Shrinking the Keys of Discrete-Log-Type Lossy Trapdoor Functions

Xavier Boyen and Brent Waters *

[1] Institut Montefiore, Université de Liège, Belgium
[2] University of Texas at Austin, USA

Abstract. To this day, realizations in the standard-model of (lossy) trapdoor functions from discrete-log-type assumptions require large public key sizes, e.g., about $\Theta(\lambda^2)$ group elements for a reduction from the decisional Diffie-Hellman assumption (where λ is a security parameter). We propose two realizations of lossy trapdoor functions that achieve public key size of only $\Theta(\lambda)$ group elements in bilinear groups, with a reduction from the decisional Bilinear Diffie-Hellman assumption.
Our first construction achieves this result at the expense of a long common reference string of $\Theta(\lambda^2)$ elements, albeit reusable in multiple LTDF instantiations. Our second scheme also achieves public keys of size $\Theta(\lambda)$, entirely in the standard model and in particular without any reference string, at the cost of a slightly more involved construction.
The main technical novelty, developed for the second scheme, is a compact encoding technique for generating compressed representations of certain sequences of group elements for the public parameters.

1 Introduction

The notion of Lossy Trapdoor Function (LTDF) is a new fundamental public-key primitive that was recently introduced by Peikert and Waters [20], henceforth PW. This notion has generated interest for two main reasons:

1. LTDFs can be constructed from widely differing hardness assumptions. These include the two recent constructions of PW from the Discrete-Log-based DDH and a worst-case Lattice assumption [20]. In addition, the Damgård-Jurik [9] variant of the decade-old Paillier cryptosystem [19], which, as was independently pointed out in [23, 5], happens to immediately give an LTDF from the Factoring-based assumption of composite residuosity.
2. LTDFs can in turn serve as black-box building blocks within more complex primitives, such as regular injective trapdoor functions, provably collision-resistant hashing, and public-key encryption with chosen-ciphertext security [20], as well as some new strong notions of security for deterministic public-key encryption [5]. We may also expect to see more such applications in the future.

* Supported by NSF CNS-0716199, Air Force Office of Scientific Research (AFOSR) under the MURI award for "Collaborative policies and assured information sharing" (Project PRESIDIO) and the U.S. DHS under Grant Number 2006-CS-001-000001.

Since LTDFs are general and admit several applications, an interesting pursuit is to seek how to realize them in the most efficient manner, from robust and hardness assumptions. Peikert and Waters (PW), who first proposed the notion, provide two LTDF constructions from appealing hardness assumptions [20]. All of their constructions provided built upon a public key that reflected a "matrix structure". One drawback of this approach is that it results in a rather large public key of $\Theta(\lambda^2)$ group elements and thus (for the DDH-based construction) grows cubically with the security parameter λ. As a concrete example, at the $\lambda = 128$ bit security level, if we assume an optimal elliptic-curve implementation with the smallest possible 256-bit element representation, this implies a public-key storage and transmission requirement in excess of $(3 \times 128)^2 \times 256$ bits, or about 36 Mb, for every single instance of the LTDF.

In subsequent work Boldyreva, Fehr, and O'Neill [5] and Rosen and Segev [23] provided constructions of Lossy Trapdoors from the composite residuosity assumption. These constructions provided greater efficiency. Their security depends upon the composite residuosity assumption, while our goal is to look for new and efficient lossy trapdoors that do not depend upon the difficulty of factoring.

Our goal is to work toward achieving efficient trapdoor functions based on discrete log assumptions. One compelling reason to find solutions in the discrete log setting is that there are potentially many concrete instances for any one constructions. For example, there are several different realizations of bilinear groups [13]. If one group them turned out to be insecure, a construction might still be viable under a different group. In contrast, the assumption of factoring is absolute.

Our Approach In this work, we aim toward making LTDFs from discrete log assumptions realizable in practice. Our guiding principle is to try to *compress* the DDH-based LTDF construction from [20] in order to shrink the public key size from $\Theta(\lambda^2)$ to $\Theta(\lambda)$ group elements.

More precisely, we propose two new LTDF constructions that extend the PW DDH-based LTDF onto pairing-friendly curves. Since with a (symmetric) pairing the DDH assumption is generally false, we use the related decisional Bilinear Diffie-Hellman (DBDH) assumption. DBDH is weaker than DDH, which is why it may hold even in bilinear groups where DDH does not, but in counterpart our constructions do require a pairing.

On pairing-friendly curves, we will show how to exploit the bilinear map in various ways in order to remove much of the redundancy from the LTDF public key, and instead allow the user to reconstruct it efficiently as needed. Not only does this save a factor of λ in the public-key size, it also makes the LTDF computation more space- and time-efficient.

We begin by observing that in the original construction there is informational theoretically much redundancy in their public key. When constructing an injective key their setup algorithm chooses $r_1, \ldots, r_n, a_1, \ldots, a_n \in \mathbb{Z}_p$ for a trapdoor function of input length n. The (i, j)-th entry for $i \neq j$ is the element $g^{r_i \cdot a_j}$ and along the diagonal the key consists of $g^{r_i \cdot a_i} \cdot g$.

In order to take advantage of this redundancy our first idea is to use a bilinear group $\mathbb{G}$ with generator g. A natural approach is to consider publishing as part of the public key $g^{a_1}, \ldots, g^{a_n}, g^{r_1}, \ldots, g^{r_n}$ consisting of only $O(n)$ group elements. Then when evaluating the trapdoor, one can dynamically generate the PW matrix by taking the (i, j)-th element as $e(g^{r_i}, g^{a_j})$. While this first technique does indeed compress the matrix, it actually gives the attacker too much information. In particular, it gives the attacker the diagonal of a *lossy* key. This allows the attacker to trivially distinguish between a lossy and an injective key; undercutting the main security guarantee. Our challenge is create a public key that allows a user to generate all the matrix elements *except* along the diagonal, but while simultaneously keeping the key short ($\Theta(n)$ group elements).

In our first construction we amortize the cost of a creating the matrix by leveraging a (fixed, reusable) common reference string (CRS) to achieve the compression. The reference string consists of $\Theta(n^2)$ group elements, and is therefore as long as one public key in the PW scheme; however, each additional public key consists of just $O(n)$ group elements. This system works since the CRS provides information when taken along with the public key to generate everything except along the diagonal. The primary downsides to this approach are the reliance on a trusted third party to generate the reference string and the large size of the reference string itself.

In our second construction we develop a new method in order to achieve true key compression without CRS. To do this we find a somewhat surprising connection to the identity-based revocation scheme of Lewko, Sahai, and Waters [17]. In their work, they describe a "two equation" technique for identity-based revocation. They show how to encrypt to an identity ID^* such that a private key for any identity $\mathrm{ID} \neq \mathrm{ID}^*$ is able to decrypt.[3] Intuitively, we will apply a (high-level) version of this idea, not for revocating IBE users but for compressing an LTDF public key.

Conceptually we will map each row i of a ciphertext to a ciphertext that is associated with a "virtual identity" i and each column j with a "virtual key" for identity j. Using this approach one is able to decompress the encoding and obtain an LTDF public key matrix for every element (i, j), except on the diagonal where $i = j$. Along the diagonal, the system generates two dependent equations and the public key will provide these components separately. Taken all together, this gives us a Lossy TDF system with $\Theta(n)$ size public keys in the standard model.

The security of both schemes follows from the DBDH assumption, respectively in the common reference string model in the first case and in the standard model in the second case.

We note that, while our keys and ciphertext only require $\Theta(\lambda)$ group elements instead of $\Theta(\lambda^2)$, half of the new elements will live in the bilinear "target" group $\mathbb{G}_t$ instead of entirely in the "source" group $\mathbb{G}$. For known pairing and known attacks, the optimal choices of $\mathbb{G}$ and $\mathbb{G}_t$ are such that the representation size of

[3] The work of Lewko et al. [17] actually shows how to efficiently revoke several users at once, but for our purposes discussing a single revocation is sufficient.

$\mathbb{G}_t$-elements grows faster than that of $\mathbb{G}$-elements. This makes our technique less useful in the asymptote that one could have expected (based on current knowledge). Nevertheless, our framework is quite advantageous for practical values of the security parameter.

1.1 Related Work

The concept of trapdoor functions was first proposed by Diffie and Hellman [10]. In an (injective) trapdoor function $f()$ a party with a trapdoor can invert the function; however, inversion should be hard for any attacker. Trapdoor functions have several interesting applications in cryptography ranging from two party computation [26], Non-Interactive Zero Knowledge Proofs [4], and Chosen-Ciphertext Secure Encryption [18, 11] among others.

The first trapdoor realization was given by Rivest, Shamir, and Adleman [22] with security based on what has become known as the RSA assumption. Other standard model construction include the factoring based one of Rabin [21] and that of Paillier [19]. We note that all these standard model constructions rely on the difficulty of factoring.

Using the random oracle heuristic it is possible to transform any secure public key encryption system into an injective trapdoor [3, 2]. In addition, Bellare, Halevi, Sahai, and Vadhan gave generic standard model transformations from one way functions to highly *non-injective* trapdoor functions; however, the applications of these forms of trapdoors is rather limited. For example, they cannot be used to realize public key encryption. For this reason, we will often implicitly assume injectiveness when discussing useful trapdoor functions.

Until recently, the only known standard model realizations of trapdoor functions (LTDFs) relied on the difficulty of factoring. Recently, Peikert and Waters [20] introduced the concept of *Lossy Trapdoor Functions.* A Lossy Trapdoor system has the property that a publicly evaluable function f can be created to either be an injective function or highly non-injective; moreover, an adversary should not be able to distinguish what type of function f is given its description. They showed that Lossy TDFs implied standard trapdoor functions and gave several other applications of LTDFs including chosen-ciphertext secure encryption. In addition, they showed different realizations from both the hardness of the decisional Diffie-Hellman problem and certain lattice-based problems. One drawback of their construction is that uses a matrix of public key components that results in large public keys of $O(\lambda^2)$ group elements.

The PW lattice-based LTDF construction fares better asymptotically than the discrete-log one, since it requires $O(\lambda^2)$ elements of Z_q where q is relatively small [4]. However, the constants associated with the lattice-based construction might make it less efficient for security parameters in the foreseeable future, and for this paper we focus on the discrete-log setting (specifically, in a pairing-enabled context).

[4] Typically, q is set to be much less than 2^λ. See Gama and Nguyen [14] for a discussion of current lattice parameters.

One potential benefit of our work is that our public key compression techniques, developed here in a pairing context, might have future applications to the lattice setting. We note that lattice-based crypto analogues of pairing-based techniques have been used to construct Identity-Based Encryption in a lattice setting: e.g., Gentry et al. [15] recently gave a lattice-based trapdoor function system with interesting applications such as hash-and-sign signatures and identity-based encryption: at first in the random-oracle model (see [15]), and more recently in the standard model (see [1, 7, 8]).

More recently, Boldyreva, Fehr, and O'Neill [5] and Rosen and Segev [23] independently showed that the Damgård Jurik [9] extension of the Paillier [19] trapdoor was actually an efficient lossy trapdoor function and therefore inherits its applications. Freeman et al. [12] gave a number of LTDF constructions based on the quadratic and composite residuosity assumptions, or on d-Linear assumptions (see [6, 24]). Hemenway and Ostrovsky [16] showed that LTDFs can also be constructed from smooth homomorphic hash proof systems.

2 Preliminaries

Before describing our constructions, it is useful to review and understand the original scheme, and also to see where it can be improved. (The paper [20] has two constructions, one from the DDH problem, the other from worst-case Lattice problems. We focus on the first. We refer to the paper for the full details of their construction.)

2.1 Simplified Definition of Lossy TDFs

First, we give a somewhat simplified definition of Lossy TDF, based on [20]. Let λ be a security prameter and $n(\lambda)$ be the length of the input on which the function is evaluated.

Definition 1. *A lossy trapdoor function (LTDF) is a collection of polynomial-time algorithms* $(S_{inj}, S_{loss}, F_{ltdf}, F_{ltdf}^{-1})$ *such that*

- S_{inj} *randomly generates an injective function along with an associated trapdoor,*
- S_{loss} *generates a lossy function (and no trapdoor),*
- F_{ltdf} *evaluates any function generated by either* S_{inj} *or* S_{loss} *on any input vector* $\mathbf{x} \in \{0,1\}^n$*,*
- F_{ltdf}^{-1} *recovers the original input* $\mathbf{x}$ *from the output* $\mathbf{y}$ *of an injective function using its trapdoor.*

Additionally, there is a security requirement that the injective and lossy functions respectively generated by S_{inj} *and* S_{loss} *be computationally indistinguishable.*

This simplified definition will cover all the cases of interest in this paper. We refer to [20] for a more precise and more complex definition. A few related notions (such as the amount of "lossiness" induced by the lossy mode of an LTDF) will be recalled as we need them.

3 Compact LTDFs from DBDH in the CRS Model

Our first LTDF construction with a compressed output is set in the common reference string (CRS) model, and uses bilinear maps.

The main interest of this construction is that it is very simple, yet produces a Lossy TDF with shorter public keys than any previous ones not based on factoring. It also features a security reduction from the decisional bilinear Diffie-Hellman assumption, which is one of the most-studied and weakest among all the very many bilinear-group assumptions that have been made to date.

Although the reference string in this scheme is about as long as the public key in the PW DDH scheme of the previous section, we stress that the CRS is reusable across users whereas public keys are clearly not.

3.1 Intuition

At a high level, this LTDF is a bit similar to the PW DDH-based PW. A generalized ElGamal scheme is used to encrypt a matrix message $\mathbf{M}$ that is either the identity matrix $\mathbf{I}$ or the zero matrix $\mathbf{0}$, with the same consequences on injectivity versus lossiness as before. The novelty lies in the distribution of the public key.

Recall that in the PW DDH-based LTDF, the public key comprises the actual ciphertext $\mathbf{C}$ of $\mathbf{M}$, and whose representation takes up to $n \times (n + 1)$ group elements. Of course, there is a lot of redundancy in $\mathbf{C}$, due to the way it is constructed, but all of this redundancy had to remain computationally hidden in order for the injective and lossy modes to remain indistinguishable.

Our general goal here will be carefully to reveal a portion of this redundancy, so that it can be reconstructed at the time of use (explicitly or implicitly) without having to be transmitted, but without compromising the computational indistinguishability between an injective LTDF and a statistically lossy one.

Our approach in this first scheme starts with a decomposition of the ciphertext $\mathbf{C} = (\mathbf{C}_1 \| \mathbf{C}_2)$ linearly into two "additive" components: $\mathbf{C} = \mathbf{C}^{bulk} \oplus \mathbf{C}^{diag}$, where $\oplus$ is the element-wise matrix addition using $\mathbb{G}$'s own group operation (which we chose to write using a multiplicative notation):

1. The first component, $\mathbf{C}^{bulk}$, will correspond to $\mathbf{C}_1$ and all the elements of $\mathbf{C}_2$ off the diagonal: this is the "bulk" of the matrix $\mathbf{C}$, which stays the same in both injective and lossy modes.
2. The second component, $\mathbf{C}^{diag}$, will comprise just the elements on the diagonal of $\mathbf{C}_2$: there are only n of them, and the only ones whose construction changes between the two modes.

Since $\mathbf{C}^{diag}$ is already relatively compact, most of the gain will come from compressing $\mathbf{C}^{bulk}$. Observe in the PW scheme that the off-diagonal elements of $\mathbf{C}_2$ are of the form $g^{r_i z_j}$, where r_i is ephemeral and z_j belongs to the private key (if any); i.e., we have a product of two secret values in the exponent of a fixed group generator. Instead of publishing all the $g^{r_i z_j}$ elements *in extenso*, this *suggests* the publication of "precursor" elements g^{r_i} and g^{z_j} and the use of a pairing to reconstruct $e(g,g)^{r_i z_j} = e(g^{r_i}, g^{z_j})$ on the receiving end. From there, one could then implement the rest of the PW scheme in the group generated by $e(g,g)$ instead of $\mathbb{G}$.

Unfortunately, this idea does not quite work yet, because it also exposes the diagonal values $e(g^{r_i}, g^{z_j})$ for $i = j$, and from this information the injective and lossy modes become easy to distinguish. We need to forbid the reconstruction of $e(g^{r_i}, g^{z_j})$ when $i = j$, and allow it only when $i \neq j$. This is where the CRS comes into play: the CRS can be constructed in such a way that it contains values of the form $e(g^{r_i}, g^{z_j})$, or even $g^{r_i z_j}$, but only for $i \neq j$ and not for $i = j$.

Alas, there is still one problem: the private key elements z_j clearly cannot be part of a CRS that is meant to be reusable across multiple users. The solution is to have the CRS contain a reusable matrix of non-diagonal elements $g^{r_i a_j}$, and have each LTDF public key contain an independent vector of g^{b_j}. A bilinear pairing can be used to reconstruct $e(g^{r_i a_j}, g^{b_j}) = e(g,g)^{r_i a_j b_j}$ as needed, for $i \neq j$. The products $a_j b_j$ play the role of the z_j in the PW scheme (though only b_j is available as trapdoor information). The r_i are analogous to the secret ephemeral randomizers in the PW scheme, except that they have been immortalized in the CRS.

The main difficulties will be to show that inversion can still be performed only with the partial trapdoor b_j (in injective mode), and to prove that the r_i can be safely reused for all LTDF instances (even with different public keys) without putting their security in jeopardy.

3.2 Construction

Consider a pair of finite, abelian, bilinear groups $(\mathbb{G}, \times)$ and $(\hat{\mathbb{G}}, \times)$ of prime order $p = |\mathbb{G}| = |\hat{\mathbb{G}}|$, respectively generated $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$. Let $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_t$ be a non-degenerate, efficiently computable, bilinear pairing into a third abelian group $\mathbb{G}_t$ also of order $p = |\mathbb{G}_t|$ and thus generated by $e(g, \hat{g}) \in \mathbb{G}_t$. We use the multiplicative notation for the group operation in all three groups. There may or may not exist efficiently computable group homomorphisms between $\mathbb{G}$ and $\hat{\mathbb{G}}$; we make no requirement of this nature either way. We assume that the Decision Bilinear Diffie-Hellman (DBDH) problem is hard in $\mathbb{G} \times \hat{\mathbb{G}}$, meaning that no probabilistic polynomial-time algorithm $\mathbf{A}$ can distinguish $(g, g^a, g^b, \hat{g}, \hat{g}^a, \hat{g}^c, e(g,\hat{g})^{abc})$ from $(g, g^a, g^b, \hat{g}, \hat{g}^a, \hat{g}^c, e(g,\hat{g})^d)$ with probability non-negligibly greater than $\frac{1}{2}$, for randomly chosen $a, b, c, d \in \mathbb{Z}_p$.

As before, in all generality we must consider not a fixed pair of groups $\mathbb{G}$ and $\hat{\mathbb{G}}$, but an infinite family of such groups sampled by a PPT instance generator $\mathcal{G}$, where $\mathcal{G}$ on input 1^λ outputs a description of $(p, \mathbb{G}, g, \hat{\mathbb{G}}, \hat{g}, e)$ such that $\lfloor \log_2 p \rceil = \Theta(\lambda)$. The DBDH assumption, rigorously speaking, pertains to the family of groups induced by $\mathcal{G}$ for sufficiently large λ.

For simplicity of notation, in the description of the scheme and its proof below, we drop all "hats", thereby blurring the distinction between the two bilinear groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ (though $\mathbb{G}_t$ must remain distinct). For example, in this case the Decision-BDH problem becomes to distinguish $(g, g^a, g^b, g^c, e(g,g)^{abc})$ from $(g, g^a, g^b, g^c, e(g,g)^d)$.

Our compact LTDF from DBDH in the CRS model is constructed as follows:

CRS Setup: The universal setup algorithm, on input 1^λ, selects a bilinear group $(p, \mathbb{G}, g, e) \leftarrow \mathcal{G}(1^\lambda)$ and fixes $n > 3\log_2 p$. It then secretly chooses random $r_i \in \mathbb{Z}_p$ for $i \in [n]$ and random $a_j \in \mathbb{Z}_p$ for $j \in [n]$, and publishes a CRS that consists of the following:

- the description of $\mathbb{G} = \langle g \rangle$, and the pairing e;
- g^{r_i} for all $i \in [n]$;
- g^{a_j} for all $j \in [n]$;
- $g^{r_i a_j}$ for all $(i, j) \in [n]^2$ such that $i \neq j$;

Everyone can check that the CRS has been computed correctly (though of course not that the exponents r_i and a_i have been actually picked at random and forgotten).

Sampling algorithm, Injective mode: The injective function generator, denoted $S_{\text{inj}}(p, \mathbb{G}, g, e)$, randomly draws $b_j \in \mathbb{Z}_p$ for $j \in [n]$. It looks up the CRS to obtain g^{r_i} for $i \in [n]$ and g^{a_j} for $j \in [n]$.

- The LTDF function index, or "public key", is published as $2n$ elements of $\mathbb{G}$ and $\mathbb{G}_t$ in total, arranged in a single-row matrix $\mathbf{B}$ and a diagonal matrix $\mathbf{D}$ (where $1 = e(g, g)^0$),

$$\mathbf{B} = \begin{pmatrix} g^{b_1} & \cdots & g^{b_n} \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} e(g^{r_1}, g^{a_1})^{b_1} \cdot e(g, g) & 1 & \cdots & 1 \\ 1 & e(g^{r_2}, g^{a_2})^{b_2} \cdot e(g, g) & \cdots & 1 \\ \vdots & & \ddots & 1 \\ 1 & 1 & \cdots & e(g^{r_n}, g^{a_n})^{b_n} \cdot e(g, g) \end{pmatrix}$$

- The LTDF trapdoor, or "private key", consists of the n-vector $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{Z}_p^n$.

Sampling algorithm, Lossy mode: The lossy function generator, denoted $S_{\text{loss}}(p, \mathbb{G}, g, e)$, looks up the CRS and proceeds in the same fashion as S_{inj}, except that the public and private keys are created differently.

- The LTDF function index, or "public key", consists of two matrices $\mathbf{B}$ and $\mathbf{D}$, where,

$$\mathbf{B} = \begin{pmatrix} g^{b_1} & \cdots & g^{b_n} \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} e(g^{r_1}, g^{a_1})^{b_1} & 1 & \cdots & 1 \\ 1 & e(g^{r_2}, g^{a_2})^{b_2} & \cdots & 1 \\ \vdots & & \ddots & 1 \\ 1 & 1 & \cdots & e(g^{r_n}, g^{a_n})^{b_n} \end{pmatrix}$$

- There is no LTDF trapdoor, or "private key" (since in this case one is statistically unable to perform an inversion, whether one is given the b_j or not).

Evaluation algorithm: The evaluation algorithm, denoted F_{ltdf}, takes as input $((\mathbf{B},\mathbf{D}),\mathbf{x})$, where $(\mathbf{B},\mathbf{D})$ is a function index, and $\mathbf{x}\in\{0,1\}^n$ is an n-bit binary input string. The evaluation algorithm F_{ltdf} naturally also has access to the CRS. To facilitate the exposition, we present two different but functionally equivalent implementations of F_{ltdf}.

- *"Pedestrian" evaluation (requires n^2-n pairings):*
  To compute the desired output, the simplest way is first to reconstruct the analogous to the matrix $\mathbf{C}$ in the PW scheme, and then proceed with the function evaluation in the analogous way. One big difference with PW is that, here, the matrix $\mathbf{C}$ does not have all of its elements in the same group $\mathbb{G}$. Rather, the zero-th column, $\mathbf{C}_1$, has elements in $\mathbb{G}$, whereas the remaining n column, $\mathbf{C}_2$, have elements in $\mathbb{G}_t$.
  Let thus $\mathbf{C}=\mathbf{C}_1\|\mathbf{C}_2$ with $\mathbf{C}_1\in\mathbb{G}^{n\times 1}$ and $\mathbf{C}_2\in\mathbb{G}_t^{n\times n}$ computed from $\mathbf{B}$, $\mathbf{D}$, and the CRS,

$$\mathbf{C}_1=\begin{pmatrix} g^{r_1}\\ \vdots \\ g^{r_n}\end{pmatrix}\qquad \mathbf{C}_2=\begin{pmatrix} D_1 & e(g^{r_1a_2},B_2) & \cdots & e(g^{r_1a_n},B_n)\\ \vdots & & \ddots & \vdots\\ e(g^{r_2a_1},B_1) & D_2 & \cdots & e(g^{r_2a_n},B_n)\\ \vdots & & \ddots & \vdots\\ e(g^{r_na_1},B_1) & e(g^{r_na_2},B_2) & \cdots & D_n\end{pmatrix}$$

  The output is the row-vector $\mathbf{y}=\mathbf{x}\cdot\mathbf{C}$, with 1 element in $\mathbb{G}$ and n elements in $\mathbb{G}_t$, given by,

$$\begin{aligned}\mathbf{y}=\mathbf{x}\cdot\mathbf{C}=\mathbf{x}\cdot(\mathbf{C}_1\|\mathbf{C}_2) &= \begin{pmatrix}x_1 \dots x_n\end{pmatrix}\cdot\begin{pmatrix} c_{1,0} & c_{1,1} & \cdots & c_{1,n}\\ \vdots & & \ddots & \vdots\\ c_{n,0} & c_{n,1} & \cdots & c_{n,n}\end{pmatrix}\\ &= \left(\ \prod_{i=1}^n c_{i,0}^{x_i}\ ,\ \prod_{i=1}^n c_{i,1}^{x_i}\ ,\ \dots \prod_{i=1}^n c_{i,n}^{x_i}\ \right) \qquad \in \mathbb{G}\times\mathbb{G}_t^n\end{aligned}$$

- *"Shortcut" evaluation (requires n pairings):*
  A careful study of the above procedure suggests a faster but equivalent way to evaluate the function on the given inputs. Instead of expanding each element of $\mathbf{C}$ using a pairing, only to have them multiplied up together down the line, we keep the multiplication in mind from the start which allows us to do but a single pairing for each output element.
  The output must be a row-vector $\mathbf{y}$ of size $n+1$, where $y_0\in\mathbb{G}$ and each of the elements $y_j\in\mathbb{G}_t$ for $j\in[n]$ are directly computable from the respective expressions,

$$y_0=\prod_{i=1}^n (g^{r_i})^{x_i}\quad\in\mathbb{G}\qquad,\qquad y_j=e\left(\prod_{i\in[n]\setminus\{j\}}(g^{r_ia_j})^{x_i}\ ,\ B_j\right)\cdot D_j^{x_j}\quad\in\mathbb{G}_t$$

  It is easy to see that both methods give the same result. The pedestrian methods clearly shows the analogy to PW, while the shortcut method requires much less memory and is more efficient.

Inversion algorithm: The inversion algorithm F_{ltdf}^{-1} is input $(\mathbf{b}, \mathbf{y})$, where $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{Z}_p^n$ is the trapdoor from the injective sampling algorithm S_{inj}, and $\mathbf{y} = (y_0, y_1, \ldots, y_n) \in \mathbb{G}_t^{n+1}$ is the output of the evaluation algorithm F_{ltdf}. The algorithm F_{ltdf}^{-1} has access to the CRS.

The bits of the preimage $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ of $\mathbf{y}$ are output as follows,

$$x_j = \begin{cases} 0 & \text{if } y_j = e(y_0, g^{a_j})^{z_j} \cdot e(g,g)^0 \\ 1 & \text{if } y_j = e(y_0, g^{a_j})^{z_j} \cdot e(g,g)^1 \end{cases}$$

Naturally, $e(y_0, g^{a_j})^{z_j}$ is only computed once for each $j \in [n]$, so the entire inversion process requires n pairing evaluations — or $n+1$, strictly speaking, though $e(g,g)$ never changes.

3.3 Security

There are three properties that must be satisfied in order for our scheme to be an LTDF: (1) injectivity in injective mode; (2) enough lossiness in lossy mode; (3) computational indistinguishability of the two modes.

Theorem 1. *The preceding algorithms define a collection of* $(n, n - \log_2 p)$*-Lossy TDFs under the Decision Bilinear Diffie-Hellman assumption for* $\mathbb{G}$*, in the Common Reference String model.*

Proof. We need to prove each of the three listed properties: (1) injectivity in injective mode; (2) enough lossiness in lossy mode; (3) computational indistinguishability of the two modes.

Injectivity in injective mode follows from that, since the matrix $\mathbf{M}$ and thus $\mathbf{C}_2$ is invertible, the information from $\mathbf{x}$ will be preserved by the evaluation function, which by construction amounts to "multiplying" the input vector $\mathbf{x}$ by the matrix $\mathbf{C}_2$ (and also $\mathbf{C}_1$ which is important to compute the function inverse but not to show injectivity).

Lossiness in lossy mode follows from that, under the parameters generated by S_{loss}, the evaluation function $F_{\text{ltdf}}(\mathbf{C}, \cdot)$ has at most p possible output values as the input vector $\mathbf{x}$ varies, for any given choice of randomness, and so the information about $\mathbf{x}$ contained in the output $\mathbf{y}$ is at most $\log_2 p$ bits. Since the domain of $\mathbf{x}$ is $\{0,1\}^n$ of size 2^n for $n > 3\log_2 p$, it follows that the lossiness (i.e., the information loss about $\mathbf{x}$ for a uniformly distributed $\mathbf{x}$) caused by $F_{\text{ltdf}}(\mathbf{C}, \cdot)$ is equal to $k = \log_2(2^n) - \log_2 p = n - \log_2 p > n - n/3 = 2n/3$.

Indistinguishability is the only property that requires a complexity assumption, and whose proof makes use of the common reference string. The result is stated in Lemma 1.

Lemma 1. *Under the Decision Bilinear Diffie-Hellman assumption in* $\mathbb{G}$*, the distributions of public keys* $\mathbf{C}$ *generated by* S_{inj} *and by* S_{loss} *cannot be distinguished with non-negligible advantage by a probabilistic polynomial-time algorithm, in the CRS model.*

Proof (Proof.). The proof is based on a hybrid-game argument; that is, we gradually switch the LTDF from an injective-mode setup S_{inj} to a lossy-mode setup S_{loss}, one element of the diagonal at a time. That is, for $k \in [n]$, one k at a time, we change D_k from the injective definition (where $m_k = 1$) into the lossy definition (where $m_k = 0$). For each k, we then show that if the adversary can successfully distinguish the transition from H_k to H_{k+1}, then we can turn it into a BDH distinguisher. (And it must distinguish at least one such transition, if it is to distinguish injective from lossy.)

We begin by defining intermediate game H'_k where in game H'_k the first $k-1$ elements are in lossy mode, D_k is a random group element, and D_i for $i > k$ are chosen according to the injective setup.

Under the BDH assumption we can show that H_k is indistinguishable from H'_k. Our reduction algorithm takes in a d-BDH challenge tuple $g, g^a, g^b, g^c \in \mathbb{G}$ and $T \in \mathbb{G}_t$. For $i \neq k$ it chooses random a_i, b_i, r_i itself. It uses these to populate the CRS and the public key $\mathbf{B}$ at all positions except the k-th position. It then sets $g^{a_k} = g^a$, $g^{r_k} = g^c$, and $g^{b_k} = g^b$. Using these knowledge of b_i for $i \neq k$ the reduction can create $D_i = (e(g^{a_i}, g^{r_i}))^{b_i}$ for $i < k$ and for $D_i = (e(g^{a_i}, g^{r_i}))^{b_i} e(g, g)$ for $i > k$. It finally creates $D_k = Te(g, g)$. If T is a bilinear tuple we are in game H_k; otherwise we are in game H'_k.

A symmetrical argument can show that for all k it is difficult to distinguish between H'_k and H_{k+1}. Putting the sequence together completes our proof.

4 Compact LTDFs from DBDH in the Standard Model

We now present a second LTDF construction with public keys about as compact as our first construction, and likewise also based on the DBDH assumption in bilinear groups, but without CRS or any setup requirement.

4.1 Intuition

The general structure is similar to that of our first scheme, which is to say that it combines the PW LTDF strategy to achieve lossiness with some algebraic manipulations to remove as much redundancy as possible from the public key.

There is a paradox that we mentioned in the intuition of our first scheme: that, in order to achieve any meaningful compression of the elements of $\mathbf{C}$ without "outside help", we would need to expose the redundancy that was hidden in the matrix $\mathbf{C}$. But we could not do that, because it would allow an adversary to reconstruct not only its off-diagonal elements , but also the "forbidden" elements on the diagonal (those that determine whether the function is injective or lossy). Our solution was to use a (thankfully reusable) common reference string to reveal (the better part of) the off-diagonal elements without saying anything much at all about the diagonal ones.

Here, we will do much better, and resolve the paradox: we construct a special compact encoding that, without any outside help, will let anyone easily reconstruct the off-diagonal elements of $\mathbf{C}$, without exposing the diagonal ones (which are sent separately).

The idea is to construct a linear system of equations, with one equation per row i and one per column j, in such a way that each element at the intersection (i, j) can be obtained by solving the pair of the i-th row and j-column equations for two variables. The trick is that the equations are constructed in such a way that they become linearly dependent for $i = j$, rendering them useless on the diagonal — and only on the diagonal. As stated in the introduction, this can be viewed as a novel twist on Lewko's et al. [17] two equation-based revocation system, where we conceptually match up rows and columns of the Lossy TDF with ciphertexts and private keys of an broadcast revocation system.

Of course, we cannot give the equations in the clear, we use a technique that is quite common in Discrete-Log-hard groups, of encoding all the coefficients we need to hide as powers of some group generator. That is, instead of revealing $\alpha \in \mathbb{Z}_p$, we reveal $g^\alpha \in \mathbb{G}$. Because $\mathbb{Z}_p$ and $\mathbb{G}$ are homomorphic through exponentiation, we can still perform any additive operation that we like on the "α"s hidden in the exponents.

Then, since our stratagem with the pairs of equations require some multiplication at some point when trying to solve for the unknowns, we need to make use of a bilinear map in order to do this one multiplication "in the exponent".

We now describe our system. The crux of the construction lies in the denominator $\frac{1}{j-i}$ that we make appear along the computation path, and that will break things down when one tries it for $i = j$.

4.2 Construction

The notation regarding bilinear groups is the same as in Section 3. Likewise, to avoid clutter, we drop all "hats" in the notation, thereby implicitly assuming a symmetric pairing $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_t$ where $\mathbb{G} = \hat{\mathbb{G}}$. The system continues to work in the asymmetric setting, for any partition of the bilinear-group elements into $\mathbb{G}$ or $\hat{\mathbb{G}}$, as long as it is self-consistent.

Our compact LTDF from DBDH in the Standard Model is constructed as follows:

Sampling algorithm, Injective mode: The injective function generator S_{inj}, on input a security parameter 1^λ, selects a bilinear group $(p, \mathbb{G}, g, e) \leftarrow \mathcal{G}(1^\lambda)$, and fixes $n > 3 \log_2 p$. It also chooses two independent random generators $u \in \mathbb{G}$ and $h \in \mathbb{G}$ in addition to $g \in \mathbb{G}$. It then chooses random $r_i \in \mathbb{Z}_p$ for each $i \in [n]$, and random $z_j \in \mathbb{Z}_p$ for each $j \in [n]$.

- The LTDF function index, or "public key", is listed as $4n$ elements of $\mathbb{G}$ and n of $\mathbb{G}_t$:

$$\begin{array}{rcrcl} \text{for each "row" index} i \in [n] & : & R_i = g^{r_i} & , & S_i = (h^i \cdot u)^{r_i} \\ \text{for each "column" index} j \in [n] & : & V_j = g^{z_j} & , & W_j = (h^j \cdot u)^{z_j} \\ \text{for each "diagonal" element} k \in [n] & : & D_k = & & e(g, u)^{r_k z_k} \cdot e(g, g) \end{array}$$

- The LTDF trapdoor, or "private key", consists of the n-vector $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{Z}_p^n$.

Sampling algorithm, Lossy mode: The lossy function generator S_{loss}, on input a security parameter 1^λ, proceeds similarly as S_{inj} above, except for the key computation:

- The LTDF function index, or "public key", is listed as $4n$ elements of $\mathbb{G}$ and n of $\mathbb{G}_t$:

$$\begin{aligned}
\text{for each “row” index} i \in [n] \quad &: \quad R_i = g^{r_i} \quad , \quad S_i = (h^i \cdot u)^{r_i} \\
\text{for each “column” index} j \in [n] \quad &: \quad V_j = g^{z_j} \quad , \quad W_j = (h^j \cdot u)^{z_j} \\
\text{for each “diagonal” element} k \in [n] \quad &: \quad D_k = \ e(g,h)^{r_k z_k}
\end{aligned}$$

- There is no LTDF trapdoor (since the evaluation output will be lossy even with the knowledge of all the secret exponents).

Evaluation algorithm: The evaluation algorithm F_{ltdf} takes as input an LTDF public key and an n-bit binary input string $\mathbf{x} \in \{0,1\}^n$. Again, there are at least two functionally equivalent ways to perform the computation; we start with the inefficient one for expository purposes.

- *"Pedestrian" evaluation (requires a total of n^2 pairings and double-pairings):* To compute the desired output, the evaluation algorithm starts by reconstructing a virtual matrix $\mathbf{C}$ very similar to that of the CRS scheme of Section 3 (in its "pedestrian" implementation), and then proceed with the actual function evaluation. With the present notation, here the virtual matrix $\mathbf{C} = \mathbf{C}_1 \| \mathbf{C}_2$ is written,

$$\mathbf{C}_1 = \begin{pmatrix} e(g,h)^{r_1} \\ \vdots \\ e(g,h)^{r_n} \end{pmatrix}$$

$$\mathbf{C}_2 = \begin{pmatrix} e(g,h)^{r_1 z_1} \cdot e(g,g)^{m_1} & e(g,h)^{r_1 z_2} & \cdots & e(g,h)^{r_1 z_n} \\ e(g,h)^{r_2 z_1} & e(g,h)^{r_2 z_2} \cdot e(g,g)^{m_2} & \cdots & e(g,h)^{r_2 z_n} \\ \vdots & \vdots & \ddots & \vdots \\ e(g,h)^{r_n z_1} & e(g,h)^{r_n z_2} & \cdots & e(g,h)^{r_n z_n} \cdot e(g,g)^{m_n} \end{pmatrix}$$

As before, the "message" $(m_1, \ldots, m_n)$ is a vector of all 0 or all 1, depending whether the LTDF is lossy or injective (which fact must remain unknown to F_{ltdf}). We now show how the evaluation algorithm can compute $\mathbf{C}$ given the information at its disposal.

The zero-th column of $\mathbf{C}$ consists of elements of $\mathbb{G}_t$ that are computed from the public key. For each index $i \in [n]$, the i-th element of $\mathbf{C}_1$ is computed as the pairing,

$$\begin{aligned}
c_{i,0} \ &:= e(R_i, h) \\
&= e(g,h)^{r_i} \ \in \mathbb{G}_t \quad \text{which is indeed the required value for } c_{i,0}
\end{aligned}$$

The diagonal elements of $\mathbf{C}$ are elements of $\mathbb{G}_t$ which are explicitly given in the public key. For $k \in [n]$, those are the given values D_k. Hence, the algorithm simply assigns, for $i \in [n]$,

$$\begin{aligned} c_{i,i} &:= D_i \\ &= e(g,h)^{r_i z_i} \cdot e(g,g)^{m_i} \in \mathbb{G}_t \quad \text{as required, with } m_i \in \{0,1\} \text{ unknown} \end{aligned}$$

The off-diagonal of $\mathbf{C}$, i.e., the values $c_{i,j}$ with indices $i, j \in [n]$ such that $i \neq j > 0$, must be computed explicitly. For each such pair (i, j), the algorithm F_{ltdf} computes a double-pairing, which is a product, or more precisely here, a ratio, of two pairings. (Such "double pairings" can be computed almost as fast as a single pairing.) It computes,

$$c'_{i,j} := \frac{e(R_i, W_j)}{e(V_j, S_i)} = \frac{e(g, h^j)^{r_i z_j} \cdot e(g, u)^{r_i z_j}}{e(g, h^i)^{r_i z_j} \cdot e(g, u)^{r_i z_j}} = \frac{e(g, h^j)^{r_i z_j}}{e(g, h^i)^{r_i z_j}} = \Big(e(g,h)^{r_i z_j}\Big)^{j-i}$$

and for each such value takes its $(j-i)$-th root,

$$c_{i,j} := \big(c'_{i,j}\big)^{\frac{1}{j-i}} = \Big(e(g,h)^{r_i z_j}\Big)^{\frac{j-i}{j-i}} = e(g,h)^{r_i z_j} \quad \text{as required for } c_{i,j}$$

Thus, for $i \neq j$ this gives the desired outcome, $e(g,h)^{r_i z_j}$, as the preceding derivations show. Crucially, this computation and the root-taking step in particular will succeed if and only if $i \neq j$. Hence, a malicious F_{ltdf} cannot use this method to compute $e(g,h)^{r_k z_k}$ on the diagonal, and, by comparison with D_k, infer whether the LTDF is lossy or injective.

The actual LTDF output is the row-vector $\mathbf{y} = \mathbf{x} \cdot \mathbf{C}$ with $n+1$ elements in $\mathbb{G}_t$, given by,

$$\mathbf{y} = \mathbf{x} \cdot \mathbf{C} = \mathbf{x} \cdot (\mathbf{C}_1 \| \mathbf{C}_2) = \Big(\ \textstyle\prod_{i=1}^n c_{i,0}^{x_i} \ , \prod_{i=1}^n c_{i,1}^{x_i} \ , \ldots \prod_{i=1}^n c_{i,n}^{x_i} \ \Big) \in \mathbb{G}_t^{n+1}$$

- *"Shortcut" evaluation (requires n double-pairings plus 1 pairing):*

  As in Section 3, it is possible to delay the pairing computation and do the group multiplications first. This reduces the number of pairings (or more precisely, two-pairing ratios) from n to 1, for each element of the output vector.

  Thus, to compute the output $\mathbf{y}$, a row-vector of $n+1$ elements, it suffices for the algorithm F_{ltdf} to compute y_j for $j = 0$ and $j \in [n]$ respectively as follows,

$$\begin{aligned} y_0 &:= e\Big(\prod_{i=1}^n R_i^{x_i} \ , \ h\Big) \\ &= \prod_{i=1}^n e(g,h)^{r_i x_i} \in \mathbb{G} \quad \text{which is the required value for the output } y_0 \end{aligned}$$

$$
\begin{aligned}
y_j \; &:= \left(e\Big(\textstyle\prod_{i\neq j} R_i^{\frac{x_i}{j-i}} \,,\, W_j \Big) / e\Big(V_j \,,\, \textstyle\prod_{i\neq j} S_i^{\frac{x_i}{j-i}} \Big) \right) \cdot D_j^{x_j} \\
&= \frac{e\Big(\prod_{i\neq j} g^{\frac{r_i x_i}{j-i}} \,,\, (h^j \cdot u)^{z_j} \Big)}{e\Big(g^{z_j} \,,\, \prod_{i\neq j} (h^i \cdot u)^{\frac{r_i x_i}{j-i}} \Big)} \cdot D_j^{x_j} = \frac{\prod_{i\neq j} e(g,h)^{j \frac{r_i x_i}{j-i} z_j}}{\prod_{i\neq j} e(g,h)^{i \frac{r_i x_i}{j-i} z_j}} \cdot D_j^{x_j} \\
&= e(g,h)^{\sum_{i\neq j} \frac{j}{j-i} r_i x_i z_j} \cdot e(g,h)^{\sum_{i\neq j} \frac{-i}{j-i} r_i x_i z_j} \cdot D_j^{x_j} \\
&= e(g,h)^{\sum_{i\neq j} \frac{j-i}{j-i} r_i x_i z_j} \cdot D_j^{x_j} = \prod_{i\neq j} e(g,h)^{r_i x_i z_j} \cdot D_j^{x_j} \\
&= \prod_{i=1}^{n} e(g,h)^{r_i x_i z_j} \cdot e(g,g)^{m_j x_j} \;\; \in \mathbb{G}_t \quad \text{as required for } y_j
\end{aligned}
$$

It is easy to see that this "Shortcut" evaluation gives the same result as the "Pedestrian" one, while requiring much less memory and being much faster.

Inversion algorithm: The inversion algorithm F^{-1}_{ltdf} is input $(\mathbf{z}, \mathbf{y})$, where $\mathbf{z} = (b_1, \ldots, b_n) \in \mathbb{Z}_p^n$ is the trapdoor output by the injective-mode S_{inj}, and $\mathbf{y} = (y_0, y_1, \ldots, y_n) \in \mathbb{G}_t^{n+1}$ is the output of the evaluation algorithm F_{ltdf}. The bits of the preimage $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ of $\mathbf{y}$ are output as follows,

$$
x_j = \begin{cases} 0 & \text{if } y_j = y_0^{z_j} \cdot e(g,g)^0 \\ 1 & \text{if } y_j = y_0^{z_j} \cdot e(g,g)^1 \end{cases}
$$

4.3 Security

Theorem 2. *The preceding algorithms define a collection of $(n, n - \log_2 p)$-Lossy TDFs under the Decision BDH assumption for $\mathbb{G}$.*

Proof. The theorem follows from the following Lemmas 2, 3, and 4.

Lemma 2. *For all LTDF parameters $(\mathbf{C}, \mathbf{z})$ produced by the injective-mode setup function S_{inj}, we have that $\forall \mathbf{x} \in \{0,1\}^n$, $F^{-1}_{ltdf}(\mathbf{z}, F_{ltdf}(\mathbf{C}, \mathbf{x})) = \mathbf{x}$, i.e., the LTDF output is invertible.*

Proof. By inspection of the algorithm specifications. Notice that in injective mode (i.e., for parameters sampled by S_{inj}), the exponents m_i on the diagonal are equal to 1, and thus preserve the information in the bit x_i, allowing the inversion algorithm to proceed.

Lemma 3. *For all LTDF parameters $\mathbf{C}$ produced by the lossy-mode setup function S_{loss}, the "lossiness" (or information loss for a uniform input vector $\mathbf{x}$) of the function $F_{ltdf}(\mathbf{C}, \cdot)$ is at least $k = n - \log_2 p > \frac{2}{3} n$ bits.*

Proof. By construction of the lossy-mode parameters generated by S_{loss}, the evaluation function $F_{\mathrm{ltdf}}(\mathbf{C}, \cdot)$ has at most p possible output values as the input vector $\mathbf{x}$ varies, for any choice of randomness. Thus the information leakage about $\mathbf{x}$ is at most $\log_2 p$ bits. Since the domain of $\mathbf{x}$ is $\{0,1\}^n$ of size 2^n for $n > 3 \log_2 p$, it follows that the lossiness of $F_{\mathrm{ltdf}}(\mathbf{C}, \cdot)$ is $k = \log_2(2^n) - \log_2 p = n - \log_2 p > n - n/3 = 2n/3$.

Lemma 4. *Under the DBDH assumption in $\mathbb{G}$, the distributions of public keys $\mathbf{C}$ generated by S_{inj} and by S_{loss} cannot be distinguished with non-negligible advantage by a probabilistic polynomial-time algorithm.*

Proof. The proof is based on a hybrid-game argument; that is, we gradually switch the LTDF from an injective-mode setup S_{inj} to a lossy-mode setup S_{loss}, one element of the diagonal at a time. That is, for $k \in [n]$, one k at a time, we change D_k from the injective definition (where $m_k = 1$) into the lossy definition (where $m_k = 0$). For each k, we then show that if the adversary can successfully distinguish the transition from H_k to H_{k+1}, then we can turn it into a BDH distinguisher. (And it must distinguish at least one such transition, if it is to distinguish injective from lossy.)

Here are the details of the hybrid argument.

We define a series of experiments $H_1, ..., H_{n+1}$, where, in the experiment H_k, the LTDF setup function sets $D_j = e(g,h)^{r_j z_j}$ for all $j < k$, and sets $D_j = e(g,h)^{r_j z_j} e(g,g)$ for all $j \geq k$. Recall that all the other components of the public key (i.e., other than the D_j) are defined identically in the injective mode and the lossy mode.

Observe that H_1 is an injective key with the correct distribution, and that H_{n+1} is a lossy key also with the correct distribution. It follows that if for no k can an adversary distinguish between H_k and H_{k+1}, then no adversary can distinguish the injective mode from the lossy mode.

Suppose then that for some k an adversary distinguishes H_k and H_{k+1}. Then we can use that adversary to solve a Decisional BDH challenge, thereby contradicting our assumption. Let thus $g, g^a, g^b, g^c \in \mathbb{G}$ and $T \in \mathbb{G}_t$ be our DBDH challenge, where T is either $e(g,g)^{abc}$ or a random element in $\mathbb{G}_t$.

For any fixed $k \in [n]$, the reduction for the transition from H_k to H_{k+1} works as follows.

For all $i \neq k$ and $j \neq k$, the simulator simply chooses $r_i \in \mathbb{Z}_p$ and $z_j \in \mathbb{Z}_p$ at random, and computes the corresponding public-key elements R_i and V_j as in the actual scheme.

For $i = k$ and $j = k$, the simulator instead chooses some random $y \in \mathbb{Z}_p$, and lets $h = (g^c)$ and $u = h^{-k} g^y$. It then sets $R_k = (g^a)$ and $V_k = (g^b)$ (thereby implictly setting $a = r_k$ and $b = z_k$, even though it does not know such values).

The simulator can now compute the public-key components S_i and W_j for all $i \neq k$ and $j \neq k$, since it knows the exponents r_i and z_j can thus compute $S_k = (g^a)^y$ and $W_k = (g^b)^y$. This works only because the contribution of $h = (g^c)$ vanishes from within S_k and W_k for this value of k, and one can check that this gives the correct values for S_k and W_k.

The simulator can also compute the public-key components D_i for all $i \neq k$. For consistency and continuity within the entire hybrid sequence of games, for $i < k$ the components D_i are set to be lossy (i.e., computed as in S_{loss}), while for $i > k$ the D_i are set to be injective (i.e., computed as in S_{inj}).

To set the one remaining public-key component, D_k, the simulator flips a coin $\beta \in \{0,1\}$ and defines $D_k = T \cdot e(g,g)^{\beta}$. The public key is given to the adversary, who must tell whether it is lossy or injective.

If the BDH instance was genuine, then $T = e(g, g)$ and the adversary will be in the situation of having to distinguish H_k (if $\beta = 1$) from H_{k+1} (if $\beta = 0$); it will thus be able to exploit its advantage, if any. However, if T is random then the adversary cannot have any advantage in this situation since the bit β is completely hidden from its view.

It follows that any advantage ϵ that the adversary has at distinguishing H_k from H_{k+1}, gives us a BDH distinguisher with advantage $\epsilon/2$.

To conclude the hybrid argument, considering the entire sequence of hybrid transitions from H_k to H_{k+1} for $k \in [n]$, we deduce that any adversary that can distinguish an injective key from a lossy key with advantage ϵ, must be able to distinguish H_k from H_{k+1} for at least one value of k with advantage ϵ/n, which in turn will give us a distinguisher for solving DBDH with advantage $\epsilon/(2n)$.

This concludes the proof of indistinguishability between the injective and lossy modes.

5 Conclusion

We have proposed two new Lossy Trapdoor Function schemes of the "discrete-log" type, that are significantly more efficient than earlier comparable constructions. We gave two schemes: with and withour common reference string. Both of them make use of pairings, and have efficient security reductions from the classic DBDH assumption.

The main advantage of "discrete-log-type" LTDF constructions, compared to more efficient ones of the "factoring" type (based on the Paillier trapdoor), is that the hardness of problems related to the discrete log generally depend on the choice of group, unlike those related to factoring that are, so to speak, absolute. In other words, even though Factoring is liable to be universally easy independently in all choices of context, there is a hope that Discrete Log and related problems can remain hard for certain choices of groups (e.g., the counterfactual generic groups [25]), provided that we can find and construct them.

References

1. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.
2. Mihir Bellare, Shai Halevi, Amit Sahai, and Salil P. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In *CRYPTO*, pages 283–298, 1998.
3. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
4. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, 1988.
5. Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *CRYPTO*, pages 335–359, 2008.

6. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, volume 3152 of *LNCS*, pages 41–55, 2004.
7. Xavier Boyen. Lattice mixing and vanishing trapdoors : A framework for fully secure short signatures and more. In *Public Key Cryptography*, LNCS, 2010.
8. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, 2010.
9. Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136, 2001.
10. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
11. Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. Preliminary version in STOC 1991.
12. David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. Cryptology ePrint Archive, Report 2009/590, 2009. `http://eprint.iacr.org/`.
13. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
14. Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, pages 31–51, 2008.
15. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.
16. Brett Hemenway and Rafail Ostrovsky. Lossy trapdoor functions from smooth homomorphic hash proof systems. Manuscript, 2010. `http://www.math.ucla.edu/~bretth/papers/uhp_ltdf.pdf`.
17. Allison Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *Security and Privacy*, 2010.
18. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.
19. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
20. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.
21. Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology, 1979.
22. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
23. Alon Rosen and Gil Segev. Efficient lossy trapdoor functions based on the composite residuosity assumption. Cryptology ePrint Archive, Report 2008/134, 2008. `http://eprint.iacr.org/`.
24. Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/`.
25. Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *LNCS*, pages 256–66, 1997.
26. Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91, 1982.