Rate-1 Trapdoor Functions from the Diffie-Hellman Problem

Nico Döttling[*] Sanjam Garg[†] Mohammad Hajiabadi[‡] Kevin Liu[§]
Giulio Malavolta[¶]

September 20, 2019

Abstract

Trapdoor functions (TDFs) are one of the fundamental building blocks in cryptography. Studying the underlying assumptions and the efficiency of the resulting instantiations is therefore of both theoretical and practical interest. In this work we improve the input-to-image rate of TDFs based on the Diffie-Hellman problem. Specifically, we present:

(a) A rate-1 TDF from the computational Diffie-Hellman (CDH) assumption, improving the result of Garg, Gay, and Hajiabadi [EUROCRYPT 2019], which achieved linear-size outputs but with large constants. Our techniques combine non-binary alphabets and high-rate error-correcting codes over large fields.

(b) A rate-1 deterministic public-key encryption satisfying block-source security from the decisional Diffie-Hellman (DDH) assumption. While this question was recently settled by Döttling et al. [CRYPTO 2019], our scheme is conceptually simpler and *concretely more efficient*. We demonstrate this fact by implementing our construction.

1 Introduction

Trapdoor functions (TDFs) are the public-key variant of the notion of one-way functions. Informally, TDFs are (families of) one-to-one functions, where each function can be computed in the forward direction using the *index key*, and in the backward direction using a corresponding *trapdoor key*. Moreover, without knowledge of a trapdoor, a randomly chosen function should be one-way. Trapdoor functions, or extensions thereof such as lossy TDFs or deterministic public-key encryption, have important applications in the construction of primitives with CCA security, selective-opening security, and more recently in the context of designated-verifier non-interactive zero knowledge [BFOR08, BHY09, BBN$^+$09, MY10, BCPT13, LQR$^+$19].

[*]CISPA Helmholtz Center for Information Security

[†]University of California, Berkeley. 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

[¶]Simons Institute for the Theory of Computing. Part of this work was done while the author was at Carnegie Mellon University.

A series of works, some quite recent, have shown how to build TDFs and related primitives based on almost any specific assumptions from which public-key encryption (PKE) is known [PW08, FGK+10, PW11, Wee12, GH18, GGH19]. However, all these constructions are less efficient than those of PKE from the corresponding assumptions, in particular with respect to the sizes of public-keys and ciphertexts. For instance, we have constructions of PKE for which ciphertext expansion factors are small constants, sometimes even approaching 1. Yet, the situation for TDFs is different: All TDFs either have quadratic ciphertext expansions or linear expansions with large constants.

In this work we build TDFs and deterministic-encryption schemes with rates approaching 1 based on standard assumptions in cyclic groups, specifically the Computational Diffie Hellman (CDH) and Decisional Diffie Hellman (DDH) assumptions. Concretely, for an image y of an input x, the ratio $|\mathsf{x}|/|\mathsf{y}|$ approaches 1 as $|\mathsf{x}|$ grows. The first TDF constructions based on DDH [PW08, FGK+10] resulted in schemes in which the size of the image is quadratic in the input size. In a nutshell, the optimized TDF construction of Peikert and Waters, due to [FGK+10], computes a linear function *in the exponent* on a binary encoding of the input. In particular, 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, then we can encode any column vector X of bits by computing $\mathbf{M} \cdot \mathsf{X}$ in the exponent. This will allow for inversion if one possesses $\mathbf{M}^{-1}$. We can argue lossiness in a very elegant way by making the matrix $\mathbf{M}$ rank-deficient. On the downside however, we need to spend an entire group element in the output for each input bit, resulting in an expansion factor of $\Omega(\lambda)$.

A recent result of Garg, Gay and Hajiabadi [GGH19] shows how to construct linearly-expanding TDFs and DE schemes based on CDH or DDH. In particular, they give schemes in which the image expansion ratio is $O(1)$. However, this linear expansion hides big constants — a rough estimate of the constant is at least 20. At a high level, the constructions of [GGH19] achieve linear-expansion rates via the following two steps:

(a) For some constant c, first build a so-called local TDF, in which the inversion algorithm for every coordinate of the input either manages to recover the underlying bit correctly or outputs $\perp$, the latter happening with probability at most $1/2^c$.

(b) Boost correctness of local TDFs by applying erasure-correcting codes.

Their local TDFs from step (a) already incur an expansion factor of at least $2c$. Also, since erasure corrections for strings over $\mathbb{F}_2$ can tolerate only relatively small erasure rates (i.e., the ratio between the maximum number of tolerated erasures and the total length), they have to choose the constant c bit enough in Step (a) — at least 10.

The problem of rate-1 (lossy) TDFs from DDH was recently resolved in the work of Döttling et al. [DGI+19], who presented a construction based on the interplay of [GGH19] and techniques developed in the context of homomorphic secret sharing [BGI16]. Their approach however results in large index keys and does not appear to extend to the more challenging CDH settings.

Our Results. In this work, we show how to build rate-1 TDFs based on CDH or DDH, satisfying stronger properties such as block-source deterministic-encryption security in the sense of [BBO07, BFO08, BFOR08].[1] This notion of security requires that the deterministic encryptions of any two

[1]We mention that building rate-1 TDFs satisfying one-wayness alone is trivial. If a TDF TDF maps n-bit inputs to n^c-bit outputs, then define a second TDF whose input is of the form $(\mathsf{x} \in \{0,1\}^n, \mathsf{x}' \in \{0,1\}^{n^{c+1}})$, and the output is $(\mathsf{TDF}(\mathsf{x}), \mathsf{x}')$. While this trivial construction achieves rate-1, it destroys stronger properties such as deterministic-encryption security.

distributions each having high min-entropy (more than a threshold k) should be computationally indistinguishable. Ideally, we want $k << n$, where n is the bit length of the input.

At a high-level, our CDH-based construction deviates from the paradigm of [GGH19] by parsing the input into elements from a poly-sized field $\mathbb{F}$ (i.e., $|\mathbb{F}| = \mathsf{poly}(\lambda)$). Then for every block $\mathsf{B}_i \in \mathbb{F}$ of the input, we provide a corresponding "hinting" block O_i in the output of almost equal size. We then show how to perform inversion in a way which allows us to recover all except a $1/\mathsf{poly}_1(\lambda)$ fraction of the input blocks, for some polynomial poly_1. By choosing an appropriate error-correcting code over $\mathbb{F}$ and by choosing poly_1 appropriately based on $|\mathbb{F}|$, we are able to achieve rate 1. The main technical novelty of our work lies in providing the hints in a succinct way. See Section 1.1 for more details.

Under the DDH assumptions, we give a more direct rate-1 construction without the need of relying on error-correction techniques. For an input $\mathsf{x} \in \{0,1\}^n$, the output of the TDF contains only one group element plus exactly n bits. The construction has perfect correctness (i.e., can be inverted with probability 1), is conceptually simple, and is *concretely efficient*. We show this by providing a proof-concept-implementation in Python. Our implementation confirms our expectation of having short ciphertexts and relatively fast encryption/decryption times. Both encryptions and decryption times take less than a second on inputs of 128 Bytes (1024 bits).

Comparison with [DGI+19]. The work of [DGI+19] also shows how to build lossy TDFs (and deterministic encryption) based on DDH achieving rate 1 as in our construction. However, our construction achieves shorter public keys, saving an additive factor of at least $3n^2$ group elements, and is much simpler. In particular, the construction of [DGI+19] relies on non-trivial techniques such as those developed in the context of homomorphic secret sharing [BGI16] as well as error-correcting code type techniques. We rely on neither of these tools.

Open Problems. Our rate-1 primitives only provide CPA security. It would be interesting to see if techniques from [GGH19], along with those developed in this work, yield a rate-1 CCA primitive. One challenge is that in [GGH19] the (constant) multiplicative overhead of ciphertexts in the CCA case is much larger than the CPA case. In particular, our current techniques do not appear to naturally yield a rate-1 CCA primitive. We leave this as an open problem.

1.1 Technical Overview

In the following we provide an informal overview of the techniques developed in this work. We first discuss how to construct a CDH-based trapdoor function with rate 1, then we turn our attention to the DDH-based settings.

The Basic Building Block. The starting point of this work is the following group-based hash function, which maps $\{0,1\}^n$ into a group $\mathbb{G}$

$$\mathsf{Hash}(\mathsf{k}, \mathsf{x}) := \prod_{j=1}^{n} g_{j,\mathsf{x}_j}$$

where the key

$$\mathsf{k} := \begin{pmatrix} g_{1,1}, & g_{2,1}, & \dots, & g_{n,1} \\ g_{1,2}, & g_{2,2}, & \dots, & g_{n,2} \end{pmatrix} \xleftarrow{\$} \mathbb{G}^{2\times n}.$$

is chosen uniformly at random and $\mathsf{x} \in \{0,1\}^n$ is the input. By choosing n larger than the representation size of a group element in $\mathbb{G}$, this function becomes compressing. This surprisingly powerful function plays a central role in recent constructions of identity based encryption [DG17b], trapdoor functions [GH18], deterministic encryption and lossy trapdoor functions [GGH19].

In a first step, we increase the alphabet size of the input x, i.e instead of taking x from $\{0,1\}^n$, we take it from Σ^n for an alphabet $\Sigma := \{1, \dots, \sigma\}$ of size $\sigma = \mathsf{poly}(\lambda)$. While the definition of the function Hash is unchanged, we need to account for the increased alphabet size by sampling the key as

$$\mathsf{k} := \begin{pmatrix} g_{1,1}, & g_{2,1}, & \dots, & g_{n,1} \\ g_{1,2}, & g_{2,2}, & \dots, & g_{n,2} \\ \dots, & \dots, & \dots, & \dots \\ g_{1,\sigma}, & g_{2,\sigma}, & \dots, & g_{n,\sigma} \end{pmatrix} \xleftarrow{\$} \mathbb{G}^{\sigma \times n}.$$

The main effect of this modification for now is that the size of the key is increased by a σ factor. While this modification seems insignificant at first, it will be instrumental in achieving rate 1.

Adding the Encryption. We now show how this function can be augmented with the encryption functionality, using techniques of [GGH19]. Let $\mathsf{y} := \mathsf{Hash}(\mathsf{k}, \mathsf{x})$, for a certain input $\mathsf{x} \in \Sigma^n$, our objective is to design an encryption algorithm such that a ciphertext encrypted under an index $i \in [n]$, a symbol $f \in \Sigma$ and y, can be decrypted with the knowledge of x only if $\mathsf{x}_i = f$. This is done by sampling a uniform $\rho \xleftarrow{\$} \mathbb{Z}_p$ and publishing

$$\mathsf{ct}_{i,f} := \begin{pmatrix} g_{1,1}^{\rho}, & g_{2,1}^{\rho}, & \dots, & \perp, & \dots, & g_{n,1}^{\rho} \\ \dots, & \dots, & \dots, & \dots, & \dots, & \dots \\ g_{1,f}^{\rho}, & g_{2,f}^{\rho}, & \dots, & g_{i,f}^{\rho}, & \dots, & g_{n,f}^{\rho} \\ \dots, & \dots, & \dots, & \dots, & \dots, & \dots \\ g_{1,\sigma}^{\rho}, & g_{2,\sigma}^{\rho}, & \dots, & \perp, & \dots, & g_{n,\sigma}^{\rho} \end{pmatrix},$$

as the ciphertext, and letting y^ρ be the underlying (secret) encapsulated value. Given x, anyone can recover y^ρ by simply computing

$$\mathsf{y}^\rho := \prod_{j=1}^{m} g_{j,\mathsf{x}_j}^{\rho}.$$

It is not hard to show that recovering the y^ρ if $\mathsf{x}_j \neq f$ is as hard as solving the Diffie-Hellman problem.

Constructing Trapdoor Functions. The key observation of [GH18] (later improved in [GGH19]) is that the same value can be recovered from y using the trapdoor ρ, without the knowledge of x. This allows us to use the above structure to construct a trapdoor function by sampling the trapdoor as a matrix

$$\mathsf{tk} := \begin{pmatrix} \rho_{1,1}, & \rho_{2,1}, & \dots, & \rho_{n,1} \\ \rho_{1,2}, & \rho_{2,2}, & \dots, & \rho_{n,2} \\ \dots, & \dots, & \dots, & \dots \\ \rho_{1,\sigma}, & \rho_{2,\sigma}, & \dots, & \rho_{n,\sigma} \end{pmatrix} \xleftarrow{\$} \mathbb{Z}_p^{\sigma \times n}$$

and setting the index key as

$$\mathsf{ik} := \mathsf{k}, \begin{pmatrix} \mathsf{ct}_{1,1}, & \mathsf{ct}_{2,1}, & \dots, & \mathsf{ct}_{n,1} \\ \mathsf{ct}_{1,2}, & \mathsf{ct}_{2,2}, & \dots, & \mathsf{ct}_{n,2} \\ \dots, & \dots, & \dots, & \dots \\ \mathsf{ct}_{1,\sigma}, & \mathsf{ct}_{2,\sigma}, & \dots, & \mathsf{ct}_{n,\sigma} \end{pmatrix}, \begin{pmatrix} a_{1,1}, & a_{2,1}, & \dots, & a_{n,1} \\ a_{1,2}, & a_{2,2}, & \dots, & a_{n,2} \\ \dots, & \dots, & \dots, & \dots \\ a_{1,\sigma}, & a_{2,\sigma}, & \dots, & a_{n,\sigma} \end{pmatrix}$$

where k and $\mathsf{ct}_{i,f}$ are defined as above and each $a_{i,f} \overset{\$}{\leftarrow} \mathbb{G}$ is a random group elements. The purpose of these random elements is to shift the (negligible) inversion error from the random choice of x to the random choice of ik (see [GGH19] for a detailed discussion). Given an input x, the output of the trapdoor function is defined to be

$$\mathsf{u} := (\mathsf{y}, v_1 := \mathsf{y}^{\rho_{1,\mathsf{x}_1}} \oplus a_{1,\mathsf{x}_1}, \dots, v_n := \mathsf{y}^{\rho_{n,\mathsf{x}_n}} \oplus a_{n,\mathsf{x}_n}).$$

Note that, as discussed before, this computation can be performed without the trapdoor tk. On the other hand, the function can be easily inverted with the knowledge of tk (and without x) by simply recomputing each $\mathsf{y}^{\rho_{i,f}} \oplus a_{i,f}$ and comparing it with v_i. If it matches, then the i-th symbol is set to f. While this gives us a trapdoor function, its rate is far from 1: To encode one symbol $\mathsf{x}_i \in \Sigma$, we need to spend one group element $v_i \in \mathbb{G}$.

Boosting the Rate. However, we can improve the rate of this construction with a surprisingly simple idea. Namely, we will use a hardcore function (in the sense of [GL89]) H to hash the element $\mathsf{y}^{\rho_{i,\mathsf{x}_i}}$ into a polynomial-size domain $\{0,1\}^w$ and sample $a_{i,f}$ from the same domain. Image values of the function now look as follows

$$\mathsf{u} := (\mathsf{y}, v_1 := \mathsf{H}(\mathsf{y}^{\rho_{1,\mathsf{x}_1}}) \oplus a_{1,\mathsf{x}_1}, \dots, v_n := \mathsf{H}(\mathsf{y}^{\rho_{n,\mathsf{x}_n}}) \oplus a_{n,\mathsf{x}_n}).$$

Inversion is done as before: Given y and the trapdoor key tk one can check whether $v_i \overset{?}{=} \mathsf{H}(\mathsf{y}^{\rho_{i,f}}) \oplus a_{i,f}$ for all possible $f \in \Sigma$. If one finds a unique $f \in \Sigma$ with this property, then it must hold that $\mathsf{x}_i = f$. However, as $\{0,1\}^w$ is a domain of polynomial size, collisions can and will occur. That is, there can occur *false positives* $f' \neq \mathsf{x}_i$ which satisfy the above condition. Given that such collisions are not too frequent, we can protect against them by pre-processing x with a suitable code which also has high rate. Our analysis shows that the number of indices i at which such collisions occur is at most $2n \cdot \sigma/2^w$, where $\sigma = |\Sigma|$ is the size of the alphabet.

Achieving Rate 1. The crucial observation now is that we can choose Σ and $\{0,1\}^w$ in such a way that $\sigma/2^w$ is sublinear, but at the same time $\log(\sigma)/w$ approaches 1. Note that $\log(\sigma)/w$ is the rate at which we encode a symbol $\mathsf{x}_i \in \Sigma$ by $\mathsf{H}(\mathsf{y}^{\rho_{i,\mathsf{x}_i}}) \oplus a_{i,\mathsf{x}_i}$. This is e.g. achieved by choosing $\sigma \geq \lambda$ and $2^w = \sigma \cdot \log(\lambda)$. This choice gives us $\sigma/2^w \leq 1/\log(\lambda)$ and

$$\frac{\log(\sigma)}{w} = \frac{\log(\sigma)}{\log(\sigma) + \log\log(\lambda)} = 1 - \frac{\log\log(\lambda)}{\log(\sigma) + \log\log(\lambda)} \geq 1 - \frac{\log\log(\lambda)}{\log(\lambda)},$$

which approaches 1. Finally, we can pre-process the input x with a code which can handle a $2 \cdot \sigma/2^w = 2/\log(\lambda)$ fraction of erasures, such as a $[n, n - 2n/\log(\lambda), 2/\log(\lambda) + 1]$ Reed Solomon code over a field Σ of size $\sigma \geq n$. This code has rate $1 - 2/\log(\lambda)$. Concluding, the image (ignoring the group element y which causes only an additive overhead) encodes a message x at rate

$$\left(1 - \frac{2}{\log(\lambda)}\right) \cdot \left(1 - \frac{\log\log(\lambda)}{\log(\lambda)}\right) \geq 1 - \Omega\left(\frac{\log\log(\lambda)}{\log(\lambda)}\right).$$

The last question to address is how to instantiate H to extract enough randomness from a CDH instance. By our choice of parameters above, $w = O(\log(\lambda))$ random bits suffice, which allows us to use the standard Goldreich-Levin [GL89] hardcore function.

DDH-Based Deterministic Encryption (DE). Recall that we say a TDF with input space $\{0,1\}^n$ has (k,n)-CPA-security if the evaluations of any two distributions with min-entropy at least k result in computationally-indistinguishable distributions. We show how to realize this notion in a very simple and ciphertext-compact way using DDH.

The index key of our TDF consists of a random vector $\mathbf{g} \in \mathbb{G}^n$ together with n vectors $\{\mathbf{g}_i \in \mathbb{G}^n\}_{i\in[n]}$, where each $\mathbf{g}_i$ is an element-wise exponentiation of $\mathbf{g}$ to a random power ρ_i. To evaluate an input $\mathsf{x} \in \{0,1\}^n$, we return a group element $g' := \mathsf{x} \cdot \mathbf{g}$ (where $\cdot$ denotes the hash $\prod_{j=1}^n g_j^{\mathsf{x}_j}$), as well as an encoded bit $b_i := \mathsf{BL}(\mathsf{x} \cdot \mathbf{g}_i) \oplus \mathsf{x}_i$ for the i-th bit of the input. Here $\mathsf{BL}\colon \mathbb{G} \to \{0,1\}$ is a balanced function, meaning that the output of $\mathsf{BL}(g_u)$ on a uniformly-random g_u is a uniformly-random bit. Inversion can be performed by knowing all the exponents ρ_i's.

We show if $k \geq \log p + \omega(\log\lambda)$ — where p is the size of the group — then we have (k,n)-CPA security. To argue this, first recall that an index key is of the form $(\mathbf{g}, \mathbf{g}_1, \ldots, \mathbf{g}_n)$, where each $\mathbf{g}_i$ is an exponentiation of $\mathbf{g}$. Say two x and x' are siblings if $\mathsf{x} \cdot \mathbf{g} = \mathsf{x}' \cdot \mathbf{g}$. (That is, if both result in the same group element in the output.) We show that for any $\mathsf{x} \in \{0,1\}^n$, one may sample the $\mathbf{g}_i$ components of the index key in a manner correlated with x to get a correlated ik^* in such a way that:

1. $(\mathsf{x}, \mathsf{ik}^*) \overset{c}{\equiv} (\mathsf{x}, \mathsf{ik})$, where ik is a real index key; and

2. ik^* will lose information w.r.t. all siblings of x. That is, if x' is a sibling of x, then $\mathsf{TDF.F}(\mathsf{ik}^*, \mathsf{x}) = \mathsf{TDF.F}(\mathsf{ik}^*, \mathsf{x}')$.

3. The joint distribution $(\mathsf{ik}^*, \mathsf{TDF.F}(\mathsf{ik}^*, \mathsf{x}))$ can be formed just by knowing $g' := \mathsf{x} \cdot \mathbf{g}$, and especially without knowing x.

Let us first sketch why the above properties imply DE security. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be the underlying high-entropy distributions. Let $\mathsf{x}_b \overset{\$}{\leftarrow} \mathcal{D}_b$, $\mathbf{g} \overset{\$}{\leftarrow} \mathbb{G}^n$ and $g'_b = \mathsf{x}_b \cdot \mathbf{g}$. Also, let ik^*_b be the corresponding correlated index key which by Item 3 can be formed just by knowing g'_b. By Item 1 we have $(\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x}_b)) \overset{c}{\equiv} (\mathsf{ik}^*_b, \mathsf{TDF.F}(\mathsf{ik}^*_b, \mathsf{x}_b))$. Now since by Item 3 the joint distribution $(\mathsf{ik}^*_b, \mathsf{TDF.F}(\mathsf{ik}^*_b, \mathsf{x}_b))$ can be sampled just by knowing g'_b and since we have $g'_0 \overset{s}{\equiv} g'_1$ (by the leftover hash lemma), we have $(\mathsf{ik}^*_0, \mathsf{TDF.F}(\mathsf{ik}^*_0, \mathsf{x}_0)) \overset{s}{\equiv} (\mathsf{ik}^*_1, \mathsf{TDF.F}(\mathsf{ik}^*_1, \mathsf{x}_1))$, establishing the desired security.

Now let us explain how to sample such "lossy" index key ik^* for x just by knowing $g_c = \mathsf{x} \cdot \mathbf{g}$. We form $\mathsf{ik}^* := (\mathbf{g}, \mathbf{g}^*_1, \ldots, \mathbf{g}^*_n)$, where each $\mathbf{g}^*_i$ is formed exactly as in $\mathbf{g}_i$, except that we multiply the i-th element of the vector $\mathbf{g}_i := \mathbf{g}^{\rho_i}$ with a random group element g'_i which satisfies $\mathsf{BL}(g_c^{\rho_i}) = 1 \oplus \mathsf{BL}(g_c^{\rho_i} \cdot g'_i)$. Namely, each $\mathbf{g}^*_i$ is an "almost" exponentiation of $\mathbf{g}$ in that we tamper with the i-th element of the resulting exponentiated vector.

Using simple inspection we can verify that Property 2 follows by the particular way in which ik^* is sampled. Also, the way ik^* is defined allows us to sample the joint distribution $(\mathsf{ik}^*, \mathsf{TDF.F}(\mathsf{ik}^*, \mathsf{x}))$ just by knowing g_c and ρ_i's, establishing Property 3. Finally, via a sequence of hybrids, we show how to establish Property 1 based on DDH.

2 Preliminaries

We denote the security parameter by λ. We use $\overset{c}{\equiv}$ to denote computational indistinguishability between two distributions and use $\equiv$ to denote two distributions are identical. We write $\overset{s}{\equiv}$ for statistical indistinguishability and we write $\approx_\varepsilon$ to denote that two distributions are statistically close, within statistical distance ε. For a distribution $\mathcal{S}$ we use $x \overset{\$}{\leftarrow} \mathcal{S}$ 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 $x \overset{\$}{\leftarrow} \mathsf{S}$ to indicate that x is chosen uniformly at random from S. The set $\{1, \ldots, n\}$ is often abbreviated as $[n]$. We say that a machine is PPT if it runs in probabilistic polynomial-time.

The min-entropy of a distribution $\mathcal{S}$ is $\mathsf{H}_\infty(\mathcal{S}) \triangleq -\log(\max_{\mathsf{x}} \Pr[\mathcal{S} = \mathsf{x}])$. For a finite alphabet Σ, we say a distribution $\mathcal{S}$ is a *k-source* over Σ^n if $\mathsf{H}_\infty(\mathcal{S}) \geq k$. When the alphabet Σ is clear from context, we say $\mathcal{S}$ is a (k, n)-source.

Lemma 2.1 (Chernoff Inequality)**.** *Let X be binomially distributed with parameters $n \in \mathbb{N}$ and $p \in [0, 1]$. Let $p' > p$. Then*

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

Lemma 2.2 (Leftover Hash Lemma [ILL89])**.** *Let $\mathcal{X}$ be a random variable over X and $h : \mathsf{S} \times \mathsf{X} \to \mathsf{Y}$ be a 2-universal hash function, where $|\mathsf{Y}| \leq 2^m$ for some $m > 0$. If $m \leq \mathsf{H}_\infty(\mathcal{X}) - 2\log\left(\frac{1}{\varepsilon}\right)$, then $(h(\mathcal{S}, \mathcal{X}), \mathcal{S}) \approx_\varepsilon (\mathcal{U}, \mathcal{S})$, where $\mathcal{S}$ is uniform over S and $\mathcal{U}$ is uniform over Y.*

Lemma 2.3 (Log-Many Bits Hardcore Functions [GL89])**.** *Let $\mathsf{f} \colon \{0, 1\}^n \to \{0, 1\}^m$ be an OWF with respect to a distribution $\mathcal{D}$. Let $\mathsf{B}_i \colon \{0, 1\}^n \times \{0, 1\}^{2n} \to \{0, 1\}$ be a function defined as $\mathsf{B}_i(\mathsf{x}, \mathsf{s}) := \langle \mathsf{x}, \mathsf{s}[i, i + n - 1]\rangle \mod 2$, where $\mathsf{s}[i, i + n - 1] := (\mathsf{s}_i, \ldots, \mathsf{s}_{i+n-1})$. Then for any constant $c > 0$, the function $\mathsf{H} \colon \{0, 1\}^n \times \{0, 1\}^{2n} \to \{0, 1\}^{c\lceil \log n\rceil}$ defined as*

$$\mathsf{H}(\mathsf{x}, \mathsf{s}) := (\mathsf{B}_1(\mathsf{x}, \mathsf{s}), \ldots, \mathsf{B}_{c\lceil \log n\rceil}(\mathsf{x}, \mathsf{s}))$$

is a hardcore function for f. That is, $(\mathsf{s}, \mathsf{f}(\mathsf{x}), \mathsf{H}(\mathsf{x}, \mathsf{s})) \overset{c}{\equiv} (\mathsf{s}, \mathsf{f}(\mathsf{x}), \mathsf{w})$, where $\mathsf{s} \overset{\$}{\leftarrow} \{0, 1\}^{2n}$, $\mathsf{x} \overset{\$}{\leftarrow} \mathcal{D}$ and $\mathsf{w} \overset{\$}{\leftarrow} \{0, 1\}^{c\lceil \log n\rceil}$.

2.1 Error Correcting Codes

For our constructions we will rely on efficiently correctable error correcting block codes. Fix a finite alphabets Σ, and two parameters k and m. We will represent codes by two efficient algorithms Encode and Decode, where Encode takes as input a message $\mathsf{x} = (\mathsf{x}_1, \ldots, \mathsf{x}_k) \in \Sigma^k$ and outputs a codeword $\mathsf{c} = (\mathsf{c}_1, \ldots, \mathsf{c}_m) \in \Sigma^m$. We refer to the support of the algorithm Encode as the code C. The algorithm Decode takes as input a string $\hat{\mathsf{c}} \in (\Sigma \cup \{\perp\})^m$ and outputs a message $\mathsf{x} \in \Sigma^k$ or $\perp$. We say that such a code jointly corrects r errors and s erasures, if it holds for every $\hat{\mathsf{c}} \in (\Sigma \cup \{\perp\})^m$

which can be obtained from a codeword $\mathsf{c} \in \mathsf{C}$ by changing at most r positions and erasing at most s positions that $\mathsf{Decode}(\hat{\mathsf{c}}) = \mathsf{Decode}(\mathsf{c})$.

The specific class of codes which we use in our constructions are Reed Solomon (RS) codes [RS60]. The alphabets of Reed Solomon codes are finite fields $\mathbb{F}_q$, and an $[m,k]$-RS code exists whenever $m \leq q$. The encoding procedure Encode of a $[m,k]$-RS code represents the message $\mathsf{x} \in \mathbb{F}_q^k$ as a polynomial P of degree $k-1$ over $\mathbb{F}_q$ via the coefficient embedding, and computes and outputs $(P(\xi_1), \ldots, P(\xi_m))$, where $\xi_1, \ldots, \xi_m$ are pairwise distinct elements of $\mathbb{F}_q$. There exists an efficient decoding algorithm, the so-called Berlekamp-Welch decoder [WB86], which can jointly decode r errors and s erasures given that $2r+s \leq m-k$. We say this RS code has minimum-distance $m-k+1$.

In abuse of notation, we will provide as input to the encoding algorithm a binary string, i.e. an element from $\{0,1\}^n$. For $\Sigma = \mathbb{F}_{2^\kappa}$, such a string $x \in \{0,1\}^n$ can be mapped to a string $\mathsf{x} \in \mathbb{F}_{2^k}$ by chopping x into blocks of length κ and letting each block represent an element of $\mathbb{F}_{2^\kappa}$. Finally, we will always assume that there is a canonical enumeration of the elements in Σ. This lets us identify each element in Σ with a corresponding element in the set $\{1, \ldots, \Sigma\}$.

2.2 Trapdoor Functions

We recall the definition of trapdoor function (TDFs).

Definition 2.4 (Trapdoor Functions). *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:***
$$\Pr_{(\mathsf{ik},\mathsf{tk}),\mathsf{x}}[\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})) \neq \mathsf{x}] = \mathsf{negl}(\lambda), \tag{1}$$
where $(\mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$ *and* $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$.
- ***One-wayness:*** *For any PPT adversary $\mathcal{A}$:* $\Pr[\mathcal{A}(\mathsf{ik}, \mathsf{u}) = \mathsf{x}] = \mathsf{negl}(\lambda)$*, where* $(\mathsf{ik}, *) \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$, $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ *and* $\mathsf{u} := \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})$.

We also define a stronger security property, called CPA block-source security*.*

- ***CPA-deterministic security:*** *We say* TDF *is (k,n)-CPA-secure if for any two (k,n)-sources $\mathcal{D}_0$ and $\mathcal{D}_1$:* $(\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathcal{D}_0)) \overset{c}{\equiv} (\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathcal{D}_1))$*, where* $(\mathsf{ik}, *) \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$.

We give the definition of rate for a TDF, which captures the asymptotic input-to-image ratio.

Definition 2.5 (Rate)**.** *A TDF* $(\mathsf{TDF.KG}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ *has rate* ρ *if for all* $\lambda \in \mathbb{N}$*, all polynomials* $n(\lambda)$*, all* ik *in the support of* $\mathsf{TDF.KG}(1^\lambda)$*, all inputs in* $\mathsf{x} \in \{0,1\}^{n(\lambda)}$*:*

$$\liminf_{\lambda \to \infty} \frac{n(\lambda)}{|\mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})|} = \rho.$$

2.3 The Diffie-Hellman Problems

We recall the classical Diffie-Hellman problem [DH76] both in its search and decisional version. 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. In favor of a simpler analysis, we consider groups such that $\log(p) = |g| = \lambda$.

Definition 2.6 (Diffie-Hellman Assumptions)**.** *We say* G *is CDH-hard if for any PPT adversary* $\mathcal{A}$ $\Pr[\mathcal{A}(\mathbb{G}, p, g, g^{a_1}, g^{a_2}) = g^{a_1 a_2}] = \mathsf{negl}(\lambda)$ *where* $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{G}(1^\lambda)$ *and* $(a_1, a_2) \xleftarrow{\$} \mathbb{Z}_p^2$*. We say* G *is DDH-hard if* $(\mathbb{G}, p, g, g^{a_1}, g^{a_2}, g^{a_3}) \overset{c}{\equiv} (\mathbb{G}, p, g, g^{a_1}, g^{a_2}, g^{a_1 a_2})$*, where* $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{G}(1^\lambda)$ *and* $(a_1, a_2, a_3) \xleftarrow{\$} \mathbb{Z}_p^3$*.*

3 Smooth Recyclable OWFE

We recall the definition of recyclable one-way function with encryption (OWFE) from [GH18]. The following definitions are taken almost in verbatim from [GGH19], except that we consider a generalized version of the primitive over any finite alphabets Σ. The notion of OWFE in turn builds on related notions known in the literature as (chameleon) hash encryption and its variants [DG17b, DG17a, BLSV18, DGHM18].

Definition 3.1 (Recyclable one-way function with encryption)**.** *Let* $\Sigma = \{1, \dots, \sigma\}$ *for some integer* σ*. A* w*-bit recyclable* (k, n)*-OWFE scheme consists of the PPT algorithms* Gen*,* Hash*,* Enc_1*,* Enc_2 *and* Dec *with the following syntax.*

- $\mathsf{Gen}(1^\lambda)$*: Takes the security parameter* 1^λ *and outputs a public parameter* k *(by tossing coins) for a function* $\mathsf{Hash}(\mathsf{k}, \cdot)$ *from* n *bits to* ν *bits.*
- $\mathsf{Hash}(\mathsf{k}, \mathsf{x})$*: Takes a public parameter* k *and a preimage* $\mathsf{x} \in \Sigma^n$*, and deterministically outputs an element* y*.*
- $\mathsf{Enc}_1(\mathsf{k}, (i, z); \rho)$*: Takes a public parameter* k*, an index* $i \in [n]$*, a word* $z \in \Sigma$ *and randomness* ρ*, and outputs a ciphertext* ct*. We implicitly assume that* ct *contains* (i, z)*.*
- $\mathsf{Enc}_2(\mathsf{k}, \mathsf{y}, (i, z); \rho)$*: Takes a public parameter* k*, a value* y*, an index* $i \in [n]$*, a word* $z \in \Sigma$ *and randomness* ρ*, and outputs a string* $\mathsf{e} \in \{0,1\}^w$*. Notice that unlike* Enc_1*, which does not take* y *as input, the algorithm* Enc_2 *does take* y *as input.*
- $\mathsf{Dec}(\mathsf{k}, \mathsf{ct}, \mathsf{x})$*: Takes a public parameter* k*, a ciphertext* ct *and a preimage* $\mathsf{x} \in \Sigma^n$*, and deterministically outputs a string* $\mathsf{e} \in \{0,1\}^w$*.*

We require the following properties.

- ***Correctness.*** *For any choice of* $\mathsf{k} \in \mathsf{Gen}(1^\lambda)$*, any index* $i \in [n]$*, any preimage* $\mathsf{x} \in \Sigma^n$ *and any randomness value* ρ*,*

$$\Pr[\mathsf{Enc}_2(\mathsf{k}, \mathsf{y}, (i, \mathsf{x}_i); \rho) = \mathsf{Dec}(\mathsf{k}, \mathsf{ct}, \mathsf{x})] = 1$$

  where $\mathsf{y} := \mathsf{Hash}(\mathsf{k}, \mathsf{x})$ *and* $\mathsf{ct} := \mathsf{Enc}_1(\mathsf{k}, (i, \mathsf{x}_i); \rho)$*.*

- (k, n)***-One-wayness:*** *For any* k*-source* $\mathcal{S}$ *over* Σ^n *and any PPT adversary* $\mathcal{A}$*:*

$$\Pr[\mathsf{Hash}(\mathsf{k}, \mathcal{A}(\mathsf{k}, \mathsf{y})) = \mathsf{y}] = \mathsf{negl}(\lambda),$$

  where $\mathsf{k} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$*,* $\mathsf{x} \xleftarrow{\$} \mathcal{S}$ *and* $\mathsf{y} := \mathsf{Hash}(\mathsf{k}, \mathsf{x})$*. If* $k = n$*, then we simply refer to an OWFE scheme (without specifying the parameters).*

- (k, n)***-Smoothness:*** *For any two* (k, n)*-sources* $\mathcal{S}_1$ *and* $\mathcal{S}_2$*:*

$$(\mathsf{k}, \mathsf{Hash}(\mathsf{k}, \mathsf{x}_1)) \overset{c}{\equiv} (\mathsf{k}, \mathsf{Hash}(\mathsf{k}, \mathsf{x}_2))$$

  where $\mathsf{k} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$*,* $\mathsf{x}_1 \xleftarrow{\$} \mathcal{S}_1$ *and* $\mathsf{x}_2 \xleftarrow{\$} \mathcal{S}_2$*.*

- ***Security for Encryption:*** *For any* $i \in [n]$*, any* $\mathsf{x} \in \Sigma^n$*, and any* $f \in \Sigma \setminus \{\mathsf{x}_i\}$*:*

$$(\mathsf{x}, \mathsf{k}, \mathsf{ct}, \mathsf{e}) \overset{c}{\equiv} (\mathsf{x}, \mathsf{k}, \mathsf{ct}, \mathsf{e}')$$

  where $\mathsf{k} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$*,* $\rho \xleftarrow{\$} \{0,1\}^*$*,* $\mathsf{ct} := \mathsf{Enc}_1(\mathsf{k}, (i, f); \rho)$*,* $\mathsf{e} := \mathsf{Enc}_2\big(\mathsf{k}, \mathsf{Hash}(\mathsf{k}, \mathsf{x}), (i, f); \rho\big)$ *and* $\mathsf{e}' \xleftarrow{\$} \{0,1\}^w$*.*

3.1 Smooth Recyclable OWFE from CDH

We generalize the recyclable OWFE from [GH18] to any finite alphabet Σ. Although this modification might look insignificant, it will be our main leverage to construct a rate-1 trapdoor function.

Construction 3.2 (Smooth recyclable OWFE from CDH)**.** *Let* G *be a CDH-hard group-generator scheme and let* $\Sigma := \{1, \ldots, \sigma\}$ *be a finite alphabet.*

- $\mathsf{Gen}(1^\lambda)$*: Sample* $(\mathbb{G}, p, g) \xleftarrow{\$} \mathbb{G}(1^\lambda)$*. For each* $j \in [n]$ *and* $f \in \Sigma$*, choose* $g_{j,f} \xleftarrow{\$} \mathbb{G}$*. Output*

$$\mathsf{k} := \begin{pmatrix} g_{1,1}, & g_{2,1}, & \ldots, & g_{n,1} \\ g_{1,2}, & g_{2,2}, & \ldots, & g_{n,2} \\ \ldots, & \ldots, & \ldots, & \ldots \\ g_{1,\sigma}, & g_{2,\sigma}, & \ldots, & g_{n,\sigma} \end{pmatrix}. \tag{2}$$

- $\mathsf{Hash}(\mathsf{k}, \mathsf{x})$*: Parse* k *as in Equation 2, and output* $\mathsf{y} := \prod_{j \in [n]} g_{j,\mathsf{x}_j}$*.*

- $\mathsf{Enc}_1(\mathsf{k}, (i, z); \rho)$*: Parse* k *as in Equation 2. Given the randomness* $\rho \xleftarrow{\$} \mathbb{Z}_p$*, proceed as follows:*
  - *For every* $j \in [n] \setminus \{i\}$*, and every* $f \in \Sigma$ *set* $c_{j,f} := g_{j,z}^{\rho}$*.*

  - *For every $f \in \Sigma \setminus \{z\}$ set $c_{i,f} := \perp$, then set $c_{i,z} := g_{i,z}^{\rho}$.*
  - *Output*

$$\mathsf{ct} := \begin{pmatrix} c_{1,1}, & c_{2,1}, & \dots, & c_{n,1} \\ c_{1,2}, & c_{2,2}, & \dots, & c_{n,2} \\ \dots, & \dots, & \dots, & \dots \\ c_{1,\sigma}, & c_{2,\sigma}, & \dots, & c_{n,\sigma} \end{pmatrix}. \tag{3}$$

- $\mathsf{Enc}_2(\mathsf{k}, (\mathsf{y}, i, z); \rho)$: *Given the randomness $\rho \xleftarrow{\$} \mathbb{Z}_p$, output $\mathsf{H}(\mathsf{y}^{\rho})$, where $\mathsf{H} : \mathbb{G} \to \{0,1\}^w$ denotes a hardcore function (e.g., the function from Lemma 2.3).*

- $\mathsf{Dec}(\mathsf{k}, \mathsf{ct}, \mathsf{x})$: *Parse ct as in Equation 3, and output $\mathsf{H}\left(\prod_{j \in [n]} c_{j,\mathsf{x}_j}\right)$.*

Correctness of the scheme is immediate. We now show that the construction satisfies all of the required security properties properties.

Theorem 3.3 (One-Wayness). *Let G generate a CDH-hard group, then for all $n \geq \omega(\log(p))$, Construction 3.2 is one-way.*

Proof. This is shown with a reduction to the discrete logarithm problem. On input a challenge random element $h \in \mathbb{G}$, sample a random pair of indices $i^* \xleftarrow{\$} [n]$ and $f^* \xleftarrow{\$} \Sigma$ and set $g_{i^*,f^*} := h$. For all $i \xleftarrow{\$} [n]$ and $f \xleftarrow{\$} \Sigma$, except for the pair (i^*, f^*), set $g_{i,f} := g^{r_{i,f}}$, for a uniform $r_{i,f} \xleftarrow{\$} \mathbb{Z}_p$. Define the public key as

$$\mathsf{k} := \begin{pmatrix} g_{1,1}, & g_{2,1}, & \dots, & g_{n,1} \\ g_{1,2}, & g_{2,2}, & \dots, & g_{n,2} \\ \dots, & \dots, & \dots, & \dots \\ g_{1,\sigma}, & g_{2,\sigma}, & \dots, & g_{n,\sigma} \end{pmatrix}.$$

Then sample a uniform $\mathsf{x} \xleftarrow{\$} \Sigma^n$ such that $\mathsf{x}_{i^*} \neq f^*$ and compute $\mathsf{y} := \prod_{j \in [n]} g_{j,\mathsf{x}_j}$. Give (k, y) to the adversary and receive some x'. By Lemma 2.2, $\mathsf{x}'_{i^*} = f^*$ with probability close to $1/\sigma$, which allows us to compute the discrete logarithm of h. □

Theorem 3.4 ((k,n)-Smoothness). *Let G generate a CDH-hard group and $k \geq \log p + \omega(\log \lambda)$, then Construction 3.2 is (k,n)-smooth.*

Proof. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two (k,n) sources. The smoothness is a direct consequence of Lemma 2.2. Namely, assuming $\mathsf{k} \xleftarrow{\$} \mathsf{Gen}(1^{\lambda})$, $\mathsf{x}_1 \xleftarrow{\$} \mathcal{S}_1$ and $\mathsf{x}_2 \xleftarrow{\$} \mathcal{S}_2$, since Hash is a 2-universal hash function, by Lemma 2.2 we know that the outputs of both $\mathsf{Hash}(\mathsf{k}, \mathsf{x}_1)$ and $\mathsf{Hash}(\mathsf{k}, \mathsf{x}_2)$ are statistically $\frac{1}{2^{\omega(\log \lambda)}}$ close to the uniform over $\mathbb{G}$, and hence negligibly close (statistically) to each other. □

Theorem 3.5 (Security for Encryption). *Let G generate a CDH-hard group, then Construction 3.2 is secure for encryption.*

Proof. Assume towards contradiction that there exists some i^*, x, and $f^* \neq \mathsf{x}_{i^*}$ such that an adversary can successfully distinguish on those input. Let (g, h_1, h_2, s) be a CDH challenge, where $s \in \{0,1\}^w$ is either a random string or the output of the hardcore function. For all $i \in [n] \setminus i^*$ and all $f \in \Sigma$ set $g_{i,f} := g^{r_{i,f}}$, for a uniform $r_{i,f} \xleftarrow{\$} \mathbb{Z}_p$. Similarly, for all $f \in \Sigma \setminus \mathsf{x}_{i^*}$ set $g_{i^*,f} := g^{r_{i^*,f}}$, for a uniform $r_{i^*,f} \xleftarrow{\$} \mathbb{Z}_p$. Finally set

$$g_{i^*,\mathsf{x}_{i^*}} := \frac{h_1}{\prod_{j \in [n] \setminus i^*} g_{j,\mathsf{x}_j}}$$

and define k accordingly. Define

$$\mathsf{ct} := \begin{pmatrix} c_{1,1}, & c_{2,1}, & \dots, & c_{n,1} \\ c_{1,2}, & c_{2,2}, & \dots, & c_{n,2} \\ \dots, & \dots, & \dots, & \dots \\ c_{1,\sigma}, & c_{2,\sigma}, & \dots, & c_{n,\sigma} \end{pmatrix}$$

where for $i \in [n] \setminus i^*$ and all $f \in \Sigma$ we have $c_{i,f} := h_2^{r_{i,f}}$ and $c_{i^*,f^*} := h_2^{r_{i^*,f^*}}$, whereas the other terms are set to $\perp$. The adversary is given $(\mathsf{x}, \mathsf{k}, \mathsf{ct}, s)$ and the reduction returns whatever the adversary returns. Security follows from the fact that

$$\mathsf{Hash}(\mathsf{k}, \mathsf{x}) = \prod_{j \in [n]} g_{j,\mathsf{x}_j} = h_1.$$

□

4 Rate-1 CDH-Based Trapdoor Function

In this section we give a construction of rate-1 TDFs based on CDH, satisfying deterministic-encryption security. The result of [GGH19] gives CDH-based TDF constructions with rates $1/c$ for a constant c. A rough estimate of the constant c is at least 20. The main reason behind the large constant is that [GGH19] first builds an intermediate *local TDF* which (1) outputs two bits for every bit of the input (i.e., a rate less than 1/2) and (2) the TDF has a local property in that for each bit of the input, the inversion algorithm either recovers the bit or gives up for that particular bit, each happening with probability 1/2. The construction of [GGH19] then performs error correction over bitstrings to boost correctness. This results in another constant blowup.

At a high level our approach for achieving rate 1 proceeds as follows. We encode the input to the TDF block-by-block, instead of bit-by-bit. Each block is a symbol of an alphabet over a field for which erasure correction with better rates can be done. We then show how to provide an almost equally-sized *hint* for every block of the input, achieving a rate 1 at the end. The main technical novelty of our work relies on how to form the hint in a succinct way.

Construction 4.1 (Rate-1 TDF from CDH)**.** *Let* $\Sigma = \{1, \dots, \sigma\}$ *be a finite alphabet, let* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{Dec})$ *be a w-bit OWFE, and let* $(\mathsf{Encode}, \mathsf{Decode})$ *be an error-correcting code, where* $\mathsf{Encode} : \{0,1\}^n \to \Sigma^m$*. We define our TDF construction* $(\mathsf{TDF.KG}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ *as follows.*

- $\mathsf{TDF.KG}(1^\lambda)$*:*
    1. *Sample* $\mathsf{k} := \mathsf{Gen}(1^\lambda)$.
    2. *For all* $i \in [m]$ *and all* $f \in \Sigma$*:*
        (a) *Sample* $\rho_{i,f} \xleftarrow{\$} \{0,1\}^\lambda$ *and* $a_{i,f} \xleftarrow{\$} \{0,1\}^w$.

   (b) Compute $\mathsf{ct}_{i,f} := \mathsf{Enc}_1(\mathsf{k}, (i, f); \rho_{i,f})$.

3. *Set the trapdoor key as*

$$\mathsf{tk} := \mathsf{k}, \begin{pmatrix} \rho_{1,1}, & \rho_{2,1}, & \dots, & \rho_{m,1} \\ \rho_{1,2}, & \rho_{2,2}, & \dots, & \rho_{m,2} \\ \dots, & \dots, & \dots, & \dots \\ \rho_{1,\sigma}, & \rho_{2,\sigma}, & \dots, & \rho_{m,\sigma} \end{pmatrix} \tag{4}$$

and the index key as

$$\mathsf{ik} := \mathsf{k}, \begin{pmatrix} \mathsf{ct}_{1,1}, & \mathsf{ct}_{2,1}, & \dots, & \mathsf{ct}_{m,1} \\ \mathsf{ct}_{1,2}, & \mathsf{ct}_{2,2}, & \dots, & \mathsf{ct}_{m,2} \\ \dots, & \dots, & \dots, & \dots \\ \mathsf{ct}_{1,\sigma}, & \mathsf{ct}_{2,\sigma}, & \dots, & \mathsf{ct}_{m,\sigma} \end{pmatrix}, \begin{pmatrix} a_{1,1}, & a_{2,1}, & \dots, & a_{m,1} \\ a_{1,2}, & a_{2,2}, & \dots, & a_{m,2} \\ \dots, & \dots, & \dots, & \dots \\ a_{1,\sigma}, & a_{2,\sigma}, & \dots, & a_{m,\sigma} \end{pmatrix}. \tag{5}$$

- $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{x} \in \{0,1\}^n)$*:*

  1. *Parse* ik *as in Equation 5.*
  2. *Let* $\mathsf{z} := \mathsf{Encode}(\mathsf{x}) \in \Sigma^m$ *and* $\mathsf{y} := \mathsf{Hash}(\mathsf{k}, \mathsf{z})$.
  3. *For all* $i \in [m]$*:*
     (a) Let $h_i := \mathsf{Dec}(\mathsf{k}, \mathsf{ct}_{i,\mathsf{z}_i}, \mathsf{z})$.
     (b) Set $v_i := h_i \oplus a_{i,\mathsf{z}_i} \in \{0,1\}^w$.
  4. *Return* $\mathsf{u} := (\mathsf{y}, v_1, \dots, v_m)$.

- $\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{u})$*:*

  1. *Parse* tk *as in Equation 4 and* $\mathsf{u} := (\mathsf{y}, v_1, \dots, v_m)$.
  2. *Retrieve* z' *element-by-element as follows. For* $i \in [m]$*, to retrieve the* i*-th element:*
     (a) If there exists one and only one index $f \in \Sigma$ *such that*

$$\mathsf{Enc}_2(\mathsf{k}, \mathsf{y}, (i, f); \rho_{i,f}) = a_{i,f} \oplus v_i, \tag{6}$$

     then set $\mathsf{z}'_i = f$.
     (b) Otherwise, set $\mathsf{z}'_i = \perp$.
  3. *Return* $\mathsf{Decode}(\mathsf{z}')$.

4.1 Analysis

In the following we show that our construction is a correct and secure TDF.

Theorem 4.2 (Correctness)**.** *Let* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{Dec})$ *be a* w*-bit OWFE, where* $2^w \geq 2\sigma/\eta$*, for some* $\eta \in (0,1]$*. Let* $(\mathsf{Encode}, \mathsf{Decode})$ *be an error-correcting code resilient against a* η*-fraction of erasures, where* $\mathsf{Encode} : \{0,1\}^n \to \Sigma^m$ *and* $m \geq 6\lambda/\eta$*. Then Construction 4.1 is correct except with probability* $e^{-\lambda}$*:* $\Pr[\exists \mathsf{x} \in \{0,1\}^n : \mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})) \neq \mathsf{x}] \leq e^{-\lambda}$*, where* $(\mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$.

Proof. Let $(\mathsf{ik}, \mathsf{tk}) := \mathsf{TDF.KG}(1^\lambda)$, let $\mathsf{u} := \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})$ for a uniform $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$, and let $\mathsf{z} := \mathsf{Encode}(\mathsf{x})$. For all $i \in [m]$, by the correctness of the OWFE we have

$$v_i = h_i \oplus a_{i,\mathsf{z}_i} = \mathsf{Dec}(\mathsf{k}, \mathsf{ct}_{i,\mathsf{z}_i}, \mathsf{z}) \oplus a_{i,\mathsf{z}_i} = \mathsf{Enc}_2(\mathsf{k}, \mathsf{y}, (i, \mathsf{z}_i); \rho_{i,\mathsf{z}_i}) \oplus a_{i,\mathsf{z}_i}.$$

Thus, the index z_i satisfies Equation 6. Now we consider the probability that some $f \neq \mathsf{z}_i$ satisfies the same condition. Since $a_{i,f}$ is chosen uniformly and independently at random, the two values $\mathsf{Enc}_2(\mathsf{k}, (\mathsf{y}, i, f); \rho_{i,f}) \oplus v_i$ and $a_{i,f}$ are independent. Thus, the probability that the index $f \neq \mathsf{z}_i$ satisfies Equation 6 is

$$\Pr\left[\mathsf{Enc}_2(\mathsf{k}, \mathsf{y}, (i, f); \rho_{i,f}) \oplus v_i = a_{i,f}\right] \leq \frac{1}{2^w}.$$

By a union bound, the probability that such an $f \neq \mathsf{z}_i$ exists is at most

$$\Pr\left[\exists f \in \Sigma \setminus \{\mathsf{z}_i\} : \mathsf{Enc}_2(\mathsf{k}, \mathsf{y}, (i, f); \rho_{i,f}) \oplus v_i = a_{i,f}\right] \leq \frac{\sigma}{2^w} \leq \frac{\eta}{2}$$

as $2^w \geq 2\sigma/\eta$. Applying the Chernoff bound (Lemma 2.1) yields that at most $\eta \cdot m$ of the indices contain a non unique decoding and therefore z'_i is set to $\perp$, except with probability $e^{-\frac{\eta \cdot m}{6}} \leq e^{-\lambda}$. Let $S \subseteq [m]$ be the set of indices for which there exists a $\mathsf{z}'_i = \perp$. By the above, $|S| \leq \eta \cdot m$, except with probability $e^{-\lambda}$. As we assume the code $(\mathsf{Encode}, \mathsf{Decode})$ is capable of handling a η-fraction of erasures, $\mathsf{Decode}(\mathsf{z}')$ will output x with overwhelming probability. □

We show that our TDF is one-way.

Theorem 4.3 (One-Wayness and DE security)**.** *Assuming* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{Dec})$ *is an* (n,m)*-OWFE scheme, then Construction 4.1 is one-way. Moreover, if* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{Dec})$ *is* (k,m)*-smooth, the resulting TDF is* (k,m)*-CPA indistinguishable.*

Proof. Let $\mathsf{x} \in \{0,1\}^n$ be the random input to the TDF, and let $\mathsf{z} := \mathsf{Encode}(\mathsf{x})$. Also, let $\mathsf{y} := \mathsf{Hash}(\mathsf{k}, \mathsf{z})$. We first construct a simulator $\mathsf{Sim}(\mathsf{k}, \mathsf{y})$, which — without knowledge of x — samples a simulated index key $\mathsf{ik}_{\mathrm{sim}}$ together with a corresponding $\mathsf{u}_{\mathrm{sim}}$ as follows.

- $\mathsf{Sim}(\mathsf{k}, \mathsf{y})$:
  1. For all $i \in [m]$: Sample $a_i \xleftarrow{\$} \{0,1\}^w$.
  2. For all $i \in [m]$ and $f \in \Sigma$:
     (a) Sample $\rho_{i,f} \xleftarrow{\$} \{0,1\}^*$.
     (b) Compute $\mathsf{ct}_{i,f} := \mathsf{Enc}_1(\mathsf{k}, (i, f); \rho_{i,f})$.
     (c) Compute $a_{i,f} := a_i \oplus \mathsf{Enc}_2(\mathsf{k}, \mathsf{y}, (i, f); \rho_{i,f})$.
  3. Set the index key as

$$\mathsf{ik}_{\mathrm{sim}} := \mathsf{k}, \begin{pmatrix} \mathsf{ct}_{1,1}, & \mathsf{ct}_{2,1}, & \ldots, & \mathsf{ct}_{m,1} \\ \mathsf{ct}_{1,2}, & \mathsf{ct}_{2,2}, & \ldots, & \mathsf{ct}_{m,2} \\ \ldots, & \ldots, & \ldots, & \ldots \\ \mathsf{ct}_{1,\sigma}, & \mathsf{ct}_{2,\sigma}, & \ldots, & \mathsf{ct}_{m,\sigma} \end{pmatrix}, \begin{pmatrix} a_{1,1}, & a_{2,1}, & \ldots, & a_{m,1} \\ a_{1,2}, & a_{2,2}, & \ldots, & a_{m,2} \\ \ldots, & \ldots, & \ldots, & \ldots \\ a_{1,\sigma}, & a_{2,\sigma}, & \ldots, & a_{m,\sigma} \end{pmatrix}$$

  and the image as $\mathsf{u} := (\mathsf{y}, a_1, \ldots, a_m)$.

We now show that for any distribution $\mathcal{S}$ over $\{0,1\}^n$

$$(\mathsf{x}, \mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})) \overset{c}{\equiv} (\mathsf{x}, \mathsf{Sim}(\mathsf{k}, \mathsf{y})) \tag{7}$$

where $\mathsf{x} \overset{\$}{\leftarrow} \mathcal{S}$, $(\mathsf{ik}, *) \overset{\$}{\leftarrow} \mathsf{TDF.KG}(1^\lambda)$, $\mathsf{k} \overset{\$}{\leftarrow} \mathsf{Gen}(1^\lambda)$, and $\mathsf{y} := \mathsf{Hash}(\mathsf{k}, \mathsf{Encode}(\mathsf{x}))$. This will yield both the one-wayness and deterministic-encryption security claims of the lemma.

We define $\mathsf{Sim}'(\mathsf{k}, \mathsf{x}, \mathsf{y})$ as follows. For all $i \in [m]$ and $f \in \Sigma$, sample $\mathsf{ct}_{i,f}$ exactly as in $\mathsf{Sim}(\mathsf{k}, \mathsf{y})$, and letting $\mathsf{z} := \mathsf{Encode}(\mathsf{x})$, sample $a_{i,j}$ as follows:

- If $f = \mathsf{z}_i$, then set $a_{i,f} := a_i \oplus \mathsf{Enc}_2(\mathsf{k}, \mathsf{y}, (i, f); \rho_{i,f})$, exactly as in $\mathsf{Sim}(\mathsf{k}, \mathsf{y})$.
- If $f \neq \mathsf{z}_i$, then sample $a_{i,f} \overset{\$}{\leftarrow} \{0,1\}^w$.

By the security-for-encryption requirement of the underlying OWFE

$$(\mathsf{x}, \mathsf{Sim}(\mathsf{k}, \mathsf{y})) \overset{c}{\equiv} (\mathsf{x}, \mathsf{Sim}'(\mathsf{k}, \mathsf{x}, \mathsf{y})).$$

By simple inspection we can see that the distribution $(\mathsf{x}, \mathsf{Sim}'(\mathsf{k}, \mathsf{x}, \mathsf{y}))$ is identically distributed to $(\mathsf{x}, \mathsf{ik}, \mathsf{TDF}(\mathsf{ik}, \mathsf{x}))$, where $(\mathsf{ik}, *) \overset{\$}{\leftarrow} \mathsf{TDF.KG}(1^\lambda)$. The proof is now complete. □

4.2 Parameters

We analyze the rate of our scheme and we discuss possible instantiations for the underlying building blocks.

Theorem 4.4 (Rate). *Let $\sigma \geq \lambda$ and let $(\mathsf{Encode}, \mathsf{Decode})$ be an error correcting code for an alphabet Σ of size σ that can correct a fraction of erasure $\eta = 1/\log(\lambda)$ and has rate $1 - 1/\log(\lambda)$. Let $2^w = 2 \cdot \sigma/\eta$ and $m \geq 6\lambda/\eta$. Then Construction 4.1 has rate 1.*

Proof. By definition we have that

$$2^w = 2 \cdot \sigma/\eta = 2 \cdot \sigma \cdot \log(\lambda).$$

Then the value $v_i \in \{0,1\}^w$ encodes the codeword symbol $\mathsf{z}_i \in \Sigma$ of the codeword x_i. Thus, each codeword symbol is encoded at rate

$$\frac{\log(\sigma)}{w} = \frac{\log(\sigma)}{\log(\sigma) + \log\log(\lambda) + 1} \geq 1 - 2\frac{\log\log(\lambda)}{\log(\sigma)} \geq 1 - 2\frac{\log\log(\lambda)}{\log(\lambda)}.$$

Recall that $(\mathsf{Encode}, \mathsf{Decode})$ has rate $1 - 1/\log(\lambda)$ and can efficiently decode from a $\eta = 1/\log(\lambda)$ fraction of errors. Taking into account that the sender message includes an additional group element $\mathsf{y} \in \mathbb{G}$ and assuming that $\log(|\mathbb{G}|) = \lambda$, this accounts for a decrease of the rate by a factor

$$\frac{m \cdot w}{m \cdot w + \log(|\mathbb{G}|)} = 1 - \frac{\log(|\mathbb{G}|)}{m \cdot w + \log(|\mathbb{G}|)} \geq 1 - \frac{1}{m \cdot \log(\lambda)}.$$

Consequently, the total rate of our scheme is lower bounded by

$$\left(1 - \frac{1}{m \cdot \log(\lambda)}\right)\left(1 - \frac{1}{\log(\lambda)}\right)\left(1 - 2\frac{\log\log(\lambda)}{\log(\lambda)}\right) \geq 1 - 3\frac{\log\log(\lambda)}{\log(\lambda)},$$

which approaches 1. □

Note that the constraints $2^w = 2\sigma \cdot \log(\lambda)$ and $\sigma \geq \lambda$ require us to instantiate a hardcore function H that extracts $O(\log(\lambda))$ random bits from a CDH instance. This is well in reach of the function given in Lemma 2.3. What is left to be shown is a code that handles a $\eta = 1/\log(\lambda)$ fraction of errors. A natural choice is a Reed Solomon code over the alphabet Σ^2, specifically a $[m, m - m/\log(\lambda), m/\log(\lambda) + 1]$ Reed Solomon code. For this code, we can efficiently decode $m/\log(\lambda) = \eta m$ erasures, ensuring correctness of our scheme. This code has rate $1 - 1/\log(\lambda)$.

5 Rate-1 DDH-based Deterministic Encryption

In this section we show how to build TDFs satisfying DE security with the following two properties: (a) the index key contains $(n^2 + 1)$ group elements and (b) the image contains one group element plus exactly n bits. We mention that a recent result of Döttling et al. [DGI+19] achieve the same image size, but at the cost of bigger index keys, containing at least $4n^2$ group elements. Moreover, the construction of [DGI+19] is highly non-trivial, using techniques from [BGI16] as well as error-correcting codes. In contrast, our construction is fairly elementary and does not need ECC-based techniques.

We will make use of a balanced predicate during our construction, defined as follows.

Definition 5.1 (Balanced predicates)**.** *We say a predicate $\mathsf{P} : \mathsf{S} \times \{0,1\}^* \to \{0,1\}$ is balanced over a set S if for all $b_1, b_2 \in \{0,1\}$: $\Pr[\mathsf{P}(x_1; r) = b_1 \wedge \mathsf{P}(x_2; r) = b_2] = 1/4$, where $x_1, x_2 \xleftarrow{\$} \mathsf{S}$ and $r \xleftarrow{\$} \{0,1\}^*$.*

An obvious example of a balanced predicate is the inner-product function mod 2. However, in some situations one may be able to give a more direct (and sometimes a deterministic) construction. For example, if the underlying set S is $\{0,1\}^n$, then we may simply define $\mathsf{P}(x) = x_1$.

Notation. For $\mathsf{x} \in \{0,1\}^n$ and a vector $\mathbf{g} := (g_1, \ldots, g_n)$ we define $\mathsf{x} \cdot \mathbf{g} = \Pi_{i \in [n]} g_i^{\mathsf{x}_i}$.

Construction 5.2 (Linear-image TDF)**.** *Let G be a group scheme and let BL be a balanced predicate for the underlying group (Definition 5.1).*

We define our TDF construction $(\mathsf{TDF.KG}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ as follows.

- $\mathsf{TDF.KG}(1^\lambda)$*:*
  1. *Sample $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{G}(1^\lambda)$ and $\mathbf{g} := (g_1, \ldots, g_n) \xleftarrow{\$} \mathbb{G}^n$.*
  2. *For all $i \in [n]$, sample $\rho_i \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathbf{g}_i := \mathbf{g}^{\rho_i}$, where $\mathbf{g}^{\rho_i}$ denotes element-wise exponentiation to the power of ρ_i.*
  3. *For each $i \in [n]$ sample random coins $r_i \xleftarrow{\$} \{0,1\}^*$ for BL.*[3]
  4. *Set $\mathsf{tk} := (\rho_1, \ldots, \rho_n, \{r_i\})$ as the trapdoor key and $\mathsf{ik} := (\mathbf{g}, \mathbf{g}_1, \ldots, \mathbf{g}_n, (r_i)_{i \in [n]})$ as the index key.*

[2]By increasing the the size of Σ to e.g. the next power of 2, the bit representation of each symbol in Σ grows by at most one bit, i.e., the rate of such an encoding is $1 - 1/\lambda$

[3]We can also prove security just by sampling a single r, but the proof will be more complicated.

- $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{x} \in \{0,1\}^n)$*: Parse* $\mathsf{ik} := (\mathbf{g}, \mathbf{g}_1, \ldots, \mathbf{g}_n, (r_i)_{i\in[n]})$*. Return*

$$\mathsf{u} := (\mathsf{x}\cdot\mathbf{g}, \mathsf{BL}(\mathsf{x}\cdot\mathbf{g}_1; r_1) \oplus \mathsf{x}_1, \ldots, \mathsf{BL}(\mathsf{x}\cdot\mathbf{g}_n; r_n) \oplus \mathsf{x}_n) \in \mathbb{G} \times \{0,1\}^n. \tag{8}$$

- $\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{u})$*:*
  1. *Parse* $\mathsf{tk} := (\rho_1, \ldots, \rho_n, (r_i)_{i\in[n]})$ *and* $\mathsf{u} := (g_c, b'_1, \ldots, b'_n)$*.*
  2. *Return* $(\mathsf{BL}(g_c^{\rho_1}; r_1) \oplus b'_1, \ldots, \mathsf{BL}(g_c^{\rho_n}; r_n) \oplus b'_n)$*.*

5.1 Analysis

The correctness of the scheme is immediate.

Lemma 5.3 (Deterministic-encryption security)**.** *Assuming the underlying group is DDH-hard, then for any $k \le n$ such that $k \ge \log p + \omega(\log\lambda)$, the TDF given in construction 5.2 provides (k,n)-CPA security.*

Proof. For any two (k,n)-sources $\mathcal{D}_0$ and $\mathcal{D}_1$ we need to show $(\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathcal{D}_0)) \overset{c}{\equiv} (\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathcal{D}_1))$, where $(\mathsf{ik}, *) \xleftarrow{\$} \mathsf{TDF.KG}(1^\lambda)$. We do this via a series of hybrids, where in each hybrid we sample $(\mathsf{ik}, \mathsf{u})$ as follows.

- Hyb_b [Real game for $\mathcal{D}_b$]: sample $\mathbf{g} \xleftarrow{\$} \mathbb{G}^n$ and set $\mathbf{g}_i := \mathbf{g}^{\rho_i}$, for $\rho_i \xleftarrow{\$} \mathbb{Z}_p$. Set $\mathsf{ik} := (\mathbf{g}, \mathbf{g}_1, \ldots, \mathbf{g}_n)$. Sample $\mathsf{x} \xleftarrow{\$} \mathcal{D}_b$ and return $(\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x}))$.
- Hyb'_b:
  1. Sample $\mathbf{g} \xleftarrow{\$} \mathbb{G}^n$, $\mathsf{x} \xleftarrow{\$} \mathcal{D}_b$, and set $g_c := \mathsf{x}\cdot\mathbf{g}$.
  2. For $i \in [n]$ sample $\rho_i \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathbf{g}'_i := \mathbf{g}^{\rho_i}$.
  3. Set $\mathbf{g}_i := \mathbf{g}'_i \cdot \mathbf{v}_i$, where
  $$\mathbf{v}_i := (1, \ldots, 1, \underbrace{g'_i}_{i\text{th position}}, 1, \ldots, 1) \tag{9}$$
  and
     (a) sample $g'_i \xleftarrow{\$} \mathbb{G}$ in such a way that $1 \oplus \mathsf{BL}(g''_i \cdot g'_i; r_i) = \mathsf{BL}(g''_i; r_i)$, where $g''_i := \mathsf{x}\cdot\mathbf{g}'_i$.
  4. Set $\mathsf{ik} := (\mathbf{g}, \mathbf{g}_1, \ldots, \mathbf{g}_n)$ and $\mathsf{u} := \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})$.

Note 1 about Hyb'_b. For x and ik sampled as in Hyb'_b, we have the following relation: $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{x}) = (g_c, \mathsf{BL}(g_c^{\rho_1}; r_1), \ldots, \mathsf{BL}(g_c^{\rho_n}; r_n))$. In particular, the output of $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{x})$ can be sampled without knowing x, and just by knowing g_c and ρ_i's. Notice that the value of $g''_i := \mathsf{x}\cdot\mathbf{g}'_i$ (Item 3a above) can alternatively be computed as $g''_i = g_c^{\rho_i}$, without knowing x. We will make use of this fact in our proofs below.

Indistinguishability in Hyb': We have $\mathsf{Hyb}'_0 \overset{c}{\equiv} \mathsf{Hyb}'_1$. This follows from two facts. First, by Note 1 above, Hyb'_b can be sampled by just knowing g_c and ρ_i's and especially without knowing x. The second fact is that the distributions of g_c in Hyb'_0 and Hyb'_1 are statistically indistinguishable, by the leftover hash lemma and the fact that $\mathsf{H}_\infty(\mathcal{D}_b) \ge k$. These two facts together imply $\mathsf{Hyb}'_0 \overset{c}{\equiv} \mathsf{Hyb}'_1$.

Proof of $\mathsf{Hyb}_b \overset{c}{\equiv} \mathsf{Hyb}'_b$ for both $b \in \{0,1\}$: We prove this for $b = 0$, and the proof for the other case is the same. To prove $\mathsf{Hyb}_0 \overset{c}{\equiv} \mathsf{Hyb}''_0$, define the following two hybrids:

- HybRnd_0: same as Hyb_0 except for every i we replace $\mathbf{g}_i$ with a random vector chosen uniformly from $\mathbb{G}^n$.
- HybRnd'_0: same as Hyb'_0 except for every i we replace $\mathbf{g}'_i$ with a random vector chosen uniformly from $\mathbb{G}^n$.

We will now show $\mathsf{Hyb}_0 \overset{c}{\equiv} \mathsf{HybRnd}_0 \overset{c}{\equiv} \mathsf{HybRnd}'_0 \overset{c}{\equiv} \mathsf{Hyb}'_0$, and this will complete the proof.

Proof for $\mathsf{Hyb}_0 \overset{c}{\equiv} \mathsf{HybRnd}_0$. The proof follows from DDH, by considering the fact that either hybrid can be simulated just by knowing $\mathbf{g}_i$ and that in one hybrid we have $\mathbf{g}_i := \mathbf{g}^{\rho_i}$ for a random exponent ρ_i, and in the other exponent $\mathbf{g}_i \xleftarrow{\$} \mathbb{G}^n$.

Proof for $\mathsf{HybRnd}_0 \overset{c}{\equiv} \mathsf{HybRnd}'_0$. These two distributions are identical, because in either distribution $\mathbf{g}_i$ is uniformly random.

Proof for $\mathsf{HybRnd}'_0 \overset{c}{\equiv} \mathsf{Hyb}'_0$. The proof follows from DDH, by considering the fact that either hybrid can be simulated just by knowing x and $\mathbf{g}'_i$'s, and that in one hybrid we have $\mathbf{g}'_i \xleftarrow{\$} \mathbb{G}^n$, and in the other hybrid $\mathbf{g}'_i := \mathbf{g}^{\rho_i}$ for a random exponent ρ_i. □

6 Experimental Results

In this section we report proof-of-concept implementations of our DDH-based TDF construction (Construction 5.2) using Python. We report the resulting parameters of the scheme in Table 1.

Our group is an elliptic curve group on Ed25519 and the size of a group element in our implementation is 32 Bytes (B) = 256 bits. The encryption and decryption algorithms take less than a second. The table shows the growth of ciphertext size based on input size. We have not optimized our code for achieving more compact ciphertexts. Essentially, we used a serialization package (Pickle) which resulted in extra overhead in ciphertext size. As expected, the key-generation algorithm is main bottleneck in our implementation, together with the resulting index/trapdoor keys.

The machine specifications are as follows.

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             4
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
```

msg	ct	tk	ik	KG time	Enc time	Dec time
64 B	274 B	18 KB	19 MB	15.6 s	0.11 s	0.04 s
128 B	338 B	35 KB	75 MB	62.3 s	0.42 s	0.07 s

Table 1: Experimental results of our TDF construction. Here B denotes bytes (8 bits). The size of the group element is 32 B.

```
Model:                 6
Model name:            QEMU Virtual CPU version 2.5+
Stepping:              3
CPU MHz:               2599.998
BogoMIPS:              5199.99
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
L3 cache:              16384K
NUMA node0 CPU(s):     0-3
```

References

[BBN$^+$09] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT 2009*, *LNCS* 5912, pages 232–249, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany. 1

[BBO07] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO 2007*, *LNCS* 4622, pages 535–552, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany. 2

[BCPT13] E. Birrell, K.-M. Chung, R. Pass, and S. Telang. Randomness-dependent message security. In *TCC 2013*, *LNCS* 7785, pages 700–720, Tokyo, Japan, March 3–6, 2013. Springer, Heidelberg, Germany. 1

[BFO08] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *CRYPTO 2008*, *LNCS* 5157, pages 335–359, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 2

[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. 1, 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. 2, 3, 16

[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. 1

[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. 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. 9

[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. 4, 9

[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. 9

[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, 16

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 9

[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. 2

[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, 3, 4, 5, 9, 12

[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, 4, 9, 10

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32, Seattle, WA, USA, May 15–17, 1989. ACM Press. 5, 6, 7

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24, Seattle, WA, USA, May 15–17, 1989. ACM Press. 7

[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. 1

[MY10] P. Mol and S. Yilek. Chosen-ciphertext security from slightly lossy trapdoor functions. In *PKC 2010*, *LNCS* 6056, pages 296–311, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany. 1

[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

[PW11] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6):1803–1844, 2011. 2

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960. 8

[WB86] L. R. Welch and E. R. Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470. 8

[Wee12] H. Wee. Dual projective hashing and its applications - lossy trapdoor functions and more. In *EUROCRYPT 2012*, *LNCS* 7237, pages 246–262, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 2