33–64, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 5

[KW19] V. Koppula and B. Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In *CRYPTO 2019, Part II*, LNCS, pages 671–700, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 5

[Lip05] H. Lipmaa. An oblivious transfer protocol with log-squared communication. In *ISC 2005*, *LNCS* 3650, pages 314–328, Singapore, September 20–23, 2005. Springer, Heidelberg, Germany. 5

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, *LNCS* 6110, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. 5

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT 2013*, *LNCS* 7881, pages 35–54, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. 5

[LQR+19] A. Lombardi, W. Quach, R. D. Rothblum, D. Wichs, and D. J. Wu. New constructions of reusable designated-verifier NIZKs. In *CRYPTO 2019, Part III*, LNCS, pages 670–700, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 2, 5

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. 9

[OS07] R. Ostrovsky and W. E. Skeith III. A survey of single-database private information retrieval: Techniques and applications (invited talk). In *PKC 2007*, *LNCS* 4450, pages 393–411, Beijing, China, April 16–20, 2007. Springer, Heidelberg, Germany. 5

[PW08] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *40th ACM STOC*, pages 187–196, Victoria, BC, Canada, May 17–20, 2008. ACM Press. 2, 4, 5, 8

[PW11] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6):1803–1844, 2011. 8

[RS09] A. Rosen and G. Segev. Chosen-ciphertext security via correlated products. In *TCC 2009*, *LNCS* 5444, pages 419–436. Springer, Heidelberg, Germany, March 15–17, 2009. 5