Trapdoor Functions from the Computational Diffie-Hellman Assumption*

Sanjam Garg† Mohammad Hajiabadi‡

June 4, 2018

Abstract

Trapdoor functions (TDFs) are a fundamental primitive in cryptography. Yet, the current set of assumptions known to imply TDFs is surprisingly limited, when compared to public-key encryption. We present a new general approach for constructing TDFs. Specifically, we give a generic construction of TDFs from any Hash Encryption (Döttling and Garg [CRYPTO '17]) satisfying a novel property which we call recyclability. By showing how to adapt current Computational Diffie-Hellman (CDH) based constructions of hash encryption to yield recyclability, we obtain the first construction of TDFs with security proved under the CDH assumption. While TDFs from the Decisional Diffie-Hellman (DDH) assumption were previously known, the possibility of basing them on CDH had remained open for more than 30 years.

1 Introduction

Trapdoor functions (TDFs) are a fundamental primitive in cryptography, historically pre-dating the complexity-based development of public key encryption (PKE) [DH76, RSA78]. Informally, TDFs are a family of functions, where each function in the family is easy to compute given the function's index key, and also easy to invert given an associated trapdoor key. The security requirement is that a randomly chosen function from the family should be hard to invert without knowledge of a trapdoor key.

A salient difference between TDFs and PKE lies in their inversion (decryption) algorithms: while the inversion algorithm of a TDF recovers the entire pre-image in full, the decryption algorithm of a PKE only recovers the corresponding plaintext, and not necessarily the randomness. This full-input recovery feature of TDFs is useful in may applications. For example, suppose we have two image points $y_1 := \mathsf{F}(\mathsf{ik}_1, x_1)$ and $y_2 := \mathsf{F}(\mathsf{ik}_2, x_2)$ of a trapdoor function F, and we want to convince Alice — who is given both y_1 and y_2 but only a trapdoor key tk_1 for ik_1 — that $x_1 = x_2$. This will be easy for Alice to do herself: retrieve x_1 from y_1 using the trapdoor key tk_1 and check whether $y_1 = \mathsf{F}(\mathsf{ik}_1, x_1)$ and $y_2 = \mathsf{F}(\mathsf{ik}_2, x_1)$.[1] This is a very useful property, especially in the context

*Research supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, AFOSR YIP Award, 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 and University of Virginia.

[1] Here we also need to assume that certifying whether a given point is in the domain of a trapdoor function can be done efficiently.

of chosen-ciphertext (CCA2) security, and is in fact the main reason behind the success of building CCA2-secure PKE in a black-box way from various forms of TDFs [PW08, RS09, KMO10]. In contrast, enabling this technique based on PKE [NY90] requires the use of expensive non-interactive zero knowledge proofs [BFM90, FLS90], which in turn require strong assumptions and lead to non-black-box constructions.

The deterministic structure of TDFs, however, comes with a price, making the construction of TDFs more challenging than that of PKE. This belief is justified by an impossibility result of Gertner, Malkin and Reingold [GMR01] showing that TDFs cannot be built from PKE in a black-box way. As another evidence, while it was known from the 80's how to build semantically-secure PKE from the Decisional Diffie-Hellman (DDH) assumption [Yao82, GM82, ElG84], it took two decades to realize TDFs based on DDH [PW08].

Despite the fundamental nature of TDFs and extensive research on them [BHSV98, BBO07, PW08, BFOR08, RS09, KMO10, FGK+10, Wee12] a long-standing question has remained open:

Can trapdoor functions be based on the Computational Diffie-Hellman (CDH) Assumption?

The main difficulty of the above question is that all known DDH-based constructions of TDFs, e.g., [PW08, FGK+10], exploit properties of DDH, such as pseudorandomness of low rank matrices of group elements, which do not hold in the CDH setting (see Section 1.1).

Apart from being a natural question, it has the following theoretical motivation: since we know that TDFs are not necessary in a black-box sense for PKE [GMR01], there may be computational assumptions that imply PKE but not TDFs. Thus, it is important to understand whether TDFs can be obtained from all existing computational assumptions that imply PKE. This provides insights into the hardness nature of TDFs as well as our computational assumptions.

1.1 Lack of CDH-Based Techniques for TDF

Diffie-Hellman related assumptions (even DDH) do not naturally lend themselves to a TDF construction. The main reason why it is more difficult to build TDFs from such assumptions, compared to, say, factoring related assumptions, is that we do not know of any generic trapdoors for the *discrete log* problem. Indeed, a long standing open problem in cryptography is whether PKE can be based on the sole hardness of the discrete log problem. To see how this makes things more difficult, consider ElGamal encryption: to encrypt a group element g_m under a public key (g, g_1), we return $(g^r, g_1^r \cdot g_m)$, where r is a random exponent. The decryption algorithm can recover g_m but not r because computing r is as hard as solving the discrete log problem.

Known DDH-based TDF constructions [PW08, FGK+10] get around the above obstacle by designing their TDFs in such a way that during inversion, one will only need to solve the discrete log problem over a small space, e.g., recovering a bit b from g^b. The main idea is as follows: the index key ik of their TDF is g^{M}, where g is a generator of the group $\mathbb{G}$ of order p and $\mathsf{M} \in \mathbb{Z}_p^{n \times n}$ is a random $n \times n$ invertible matrix and g^{M} denotes entry-wise exponentiation. Let $\mathsf{tk} := \mathsf{M}^{-1}$ be the trapdoor key. Using ik, the evaluation algorithm on input $\mathsf{x} \in \{0,1\}^n$ may use the algebraic property of the group to compute $\mathsf{y} := g^{\mathsf{M}\mathsf{x}} \in \mathbb{G}^n$. Now using tk and y one can compute $g^{\mathsf{x}} \in \mathbb{G}^n$, hence retrieving x.

To argue about one-wayness, one uses the following property implied by DDH: the matrix g^{M} is computationally indistinguishable from a matrix g^{M_1}, where M_1 is a random matrix of rank one. If

the index key is now set to g^{M_1} and if we have $2^n \gg p$, then even an unbounded adversary cannot retrieve the original x from y. This argument is used to establish one-wayness for the TDF.

Unfortunately, the above rank indistinguishability property used to prove one-wayness is not known (and not believed) to be implied by CDH. Thus, designing TDFs based on CDH requires new techniques.

Finally, we mention that even from the Computational Bilinear Assumption [BB04] (i.e., pairing-based CDH) no TDF constructions are known. The closest is a result of Wee [Wee10], showing that trapdoor relations, which are much weaker than TDFs, can be built from CDH. Roughly, trapdoor relations are a relaxed version of TDFs, in that the function might not be efficiently computable on individual points but one may sample efficiently a *random* input element together with its corresponding image.

1.2 Our Results and Techniques

We give the first construction of TDFs under the CDH assumption. Our construction is black-box and is obtained through a general construction of TDFs from a primitive we call a *recyclable one-way function with encryption (OWFE)*. Moreover, we show that an *adaptive* strengthening of our notion of recyclable OWEF yields a black-box construction of CCA2-secure PKE.

OWFE notion. An OWFE is described by a one-way function $\mathsf{f}_{\mathsf{pp}}\colon \{0,1\}^n \to \{0,1\}^\nu$, where pp is a public parameter, together with encapsulation/decapsulation algorithms (E, D). Specifically, E takes as input pp, an image $\mathsf{y} \in \{0,1\}^\nu$ of f_{pp}, an index $i \in [n]$ and a selector bit $b \in \{0,1\}$, and produces an encapsulated ciphertext ct and a corresponding *key bit* $\mathsf{e} \in \{0,1\}$. The algorithm D allows anyone to retrieve e from ct using any pre-image x of y whose ith bit is b. For security, letting $\mathsf{y} := \mathsf{f}_{\mathsf{pp}}(\mathsf{x})$, we require that if $(\mathsf{ct}, \mathsf{e}) \xleftarrow{\$} \mathsf{E}(\mathsf{pp}, \mathsf{y}, (i, b))$ and $\mathsf{x}_i \neq b$, then even knowing x one cannot recover e from ct with a probability better than $\frac{1}{2} + \mathsf{negl}(\lambda)$, where λ is the security parameter. That is, for any $\mathsf{x} \in \{0,1\}^n$, $i \in [n]$, we have $(\mathsf{x}, \mathsf{ct}, \mathsf{e}) \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{ct}, \mathsf{e}')$, where $\mathsf{e}' \xleftarrow{\$} \{0,1\}$, $(\mathsf{ct}, \mathsf{e}) \xleftarrow{\$} \mathsf{E}(\mathsf{pp}, \mathsf{f}(\mathsf{pp}, \mathsf{x}), (i, 1-\mathsf{x}_i))$ and $\stackrel{c}{\equiv}$ denotes computational indistinguishability. Our OWFE notion is a weakening of the chameleon-encryption notion [DG17b] in that we do not require f to be collision resistant. The following is a variant of the CDH-based construction of [DG17b].

CDH-based instantiation of OWFE [DG17b]. Let $\mathbb{G}$ be a group of prime order p. The public parameter is a $2 \times n$ matrix of random group elements $\mathsf{pp} := \left(\begin{smallmatrix} g_{1,0}, g_{2,0}, \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{smallmatrix}\right)$ and $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x} \in \{0,1\}^n) = \prod_{j \in [n]} g_{j,\mathsf{x}_j}$.

To perform $\mathsf{E}(\mathsf{pp}, \mathsf{y}, (i, b))$, sample $\rho \xleftarrow{\$} \mathbb{Z}_p$ and return $(\mathsf{ct}, \mathsf{e})$, where

$$\mathsf{ct} := \left(\begin{smallmatrix} g'_{1,0}, g'_{2,0}, \ldots, g'_{n,0} \\ g'_{1,1}, g'_{2,1}, \ldots, g'_{n,1} \end{smallmatrix}\right), \text{ where } g'_{i,1-b} := \perp, g'_{i,b} := g^\rho_{i,b}$$

$$\text{and for all } j \neq i : g'_{j,0} := g^\rho_{j,0} \text{ and } g'_{j,1} := g^\rho_{j,1}$$

and $\mathsf{e} := \mathsf{HC}(\mathsf{y}^\rho)$, where HC is a hardcore bit function. The function D is now derived easily. See [DG17b] (or Appendix A) for the proof of security.

Recyclability. Our recyclability notion asserts that the ciphertext part output, ct, of the key encapsulation algorithm E is independent of the corresponding image input part y. That is, letting E_1 and E_2 refer to the first and second output of E, for any values of y_1 and y_2, we always have $\mathsf{E}_1(\mathsf{pp}, \mathsf{y}_1, (i,b); \rho) = \mathsf{E}_1(\mathsf{pp}, \mathsf{y}_2, (i,b); \rho)$. It is easy to verify that the above CDH-based OWFE satisfies this property. Thus, we may drop y as an input to E_1 and obtain the following:

Property 1. Letting $\mathsf{x} \in \{0,1\}^n$, $\mathsf{x}_i = b$, $\mathsf{ct} := \mathsf{E}_1(\mathsf{pp}, (i,b); \rho)$ and $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$ we have

$$\mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct}) = \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,b); \rho).$$

1.3 Sketch of our OWFE-Based Construction and Techniques

Let $(\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$ be a recyclable OWFE scheme.[2] Our TDF construction is based on a new technique that we call *bits planting*. Briefly, the input X to our TDF consists of a domain element $\mathsf{x} \in \{0,1\}^n$ of $\mathsf{f}(\mathsf{pp}, \cdot)$ and a *blinding* string $\mathbf{b} \in \{0,1\}^{n \times r}$, for some r that we will specify later. The output Y is comprised of $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$, as well as a matrix of bits in which we copy all the bits of $\mathbf{b}$ in the clear but in hidden spots determined by x; we fill up the rest of the matrix with key bits that somehow correspond to bit-by-bit encryption of x under y. To an adversary, the matrix is "unrevealing," with no indicative signs of what spots corresponding to the blinding part — which contain $\mathbf{b}$ in the clear. However, using our designed trapdoor, an inverter can pull out both x and $\mathbf{b}$ from Y with all but negligible probability.

Warm-up construction. We first give a warm up construction in which our inversion algorithm only recovers half of the input bits (on average). Our TDF input is of the form $(\mathsf{x}, \mathbf{b}) \in \{0,1\}^n \times \{0,1\}^n$. This warm-up construction contains most of the ideas behind the full-blown construction.

- **Key generation:** The trapdoor key is $\mathsf{tk} := \left(\begin{smallmatrix} \rho_{1,0}, \ldots, \rho_{n,0} \\ \rho_{1,1}, \ldots, \rho_{n,1} \end{smallmatrix}\right)$ a matrix of randomness values, and the index key is $\mathsf{ik} := \mathsf{pp}, \left(\begin{smallmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{smallmatrix}\right)$, formed as:

$$\mathsf{ik} := \mathsf{pp}, \begin{pmatrix} \mathsf{ct}_{1,0} := \mathsf{E}_1(\mathsf{pp}, (1,0); \rho_{1,0}), \ldots, \mathsf{ct}_{n,0} := \mathsf{E}_1(\mathsf{pp}, (n,0); \rho_{n,0}) \\ \mathsf{ct}_{1,1} := \mathsf{E}_1(\mathsf{pp}, (1,1); \rho_{1,1}), \ldots, \mathsf{ct}_{n,1} := \mathsf{E}_1(\mathsf{pp}, (n,1); \rho_{n,1}) \end{pmatrix}.$$

- **Evaluation $\mathsf{F}(\mathsf{ik}, \mathsf{X})$:** Parse $\mathsf{ik} := \mathsf{pp}, \left(\begin{smallmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{smallmatrix}\right)$ and parse the input X as $(\mathsf{x} \in \{0,1\}^n, \mathbf{b} := \mathsf{b}_1 \cdots \mathsf{b}_n \in \{0,1\}^n)$. Set $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$. For $i \in [n]$ set M_i as follows:

$$\begin{aligned} \mathsf{M}_i := \begin{pmatrix} \mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct}_{i,0}) \\ \mathsf{b}_i \end{pmatrix} \overset{*}{=} \begin{pmatrix} \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,0); \rho_{i,0}) \\ \mathsf{b}_i \end{pmatrix} \text{ if } \mathsf{x}_i = 0 \\ \mathsf{M}_i := \begin{pmatrix} \mathsf{b}_i \\ \mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct}_{i,1}) \end{pmatrix} \overset{*}{=} \begin{pmatrix} \mathsf{b}_i \\ \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,1); \rho_{i,1}) \end{pmatrix} \text{ if } \mathsf{x}_i = 1 \end{aligned} \tag{1}$$

  The matrix M_i is computed using the *deterministic* algorithm D and the equalities specified as $\overset{*}{=}$ follow by Property (1).

  Return $\mathsf{Y} := (\mathsf{y}, \mathsf{M}_1 || \ldots || \mathsf{M}_n)$.

[2] K is the public-parameter generation algorithm. See Definition 3.1.

- **Inversion $\mathsf{F}^{-1}(\mathsf{tk}, \mathsf{Y})$:** Parse $\mathsf{Y} := (\mathsf{y}, \mathsf{M}_1 || \ldots || \mathsf{M}_n)$ and $\mathsf{tk} := \binom{\rho_{1,0}, \ldots, \rho_{n,0}}{\rho_{1,1}, \ldots, \rho_{n,1}}$. Set

$$(\mathsf{M}'_1 || \ldots || \mathsf{M}'_n) := \begin{pmatrix} \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (1,0); \rho_{1,0}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (n,0); \rho_{n,0}) \\ \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (1,1); \rho_{1,1}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (n,1); \rho_{n,1}) \end{pmatrix}. \tag{2}$$

  Output $(\mathsf{x}, \mathbf{b} := \mathsf{b}_1 \ldots \mathsf{b}_n)$, where we retrieve x_i and b_i as follows. If $\mathsf{M}'_{i,1} = \mathsf{M}_{i,1}$ and $\mathsf{M}'_{i,2} \neq \mathsf{M}_{i,2}$ (where $\mathsf{M}_{i,1}$ is the first element of of M_i), then set $\mathsf{x}_i := 0$ and $\mathsf{b}_i := \mathsf{M}_{i,2}$. If $\mathsf{M}'_{i,1} \neq \mathsf{M}_{i,1}$ and $\mathsf{M}'_{i,2} = \mathsf{M}_{i,2}$, then set $\mathsf{x}_i := 1$ and $\mathsf{b}_i := \mathsf{M}_{i,1}$. Else, set $\mathsf{x}_i := \perp$ and $\mathsf{b}_i := \perp$.

One-wayness (sketch). We show $(\mathsf{ik}, \mathsf{Y}) \stackrel{c}{\equiv} (\mathsf{ik}, \mathsf{Y}_{\text{sim}})$, where $(\mathsf{ik}, \mathsf{Y})$ is as above and

$$\mathsf{Y}_{\text{sim}} := \mathsf{y}, \begin{pmatrix} \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (1,0); \rho_{1,0}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (n,0); \rho_{n,0}) \\ \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (1,1); \rho_{1,1}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (n,1); \rho_{n,1}) \end{pmatrix}.$$

Noting that we may produce $(\mathsf{ik}, \mathsf{Y}_{\text{sim}})$ using only pp and y, the one-wayness of $\mathsf{f}(\mathsf{pp}, \cdot)$ implies it is hard to recover x from $(\mathsf{ik}, \mathsf{Y}_{\text{sim}})$, and so also from $(\mathsf{ik}, \mathsf{Y})$.

Why $(\mathsf{ik}, \mathsf{Y}) \stackrel{c}{\equiv} (\mathsf{ik}, \mathsf{Y}_{\text{sim}})$? Consider $\mathsf{Y}_{\text{sim},1}$, whose first column is the same as Y_{sim} and whose subsequent columns are the same as Y. We prove $(\mathsf{x}, \mathsf{ik}, \mathsf{Y}) \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{ik}, \mathsf{Y}_{\text{sim},1})$; the rest will follow using a hybrid argument.

Letting M_1 be formed as in Equations 1, to prove $(\mathsf{x}, \mathsf{ik}, \mathsf{Y}) \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{ik}, \mathsf{Y}_{\text{sim},1})$ it suffices to show

$$\mathsf{x}, \begin{pmatrix} \mathsf{E}_1(\mathsf{pp}, \mathsf{y}, (1,0); \rho_{1,0}) \\ \mathsf{E}_1(\mathsf{pp}, (1,1); \rho_{1,1}) \end{pmatrix}, \mathsf{M}_1 \stackrel{c}{\equiv} \mathsf{x}, \begin{pmatrix} \mathsf{E}_1(\mathsf{pp}, (1,0); \rho_{1,0}) \\ \mathsf{E}_1(\mathsf{pp}, (1,1); \rho_{1,1}) \end{pmatrix}, \begin{pmatrix} \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (1,0); \rho_{1,0}) \\ \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (1,1); \rho_{1,1}) \end{pmatrix}. \tag{3}$$

We prove Equation 3 using the security property of OWFE, which says

$$\left(\mathsf{x}, \mathsf{E}_1(\mathsf{pp}, (1, 1-\mathsf{x}_1); \rho), \mathsf{b}'\right) \stackrel{c}{\equiv} \left(\mathsf{x}, \mathsf{E}_1(\mathsf{pp}, (1, 1-\mathsf{x}_1); \rho), \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (1, 1-\mathsf{x}_1); \rho)\right), \tag{4}$$

where $\mathsf{b}' \stackrel{\$}{\leftarrow} \{0,1\}$ and ρ is random. We give an algorithm that converts a sample from either side of Equation 4 into a sample from the same side of Equation 3. On input $(\mathsf{x}, \mathsf{ct}_1, \mathsf{b}_1)$, sample $(\mathsf{ct}_2, \mathsf{b}_2) \stackrel{\$}{\leftarrow} \mathsf{E}(\mathsf{pp}, \mathsf{y}, (1, \mathsf{x}_1))$ and

- if $\mathsf{x}_1 = 0$, then return $\mathsf{x}, \binom{\mathsf{ct}_2}{\mathsf{ct}_1}, \binom{\mathsf{b}_2}{\mathsf{b}_1}$;
- else if $\mathsf{x}_1 = 1$, then return $\mathsf{x}, \binom{\mathsf{ct}_1}{\mathsf{ct}_2}, \binom{\mathsf{b}_1}{\mathsf{b}_2}$.

The claimed property of the converter follows by inspection. Finally, we mention that the argument used to prove Equation 3 is similar to a previous technique used by Brakerski et al. [BLSV18] to build circularly-secure PKE.

Correctness. F^{-1} recovers on average half of the input bits: F^{-1} fails for an index $i \in [n]$ if $\mathsf{b}_i = \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i, 1-\mathsf{x}_i); \rho_{i,1-\mathsf{x}_i})$. This happens with probability $\frac{1}{2}$ because b_i is a completely random bit.

Boosting correctness. To boost correctness, we provide r blinding bits for each index i of $\mathsf{x} \in \{0,1\}^n$. That is, the input to the TDF is $(\mathsf{x}, \mathbf{b}) \in \{0,1\}^n \times \{0,1\}^{rn}$. We will also expand ik by providing r encapsulated ciphertexts for each position $(i, b) \in [n] \times \{0,1\}$. This extra information will bring the inversion error down to 2^{-r}. We will show that one-wayness is still preserved.

Hardcore bits. Having a TDF, we may use standard techniques to derive hardcore functions [GL89]. In Section D we show more direct ways of doing so.

On the role of blinding. One may wonder why we need to put a blinding string $\mathbf{b}$ in the TDF input. Why do not we simply let the TDF input be x and derive multiple key bits for every index i of x by applying D to the corresponding ciphertexts provided for that position i in the index key ik; the inverter can still find the matching bit for every index. The reason behind our design choice is that by avoiding blinders, it seems very difficult (if not impossible) to give a reduction to the security of the OWFE scheme.

1.4 CCA2 Security

Rosen and Segev [RS09] show that an extended form of one-wayness for TDFs, which they term *k-repetition security*, leads to a black-box construction of CCA2-secure PKE. Informally, a TDF is k-repetition secure if it is hard to recover a random input X from $\mathsf{F}(\mathsf{ik}_1, \mathsf{X}), \ldots, \mathsf{F}(\mathsf{ik}_k, \mathsf{X})$, where $\mathsf{ik}_1, \ldots, \mathsf{ik}_k$ are sampled independently. They show that k-repetition security, for $k \in \Theta(\lambda)$, suffices for CCA2 security.

We give a CCA2 secure PKE by largely following [RS09], but we need to overcome two problems. The first problem is that our TDF is not k-repetition secure, due to the blinding part of the input. We overcome this by observing that a weaker notion of k-repetition suffices for us: one in which we should keep x the same across all k evaluations but may sample $\mathbf{b}$ freshly each time. A similar weakening was also used in [Pei09].

The second problem is that our inversion may fail with negligible probability for every choice of $(\mathsf{ik}, \mathsf{tk})$ and a bit care is needed here. In particular, the simulation strategy of [RS09] will fail if the adversary can create an image Y, which is a true image of a domain point X, but which the inversion algorithm fails to invert. To overcome this problem, we slightly extend the notion of security required by OWFE, calling it *adaptive OWFE*, and show that if our TDF is instantiated using this primitive, it satisfies all the properties needed to build CCA2 secure PKE.

Comparison with related CCA2 constructions. We note that CCA2-secure PKE constructions from CDH are already known, e.g., [CKS08, Wee10, HJKS10], which are more efficient than the one obtained by instantiating our construction using CDH. We presented a CCA-2 secure construction just to show the black-box utility of our base general primitive. The recent results of [DG17b, DG17a, BLSV18, DGHM18], combined with [CHK04], show that CCA-secure PKE can be built from a related primitive called chameleon/batch encryption, but in a non-black-box way.

1.5 Discussion

Black-box power of chameleon encryption and related primitives. Our work is a contribution toward understanding the black-box power of the notion of chameleon encryption. Recent works [DG17b, DG17a, BLSV18] show that chameleon encryption (and its variants) may be used in a non-black-box way to build strong primitives such as identity-based encryption (IBE). The work of Brakerski et al. [BLSV18] shows also black-box applications of (a variant of) this notion, obtaining in turn circularly-secure and leakage-resilient PKE from CDH. Our work furthers the progress in this area, by giving a black-box construction of TDFs.

Related work. Hajiabadi and Kapron [HK15] show how to build TDFs from any *reproducible* circularly secure single-bit PKE. Informally, a PKE is reproducible if given a public key pk', a public/secret key $(\mathsf{pk}, \mathsf{sk})$ and a ciphertext $\mathsf{c} := \mathsf{PKE.E}(\mathsf{pk}', \mathsf{b}'; r)$, one can recycle the randomness of c to obtain $\mathsf{PKE.E}(\mathsf{pk}, \mathsf{b}; r)$ for any bit $\mathsf{b} \in \{0,1\}$. Brakerski et al. [BLSV18] recently built a circularly secure single-bit PKE using CDH. Their construction is not reproducible, however. (The following assumes familiarity with [BLSV18].) In their PKE, a secret key x of their PKE is an input to their hash function and the public key y is its corresponding image. To encrypt a bit b they (a) additively secret-share b into $(\mathsf{b}_1, \ldots, \mathsf{b}_n)$, where $n = |\mathsf{x}|$ and (b) form $2n$ ciphertext $\mathsf{ct}_{i,b}$, where $\mathsf{ct}_{i,b}$ encrypts b_i using y relative to (i, b). Their scheme is not reproducible because the randomness used for step (a) cannot be recycled and also half of the randomness used to create hash encryption ciphertexts in step (b) cannot be recycled. (This half corresponds to the bits of the target secret key w.r.t. which we want to recycle randomness.) It is not clear whether their scheme can be modified to yield reproducibility.

Open problems. Our work leads to several open problems. Can our TDF be improved to yield perfect correctness? Our current techniques leave us with a negligible inversion error. Can we build lossy trapdoor functions (LTDF) [PW08] from recyclable-OWFE/CDH? Given the utility of LTDFs, a construction based on CDH will be interesting. Can we build deterministic encryption based on CDH matching the parameters of those based on DDH [BFO08]?

2 Preliminaries

Notation. We use λ for the security parameter. We use $\stackrel{c}{\equiv}$ to denote computational indistinguishability between two distributions and use $\equiv$ to denote two distributions are identical. For a distribution D we use $x \stackrel{\$}{\leftarrow} D$ to mean x is sampled according to D and use $y \in D$ to mean y is in the support of D. For a set S we overload the notation to use $x \stackrel{\$}{\leftarrow} \mathsf{S}$ to indicate that x is chosen uniformly at random from S.

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

- $\mathsf{TDF.K}(1^\lambda)$*: 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 $\mathsf{X} \in \{0,1\}^w$ and outputs an image element Y.*
- $\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{Y})$*: Takes a trapdoor key tk and an image element Y and outputs a value $\mathsf{X} \in \{0,1\}^w \cup \{\perp\}$.*

We require the following properties.

- ***Correctness:*** *For any $(\mathsf{ik}, \mathsf{tk}) \in \mathsf{TDF.K}(1^\lambda)$*

$$\Pr[\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X})) \neq \mathsf{X}] = \mathsf{negl}(\lambda), \tag{5}$$

where the probability is taken over $\mathsf{X} \stackrel{\$}{\leftarrow} \{0,1\}^w$.

- ***One-wayness:*** *For any PPT adversary* $\mathcal{A}$*, we have* $\Pr[\mathcal{A}(\mathsf{ik}, \mathsf{Y}) = \mathsf{X}] = \mathsf{negl}(\lambda)$*, where* $(\mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$*,* $\mathsf{X} \xleftarrow{\$} \{0,1\}^w$ *and* $\mathsf{Y} = \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X})$*.*

A note about the correctness condition. Our correctness notion relaxes that of perfect correctness by allowing the inversion algorithm to fail (with respect to any trapdoor key) for a negligible fraction of evaluated elements. This relaxation nonetheless suffices for all existing applications of perfectly-correct TDFs. Our correctness notion, however, implies a weaker notion under which the correctness probability is also taken over the choice of the index/trapdoor keys. This makes our result for constructing TDFs stronger.

Definition 2.2 (Computational Diffie-Hellman (CDH) assumption)**.** *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 of the group. We say that* 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* $a_1, a_2 \xleftarrow{\$} \mathbb{Z}_p$*.*

3 Recyclable One-Way Function with Encryption

We will start by defining the notion of a one-way function with encryption. This notion is similar to the chameleon encryption notion of Döttling and Garg [DG17b]. However, it is weaker in the sense that it does not imply collision-resistant hash functions.

Next, we will define a novel *ciphertext-randomness recyclability* property for one-way function with encryption schemes. We will show that a variant of the chameleon encryption construction of Döttling and Garg [DG17b] satisfies this ciphertext-randomness recyclability property.

3.1 Recyclable One-Way Function with Encryption

We provide the definition of a one-way function with encryption. We define the notion as a key-encapsulation mechanism with single bit keys.

Definition 3.1 (One-way function with encryption (OWFE))**.** *An OWFE scheme consists of four PPT algorithms* K*,* f*,* E *and* D *with the following syntax.*

- $\mathsf{K}(1^\lambda)$*: Takes the security parameter* 1^λ *and outputs a public parameter* pp *for a function* f *from* n *bits to* ν *bits.*
- $\mathsf{f}(\mathsf{pp}, \mathsf{x})$*: Takes a public parameter* pp *and a preimage* $\mathsf{x} \in \{0,1\}^n$*, and outputs* $\mathsf{y} \in \{0,1\}^\nu$*.*
- $\mathsf{E}(\mathsf{pp}, \mathsf{y}, (i, b); \rho)$*: Takes a public parameter* pp*, a value* y*, an index* $i \in [n]$*, a bit* $b \in \{0,1\}$ *and randomness* ρ*, and outputs a ciphertext* ct *and a bit* e*.*[3]
- $\mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct})$*: Takes a public parameter* pp*, a value* x *and a ciphertext* ct*, and deterministically outputs* $\mathsf{e}' \in \{0,1\} \cup \{\perp\}$*.*

We require the following properties.

[3] ct is assumed to contain (i, b).

- ***Correctness:*** *For any* $\mathsf{pp} \in \mathsf{K}(1^\lambda)$*, any* $i \in [n]$*, any* $\mathsf{x} \in \{0,1\}^n$ *and any randomness value* ρ*, the following holds: letting* $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$*,* $b := \mathsf{x}_i$ *and* $(\mathsf{ct}, \mathsf{e}) := \mathsf{E}(\mathsf{pp}, \mathsf{y}, (i, b); \rho)$*, we have* $\mathsf{e} = \mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct})$*.*

- ***One-wayness:*** *For any PPT adversary* $\mathcal{A}$*:*

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

  where $\mathsf{pp} \xleftarrow{\$} \mathsf{K}(1^\lambda)$*,* $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ *and* $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$*.*

- ***Security for encryption:*** *For any* $i \in [n]$ *and* $\mathsf{x} \in \{0,1\}^n$*:*

$$(\mathsf{x}, \mathsf{pp}, \mathsf{ct}, \mathsf{e}) \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{pp}, \mathsf{ct}, \mathsf{e}')$$

  where $\mathsf{pp} \xleftarrow{\$} \mathsf{K}(1^\lambda)$*,* $(\mathsf{ct}, \mathsf{e}) \xleftarrow{\$} \mathsf{E}(\mathsf{pp}, \mathsf{f}(\mathsf{pp}, \mathsf{x}), (i, 1 - \mathsf{x}_i))$ *and* $\mathsf{e}' \xleftarrow{\$} \{0,1\}$*.*

Definition 3.2 (Recyclability)**.** *We say that an OWFE scheme* $(\mathsf{f}, \mathsf{K}, \mathsf{E}, \mathsf{D})$ *is recyclable if the following holds. Letting* E_1 *and* E_2 *refer to the first and second output of* E*, the value of* $\mathsf{E}_1(\mathsf{pp}, \mathsf{y}, (i, b); \rho)$ *is always independent of* y*. That is, for any* $\mathsf{pp} \in \mathsf{K}(1^\lambda)$*,* $\mathsf{y}_1, \mathsf{y}_2 \in \{0,1\}^\nu$*,* $i \in [n]$*,* $b \in \{0,1\}$ *and randomness* ρ*:* $\mathsf{E}_1(\mathsf{pp}, \mathsf{y}_1, (i, b); \rho) = \mathsf{E}_1(\mathsf{pp}, \mathsf{y}_2, (i, b); \rho)$*.*

We now conclude the above definitions with two remarks.

Remark 3.3 (Simplified recyclability)**.** *Since under the recyclability notion the ciphertext output* ct *of* E *is independent of the input value* y*, when referring to* E_1*, we may omit the inclusion of* y *as an input and write* $\mathsf{ct} = \mathsf{E}_1(\mathsf{pp}, (i, b); \rho)$*.*

Remark 3.4. *If the function* $\mathsf{f}(\mathsf{pp}, \cdot)$ *is length decreasing (e.g.,* $\mathsf{f}(\mathsf{pp}, \cdot)\colon \{0,1\}^n \mapsto \{0,1\}^{n-1}$*), then the one-wayness condition of Definition 3.1 is implied by the combination of the security-for-encryption and correctness conditions. In our definition, however, we do not place any restriction on the structure of the function* f*, and it could be, say, a one-to-one function. As such, under our general definition, the one-wayness condition is not necessarily implied by those two other conditions.*

3.2 Adaptive One-Way Function with Encryption

For our CCA application we need to work with an adaptive version of the notion of OWFE. Recall by Note 3.3 that a ciphertext ct does not depend on the corresponding y. The security for encryption notion (Definition 3.1) says if $(\mathsf{ct}, \mathsf{e})$ is formed using an image $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$ and parameters (i, b), and if $\mathsf{x}_i \neq b$, then even knowing x does not help an adversary in distinguishing e from a random bit. The adaptive version of this notion allows the adversary to choose x after seeing ct. This notion makes sense because ct does not depend on the image y, and so ct may be chosen first.

Definition 3.5 (Adaptive OWFE)**.** *We say that* $\mathcal{E} = (\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$ *is an adaptive one-way function with encryption scheme if* $\mathcal{E}$ *is correct in the sense of Definition 3.1,* f *is one-way in the sense of Definition 3.1 and that* $\mathcal{E}$ *is adaptively secure in the following sense.*

- ***Adaptive Security:*** *For any PPT adversary* $\mathcal{A}$ *the probability that* $\mathsf{AdapOWFE}[t = 1](\mathcal{E}, \mathcal{A})$ *outputs* 1 *is* $\frac{1}{2} + \mathsf{negl}(\lambda)$*, where the experiment* $\mathsf{AdapOWFE}[t]$ *is defined in Fig. 1.*

Experiment $\mathsf{AdapOWFE}[t](\mathcal{E}, \mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3))$:

1. $(i^* \in [n], b^* \in \{0,1\}, \mathsf{st}_1) \xleftarrow{\$} \mathcal{A}_1(1^\lambda)$
2. Sample $\mathsf{pp} \xleftarrow{\$} \mathsf{K}(1^\lambda)$ and $\rho_1, \ldots, \rho_t \xleftarrow{\$} \{0,1\}^*$
3. Set $\mathsf{ct}_1 := \mathsf{E}_1(\mathsf{pp}, (i^*, b^*); \rho_1), \ \ldots, \mathsf{ct}_t := \mathsf{E}_1(\mathsf{pp}, (i^*, b^*); \rho_t)$
4. $(\mathsf{x}, \mathsf{st}_2) \xleftarrow{\$} \mathcal{A}_2(\mathsf{st}_1, \mathsf{pp}, \mathsf{ct}_1, \ldots, \mathsf{ct}_t)$.
5. If $\mathsf{x}_{i^*} = b^*$, HALT.
6. For $j \in [t]$: $\mathsf{e}_j := \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i^*, b^*); \rho_j)$, where $\mathsf{y} = \mathsf{f}(\mathsf{pp}, \mathsf{x})$.
7. $\mathsf{ch} \xleftarrow{\$} \{0,1\}$. If $\mathsf{ch} = 0$, set $\mathbf{e}' := (\mathsf{e}_1, \ldots, \mathsf{e}_t)$; else, $\mathbf{e}' \xleftarrow{\$} \{0,1\}^t$.
8. $\mathsf{out} \xleftarrow{\$} \mathcal{A}_3(\mathsf{st}_2, \mathbf{e}')$.
9. Output 1 if $\mathsf{out} = \mathsf{ch}$ and 0 otherwise.

Figure 1: The $\mathsf{AdapOWFE}[t](\mathcal{E}, \mathcal{A})$ Experiment

We remind the reader that in Step 3 of Fig. 1 the algorithm E_1 does not take any y as input because of Note 3.3. The following lemma is obtained using a straightforward hybrid argument, so we omit the proof.

Lemma 3.6. *Let $\mathcal{E} = (\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$ be an adaptive OWFE scheme. For any polynomial $t := t(\lambda)$ and any PPT adversary $\mathcal{A}$, we have* $\Pr[\mathsf{AdapOWFE}[t](\mathcal{E}, \mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

3.3 Construction from CDH

We give a CDH-based construction of a recyclable adaptive OWFE based on a group scheme G (Definition 2.2), which is a close variant of constructions given in [CDG+17, DG17b].

- $\mathsf{K}(1^\lambda)$: Sample $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{G}(1^\lambda)$. For each $j \in [n]$ and $b \in \{0,1\}$, choose $g_{j,b} \xleftarrow{\$} \mathbb{G}$. Output

$$\mathsf{pp} := \mathbb{G}, p, g, \begin{pmatrix} g_{1,0}, g_{2,0}, \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{pmatrix}. \tag{6}$$

- $\mathsf{f}(\mathsf{pp}, \mathsf{x})$: Parse pp as in Equation 6, and output $\mathsf{y} := \prod_{j \in [n]} g_{j,\mathsf{x}_j}$.

- $\mathsf{E}(\mathsf{pp}, \mathsf{y}, (i, b))$: Parse pp as in Equation 6. Sample $\rho \xleftarrow{\$} \mathbb{Z}_p$ and proceed as follows:
  1. For every $j \in [n] \backslash \{i\}$, set $c_{j,0} := g_{j,0}^\rho$ and $c_{j,1} := g_{j,1}^\rho$.
  2. Set $c_{i,b} := g_{i,b}^\rho$ and $c_{i,1-b} := \perp$.

3. Set $\mathsf{e} := \mathsf{HC}(\mathsf{y}^{\rho})$.[4]

4. Output $(\mathsf{ct}, \mathsf{e})$ where $\mathsf{ct} := \begin{pmatrix} c_{1,0}, c_{2,0}, \ldots, c_{n,0} \\ c_{1,1}, c_{2,1}, \ldots, c_{n,1} \end{pmatrix}$.

- $\mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct})$: Parse $\mathsf{ct} := \begin{pmatrix} c_{1,0}, c_{2,0}, \ldots, c_{n,0} \\ c_{1,1}, c_{2,1}, \ldots, c_{n,1} \end{pmatrix}$. Output $\mathsf{HC}(\prod_{j \in [n]} c_{j,\mathsf{x}_j})$.

We prove the following lemma in Appendix A.

Lemma 3.7. *Assuming that* G *is CDH-hard and* $n \in \omega(\log p)$*, the construction described above is an adaptive one-way function with encryption scheme satisfying the recyclability property.*

4 TDF Construction

In this section we describe our TDF construction. We first give the following notation.

Extending the notation for D**.** For a given pp, a sequence $\mathbf{ct} := (\mathsf{ct}_1, \ldots, \mathsf{ct}_r)$ of encapsulated ciphertexts and a value x, we define $\mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathbf{ct})$ to be the concatenation of $\mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct}_i)$ for $i \in [r]$.

Algorithm Perm**.** For two lists $\mathbf{u_1}$ and $\mathbf{u_2}$ and a bit b we define $\mathsf{Perm}(\mathbf{u_1}, \mathbf{u_2}, b)$ to output $(\mathbf{u_1}, \mathbf{u_2})$ if $b = 0$, and $(\mathbf{u_2}, \mathbf{u_1})$ otherwise.

Construction 4.1 (TDF construction)**.** *We now describe our TDF construction.*

Base primitive. *A recyclable OWFE scheme* $\mathcal{E} = (\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$*. Let* Rand *be the randomness space of the encapsulation algorithm* E*.*

Construction. *The construction is parameterized over two parameters* $n = n(\lambda)$ *and* $r = r(\lambda)$*, where* n *is the input length to the function* f*, and* r *will be instantiated in the correctness proof. The input space of each TDF is* $\{0,1\}^{n+nr}$*. We will make use of the fact explained in Note 3.3.*

- $\mathsf{TDF.K}(1^{\lambda})$*:*

  1. *Sample* $\mathsf{pp} \leftarrow \mathsf{K}(1^{\lambda})$.
  2. *For each* $i \in [n]$ *and selector bit* $b \in \{0,1\}$*:*

$$\boldsymbol{\rho}_{i,b} := (\rho_{i,b}^{(1)}, \ldots, \rho_{i,b}^{(r)}) \xleftarrow{\$} \mathsf{Rand}^r$$
$$\mathbf{ct}_{i,b} := (\mathsf{E}_1(\mathsf{pp}, (i,b); \rho_{i,b}^{(1)}), \ldots, \mathsf{E}_1(\mathsf{pp}, (i,b); \rho_{i,b}^{(r)})).$$

  3. *Form the index key* ik *and the trapdoor key* tk *as follows:*

$$\mathsf{ik} := (\mathsf{pp}, \mathbf{ct}_{1,0}, \mathbf{ct}_{1,1}, \ldots, \mathbf{ct}_{n,0}, \mathbf{ct}_{n,1}) \tag{7}$$
$$\mathsf{tk} := \left(\mathsf{pp}, \boldsymbol{\rho}_{1,0}, \boldsymbol{\rho}_{1,1}, \ldots, \boldsymbol{\rho}_{n,0}, \boldsymbol{\rho}_{n,1}\right). \tag{8}$$

[4] We assume that the $\mathsf{HC}(\cdot)$ is a hardcore bit function. If a deterministic hard-core bit for the specific function is not known then we can use the Goldreich-Levin [GL89] construction. We skip the details of that with the goal of keeping exposition simple.

- $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{X})$*:*

  1. *Parse* ik *as in Equation 7 and parse*
  $$\mathsf{X} := (\mathsf{x} \in \{0,1\}^n, \mathbf{b}_1 \in \{0,1\}^r, \ldots, \mathbf{b}_n \in \{0,1\}^r).$$
  2. *Set* $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$.
  3. *For all* $\mathsf{i} \in [n]$ *set*
  $$\mathbf{e}_i := \mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathbf{ct}_{\mathsf{i},\mathsf{x}_i}).$$
  4. *Return*
  $$\mathsf{Y} := \big(\mathsf{y}, \mathsf{Perm}(\mathbf{e}_1, \mathbf{b}_1, \mathsf{x}_1), \ldots, \mathsf{Perm}(\mathbf{e}_n, \mathbf{b}_n, \mathsf{x}_n)\big).$$

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

  1. *Parse* tk *as in Equation 8 and* $\mathsf{Y} := (\mathsf{y}, \widetilde{\mathbf{b}_{1,0}}, \widetilde{\mathbf{b}_{1,1}}, \ldots, \widetilde{\mathbf{b}_{n,0}}, \widetilde{\mathbf{b}_{n,1}})$.
  2. *Reconstruct* $\mathsf{x} := \mathsf{x}_1 \cdots \mathsf{x}_n$ *bit-by-bit and* $\mathbf{b} := (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ *vector-by-vector as follows. For* $i \in [n]$*:*
     (a) *Parse* $\boldsymbol{\rho}_{i,0} := (\rho_{i,0}^{(1)}, \ldots, \rho_{i,0}^{(r)})$ *and* $\boldsymbol{\rho}_{i,1} := (\rho_{i,1}^{(1)}, \ldots, \rho_{i,1}^{(r)})$.
     (b) *If*
     $$\widetilde{\mathbf{b}_{i,0}} = \Big(\mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,0); \rho_{\mathsf{i},0}^{(1)}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,0); \rho_{\mathsf{i},0}^{(\mathsf{r})})\Big) \text{ and}$$
     $$\widetilde{\mathbf{b}_{i,1}} \neq \Big(\mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,1); \rho_{\mathsf{i},1}^{(1)}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,1); \rho_{\mathsf{i},1}^{(\mathsf{r})})\Big), \quad (9)$$
     then set $\mathsf{x}_i = 0$ *and* $\mathbf{b}_i = \widetilde{\mathbf{b}_{i,1}}$.
     (c) *Else, if*
     $$\widetilde{\mathbf{b}_{i,0}} \neq \Big(\mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,0); \rho_{\mathsf{i},0}^{(1)}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,0); \rho_{\mathsf{i},0}^{(\mathsf{r})})\Big) \text{ and}$$
     $$\widetilde{\mathbf{b}_{i,1}} = \Big(\mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,1); \rho_{\mathsf{i},1}^{(1)}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,1); \rho_{\mathsf{i},1}^{(\mathsf{r})})\Big), \quad (10)$$
     then set $\mathsf{x}_i = 1$ *and* $\mathbf{b}_i = \widetilde{\mathbf{b}_{i,0}}$.
     (d) *Else, halt and return* $\perp$.
  3. *If* $\mathsf{y} \neq \mathsf{f}(\mathsf{pp}, \mathsf{x})$*, then return* $\perp$*. Otherwise, return* $(\mathsf{x}, \mathbf{b})$.

We will now give the correctness and one-wayness statements about our TDF, and will prove them in subsequent subsections.

Lemma 4.2 (TDF correctness)**.** *The inversion error of our constructed* TDF *is at most* $\frac{n}{2^r}$*. That is, for any* $(\mathsf{ik}, \mathsf{tk}) \in \mathsf{TDF.K}(1^\lambda)$ *we have*

$$\beta := \Pr[\mathsf{TDF.F}^{-1}(\mathsf{tk}, (\mathsf{TDF.F}(\mathsf{ik}, \mathsf{X}))) \neq \mathsf{X}] \leq \frac{n}{2^r}, \quad (11)$$

where the probability is taken over $\mathsf{X} := (\mathsf{x}, \mathbf{b}_1, \ldots, \mathbf{b}_n) \xleftarrow{\$} \{0,1\}^{n+nr}$*. By choosing* $r \in \omega(\log \lambda)$ *we will have a negligible inversion error.*

For one-wayness we will prove something stronger: parsing $\mathsf{X} := (\mathsf{x}, \dots)$, then recovering any x' satisfying $\mathsf{f}(\mathsf{pp}, \mathsf{x}) = \mathsf{f}(\mathsf{pp}, \mathsf{x}')$ from $(\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X}))$ is infeasible.

Lemma 4.3 (TDF one-wayness)**.** *The TDF* $(\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ *given in Construction 4.1 is one-way. That is, for any PPT adversary* $\mathcal{A}$

$$\Pr[\mathcal{A}(\mathsf{ik}, \mathsf{Y}) = \mathsf{x}' \text{ and } \mathsf{f}(\mathsf{pp}, \mathsf{x}') = \mathsf{y}] = \mathsf{negl}(\lambda), \tag{12}$$

where $(\mathsf{ik} := (\mathsf{pp}, \dots), \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$, $\mathsf{X} := (\mathsf{x}, \dots) \xleftarrow{\$} \{0,1\}^{n+nr}$ *and* $\mathsf{Y} := (\mathsf{y}, \dots) := \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X})$.

By combining Lemmas 3.7, 4.2 and 4.3 we will obtain our main result below.

Theorem 4.4 (CDH implies TDFs)**.** *There is a black-box construction of TDFs from CDH-hard groups.*

4.1 Proof of Correctness: Lemma 4.2

Proof of Lemma 4.2. Let $\mathsf{X} := (\mathsf{x}, \mathbf{b}_1, \dots, \mathbf{b}_n) \xleftarrow{\$} \{0,1\}^{n+nr}$ be as in the lemma and

$$\mathsf{Y} := \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X}) := (\mathsf{y}, \widetilde{\mathbf{b}_{1,0}}, \widetilde{\mathbf{b}_{1,1}}, \dots, \widetilde{\mathbf{b}_{n,0}}, \widetilde{\mathbf{b}_{n,1}}). \tag{13}$$

By design, for all $i \in [n]$: $\widetilde{\mathbf{b}_{i,1-\mathsf{x}_i}} = \mathbf{b}_i$. Parse

$$\mathsf{tk} := \left(\boldsymbol{\rho}_{1,0}, \boldsymbol{\rho}_{1,1}, \dots, \boldsymbol{\rho}_{n,0}, \boldsymbol{\rho}_{n,1}\right),$$
$$\boldsymbol{\rho}_{i,b} := (\rho_{i,b}^{(1)}, \dots, \rho_{i,b}^{(r)}), \text{ for } i \in [n] \text{ and } b \in \{0,1\}.$$

Consider the execution of $\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{Y})$. By the correctness of our recyclable OWFE scheme $\mathcal{E}$ we have the following: the probability that $\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{Y}) \neq \mathsf{X}$ is the probability that for some $i \in [n]$:

$$\mathbf{b}_i = \left(\mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i, 1-\mathsf{x}_i); \rho_{i,1-\mathsf{x}_i}^{(1)}), \dots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i, 1-\mathsf{x}_i); \rho_{i,1-\mathsf{x}_i}^{(r)})\right). \tag{14}$$

Now since $\mathbf{b}_i$, for all i, is chosen uniformly at random and independently of x, the probability of the event in Equation 14 is $\frac{1}{2^r}$. A union bound over $i \in [n]$ gives us the claimed error bound. □

4.2 Proof of One-wayness: Lemma 4.3

We prove Lemma 4.3 via a couple of hybrids, corresponding to the real and a simulated view. We first give the following definition which will help us describe the two hybrids in a compact way.

Definition 4.5. *Fix* pp, $\mathsf{x} \in \{0,1\}^n$ *and* $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$. *We define two PPT algorithms* Real *and* Sim*, where* Real *takes as input* $(\mathsf{pp}, \mathsf{x})$ *and* Sim *takes as input* $(\mathsf{pp}, \mathsf{y})$. *We stress that* Sim *does not take* x *as input.*

The algorithm $\mathsf{Real}(\mathsf{pp}, \mathsf{x})$ *outputs* $(\mathbf{CT}, \mathbf{E})$ *and the algorithm* $\mathsf{Sim}(\mathsf{pp}, \mathsf{y})$ *outputs* $(\mathbf{CT}, \mathbf{E}_{\mathsf{sim}})$*, sampled in the following way.*

- *Sample* $\begin{pmatrix} \rho_{1,0}, \dots, \rho_{\mathsf{n},0} \\ \rho_{1,1}, \dots, \rho_{\mathsf{n},1} \end{pmatrix} \xleftarrow{\$} \mathsf{Rand}^{2\times n}$.

- *Set*
$$\mathbf{CT} := \begin{pmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{pmatrix} := \begin{pmatrix} \mathsf{E}_1(\mathsf{pp}, (1,0); \rho_{1,0}), \ldots, \mathsf{E}_1(\mathsf{pp}, (n,0); \rho_{n,0}) \\ \mathsf{E}_1(\mathsf{pp}, (1,1); \rho_{1,1}), \ldots, \mathsf{E}_1(\mathsf{pp}, (n,1); \rho_{n,1}) \end{pmatrix}.$$
- *Set*
$$\mathbf{E} := \begin{pmatrix} \mathsf{b}_{1,0}, \ldots, \mathsf{b}_{n,0} \\ \mathsf{b}_{1,1}, \ldots, \mathsf{b}_{n,1} \end{pmatrix},$$
where, for all $i \in [n]$*:*
  - *if* $\mathsf{x}_i = 0$*, then* $\mathsf{b}_{i,0} := \mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct}_{i,0})$ *and* $\mathsf{b}_{i,1} \xleftarrow{\$} \{0,1\}$.
  - *if* $\mathsf{x}_i = 1$*, then* $\mathsf{b}_{i,0} \xleftarrow{\$} \{0,1\}$ *and* $\mathsf{b}_{i,1} := \mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct}_{i,1})$.
- *Set*
$$\mathbf{E}_{\text{sim}} := \begin{pmatrix} \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (1,0); \rho_{1,0}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (n,0); \rho_{n,0}) \\ \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (1,1); \rho_{1,1}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (n,1); \rho_{n,1}) \end{pmatrix}$$

We now prove the following lemma which will help us to prove the indistinguishability of the two hybrids in our main proof.

Lemma 4.6. *Fix polynomial* $r := r(\lambda)$ *and let* $\mathsf{x} \in \{0,1\}^n$*. We have*
$$(\mathsf{pp}, \mathsf{x}, \mathbf{CT}_1, \mathbf{E}_1, \ldots, \mathbf{CT}_r, \mathbf{E}_r) \stackrel{c}{\equiv} (\mathsf{pp}, \mathsf{x}, \mathbf{CT}_1, \mathbf{E}_{\text{sim},1}, \ldots, \mathbf{CT}_r, \mathbf{E}_{\text{sim},r}), \tag{15}$$
where $\mathsf{pp} \xleftarrow{\$} \mathsf{K}(1^\lambda)$*, and for all* $i \in [r]$*,* $(\mathbf{CT}_i, \mathbf{E}_i) \xleftarrow{\$} \mathsf{Real}(\mathsf{pp}, \mathsf{x})$ *and* $(\mathbf{CT}_i, \mathbf{E}_{\text{sim},i}) \xleftarrow{\$} \mathsf{Sim}(\mathsf{pp}, \mathsf{f}(\mathsf{pp}, \mathsf{x}))$.

Proof. Fix $\mathsf{x} \in \{0,1\}^n$ and let $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$. For the purpose of doing a hybrid argument we define two algorithms SReal and SSim below.

- $\mathsf{SReal}(i, \mathsf{pp}, \mathsf{x})$: sample $\rho_0, \rho_1 \xleftarrow{\$} \mathsf{Rand}$ and return $(\mathbf{ct}, \mathbf{e})$, where
$$\mathbf{ct} := \begin{pmatrix} \mathsf{ct}_0 \\ \mathsf{ct}_1 \end{pmatrix} := \begin{pmatrix} \mathsf{E}_1(\mathsf{pp}, (i,0); \rho_0) \\ \mathsf{E}_1(\mathsf{pp}, (i,1); \rho_1) \end{pmatrix} \tag{16}$$
and $\mathbf{e}$ is defined as follows:
  - if $\mathsf{x}_i = 0$, then $\mathbf{e} := \begin{pmatrix} \mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct}_0) \\ b \end{pmatrix}$, where $b \xleftarrow{\$} \{0,1\}$;
  - if $\mathsf{x}_i = 1$, then $\mathbf{e} := \begin{pmatrix} b \\ \mathsf{D}(\mathsf{pp}, \mathsf{x}, \mathsf{ct}_1) \end{pmatrix}$, where $b \xleftarrow{\$} \{0,1\}$.
- $\mathsf{SSim}(i, \mathsf{pp}, \mathsf{y})$: Return $(\mathbf{ct}, \mathbf{e}_{\text{sim}})$, where $\mathbf{ct}$ is sampled as in Equation 16 and
$$\mathbf{e}_{\text{sim}} := \begin{pmatrix} \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,0); \rho_0) \\ \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,1); \rho_1) \end{pmatrix}.$$

We will show that for all $i \in [n]$ and $\mathsf{x} \in \{0,1\}^n$

$$(\mathsf{pp}, \mathsf{x}, \mathbf{ct}, \mathbf{e}) \stackrel{c}{\equiv} (\mathsf{pp}, \mathsf{x}, \mathbf{ct}, \mathbf{e}_{\mathrm{sim}}), \tag{17}$$

where

$$\mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{K}(1^\lambda), \ (\mathbf{ct}, \mathbf{e}) \stackrel{\$}{\leftarrow} \mathsf{SReal}(i, \mathsf{pp}, \mathsf{x}) \text{ and } (\mathbf{ct}, \mathbf{e}_{\mathrm{sim}}) \stackrel{\$}{\leftarrow} \mathsf{SSim}(i, \mathsf{pp}, \mathsf{f}(\mathsf{pp}, \mathsf{x})).$$

From Equation 17 using a simple hybrid argument the indistinguishability claimed in the lemma (Equation 15) is obtained. Note that for the hybrid argument we need to make use of the fact that that x is provided in both sides of Equation 17, because we need to know x to be able to build the intermediate hybrids between those of Equation 15. Thus, in what follows we will focus on proving Equation 17.

To prove Equation 17, first note that by the correctness of the OWFE scheme $\mathcal{E}$, we have

$$(\mathsf{pp}, \mathsf{x}, \mathbf{ct}, \mathbf{e}) \equiv (\mathsf{pp}, \mathsf{x}, \mathbf{ct}, \mathbf{e}'),$$

where $\mathbf{ct}$ and $\mathbf{e}$ are sampled according to $\mathsf{SReal}(i, \mathsf{pp}, \mathsf{x})$ as above (using randomness values ρ_0 and ρ_1), and $\mathbf{e}'$ is sampled as:

- if $\mathsf{x}_i = 0$, then $\mathbf{e}' := \begin{pmatrix} \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,0); \rho_0) \\ b \end{pmatrix}$, where $b \stackrel{\$}{\leftarrow} \{0,1\}$;
- if $\mathsf{x}_i = 1$, then $\mathbf{e}' := \begin{pmatrix} b \\ \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,1); \rho_1) \end{pmatrix}$, where $b \stackrel{\$}{\leftarrow} \{0,1\}$.

Thus, we will prove

$$(\mathsf{pp}, \mathsf{x}, \mathbf{ct}, \mathbf{e}') \stackrel{c}{\equiv} (\mathsf{pp}, \mathsf{x}, \mathbf{ct}, \mathbf{e}_{\mathrm{sim}}). \tag{18}$$

We derive Equation 18 from the security-for-encryption requirement of the scheme $(\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$.

Recall that the security for encryption requirement asserts that no PPT adversary can distinguish between $(\mathsf{x}, \mathsf{ct}_1, \mathsf{e}_1)$ and $(\mathsf{x}, \mathsf{ct}_1, \mathsf{e}_2)$, where $\mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{K}(1^\lambda)$, $(\mathsf{ct}_1, \mathsf{e}_1) \stackrel{\$}{\leftarrow} \mathsf{E}(\mathsf{pp}, \mathsf{f}(\mathsf{pp}, \mathsf{x}), (i, 1-\mathsf{x}_i))$ and $\mathsf{e}_2 \stackrel{\$}{\leftarrow} \{0,1\}$. Let us call $(\mathsf{x}, \mathsf{ct}_1, \mathsf{e}_1)$ the simulated challenge and $(\mathsf{x}, \mathsf{ct}_1, \mathsf{e}_2)$ the random challenge.

To give the reduction we present a procedure Turn which generically turns a simulated challenge into a sample of $\mathsf{SSim}(i, \mathsf{pp}, \mathsf{x})$ and turns a random challenge into a sample of $\mathsf{SReal}(i, \mathsf{pp}, \mathsf{y})$.

The algorithm $\mathsf{Turn}(\mathsf{x}, \mathsf{ct}, \mathsf{e})$ returns $(\mathbf{ct}_1, \mathbf{e}_1)$, formed as follows:

- Sample $\rho \stackrel{\$}{\leftarrow} \mathsf{Rand}$. Then
  - if $\mathsf{x}_i = 0$, then return
$$\mathbf{ct}_1 = \begin{pmatrix} \mathsf{E}_1(\mathsf{pp}, (i,0); \rho) \\ \mathsf{ct} \end{pmatrix} \qquad \mathbf{e}_1 = \begin{pmatrix} \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,0); \rho) \\ \mathsf{e} \end{pmatrix}$$
  - if $\mathsf{x}_i = 1$, then return
$$\mathbf{ct}_1 = \begin{pmatrix} \mathsf{ct} \\ \mathsf{E}_1(\mathsf{pp}, (i,0); \rho) \end{pmatrix} \qquad \mathbf{e}_1 = \begin{pmatrix} \mathsf{e} \\ \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i,0); \rho) \end{pmatrix}$$

It should be clear by inspection that the output of $\mathsf{Turn}(\mathsf{x}, \mathsf{ct}, \mathsf{e})$ is identically distributed to $\mathsf{SReal}(i, \mathsf{pp}, \mathsf{x})$ if $(\mathsf{x}, \mathsf{ct}, \mathsf{e})$ is a random challenge (defined above), and identically distributed to $\mathsf{SSim}(i, \mathsf{pp}, \mathsf{x})$ if $(\mathsf{x}, \mathsf{ct}, \mathsf{e})$ is a simulated challenge. The proof is now complete. □

Proof of Lemma 4.3. To prove Lemma 4.3 we define two hybrids and will use the notation view_i to refer to the view sampled in **Hybrid** i.

Hybrid 0. The view $(\mathsf{ik}, \mathsf{Y})$ is produced honestly as in the real executions of the scheme TDF. That is,

- Sample $\mathsf{pp} \xleftarrow{\$} \mathsf{K}(1^\lambda)$, $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ and let $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$.
- For all $j \in [r]$ sample $(\mathbf{CT}^{(j)}, \mathbf{E}^{(j)}) \xleftarrow{\$} \mathsf{Real}(\mathsf{pp}, \mathsf{x})$. Parse

$$\mathbf{CT}^{(j)} := \begin{pmatrix} \mathsf{ct}_{1,0}^{(j)}, \ldots, \mathsf{ct}_{n,0}^{(j)} \\ \mathsf{ct}_{1,1}^{(j)}, \ldots, \mathsf{ct}_{n,1}^{(j)} \end{pmatrix} \qquad \mathbf{E}^{(j)} := \begin{pmatrix} \mathsf{b}_{1,0}^{(j)}, \ldots, \mathsf{b}_{n,0}^{(j)} \\ \mathsf{b}_{1,1}^{(j)}, \ldots, \mathsf{b}_{n,1}^{(j)} \end{pmatrix}.$$

- For all $i \in [n]$ and $d \in \{0,1\}$ set

$$\mathbf{ct}_{i,d} := (\mathsf{ct}_{i,d}^{(1)}, \ldots, \mathsf{ct}_{i,d}^{(r)})$$
$$\mathbf{b}_{i,d} := (\mathsf{b}_{i,d}^{(1)}, \ldots, \mathsf{b}_{i,d}^{(r)}).$$

- Form the view $(\mathsf{ik}, \mathsf{Y})$ as follows:

$$(\underbrace{(\mathsf{pp}, \mathbf{ct}_{1,0}, \mathbf{ct}_{1,1}, \ldots, \mathbf{ct}_{n,0}, \mathbf{ct}_{n,1})}_{\mathsf{ik}}, \underbrace{(\mathsf{y}, \mathbf{b}_{1,0}, \mathbf{b}_{1,1}, \ldots, \mathbf{b}_{n,0}, \mathbf{b}_{n,1})}_{\mathsf{Y}}) \tag{19}$$

Hybrid 1. The view $(\mathsf{ik}, \mathsf{Y})$ is produced the same as **Hybrid** 0 except that for all $j \in [r]$ we sample $(\mathbf{CT}^{(j)}, \mathbf{E}^{(j)})$ now as $(\mathbf{CT}^{(j)}, \mathbf{E}^{(j)}) \xleftarrow{\$} \mathsf{Sim}(\mathsf{pp}, \mathsf{y})$.

We prove that the two views above are indistinguishable and then we will show that inverting the image under view_1 is computationally infeasible.

Indistinguishability of the views: By Lemma 4.6 we have $\mathsf{view}_0 \stackrel{c}{\equiv} \mathsf{view}_1$. The reason is that the view in either hybrid is produced entirely based on $(\mathbf{CT}^{(1)}, \mathbf{E}^{(1)}, \ldots, \mathbf{CT}^{(r)}, \mathbf{E}^{(r)})$ and that this tuple is sampled from the distribution $\mathsf{Real}(\mathsf{pp}, \mathsf{x})$ in one hybrid and from $\mathsf{Sim}(\mathsf{pp}, \mathsf{y})$ in the other.

One-wayness in Hybrid 1**:** We claim that for any PPT adversary $\mathcal{A}$

$$\Pr[\mathcal{A}(\mathsf{view}_1) = \mathsf{x}' \text{ and } \mathsf{f}(\mathsf{pp}, \mathsf{x}') = \mathsf{y}] = \mathsf{negl}(\lambda). \tag{20}$$

Recall that $\mathsf{view}_1 := (\mathsf{ik}, \mathsf{Y})$ is the view in **Hybrid** 1 and that the variables pp and y are part of $\mathsf{ik} := (\mathsf{pp}, \ldots)$ and $\mathsf{Y} := (\mathsf{y}, \ldots)$. The proof of Equation 20 follows from the one-wayness of f, taking into account the fact that view_1 in its entirety is produced solely based on pp and $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$ (and especially without knowing x). This is because all the underlying variables $(\mathbf{CT}^{(j)}, \mathbf{E}^{(j)})$ — for all j — are produced as $(\mathbf{CT}^{(j)}, \mathbf{E}^{(j)}) \xleftarrow{\$} \mathsf{Sim}(\mathsf{pp}, \mathsf{y})$, which can be formed without knowledge of x.

Completing the Proof of Lemma 4.3. Let $\mathsf{view}_0 := (\mathsf{ik}, \mathsf{Y})$ and parse $\mathsf{ik} := (\mathsf{pp}, \ldots)$ and $\mathsf{Y} := (\mathsf{y}, \ldots)$. For any PPT adversary $\mathcal{A}$ we need to show that the probability that $\mathcal{A}$ on input view_0 outputs $\mathsf{x}' \in \{0,1\}^n$ such that $\mathsf{f}(\mathsf{pp}, \mathsf{x}') = \mathsf{y}$ is negligible. We know that $\mathcal{B}$ fails to compute such a string x' with non-negligible probability if the view $((\mathsf{pp}, \ldots), (\mathsf{y}, \ldots))$ is sampled according to view_1. Since $\mathsf{view}_0 \stackrel{c}{\equiv} \mathsf{view}_1$, the claim follows. □

4.3 Extended One-Wayness

For our CCA2 application we need to prove a stronger property than the standard one-wayness for our constructed TDF. This extension requires that if we evaluate m correlated inputs under m independent functions from the TDF family, the result still cannot be inverted.

Lemma 4.7 (m-Extended one-wayness)**.** *Let* $\mathsf{TDF} = (\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ *be the TDF built in Construction 4.1 based on an arbitrary parameter* $r = r(\lambda)$*. Let* $m := m(\lambda)$*. For any PPT adversary* $\mathcal{A}$

$$\Pr[\mathcal{A}(\mathsf{view} := (\mathsf{ik}_1, \ldots, \mathsf{ik}_m, \mathsf{Y}_1, \ldots, \mathsf{Y}_m)) = \mathsf{x}] = \mathsf{negl}(\lambda),$$

where $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ *and for* $i \in [m]$, $(\mathsf{ik}_i, \mathsf{tk}_i) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$, $\mathbf{b}_i \xleftarrow{\$} \{0,1\}^{nr}$ *and* $\mathsf{Y}_i := \mathsf{TDF.F}(\mathsf{ik}_i, \mathsf{x}||\mathbf{b}_i)$*. Thus, there exists a hardcore function* HC *such that* $\mathsf{HC}(\mathsf{x})$ *is pseudorandom in the presence of* view*.*

Proof. For any PPT adversary $\mathcal{A}$ we need to show that the probability that $\mathcal{A}(\mathsf{view})$ outputs x is negligible. It is easy to verify by inspection that the distribution of view can be perfectly formed based on the view $(\mathsf{ik}^*, \mathsf{Y}^*)$ of an inverter against the standard one-wayness of the trapdoor function $(\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ of Construction 4.1 but under the new parameter $r' = m \times r$. Invoking Lemma 4.3 our claimed one-wayness extension follows. □

5 CCA2-Secure Public-Key Encryption

In this section we show how to use our constructed TDF to build a CCA2 secure PKE. For the proof of CCA2 security we need to assume that the OWFE scheme underlying the TDF is adaptively secure (Definition 3.5).

Notation. Let $\mathsf{TDF} := (\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ be as in Section 4. We will interpret the input X to the TDF as (x, s), where $\mathsf{x} \in \{0,1\}^n$ corresponds to f's pre-image part and $\mathsf{s} \in \{0,1\}^{n_1}$ corresponds to the *blinding* part. In particular, if r is the underlying parameter of the constructed TDF as in Construction 4.1, then $n_1 = n \times r$.

Ingredients of our CCA2-secure PKE. Apart from a TDF with the above syntax, our CCA2 secure construction also makes use of a one-time signature scheme $\mathsf{SIG} = (\mathsf{SIG.K}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$ with prefect correctness, which in turn can be obtained from any one-way function. See Appendix C for the description of this notion.

Our CCA2 primitive. We will build a CCA2 secure single-bit PKE, which by the result of [Ms09] can be boosted into many-bit CCA2 secure PKE. Since we deal with single-bit CCA2 PKE, we may assume without loss of generality that the CCA adversary issues all her CCA oracles after seeing the challenge ciphertext.

We will now describe our CCA2-secure PKE scheme.

Construction 5.1 (IND-CCA2-secure PKE)**.** *The construction is parameterized over a parameter* $m := m(\lambda)$*, which denotes the size of the verification key of the underlying signature scheme* SIG*. Let* HC *be a bit-valued hardcore function whose existence was proved in Lemma 4.7.*

- $\mathsf{PKE.K}(1^\lambda)$: *For* $i \in [m]$ *and* $b \in \{0,1\}$, *sample* $(\mathsf{ik}_i^b, \mathsf{tk}_i^b) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$. *Form* $(\mathsf{pk}, \mathsf{sk})$ *the public/secret key as follows:*

$$\mathsf{pk} := (\mathsf{ik}_1^0, \mathsf{ik}_1^1, \ldots, \mathsf{ik}_m^0, \mathsf{ik}_m^1), \ \mathsf{sk} := (\mathsf{tk}_1^0, \mathsf{tk}_1^1, \ldots, \mathsf{tk}_m^0, \mathsf{tk}_m^1). \tag{21}$$

- $\mathsf{PKE.E}(\mathsf{pk}, \mathsf{b})$: *Parse* pk *as in Equation 21. Sample* $(\mathsf{vk}, \mathsf{sgk}) \xleftarrow{\$} \mathsf{SIG.K}(1^\lambda)$, $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ *and set*

$$\mathsf{X}_1 := (\mathsf{x}, \mathsf{s}_1 \xleftarrow{\$} \{0,1\}^{n_1}), \quad \ldots, \quad \mathsf{X}_m := (\mathsf{x}, \mathsf{s}_m \xleftarrow{\$} \{0,1\}^{n_1}). \tag{22}$$

Let $\mathsf{b}' = \mathsf{b} \oplus \mathsf{HC}(\mathsf{x})$ *and for* $i \in [m]$ *let* $\mathsf{Y}_i = \mathsf{TDF.F}(\mathsf{ik}_i^{\mathsf{vk}_i}, \mathsf{X}_i)$. *Return*

$$\mathsf{c} := \left(\mathsf{vk}, \mathsf{Y}_1, \ldots, \mathsf{Y}_m, \mathsf{b}', \mathsf{SIG.Sign}(\mathsf{sgk}, \mathsf{Y}_1 || \ldots || \mathsf{Y}_m || \mathsf{b}')\right). \tag{23}$$

- $\mathsf{PKE.D}(\mathsf{sk}, \mathsf{c})$: *Parse* sk *as in Equation 21 and parse*

$$\mathsf{c} := (\mathsf{vk}, \mathsf{Y}_1, \ldots, \mathsf{Y}_m, \mathsf{b}', \sigma). \tag{24}$$

  1. *Set* $\mathsf{msg} := \mathsf{Y}_1 || \cdots \mathsf{Y}_m || \mathsf{b}'$. *If* $\mathsf{SIG.Ver}(\mathsf{vk}, \mathsf{msg}, \sigma) = \bot$, *then return* $\bot$.
  2. *Otherwise, for* $i \in [m]$ *set* $\mathsf{X}_i := \mathsf{TDF.F}^{-1}(\mathsf{tk}_i^{\mathsf{vk}_i}, \mathsf{Y}_i)$. *Check that for all* $i \in [n]$: $\mathsf{Y}_i = \mathsf{TDF.F}(\mathsf{ik}_i^{\mathsf{vk}_i}, \mathsf{X}_i)$. *If not, return* $\bot$.
  3. *If there exists* $\mathsf{x} \in \{0,1\}^n$ *and* $\mathsf{s}_1, \ldots, \mathsf{s}_m \in \{0,1\}^{n_1}$ *such that for all* $i \in [m]$, $\mathsf{X}_i = (\mathsf{x}, \mathsf{s}_i)$, *then return* $\mathsf{b}' \oplus \mathsf{HC}(\mathsf{x})$. *Otherwise, return* $\bot$.

Correctness. Assuming that the underlying signature scheme $\mathsf{SIG} = (\mathsf{SIG.K}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$ is correct and also that the underlying TDF $(\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ is correct in the sense of Definition 2.1, the above constructed PKE is correct in a similar sense: for any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{PKE.K}(1^\lambda)$ and plaintext bit $\mathsf{b} \in \{0,1\}$ we have $\Pr[\mathsf{PKE.D}(\mathsf{sk}, \mathsf{PKE.E}(\mathsf{pk}, \mathsf{b}))] = \mathsf{negl}(\lambda)$. The proof of this is straightforward.

6 Proof of CCA2 Security

We will prove the following theorem.

Theorem 6.1 (CCA2 security)**.** *Let* $(\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ *be the TDF that results from Construction 4.1 based on a recyclable OWFE* $(\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$. *Assuming* $(\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$ *is adaptively secure, the PKE given in Construction 5.1 is CCA2 secure.*

We need to show that the probability of success of any CCA2 adversary is the CCA2 game is at most $\frac{1}{2} + \mathsf{negl}(\lambda)$. Fix the adversary $\mathcal{A}$ in the remainder of this section. We give the following event that describes exactly the success of $\mathcal{A}$.

Event Success**.** Sample $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PKE.K}(1^\lambda)$, $\mathsf{b}_{\text{plain}} \xleftarrow{\$} \{0,1\}$ and $\mathsf{c} \xleftarrow{\$} \mathsf{PKE.E}(\mathsf{pk}, \mathsf{b}_{\text{plain}})$. Run the CCA2 adversary $\mathcal{A}$ on input $(\mathsf{pk}, \mathsf{c})$ and reply to any query $\mathsf{c}' \neq \mathsf{c}$ of $\mathcal{A}$ with $\mathsf{PKE.D}(\mathsf{sk}, \mathsf{c}')$. We say that the event Success holds if $\mathcal{A}$ outputs $\mathsf{b}_{\text{plain}}$.

Roadmap. To prove Theorem 6.1, in Section 6.1 we define a simulated experiment Sim and we show that the probability of success of any CCA2 adversary in this experiment is $\frac{1}{2}+\mathsf{negl}(\lambda)$. Next, in Section 6.2 we will show that the probabilities of success of any CCA2 adversary in the real and simulated experiments are negligibly close, establishing Theorem 6.1.

6.1 Simulated CCA2 Experiment

We define a simulated way of doing the CCA2 experiment. Roughly, our simulator does not have the full secret key (needed to reply to CCA2 queries of the adversary), but some part of it. Our simulation is enabled by a syntactic property of our constructed TDF. We first state the property and then prove that it is satisfied by our TDF. We require the existence of an efficient algorithm $\mathsf{Recover}$ for our constructed TDF $(\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ that satisfies the following property.

Algorithm $\mathsf{Recover}$**:** The input to the algorithm is an index key ik, a pre-fix input $\mathsf{x} \in \{0,1\}^n$ and a possible image Y. The output of the algorithm is $\mathsf{X} \in \{0,1\}^{n+n_1} \cup \{\perp\}$. As for correctness we requite the following. For any $(\mathsf{ik}, *) \in \mathsf{TDF.K}(1^\lambda)$, $\mathsf{x} \in \{0,1\}^n$ and Y both the following two properties hold:

1. If there exists no $\mathsf{s} \in \{0,1\}^{n_1}$ such that $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{x}||\mathsf{s}) = \mathsf{Y}$, then $\mathsf{Recover}(\mathsf{ik}, \mathsf{x}, \mathsf{Y}) = \perp$.
2. If for some s, $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{x}||\mathsf{s}) = \mathsf{Y}$, then $\mathsf{Recover}(\mathsf{ik}, \mathsf{x}, \mathsf{Y})$ returns (x, s).

Lemma 6.2 (Existence of $\mathsf{Recover}$)**.** *There exists an efficient algorithm* $\mathsf{Recover}$ *with the above properties for our constructed TDF .*

Proof. To build $\mathsf{Recover}$, first parse the given inputs as follows: $\mathsf{ik} = (\mathsf{pp}, \dots)$, $\mathsf{x} \in \{0,1\}^n$ and $\mathsf{Y} := (\mathsf{y}, \widetilde{\mathbf{b}_{1,0}}, \widetilde{\mathbf{b}_{1,1}}, \dots, \widetilde{\mathbf{b}_{n,0}}, \widetilde{\mathbf{b}_{n,1}})$. Do the following steps:

1. For all $i \in [n]$ set $\mathbf{b}_i := \widetilde{\mathbf{b}_{i,1-\mathsf{x}_i}}$.
2. Let $\mathsf{s} = \mathbf{b}_1 || \cdots || \mathbf{b}_n$
3. Check if $\mathsf{Y} = \mathsf{TDF.F}(\mathsf{ik}, \mathsf{x}||\mathsf{s})$. If the check holds, return (x, s). Otherwise, return $\perp$.

The correctness of the algorithm $\mathsf{Recover}$ follows by inspection. □

We now define our simulation algorithm.

$\mathsf{Sim}(\mathsf{ik}_1, \dots, \mathsf{ik}_m, \mathsf{Y}_1, \dots, \mathsf{Y}_m, \mathsf{b})$**:** The simulated experiment differs from the real experiment in that the challenger does not know the trapdoor keys of half of the index keys that are given to the CCA adversary as part of the public key. The challenger, however, tries to produce accurate answers based on her partial knowledge.

Formally, do the following steps.

1. **Initializing the CCA adversary:**
   (a) Sample $(\mathsf{vk}, \mathsf{sgk}) \xleftarrow{\$} \mathsf{SIG.K}(1^\lambda)$.
   (b) For all $i \in [m]$ set $\mathsf{ik}_i^{\mathsf{vk}_i} := \mathsf{ik}_i$ and sample $(\mathsf{ik}_i^{1-\mathsf{vk}_i}, \mathsf{tk}_i^{1-\mathsf{vk}_i}) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$.

(c) Sample a challenge bit $\mathsf{b}_{\text{plain}} \xleftarrow{\$} \{0,1\}$ and let $\mathsf{b}_1 := \mathsf{b}_{\text{plain}} \oplus \mathsf{b}$. Set

$$\mathsf{msg} := \mathsf{Y}_1 || \cdots || \mathsf{Y}_m || \mathsf{b}_1.$$

Sample $\sigma \xleftarrow{\$} \mathsf{SIG.Sign}(\mathsf{sgk}, \mathsf{msg})$ and set

$$\mathsf{pk} := (\mathsf{ik}_1^0, \mathsf{ik}_1^1, \ldots, \mathsf{ik}_m^0, \mathsf{ik}_m^1),\ \mathsf{c} := (\mathsf{vk}, \mathsf{Y}_1, \ldots, \mathsf{Y}_m, \mathsf{b}_1, \sigma).$$

Run the CCA2 adversary $\mathcal{A}$ on $(\mathsf{pk}, \mathsf{c})$.

2. **Simulating the CCA responses:**

Respond to a CCA2-oracle query $\mathsf{c}' := (\mathsf{vk}', \mathsf{Y}'_1, \ldots, \mathsf{Y}'_m, \mathsf{b}', \sigma')$ as follows:

(a) Letting $\mathsf{msg}' := \mathsf{Y}'_1 || \cdots || \mathsf{Y}'_m || \mathsf{b}'$ if $\mathsf{SIG.Ver}(\mathsf{vk}', \mathsf{msg}', \sigma') = \perp$, then return $\perp$. Otherwise, if $\mathsf{vk}' = \mathsf{vk}$, then halt and return $\perp$.

(b) Otherwise, let Q consist all of all indices $i \in [m]$ for which we have $\mathsf{vk}'_i \neq \mathsf{vk}_i$. For $i \in \mathsf{Q}$ set $\mathsf{X}'_i := \mathsf{TDF.F}^{-1}(\mathsf{tk}_i^{\mathsf{vk}'_i}, \mathsf{Y}'_i)$ and check if $\mathsf{Y}'_i = \mathsf{TDF.F}(\mathsf{ik}_i^{\mathsf{vk}'_i}, \mathsf{X}'_i)$; if this fails for any $i \in \mathsf{Q}$, then return $\perp$. Now if there exists $\mathsf{x}' \in \{0,1\}^n$ such that for all $i \in \mathsf{Q}$ we have $\mathsf{X}'_i = (\mathsf{x}', *)$ then continue with the following steps and otherwise return $\perp$.

i. for all $j \in [m] \setminus \mathsf{Q}$, let $\mathsf{X}'_j := \mathsf{Recover}(\mathsf{ik}_j^{\mathsf{vk}'_j}, \mathsf{x}', \mathsf{Y}'_j)$. Reply to the query c' with $\mathsf{HC}(\mathsf{x}')$ if for all j we have $\mathsf{X}'_j \neq \perp$; otherwise, reply to the query with $\perp$.

3. **Forming the output of the experiment:** The experiment outputs 1 if $\mathcal{A}$ outputs $\mathsf{b}_{\text{plain}}$; otherwise, the experiments outputs 0.

Event $\mathsf{Success}_{\text{sim}}$. The event that $\mathsf{Sim}(\mathsf{ik}_1, \ldots, \mathsf{ik}_m, \mathsf{Y}_1, \ldots, \mathsf{Y}_m, \mathsf{b})$ outputs 1 where $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$, $\mathsf{b} := \mathsf{HC}(\mathsf{x})$ and for $i \in [m]$, $(\mathsf{ik}_i, \mathsf{tk}_i) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$, $\mathsf{s}_i \xleftarrow{\$} \{0,1\}^{nr}$ and $\mathsf{Y}_i := \mathsf{TDF.K}(\mathsf{ik}_i, \mathsf{x} || \mathsf{s}_i)$.

We show that the probability of the event $\mathsf{Success}_{\text{sim}}$ is $\frac{1}{2} + \mathsf{negl}(\lambda)$. We will then show in the next section that the probability of the event $\mathsf{Success}$ is close to that of $\mathsf{Success}_{\text{sim}}$, hence obtaining our main result.

Lemma 6.3.

$$\alpha := \Pr[\mathsf{Success}_{\text{sim}}] \leq \frac{1}{2} + \mathsf{negl}(\lambda). \tag{25}$$

Proof. This lemma follows by Lemma 4.7. Suppose the input to Sim is sampled exactly as done in the event $\mathsf{Success}_{\text{sim}}$, except that we sample $\mathsf{b} \xleftarrow{\$} \{0,1\}$ (instead of setting $\mathsf{b} := \mathsf{HC}(\mathsf{x})$). In this case the output of the simulation is 1 with probability $1/2$. Now Lemma 4.7 implies that $\alpha = \frac{1}{2} + \mathsf{negl}(\lambda)$ (Equation 25), and the proof is complete. □

6.2 Relating the Simulated and Real Experiments

We will now show that the probabilities of the events $\mathsf{Success}$ and $\mathsf{Success}_{\text{sim}}$ are negligibly close, hence obtaining Theorem 6.1 from Lemma 6.3. To this end, we define below two events Forge and Spoof, and show that the difference of the probabilities of $\mathsf{Success}$ and $\mathsf{Success}_{\text{sim}}$ is at most the sum of the probabilities of Forge and Spoof. We will then show that both these events happen with negligible probability.

Event Forge**:** In the experiment $\mathsf{Sim}(\mathsf{ik}_1, \ldots, \mathsf{ik}_m, \mathsf{Y}_1, \ldots, \mathsf{Y}_m, \mathsf{b})$ we let Forge be the event that $\mathcal{A}$ issues a CCA2 query

$$\mathsf{c}' := (\mathsf{vk}', \mathsf{Y}'_1, \ldots, \mathsf{Y}'_m, \mathsf{b}', \sigma')$$

such that $\mathsf{vk} = \mathsf{vk}'$ and $\mathsf{Ver}(\mathsf{vk}, \mathsf{Y}'_1 || \cdots || \mathsf{Y}'_m || \mathsf{b}', \sigma') = \top$. Recall that vk is part of the challenge ciphertext $\mathsf{c} := (\mathsf{vk}, \ldots)$ given to the adversary.

Informally, the event Spoof below describes a situation in which a decryption query during the execution of Sim is answered with a non $\perp$ string, but the same query may be replied to with $\perp$ under the real decryption oracle.

Event Spoof**:** Let $\mathsf{c} := (\mathsf{vk}, \ldots)$ be the challenge ciphertext formed during the execution of $\mathsf{Sim}(\mathsf{ik}_1, \ldots, \mathsf{ik}_m, \mathsf{Y}_1, \ldots, \mathsf{Y}_m, \mathsf{b})$. We let Spoof be the event that $\mathcal{A}$ issues a CCA2 query

$$\mathsf{c}' := (\mathsf{vk}', \mathsf{Y}'_1, \ldots, \mathsf{Y}'_m, \mathsf{b}', \sigma')$$

for which the following holds. Let Q be the set of indices $i \in [m]$ for which we have $\mathsf{vk}_i \neq \mathsf{vk}'_i$. For some $h \in \mathsf{Q}$ and for some $w \in [m] \setminus \mathsf{Q}$ we have

1. $\mathsf{TDF.F}^{-1}(\mathsf{tk}_h^{\mathsf{vk}'_h}, \mathsf{Y}'_h) = (\mathsf{x}', *) \neq \perp$; and
2. $\mathsf{s}'_j := \mathsf{Recover}(\mathsf{ik}_w^{\mathsf{vk}'_w}, \mathsf{x}', \mathsf{Y}'_w) \neq \perp$ but $\mathsf{TDF.F}^{-1}(\mathsf{tk}_w^{\mathsf{vk}'_w}, \mathsf{Y}'_w) = \perp$.

We will now prove the following three lemmas.

Lemma 6.4. *We have*

$$|\Pr[\mathsf{Success}] - \Pr[\mathsf{Success}_{\mathrm{sim}}]| \leq \Pr[\mathsf{Forge}] + \Pr[\mathsf{Spoof}].$$

Lemma 6.5.

$$\Pr[\mathsf{Forge}] \leq \mathsf{negl}(\lambda).$$

Lemma 6.6.

$$\Pr[\mathsf{Spoof}] \leq \mathsf{negl}(\lambda).$$

Let us first derive the proof of Theorem 6.1 and then prove each of the lemmas.

Proof of Theorem 6.1. Follows from Lemmas 6.3 and 6.4, taking into account the fact that the experiment $\mathsf{Real}(1^\lambda)$ is the real CCA2 experiment. □

We prove Lemmas 6.4 and 6.5 and will prove Lemma 6.6 in Section 6.3.

Proof of Lemma 6.4. First of all, note that the input $(\mathsf{pk}, \mathsf{c})$ given to the CCA2 adversary under the simulated experiment is identically distributed to that under the real CCA2 experiment. Thus, any possible difference between the simulated and real experiments must be due to the ways in which decryption queries are answered. We will now show that if at any point a decryption query is answered to differently under Sim and Real, then either Forge or Spoof must happen. First, in order for a query c' to be answered differently, either (1) the query c' is replied to with $\perp$ under Sim and with some $\mathsf{b}' \in \{0,1\}$ under Real; or (2) c' is replied to with some $\mathsf{b}' \in \{0,1\}$ under Sim and with $\perp$ under Real. In particular, we cannot have a situation in which c' is replied to with some

$\mathsf{b}' \in \{0,1\}$ under Sim and with $1-\mathsf{b}'$ under Real. The reason for this is that if both experiments reply to a query with something other than $\perp$, then the underlying recovered pre-image x' must be the same, hence both will end up replying with the same bit.

Let $\mathsf{c} := (\mathsf{vk}, \dots)$ be the challenge ciphertext of the underlying CCA2 adversary and $\mathsf{c}' := (\mathsf{vk}', \mathsf{msg}', \sigma')$ be an issued query. We now consider all possible cases:

- If Sim replies to $\mathsf{c}' := (\mathsf{vk}', \mathsf{msg}', \sigma')$ with $\perp$, then one of the following must hold.
  1. $\mathsf{SIG.Ver}(\mathsf{vk}', \mathsf{msg}', \sigma') = \perp$: in this case Real also replies to with $\perp$;
  2. $\mathsf{SIG.Ver}(\mathsf{vk}', \mathsf{msg}', \sigma') = \top$ and $\mathsf{vk} = \mathsf{vk}'$: in this case the event Forge happens;
  3. Sim returns $\perp$ as a result of Step 2b of the execution of Sim: that is, $\mathsf{TDF.F}^{-1}(\mathsf{tk}_i^{vk_i'}, \mathsf{Y}_i') = \perp$: in this case Real also replies with $\perp$.
  4. Sim replies with $\perp$ as a result of Step 2(b)i: in this case by correctness of $\mathsf{Recover}$ we will know Real will also reply with $\perp$.
- If Real replies to c' with $\perp$ and Sim replies to c' with some $\mathsf{b}' \in \{0,1\}$, then we may easily verify that the event Spoof must necessarily hold. We omit the details.

□

Proof of Lemma 6.5. Suppose $\Pr[\mathsf{Forge}] > \mathsf{negl}(\lambda)$. We show how to build an adversary $\mathcal{B}$ against the one-time unforgeability property of $\mathsf{SIG} = (\mathsf{SIG.K}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$. Build $\mathcal{B}^{\mathsf{SgnOracle}[\mathsf{sgk}](\cdot)}(\mathsf{vk})$ as follows. Sample the input $(\mathsf{ik}_1, \dots, \mathsf{ik}_m, \mathsf{Y}_1, \dots, \mathsf{Y}_m, \mathsf{b})$ to Sim and form the tuple msg as in the execution of Sim on this input. Then request a signature σ for the message msg by calling $\mathsf{SgnOracle}[\mathsf{sgk}](\cdot)$ on msg. Form $(\mathsf{pk}, \mathsf{c})$ as in Sim and run the CCA2 adversary $\mathcal{A}$ on $(\mathsf{pk}, \mathsf{c})$. Let q be the number queries that $\mathcal{A}$ asks. Choose $i \xleftarrow{\$} [q]$ to be a guess for the index of the first query for which the event Forge occurs and output the pair of message/signature contained in that query. Note that $\mathcal{B}$ can perfectly reply to all the previous $i-1$ queries of $\mathcal{A}$, because all of those can be replied to without knowing sgk. If $\alpha := \Pr[\mathsf{Forge}]$, then $\mathcal{B}$ will win with probability at least $\frac{\alpha}{q}$. □

6.3 Proof of Lemma 6.6

The proof of Lemma 6.6 is based on a property of our TDF that we now state and prove. Informally, if the OWFE scheme used in our TDF construction is adaptively secure, then the constructed TDF has the property that given a random index key ik, it is infeasible to produce an image element which is in the range of the trapdoor function $\mathsf{TDF.F}(\mathsf{ik}, \cdot)$, but which "inverts" to $\perp$.

Lemma 6.7. *Let* $\mathsf{TDF} = (\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ *be the TDF built in Section 4, with the underlying parameter* $r := r(\lambda) \in \omega(\log \lambda)$*, based on a recyclable OWFE scheme* $\mathsf{OWFE} = (\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$*. Assuming* OWFE *is adaptive (Definition 3.5), for any PPT adversary* $\mathcal{A}$*:*

$$\Pr[(\mathsf{X}, \mathsf{Y}) \xleftarrow{\$} \mathcal{A}(\mathsf{ik}) \text{ s.t. } \mathsf{Y} = \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X}), \mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{Y}) = \perp] = \mathsf{negl}(\lambda), \tag{26}$$

where $(\mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$.

Proof. Let $\mathsf{Surprise}$ be the event of the lemma. Parse

$$\mathsf{ik} := (\mathsf{pp}, \mathbf{ct}_{1,0}, \mathbf{ct}_{1,1}, \ldots, \mathbf{ct}_{n,0}, \mathbf{ct}_{n,1}), \mathsf{tk} := (\boldsymbol{\rho}_{1,0}, \boldsymbol{\rho}_{1,1}, \ldots, \boldsymbol{\rho}_{n,0}, \boldsymbol{\rho}_{n,1}), \tag{27}$$

and for all $i' \in [n]$ and $b' \in \{0,1\}$ parse

$$\begin{aligned} \mathbf{ct}_{i',b'} &:= (\mathsf{ct}^{(1)}_{i',b'}, \ldots, \mathsf{ct}^{(r)}_{i',b'}) \\ \boldsymbol{\rho}_{i',b'} &:= (\rho^{(1)}_{i',b'}, \ldots, \rho^{(r)}_{i',b'}). \end{aligned} \tag{28}$$

Recall that for all $i' \in [n]$, $b' \in \{0,1\}$ and $j \in [r]$ we have

$$\mathsf{ct}^{(j)}_{i',b'} = \mathsf{E}_1(\mathsf{pp}, (i', b'); \rho^{(j)}_{i',b'}). \tag{29}$$

Also, parse (X, Y), the output of $\mathcal{A}(\mathsf{ik})$, as

$$\begin{aligned} \mathsf{X} &:= (\mathsf{x} \in \{0,1\}^n, \mathbf{b}_1 \in \{0,1\}^r, \ldots, \mathbf{b}_n \in \{0,1\}^r) \\ \mathsf{Y} &:= (\mathsf{y}, \mathbf{b}_{1,0} \in \{0,1\}^r, \mathbf{b}_{1,1} \in \{0,1\}^r, \ldots, \mathbf{b}_{n,0} \in \{0,1\}^r, \mathbf{b}_{n,1} \in \{0,1\}^r). \end{aligned} \tag{30}$$

If the event $\mathsf{Surprise}$ happens, then by definition we have $\mathsf{Y} = \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X})$ and $\mathsf{TDF.F}^{-1}(\mathsf{tk}, \mathsf{Y}) = \perp$. Thus, by definition of $\mathsf{TDF.F}^{-1}$, for some $i \in [n]$ we must have

$$\mathbf{b}_i = \left(\mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i, 1-\mathsf{x}_i); \rho^{(1)}_{i,1-\mathsf{x}_i}), \ldots, \mathsf{E}_2(\mathsf{pp}, \mathsf{y}, (i, 1-\mathsf{x}_i); \rho^{(r)}_{i,1-\mathsf{x}_i})\right). \tag{31}$$

We show how to use Equation 31 to break the adaptive security of OWFE.

We show how to build an adversary against the adaptive security of OWFE in the sense of Lemma 3.6. Sample $i \xleftarrow{\$} [n]$ and $b \xleftarrow{\$} \{0,1\}$ — The value of b will serve as a guess bit for $1 - \mathsf{x}_i$ (see Equation 31). Give the pair (i, b) to the challenger to receive $(\mathsf{pp}, \mathsf{ct}_1, \ldots, \mathsf{ct}_r)$. Set $\mathbf{ct}_{i,b} := (\mathsf{ct}_1, \ldots, \mathsf{ct}_r)$ and sample all other $\mathbf{ct}_{i',b'}$, for $(i', b') \neq (i, b)$, as in Equation 28 and form the index ik as in Equation 27. Call $\mathcal{A}(\mathsf{ik})$ to receive (X, Y) and parse them as in Equation 30. If $\mathsf{x}_i = b$ then return $\perp$. Otherwise, give x to the challenger to receive some $\mathbf{e}' \in \{0,1\}^r$. If $\mathbf{e}' = \mathbf{b}_i$, then return 0, and otherwise return 1.

If the probability of the event $\mathsf{Surprise}$ is non-negligible, then $\mathcal{A}$ wins with a probability non-negligibly greater than $\frac{1}{2}$. The reason is if $\mathbf{e}'$ was generated uniformly at random from $\{0,1\}^r$, then the probability that $\mathbf{e}' = \mathbf{b}_i$ is $\frac{1}{2^r} = \mathsf{negl}(\lambda)$. On the other hand, if $\mathbf{e}'$ was generated as a result of true encapsulation encryptions (see the description of the game in Lemma 3.6), then the probability that $\mathbf{e}' = \mathbf{b}_i$ is the probability of the event in Equation 31, which is non-negligible. Thus, we break the adaptive security of OWFE, a contradiction to Lemma 3.6. □

Proof of Lemma 6.6. The proof of this lemma follows easily from Lemma 6.7, so we will give a sketch of the proof. Let $\beta := \Pr[\mathsf{Spoof}]$. We show how to build an adversary $\mathcal{B}$ in the sense of Lemma 6.7 that wins with probability $\frac{\beta}{poly(\lambda)}$.

Recall h and w from the event Spoof. The adversary $\mathcal{B}(\mathsf{ik})$ acts as follows.

1. Sample $(\mathsf{vk}, \mathsf{sgk}) \xleftarrow{\$} \mathsf{SIG.K}(1^\lambda)$.
2. Guess $w \xleftarrow{\$} [m]$

3. Set $\mathsf{ik}_w^{\mathsf{vk}_w} := \mathsf{ik}$. Also, sample $(\mathsf{ik}_w^{1-\mathsf{vk}_w}, \mathsf{tk}_w^{1-\mathsf{vk}_w}) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$

4. For all $i \in [m] \setminus \{w\}$ and $b \in \{0,1\}$, sample $(\mathsf{ik}_i^b, \mathsf{tk}_w^b) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$.

5. Sample $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$, $\mathsf{b} := \mathsf{HC}(\mathsf{x})$ and for $i \in [m]$, $\mathsf{Y}_i := \mathsf{TDF.K}(\mathsf{ik}_i^{\mathsf{vk}_i}, \mathsf{x}||\mathsf{s_i})$.

6. Sample $\mathsf{b}_{\text{plain}} \xleftarrow{\$} \{0,1\}$ and set $\mathsf{b}_1 := \mathsf{b}_{\text{plain}} \oplus \mathsf{b}$.

7. Set the challenge public key and ciphertext $(\mathsf{pk}, \mathsf{c})$ as in Sim and run the CCA2 adversary $\mathcal{A}$ on $(\mathsf{pk}, \mathsf{c})$.

Now guess η to be the index of the first query of $\mathcal{A}$ that causes Spoof to happen and guess $\mathsf{h} \xleftarrow{\$} [m]$ be the underlying index defined in the event Forge. Note that $\mathcal{B}$ can perfectly simulate the response all the first $\eta - 1$ queries of $\mathcal{A}$ as in Sim. The reason is that $\mathcal{B}$ has the trapdoor key for all $\mathsf{ik}_i^{1-\mathsf{vk}_i}$, and so it can perform as in Sim.

The ηth query. Letting the ηth query be

$$\mathsf{c}' := (\mathsf{vk}', \mathsf{Y}'_1, \ldots, \mathsf{Y}'_m, \mathsf{b}'_{ciph}, \sigma')$$

$\mathcal{B}$ acts as follows: it sets $(\mathsf{x}', \mathsf{s}_h) := \mathsf{TDF.F}^{-1}(\mathsf{tk}_h^{\mathsf{vk}'_h}, \mathsf{Y}'_h)$ and sets $\mathsf{X}'_\mathsf{w} := \mathsf{Recover}(\mathsf{ik}, \mathsf{x}', \mathsf{Y}'_w)$. Finally, $\mathcal{B}$ returns $(\mathsf{X}'_w, \mathsf{Y}'_w)$.

It is now easy to verify if Spoof occurs and that all the guesses of the adversary $\mathcal{B}$ were correct (i.e., the guessed values for h, w and vk'_w) — which happens with probability $\frac{1}{poly(\lambda)}$ — the adversary $\mathcal{B}$ wins in the sense of Lemma 6.7. □

Acknowledgments. We would like to thank the anonymous reviewers for their useful comments, and thank Mohammad Mahmoody and Adam O'Neill for useful discussions.

References

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. 3

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany. 2

[BFM90] Manuel Blum, Paul Feldman, and Silvio Micali. Proving security against chosen cyphertext attacks. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 256–268, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Heidelberg, Germany. 2

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 7

[BFOR08] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 360–378, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 2

[BHSV98] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil P. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 283–298, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany. 2

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 5, 6, 7

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 10

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. 6

[CKS08] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 127–145, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. 6

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 372–408, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. 6

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 3, 6, 8, 10, 30

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*,

pages 3–31, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. 6

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 1

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. 31

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany. 2

[FGK+10] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 279–295, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany. 2

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317, St. Louis, Missouri, October 22–24, 1990. IEEE Computer Society Press. 2

[GL89] Oded Goldreich and Leonid 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. 6, 11, 30

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377, San Francisco, CA, USA, May 5–7, 1982. ACM Press. 2

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd FOCS*, pages 126–135, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. 2

[HJKS10] Kristiyan Haralambiev, Tibor Jager, Eike Kiltz, and Victor Shoup. Simple and efficient public-key encryption from computational Diffie-Hellman in the standard model. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 1–18, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany. 6

[HK15] Mohammad Hajiabadi and Bruce M. Kapron. Reproducible circularly-secure bit encryption: Applications and realizations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 224–243, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 7

[KMO10] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110

of *LNCS*, pages 673–692, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. 2

[Ms09] Steven Myers and abhi shelat. Bit encryption is complete. In *50th FOCS*, pages 607–616, Atlanta, GA, USA, October 25–27, 2009. IEEE Computer Society Press. 17

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437, Baltimore, MD, USA, May 14–16, 1990. ACM Press. 2

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 6

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press. 2, 7

[RS09] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 419–436. Springer, Heidelberg, Germany, March 15–17, 2009. 2, 6

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978. 1

[Wee10] Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 314–332, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 3, 6

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

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press. 2

A Proof of Lemma 3.7

Proof. We prove that all the required properties hold.

One-wayness. The fact that f_{pp} for a random pp is one-way follows by the discrete-log hardness (and hence CDH hardness) of $\mathbb{G}$. Let g^* be a random group element for which we want to find r^* such that $g^{r^*} = g^*$. Sample $i_1 \xleftarrow{\$} [n]$ and $b_1 \xleftarrow{\$} \{0,1\}$ and set $g_{i_1,b_1} := g^*$. For all $i \in [n]$ and $b \in \{0,1\}$ where $(i,b) \neq (i_1,b_1)$, sample $r_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ and set $g_{i,b} := g^{r_{i,b}}$. Set $\mathsf{pp} := \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix}$. Sample x' at random from $\{0,1\}^n$ subject to the condition that $\mathsf{x}'_{i_1} = 1 - b_1$. Set $\mathsf{y} := \prod_{j \in [n]} g_{j,\mathsf{x}'_j}$.

Call the inverter adversary on $(\mathsf{pp}, \mathsf{y})$ to receive $\mathsf{x} \in \{0,1\}^n$. Now if $n \in \omega(\log p)$, then by the leftover hash lemma with probability negligibly close to $\frac{1}{2}$ we have $\mathsf{x}_{i_1} = b_1$, allowing us to find r^* from $r_{i,b}$'s.

Recyclability. We need to show that the ciphertext output ct of E is independent of the input value y. This follows immediately by inspection.

Notation. For a matrix $\mathbf{M} := \begin{pmatrix} a_{1,0}, a_{2,0}, \ldots, a_{n,0} \\ a_{1,1}, a_{2,1}, \ldots, a_{n,1} \end{pmatrix}$, $i \in [n]$ and $b \in \{0,1\}$, we define the matrix $\mathbf{M}' := \mathbf{M}|(i,b)$ to be the same as $\mathbf{M}$ except that instead of $a_{i,b}$ we put $\perp$ in $\mathbf{M}'$. If $\mathbf{M}$ is matrix of group elements, then $\mathbf{M}^r$ denotes element-wise exponentiation to the power of r.

Security for encryption. We show that assuming $\mathbb{G}$ is CDH-hard, the scheme is adaptively secure. Suppose there exists an adversary $\mathcal{A}$ such that $\Pr[\mathsf{AdapOWFE}[t=1](\mathcal{E}, \mathcal{A})] = \frac{1}{2} + \frac{1}{q} > \frac{1}{2} + \mathsf{negl}(\lambda)$. Using standard techniques we may transform $\mathcal{A}$ into a predictor $\mathcal{B}$ who wins with probability at least $\frac{1}{2} + \frac{1}{q}$ in the following experiment:

1. $(i^*, b^*) \xleftarrow{\$} \mathcal{B}(1^\lambda)$.
2. Sample $\mathsf{pp} := \left(\begin{smallmatrix} g_{1,0}, g_{2,0}, \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{smallmatrix} \right) \xleftarrow{\$} \mathbb{G}^{2\times n}$.
3. Sample $\rho \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathsf{ct} := \mathsf{pp}^\rho | (i^*, b^*)$.
4. $(\mathsf{x}, b) \xleftarrow{\$} \mathcal{B}(\mathsf{pp}, \mathsf{ct})$.
5. $\mathcal{B}$ wins if $\mathsf{x}_{i^*} = b^*$ and $b = \mathsf{HC}(\mathsf{y}^\rho)$, where $\mathsf{y} := \prod_{j \in [n]} g_{j,\mathsf{x}_j}$.

Using the Goldreich-Levin theorem we know that there should be an adversary $\mathcal{B}_1$ that wins with non-negligible probability in the following:

1. $(i^*, b^*) \xleftarrow{\$} \mathcal{B}_1(1^\lambda)$.
2. Sample $\mathsf{pp} := \left(\begin{smallmatrix} g_{1,0}, g_{2,0} \cdots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{smallmatrix} \right) \xleftarrow{\$} \mathbb{G}^{2\times n}$.
3. Sample $\rho \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathsf{ct} := \mathsf{pp}^\rho | (i^*, b^*)$.
4. $(\mathsf{x}, g^*) \xleftarrow{\$} \mathcal{B}_1(\mathsf{pp}, \mathsf{ct})$.
5. $\mathcal{B}_1$ wins if $\mathsf{x}_{i^*} = b^*$ and $g^* = \mathsf{y}^\rho$, where $\mathsf{y} := \prod_{j \in [n]} g_{j,\mathsf{x}_j}$.

We now show how to use $\mathcal{B}_1$ to solve the CDH problem.

CDH adversary $\mathcal{A}_1(g, g_1, g_2)$:

1. Run $\mathcal{B}_1(1^\lambda)$ to get (i^*, b^*).
2. For any $j \in [n] \setminus \{i^*\}$ and $b \in \{0,1\}$ sample $\alpha_{j,b} \xleftarrow{\$} \mathbb{Z}_p$ and set $g_{j,b} = g^{\alpha_{j,b}}$. Set $g_{i^*,b^*} := g_1$ and $g_{i^*,1-b^*} = g^\alpha$, where $\alpha \xleftarrow{\$} \mathbb{Z}_p$. Set $\mathsf{pp} := \left(\begin{smallmatrix} g_{1,0}, g_{2,0} \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{smallmatrix}\right)$.
3. Set $g'_{i^*,1-b^*} = g_2$ and $g'_{i^*,b^*} = \perp$. For any $j \in [n] \setminus \{i^*\}$ and $b \in \{0,1\}$ set $g'_{j,b} = g_2^{(\alpha^{-1} \cdot \alpha_{j,b})}$. Set $\mathsf{ct} := \left(\begin{smallmatrix} g'_{1,0}, g'_{2,0} \ldots, g'_{n,0} \\ g'_{1,1}, g'_{2,1}, \ldots, g'_{n,1} \end{smallmatrix}\right)$.
4. Run $\mathcal{B}_1(\mathsf{pp}, \mathsf{ct})$ to get (x, g^*). If $\mathsf{x}_i \neq b_i^*$ then return $\perp$. Otherwise
   (a) Set
   $$g_u := \frac{g^*}{\prod_{j=1}^{i^*-1} g'_{j,\mathsf{x}_j} \cdot \prod_{j=i^*+1}^{n} g'_{j,\mathsf{x}_j}}.$$
   (b) Return g_u^α.

By inspection one may easily verify that whenever $\mathcal{B}_1$ wins, $\mathcal{A}_1$ also wins. The proof is now complete. □

B Description of Our CDH-Based TDF

In this section we describe our resulting CDH-based TDF, obtained by instantiating our general TDF construction using the CDH-based recyclable OWFE given in Section 3.3. We first start with some notation.

Notation. Letting $\mathsf{x} \in \{0,1\}^n$ and $\mathbf{M} := \begin{pmatrix} g_{1,0}, g_{2,0} \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{pmatrix}$ we define $\mathsf{x} \odot \mathbf{M} = \prod_{j \in [n]} g_{j,\mathsf{x}_j}$. For $i \in [n]$ and $b \in \{0,1\}$, we define the matrix $\mathbf{M}' := \mathbf{M}|(i,b)$ to be the same as $\mathbf{M}$ except that instead of $g_{i,b}$ we put $\perp$ in $\mathbf{M}'$. If $\mathbf{M}$ is matrix of group elements, then $\mathbf{M}^r$ denotes entry-wise exponentiation to the power of r.

CDH-based TDF. We will only describe the key generation and evaluation algorithms. The inversion algorithm follows easily from the other two algorithms.

- $\mathsf{TDF.K}(1^\lambda)$:
  - Sample $\mathsf{pp} := \left(\begin{smallmatrix} g_{1,0}, g_{2,0} \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{smallmatrix}\right) \xleftarrow{\$} \mathbb{G}^{2 \times n}$.
  - For each $i \in [n]$ and $b \in \{0,1\}$

$$\boldsymbol{\rho}_{i,b} := (\rho_{i,b}^{(1)}, \ldots, \rho_{i,b}^{(r)}) \xleftarrow{\$} \mathbb{Z}^r \tag{32}$$

$$\text{For } j \in [r] : \mathbf{M}_{i,b}^j := \left(\mathsf{pp}^{\rho_{i,b}^{(j)}} | (i,b)\right) \in \mathbb{G}^{2 \times n} \tag{33}$$

$$\mathbf{ct}_{i,b} := (\mathbf{M}_{i,b}^1, \ldots, \mathbf{M}_{i,b}^r). \tag{34}$$

– Form the index key ik and the trapdoor key tk as follows:

$$\mathsf{ik} := (\mathsf{pp}, \mathbf{ct}_{1,0}, \mathbf{ct}_{1,1}, \ldots, \mathbf{ct}_{n,0}, \mathbf{ct}_{n,1}) \tag{35}$$

$$\mathsf{tk} := \left(\boldsymbol{\rho}_{1,0}, \boldsymbol{\rho}_{1,1}, \ldots, \boldsymbol{\rho}_{n,0}, \boldsymbol{\rho}_{n,1}\right). \tag{36}$$

- $\mathsf{TDF.F}(\mathsf{ik}, \mathsf{X})$: Parse ik as in Equation 35 and $\mathbf{ct}_{i,b}$ as in Equation 34.
  - Parse
  $$\mathsf{X} := (\mathsf{x} \in \{0,1\}^n, \mathbf{b}_1 \in \{0,1\}^r, \ldots, \mathbf{b}_n \in \{0,1\}^r).$$
  - Set $\mathsf{y} := \mathsf{x} \odot \mathsf{pp}$. Note that $\mathsf{y} \in \mathbb{G}$.
  - For all $\mathsf{i} \in [n]$ set
  $$\mathbf{e}_i := \left(\mathsf{HC}(\mathsf{x} \odot \mathbf{M}^1_{i,\mathsf{x}_i}), \ldots, \mathsf{HC}(\mathsf{x} \odot \mathbf{M}^r_{i,\mathsf{x}_i})\right) \in \{0,1\}^r.$$
  - Return
  $$\mathsf{Y} := \left(\mathsf{y}, \mathsf{Perm}(\mathbf{e}_1, \mathbf{b}_1, \mathsf{x}_1), \ldots, \mathsf{Perm}(\mathbf{e}_n, \mathbf{b}_1, \mathsf{x}_n)\right) \in \mathbb{G} \times \{0,1\}^{2nr}.$$

C Standard Primitives

One-time signature schemes. A one-time signature scheme SIG with message space $\{0,1\}^\eta$ is given by three PPT algorithms $\mathsf{SIG.K}$, $\mathsf{SIG.Sign}$ and $\mathsf{SIG.Ver}$ satisfying the following syntax. The algorithm $\mathsf{SIG.K}$ on input a security parameter 1^λ outputs a pair $(\mathsf{vk}, \mathsf{sgk})$ consisting of a verification key vk and a signing key sgk. The signing algorithm $\mathsf{SIG.Sign}$ on input a signing key sgk and a message $\mathsf{m} \in \{0,1\}^\eta$ outputs a signature σ. For correctness, we require that for any $(\mathsf{vk}, \mathsf{sgk}) \in \mathsf{SIG.K}(1^\lambda)$, any message $\mathsf{m} \in \{0,1\}^\eta$ and any signature $\sigma \in \mathsf{SIG.Sign}(\mathsf{sgk}, \mathsf{m})$: $\mathsf{SIG.Ver}(\mathsf{vk}, \mathsf{m}, \sigma) = \top$. The one-time unforgeability property requires that the success probability of any PPT adversary $\mathcal{A}$ in the following game be at most negligible. Sample $(\mathsf{vk}, \mathsf{sgk}) \xleftarrow{\$} \mathsf{SIG.K}(1^\lambda)$ and give vk to $\mathcal{A}$. Now, $\mathcal{A}(\mathsf{vk})$ may call a signing oracle $\mathsf{SgnOracle}[\mathsf{sgk}](\cdot)$ only once, where the oracle $\mathsf{SgnOracle}[\mathsf{sgk}](\cdot)$ on input m returns $\sigma \xleftarrow{\$} \mathsf{SIG.Sign}(\mathsf{sgk}, \mathsf{m})$. Finally, $\mathcal{A}(\mathsf{vk})$ should return a pair (m', σ') of message/signature and will win if $(\mathsf{m}, \sigma) \neq (\mathsf{m}', \sigma')$ and that $\mathsf{SIG.Ver}(\mathsf{vk}, \mathsf{m}', \sigma') = \top$.

D Extracting Hardcore Bits From Our TDF

Since our constructed TDF is one-way, we may use standard techniques, e.g., [GL89], to obtain hardcore bits. In this section we show that any hardcore function h for f is also a hardcore function for our constructed TDF. We will then use this fact to show that if f is length decreasing (as in [DG17b]), then one may use standard randomness extraction techniques to obtain many hardcore bits.

Definition D.1 (Hardcore functions)**.** *Let* $(\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$ *be a OWFE scheme. We say that a function* $\mathsf{h}(\cdot,\cdot)$ *with* $u = u(\lambda)$ *bit-long outputs is a* hardcore function *for* f *if* $(\mathsf{pp}, \mathsf{y}, \mathsf{h}(\mathsf{pp}, \mathsf{x})) \stackrel{c}{\equiv} (\mathsf{pp}, \mathsf{y}, \mathsf{w})$*, where* $\mathsf{pp} \xleftarrow{\$} \mathsf{K}(1^\lambda)$*,* $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$*,* $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$ *and* $\mathsf{w} \xleftarrow{\$} \{0,1\}^u$*.*

Similarly, we say that a function $\mathsf{h}(\cdot,\cdot)$ *with* $u = u(\lambda)$ *bit-long outputs is a* hardcore function *for a TDF* $(\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ *if*

$$(\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X}), \mathsf{h}(\mathsf{ik}, \mathsf{X})) \stackrel{c}{\equiv} (\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X}), \mathsf{w}), \tag{37}$$

where $(\mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$, $\mathsf{X} \xleftarrow{\$} \{0,1\}^*$ *and* $\mathsf{w} \xleftarrow{\$} \{0,1\}^u$.

We will now show that any hardcore function h for f gives rise to a similar hardcore function for our constructed TDF $(\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ built based on $(\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$ according to Construction 4.1.

Lemma D.2. *Suppose* $(\mathsf{K}, \mathsf{f}, \mathsf{E}, \mathsf{D})$ *is a recyclable OWFE scheme and* h *is a* u*-bit-output hardcore function for* f*. Let* $\mathsf{TDF} = (\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$ *be the TDF built according to Construction 4.1.*

Define a hardcore function $\mathsf{TDF.h}$ *for the trapdoor function* TDF *as follows. On input* $\mathsf{ik} := (\mathsf{pp}, *)$ *and* $\mathsf{X} := (\mathsf{x}, *)$*, define*

$$\mathsf{TDF.h}(\mathsf{ik}, \mathsf{X}) \triangleq \mathsf{h}(\mathsf{pp}, \mathsf{x}).$$

Then, $\mathsf{TDF.h}$ *is a hardcore function for* $(\mathsf{TDF.K}, \mathsf{TDF.F}, \mathsf{TDF.F}^{-1})$*, namely*

$$(\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X}), \mathsf{TDF.h}(\mathsf{ik}, \mathsf{X})) \stackrel{c}{\equiv} (\mathsf{ik}, \mathsf{TDF.F}(\mathsf{ik}, \mathsf{X}), \mathsf{w}), \tag{38}$$

where $(\mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{TDF.K}(1^\lambda)$, $\mathsf{X} := (\mathsf{x}, *) \xleftarrow{\$} \{0,1\}^{n+nr}$ *and* $\mathsf{w} \xleftarrow{\$} \{0,1\}^u$.

Proof. Let view_0 and view_1 be as in the proof of Lemma 4.3. To prove Equation 38 we need to show that

$$(\mathsf{view}_0, \mathsf{TDF.h}(\mathsf{ik}, \mathsf{X})) \stackrel{c}{\equiv} (\mathsf{view}_0, \mathsf{u}),$$

where ik, $\mathsf{X} := (\mathsf{x}, *)$ and u are sampled as above. To prove this, first recall that in the proof of Lemma 4.3 we showed that $(\mathsf{view}_0, \mathsf{x}) \stackrel{c}{\equiv} (\mathsf{view}_1, \mathsf{x})$. Therefore, we have $(\mathsf{view}_0, \mathsf{TDF.h}(\mathsf{ik}, \mathsf{X})) \stackrel{c}{\equiv} (\mathsf{view}_1, \mathsf{TDF.h}(\mathsf{ik}, \mathsf{X}))$. This is because we have $\mathsf{TDF.h}(\mathsf{ik}, \mathsf{X}) = \mathsf{h}(\mathsf{pp}, \mathsf{x})$, which can be constructed with knowledge of pp and x.

Now using the fact that view_1 can be perfectly formed using pp and $\mathsf{y} := \mathsf{f}(\mathsf{pp}, \mathsf{x})$, and since h is a hardcore function for f, we obtain $(\mathsf{view}_1, \mathsf{TDF.h}(\mathsf{ik}, \mathsf{X})) \stackrel{c}{\equiv} (\mathsf{view}_1, \mathsf{u})$. Moreover, we know that $(\mathsf{view}_1, \mathsf{u}) \stackrel{c}{\equiv} (\mathsf{view}_0, \mathsf{u})$. Thus, we have

$$(\mathsf{view}_0, \mathsf{TDF.h}(\mathsf{ik}, \mathsf{X})) \stackrel{c}{\equiv} (\mathsf{view}_1, \mathsf{TDF.h}(\mathsf{ik}, \mathsf{X})) \stackrel{c}{\equiv} (\mathsf{view}_1, \mathsf{u}) \stackrel{c}{\equiv} (\mathsf{view}_0, \mathsf{u}),$$

as desired. □

Length shrinking functions f. If the function f is sufficiently length shrinking, it admits hardcore functions with many hardcore bits via random extraction methods, e.g., pairwise independent hash functions, see, e.g., [DRS04].