Electronic Colloquium on Computational Complexity, Report No. 80 (2007)

Lossy Trapdoor Functions and Their Applications

Chris Peikert
SRI International

Brent Waters[*]
SRI International

Abstract

We propose a new general primitive called *lossy trapdoor functions* (lossy TDFs), and realize it under a variety of different number theoretic assumptions, including hardness of the decisional Diffie-Hellman (DDH) problem and the *worst-case* hardness of standard lattice problems.

Using lossy TDFs, we develop a new approach for constructing many important cryptographic primitives, including standard trapdoor functions, CCA-secure cryptosystems, collision-resistant hash functions, and more. All of our constructions are simple, efficient, and black-box.

Taken all together, these results resolve some long-standing open problems in cryptography. They give the first known (injective) trapdoor functions based on problems not directly related to integer factorization, and provide the first known CCA-secure cryptosystem based solely on worst-case lattice assumptions.

1 Introduction

A central goal in cryptography is to realize a variety of security notions based on plausible computational assumptions. Historically, such assumptions have typically been concerned with number theoretic problems from one of three broad categories: those related to *integer factorization*, those related to the *discrete logarithm* in cyclic groups, and more recently, those related to computing short vectors in *lattices*.

For several reasons, it is important to design secure cryptographic schemes based on all three categories. First, to act as a hedge against unanticipated advances in cryptanalysis, e.g., improved algorithms for one class of problems, or the construction of a large-scale quantum computer. Second, to justify the generality of abstract notions. And third, to develop new outlooks and techniques that can cross-pollinate and advance the field as a whole.

In public key cryptography in particular, two of the most important notions are *trapdoor functions* (TDFs) and *security under chosen ciphertext attack* (CCA security) [29, 32, 16]. The former is an idea going all the way back to the seminal paper of Diffie and Hellman [13], while the latter has become the *de facto* notion of security for public key encryption.

Unfortunately, it is still not known how to realize TDFs and CCA security under all of the categories of problems listed above. For CCA security, there are secure constructions based on problems related to factoring and discrete log [29, 16, 11, 12], but not lattices. For trapdoor functions, the situation is even worse: though TDFs are widely viewed as a general primitive, they have so far been realized only from problems related to factoring [35, 31, 30].

[*]Supported by NSF CNS-0524252 and the US Army Research Office under the CyberTA Grant No. W911NF-06-1-0316.

ISSN 1433-8092

In this paper, we make the following contributions:

- We introduce a new general primitive called *lossy trapdoor functions*, and show how to realize it based on the hardness of the decisional Diffie-Hellman (DDH) problem in cyclic groups, and the hardness of *worst-case* problems on lattices.
- We show that lossy trapdoor functions imply trapdoor functions in the traditional sense. This yields the first known trapdoor functions based on number theoretic problems that are not directly related to integer factorization, and that do not appear to have trapdoors naturally "built in."
- We present a *black-box* construction of a CCA-secure cryptosystem based on lossy TDFs. Notably, our decryption algorithm is *witness recovering*, i.e., it first recovers the randomness that was used to create the ciphertext, and then tests the validity of the ciphertext simply by reencrypting the message under the retrieved randomness. Until now, witness-recovering CCA-secure cryptosystems were only known to exist in the random oracle model [4, 19].

  Our approach has two main benefits: first, our construction is black-box, making it more efficient than those following the general NIZK paradigm [5, 29, 15, 36].[1] Second, ours is the first known construction of a CCA-secure cryptosystem based entirely on worst-case lattice assumptions, for which there is currently no known realization in the NIZK framework.
- We further demonstrate the utility of lossy TDFs by constructing collision-resistant hash functions and semi-honest oblivious transfer (OT) protocols, in a black-box manner. Using standard (but non-black-box) transformations [24, 25], we obtain OT protocols for malicious adversaries and general secure multiparty computation (MPC).

1.1 Trapdoor Functions and Witness-Recovering Decryption

One interesting and long-standing question in cryptography is whether it is possible to construct trapdoor functions from a semantically secure cryptosystem [3]. One tempting approach is to encrypt the function's random input x using x itself as the randomness, so that decrypting with the secret key (i.e., the trapdoor) returns x. This method has several potential benefits. First, the construction is very straightforward and efficient. Second, the technique could be extended to build a CCA-secure cryptosystem: the encryption algorithm would simply choose a random string r and encrypt it along with the "true" message m, also using r as the randomness to the encryption. The decryption algorithm would check for well-formedness of a ciphertext by first decrypting, yielding the message m and randomness r, and then would simply recompute the ciphertext to verify that it matches the input ciphertext. Indeed, approaches like these have proved fruitful in the random oracle model [3, 4, 19].

Unfortunately, the technique of encrypting a ciphertext's own randomness has so far met with less success in the standard model, because semantic security is guaranteed only if the randomness is chosen *independently* of the encrypted message. For example, consider a (pathological) encryption algorithm E', which is built from another (secure) encryption algorithm E: the algorithm $E'(m; r)$ normally returns $E(m; r)$, except if $m = r$ it simply outputs r. Then the candidate trapdoor function $f(x) = E'(x; x)$ is simply the identity function, which is trivial to invert.

[1]We note that Cramer and Shoup [11, 12] showed efficient constructions based on specific number theoretic assumptions.

While this is just a counterexample to one particular construction, Gertner *et al.* [21] in fact demonstrated a *black-box separation* between (poly-to-one) trapdoor functions and semantically secure encryption. More precisely, it is impossible to construct a (poly-to-one) trapdoor function from a semantically secure cryptosystem in a black-box fashion, using a black-box security reduction. The chief difficulty is that inverting a trapdoor function requires the recovery of its *entire* input, whereas decrypting a ciphertext recovers only the input message, not the randomness. For similar reasons, there is also some evidence that achieving CCA security from semantic security (in a black-box manner) would be difficult [20].

Perhaps for these reasons, constructions of CCA-secure encryption in the standard model [29, 15, 36, 11, 12] have followed a different path. As explained in [17], all the techniques used so far have employed a "two-key" construction, where the well-formedness of a ciphertext is guaranteed by a (simulation-sound) non-interactive zero knowledge (NIZK) proof. The zero-knowledge paradigm has the benefit that the decryption algorithm can be assured that the ciphertext is well formed without needing to know a witness to that fact (e.g., the input randomness). The two-key/NIZK paradigm has led to CCA-secure encryption based on general assumptions, such as trapdoor permutations [15], and efficient systems based on specific number theoretic assumptions [11, 12], such as the decisional Diffie-Hellman (DDH) [6] and composite residuosity [30] assumptions. However, the NIZK approach has two significant drawbacks. First, the constructions from general assumptions are *inefficient*, as they are inherently non-black-box. Second, while there exist semantically secure public key cryptosystems based on *worst-case lattice assumptions* [2, 33, 34], there are still no known *CCA-secure* systems, because it is unknown how to realize NIZKs from such assumptions.

1.2 The Power of Losing Information

In this paper we revisit the idea of building trapdoor functions and witness-recovering CCA-secure encryption, in the standard model. As discussed above, past experience seems to suggest that a stronger notion than semantic security might be needed.

We introduce a new approach that is centered around the idea of *losing information*. Specifically, we introduce a new primitive called a *lossy trapdoor function*, which is a public function f that is created to behave in one of two *indistinguishable* ways. The first way corresponds to the usual correctness condition for an (injective) trapdoor function, i.e., the *entire* input x can be recovered from $f(x)$, given a suitable trapdoor value. In the second way, f *loses* a significant amount of information about its input, i.e., the size of f's image is significantly smaller than its domain.

Using lossy trapdoor functions as a general tool, we develop new techniques for constructing (standard) trapdoor functions and CCA-secure cryptosystems, and proving their security. In essence, lossy TDFs allow us to prove security via indistinguishability arguments over the *public parameters* of a scheme (e.g., the public key of a cryptosystem), as opposed to the adversary's *challenge value* (e.g., the challenge ciphertext in the CCA game).

In more detail, the public parameters of our schemes will include some function f that is either injective or lossy. In the former case (typically corresponding to a real instantiation), the invertibility of f will allow recovery of its entire input and will ensure correctness of (say) decryption. In the latter case, the lossiness of f will imply that the scheme becomes *unconditionally* secure. The advantage of this approach is that when distinguishing between injective and lossy f in the security reduction, the simulator can always create the adversary's challenge "honestly," i.e., knowing its underlying randomness.

We now demonstrate the utility of lossy TDFs by informally sketching constructions of standard

TDFs, CPA-secure encryption, and CCA-secure encryption. (Formal definitions, constructions, and proofs are given in Sections 3 and 4.)

1.2.1 Trapdoor Functions

Suppose we have a collection of lossy TDFs having input length n and lossiness (say) $k = n/2$. Then the *injective* functions from this collection are themselves a collection of standard trapdoor functions. To see this, first consider the behavior of a hypothetical inverter $\mathcal{I}$ for an injective function f. If we choose $x \leftarrow \{0,1\}^n$ uniformly and give $(f, f(x))$ to $\mathcal{I}$, it must output that *unique* x because f is injective. Now by indistinguishability of lossy and injective functions, the same must be true when f is replaced with a lossy function f'. However, $f'(x)$ *statistically* hides the value of x, because there are (on average) about $2^k = 2^{n/2}$ other values x' such that $f'(x') = f'(x)$, all equally likely. Therefore even an *unbounded* inverter $\mathcal{I}$ cannot guess the unique value of x. We conclude that no (efficient) inverter exists for the injective functions either.

1.2.2 CPA-Secure Encryption

Using the fact that lossy TDFs imply standard TDFs, we could construct CPA-secure encryption using standard techniques, e.g., using a hard-core predicate. However, it will be an instructive warm-up for our CCA-secure construction (and useful in its own right) to give a more direct construction and security proof.

Our construction uses a pairwise independent family of hash function $\mathcal{H}$ from $\{0,1\}^n$ to $\{0,1\}^\ell$, where n is the length of the input to the lossy TDFs and ℓ is the length of the plaintexts (which will depend on the lossiness; we defer details for the purposes of this sketch).

The key generator samples an injective function f and its trapdoor t, and selects a hash function h from $\mathcal{H}$. The public key is (f, h), and the trapdoor t is the decryption key (along with the public key itself). To encrypt a message $m \in \{0,1\}^\ell$, choose $x \in \{0,1\}^n$ uniformly at random, and output the ciphertext $c = (c_1, c_2) = (f(x), m \oplus h(x))$. (Note that h acts here as a hard-core function for f.) Decrypt a ciphertext $c = (c_1, c_2)$, using the trapdoor t, as $m = c_2 \oplus h(f^{-1}(c_1))$.

Now consider an efficient adversary who mounts a chosen-plaintext attack. By indistinguishability of injective and lossy functions, the adversary's advantage is essentially the same if we instead provide it with a (malformed) public key (f, h) in which f is *lossy*. In this world, the value of x still remains *statistically* well hidden given $c_1 = f(x)$. It can therefore be shown that $h(x)$ acts as a nearly uniform and independent one-time pad, so even an unbounded adversary's advantage is negligible. We conclude that the adversary's advantage in the original scheme is also negligible, and that the cryptosystem is CPA-secure.

1.2.3 CCA-Secure Encryption

The construction of CCA-secure cryptosystems is more challenging, because the adversary is allowed to make decryption (i.e., inversion) queries. If we simply replace an injective function with a lossy function in a simulation step, then the simulator will not be able to answer (even well-formed) decryption queries, because the plaintext information will be lost. Therefore we introduce a richer abstraction called *all-but-one* (ABO) trapdoor functions. An ABO collection will be associated with a (possibly large) set B that we call *branches* of the function. When generating an ABO function, the sampling algorithm will take an extra parameter $b^* \in B$, called the *lossy* branch, and will output a function $g(\cdot,\cdot)$ and a trapdoor t. The function g has the property that for any branch $b \neq b^*$, the function $g(b,\cdot)$ is injective (and can be inverted with t), but the function $g(b^*,\cdot)$ is lossy. Moreover, the function g hides its lossy branch from any computationally bounded adversary.

Our construction of a CCA-secure cryptosystem uses an ABO collection and a lossy TDF collection, both having domain $\{0,1\}^n$. The construction also uses a one-time strongly unforgeable signature scheme, where the set of possible verification keys is equal to the set of branches B in ABO collection. As before, we also use a pairwise independent family of hash functions $\mathcal{H}$ from $\{0,1\}^n$ to $\{0,1\}^\ell$, where ℓ is the length of the plaintext. In this sketch we will give only the main ideas, and defer the exact selection of parameters to Section 4.

The key generator for the cryptosystem samples an injective function f from the lossy TDF collection, along with its trapdoor t. Next, it samples an ABO function g whose lossy branch is arbitrarily set to $b^* = 0$ (the actual system will not need the trapdoor for g, so it can be discarded). Finally, it selects a hash function h from $\mathcal{H}$. The public key is $pk = (f, g, h)$, and the trapdoor t for f is kept as the secret decryption key (along with the public key itself).

The encryption algorithm encrypts a message $m \in \{0,1\}^\ell$ as follows: it first generates a verification key vk and corresponding signing key sk_σ for the signature scheme. It then chooses a uniformly random $x \in \{0,1\}^n$. The ciphertext is generated as

$$c = (vk, \quad c_1 = f(x), \quad c_2 = g(vk, x), \quad c_3 = m \oplus h(x), \quad \sigma),$$

where σ is a signature of (c_1, c_2, c_3) under the signing key sk_σ. We point out that both f and g are evaluated on the same x.

The decryption algorithm attempts to decrypt a ciphertext $c = (vk, c_1, c_2, c_3, \sigma)$ as follows: it begins by checking that the signature σ is valid, and aborts if not. Next, it computes $x' = f^{-1}(c_1)$ using the trapdoor t, obtaining an encryption *witness* x'. Then the decrypter "recomputes" the ciphertext to check that it is well formed, by checking that $c_1 = f(x')$ and $c_2 = g(vk, x')$, and aborting if not. Finally, it outputs the message $m' = h(x') \oplus c_3$.

The proof of security follows a hybrid two key-type argument, although without zero knowledge due to the recovery of the encryption witness. The key steps of the proof involve a hybrid experiment in which the ABO lossy branch b^* is instead set to $b^* = vk^*$, where vk^* is a verification key that will eventually appear in the challenge ciphertext. In the next step of the hybrid, the decryption oracle decrypts using the trapdoor for the ABO function g, rather than lossy function f. The decryption oracle will thus be able to decrypt successfully for *all branches but one*, namely, the vk^* branch — but by unforgeability of the signature scheme, any query involving vk^* will have an invalid signature and can be rejected out of hand. The final step of the hybrid involves replacing the injective function f with a lossy one. At this point, we observe that in the challenge ciphertext, both components $c_1 = f(x)$ and $c_2 = g(vk^*, x)$ lose information about x. We can therefore show that the hash function h extracts a uniform and independent pad, which conceals the encrypted message *statistically* and implies that even an unbounded adversary has only negligible advantage.

We conclude this summary with a few remarks about our construction. First, we note that in practice one would likely use our techniques as a public-key key extraction mechanism (KEM) where the key would be derived from the hash function as $h(x)$. Second, while our system falls outside of the NIZK paradigm, we do use some techniques that are reminiscent of previous work. Our construction and proof use a two-key method [29], where during hybrid experiments the simulator uses one key to decrypt the ciphertext, while it participates in a distinguishing experiment related to the other key. The major difference is that in the NIZK paradigm, the distinguishing experiment is over a *ciphertext* corresponding to the other key, and well-formedness is guaranteed by the NIZK. In constrast, our simulation participates in a distinguishing experiment on the other key *itself*, and checks validity simply by reencrypting. Additionally, our use of a one-time verification key to specify a branch of the ABO function is similar to the method of Canetti, Halevi, and Katz [10] for constructing CCA-secure encryption from Identity-Based Encryption. Finally, we point out that our decryption algorithm does not recover the *entire* randomness of the ciphertext, because it does not recover the randomness used to generate the signing key or the signature itself. This is a minor technical point, as the decryption algorithm does in fact recover all the randomness that is used to conceal the message itself.

1.3 Realizing Lossy TDFs

We now sketch our basic framework for constructing lossy trapdoor functions. The framework is based on any (semantically secure) cryptosystem having a few special properties, which we later instantiate based on various concrete number theoretic assumptions.

A function f (whether injective or lossy) on $\{0,1\}^n$ will be specified by an entry-wise encryption of some $n \times n$ matrix $\mathbf{M}$, under a cryptosystem that is *additively homomorphic* (and has some other special properties, as described below). To evaluate $f(x)$, we view the input $x \in \{0,1\}^n$ as an n-dimensional bit vector $\mathbf{x}$, and compute an (entry-wise) encryption of the linear product $\mathbf{x} \cdot \mathbf{M}$ using the homomorphism.

For an injective function, we let the trapdoor be the decryption key for the cryptosystem, and let the encrypted matrix $\mathbf{M}$ be the identity $\mathbf{I}$ (more generally, we can let $\mathbf{M}$ be any invertible matrix). The function is therefore injective and invertible with the trapdoor, because $f(x)$ is an entry-wise encryption of $\mathbf{x} \cdot \mathbf{I} = \mathbf{x}$, which can be decrypted to recover each bit of x.

A lossy function is very similar, except we encrypt the all-zeros matrix $\mathbf{M} = \mathbf{0}$. Then for *every* input $\mathbf{x}$, the function evaluates to an entry-wise encryption of the all-zeros vector. However, this alone is not enough to ensure lossiness, because the output ciphertexts still carry some internal *randomness* that might leak information about the input. Therefore we need some additional ideas to control the behavior of this randomness.

We rely on two other special properties of the cryptosystem. First, we require that it remains secure to *reuse randomness* when encrypting under different keys. Second, we require that the homomorphic operation *isolates* the randomness, i.e., that the randomness in the output ciphertext depends only on the randomness of the input ciphertexts (and not, say, on the key or the encrypted messages). For example, many "natural" cryptosystems are also homomorphic with respect to the *randomness* of the ciphertexts, which suffices for our purposes.

With these two properties, we encrypt the matrix $\mathbf{M}$ in a special way. Each column j of the matrix is associated with a different key pk_j; the trapdoor is the set of corresponding decryption keys. Across each row i, we encrypt entry $m_{i,j}$ under key pk_j using the *same randomness* r_i (and using fresh randomness for each row). By assumption, it is secure to reuse randomness in this way, so the entries of the matrix remain hidden. Additionally, because the homomorphism isolates randomness, all the ciphertexts in the output vector $f(x)$ are encrypted under the same randomness (which depends only on $r_1, \ldots, r_n$ and x).

When $\mathbf{M} = \mathbf{I}$, we can still invert the function (given the trapdoor) by decrypting each bit of x. When $\mathbf{M} = \mathbf{0}$, the function output is always a vector of encrypted zero messages, *where each entry is encrypted under the same randomness.* Therefore the number of possible outputs is bounded by the number of different random strings that can arise. By choosing n so that 2^n (the number of inputs) is significantly larger than the number of random strings, we can guarantee that the function is lossy.

We remark that the above summary hides many technical difficulties that arise when working with existing lattice-based cryptosystems [2, 33, 34]. The primary issue is that these systems incorporate two kinds of randomness: a "linear" component that can be securely reused, and a "noise" component that cannot. The noise components affect the correctness of decryption after performing homomorphic operations. More importantly, they lead to additional information leakage in the lossy case. Surmounting these difficulties requires careful trade-offs and some additional techniques; see Section 6 for details.

1.4 Lossy Trapdoors in Context

It is informative to consider lossy trapdoors in the context of previous systems. A crucial technique in using lossy trapdoors is that security is typically demonstrated via indistinguishability arguments over the *public parameters* of a scheme, as opposed to arguments about the scheme's outputs.

Our approach can be contrasted with the oblivious transfer (OT) paradigm of Even, Goldreich, and Lempel [18] and the efficient OT protocols of Naor and Pinkas [28]. In the EGL paradigm, one can construct (semi-honest) oblivious transfer protocols from any public key cryptosystem in which one can (perfectly) sample a public key without knowing the corresponding decryption key (or its equivalent). In the OT protocol, one of the messages is encrypted under such a public key, thereby hiding it computationally from the receiver. While lossy TDFs can be employed in the EGL framework, a crucial difference is that the security properties are reversed: using lossy TDFs, one can sample a lossy public key that is only *computationally* indistinguishable from a "real" one, but messages encrypted under the lossy key are *statistically* hidden.[2]

Another interesting comparison is to the simulation techniques used to prove CCA security from Identity-Based Encryption (IBE) [38] that were first pioneered by Canetti, Halevi, and Katz [10] and improved in later work [8, 9, 7]. Our construction and simulation are inspired by techniques from these works, but also differ in some important ways. In the constructions based on IBE, the simulator is able to generate secret keys for all identities except for one special identity ID^*, and can therefore answer decryption queries in the CCA game; the decrypter does not recover the encryption witness, but the integrity of the ciphertexts is checked (in the concrete instantiations) using a bilinear map. In the simulation, ID^* is hidden *perfectly* by the public key, while the challenge ciphertext encrypted under ID^* hides its message only *computationally.* By contrast, in our simulation using lossy and all-but-one TDFs, the decrypter checks ciphertext integrity by recovering the witness. Additionally, the security properties in the simulation are once again reversed: the lossy branch b^* is hidden only computationally by the public key, but the challenge ciphertext hides its message perfectly.

2 Background

Here we review some standard notation and cryptographic definitions. We also give relevant background relating to entropy of distributions and extraction of nearly uniform strings.

[2]We point out that the final OT protocol in [28] (based on DDH) actually does lose information about the message itself, but it does not aspire to any trapdoor property, as this is not needed for OT.

2.1 Notation

We let $\mathbb{N}$ denote the natural numbers. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. For positive functions $f = f(n), g = g(n)$, we say that $f = O(g)$ if $\lim_{n\to\infty} f(n)/g(n) = c$ for some fixed constant c. We say that $f = o(g)$ if $\lim_{n\to\infty} f(n)/g(n) = 0$. We say that $f = \Omega(g)$ (or $f = \omega(g)$) if $g = O(f)$ (or $g = o(f)$, respectively). We say that $f = \tilde{O}(g)$ if $f = O(g \log^c n)$ for some fixed constant c. We let $\text{poly}(n)$ denote an unspecified function $f(n) = O(n^c)$ for some constant c.

The security parameter will be denoted by λ throughout the paper (except in Section 6, where we use d). We let $\mathsf{negl}(\lambda)$ denote some unspecified function $f(\lambda)$ such that $f = o(\lambda^{-c})$ for *every* fixed constant c, saying that such a function is *negligible* (in λ). We say that a probability is *overwhelming* if it is $1 - \mathsf{negl}(\lambda)$.

Let $\mathcal{X} = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ denote two ensembles of probability distributions indexed by λ. Given an algorithm $\mathcal{D}$, define its *advantage* in distinguishing between $\mathcal{X}$ and $\mathcal{Y}$ as $\left|\Pr_{x\leftarrow X_\lambda}[D(1^\lambda, x) = 1] - \Pr_{y\leftarrow Y_\lambda}[D(1^\lambda, y) = 1]\right|$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are *computationally indistinguishable* if the advantage of any probabilistic polynomial-time algorithm $\mathcal{D}$ is $\mathsf{negl}(\lambda)$.[3]

2.2 Trapdoor Functions

Here we review a standard definition of a collection of injective trapdoor functions. For generality, let $n = n(\lambda) = \text{poly}(\lambda)$ denote the input length of the trapdoor functions in terms of the security parameter. A collection of trapdoor functions is given by a tuple of (possibly probabilistic) poly-time algorithms (S, F, F^{-1}) having the following properties:

1. *Easy to sample, compute, and invert with trapdoor:* $S(1^\lambda)$ outputs (s, t) where s is a function index and t is its trapdoor, $F(s, \cdot)$ computes an injective function $f_s(\cdot)$ over the domain $\{0,1\}^n$, and $F^{-1}(t, \cdot)$ computes $f_s^{-1}(\cdot)$.

2. *Hard to invert without trapdoor:* for any PPT inverter $\mathcal{I}$, the probability that $\mathcal{I}(1^\lambda, s, f_s(x))$ outputs x is at most negligible, where the probability is taken over the choice of $(s, t) \leftarrow S(1^\lambda)$, $x \leftarrow \{0,1\}^n$, and $\mathcal{I}$'s randomness.

2.3 Secure Cryptosystems

Here we review the standard definitions of a cryptosystem and notions of security, including chosen-plaintext (CPA) and chosen-ciphertext (CCA). A public key cryptosystem consists of three (randomized) algorithms that are modeled as follows:

- $\mathcal{G}(1^\lambda)$ outputs a public key pk and secret key sk.

- $\mathcal{E}(pk, m)$ takes as input a public key pk and a message $m \in \mathcal{M}$ (where $\mathcal{M}$ is some fixed message space, possibly depending on λ), and outputs a ciphertext c.

- $\mathcal{D}(sk, c)$ takes as input a private key sk and a ciphertext c, and outputs a message $m \in \mathcal{M} \cup \{\perp\}$, where $\perp$ is a distinguished symbol indicating decryption failure.

[3]For simplicity, throughout the paper we opt for security under *uniform* adversaries. This can be easily adapted to a non-uniform treatment without affecting the results.

The standard completeness requirement is that for any $(pk, sk) \leftarrow \mathcal{G}(1^\lambda)$ and any $m \in \mathcal{M}$, we have $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$. This notion can be relaxed to hold with overwhelming probability over the choice of (pk, sk).

A basic notion of security for a public key cryptosystem is indistinguishability under a chosen plaintext attack. This notion, called CPA security, is defined by the following game between a challenger and an adversary $\mathcal{A}$. Both are given the security parameter 1^λ as input. The challenger generates a key pair $(pk, sk) \leftarrow \mathcal{G}(1^\lambda)$, and gives pk to $\mathcal{A}$. $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$. The challenger picks a random $b \in \{0, 1\}$, lets $c^* \leftarrow \mathcal{E}(pk, m_b)$, and gives c^* to $\mathcal{A}$. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins if $b' = b$. The *advantage* of $\mathcal{A}$ as $\left|\Pr[b' = b] - \frac{1}{2}\right|$. A cryptosystem is said to be CPA-secure if every probabilistic polynomial-time adversary $\mathcal{A}$ has only negligible (in λ) advantage in the CPA game.

A much stronger and commonly accepted notion of security for a public key cryptosystem is that of indistinguishability under an adaptive chosen ciphertext attack. This notion, called CCA security, is defined by a game similar to CPA game, where the adversary $\mathcal{A}$ is additionally given access to an oracle $\mathcal{O}$ that computes $\mathcal{D}(sk, \cdot)$, with the exception that $\mathcal{O}$ returns $\perp$ if queried on the challenge ciphertext c^*. The advantage of $\mathcal{A}$ in the game is $\left|\Pr[b' = b] - \frac{1}{2}\right|$. A cryptosystem is said to be CCA-secure if every probabilistic polynomial-time adversary $\mathcal{A}$ has only negligible advantage in the CCA game.

2.4 Strongly Unforgeable Signatures

Here we review standard definitions of signature schemes and a security notion called *strong* unforgeability. A signature system consists of three algorithms Gen, Sign, and Ver:

- $\mathsf{Gen}(1^\lambda)$ outputs a verification key vk and a signing key sk_σ.
- $\mathsf{Sign}(sk_\sigma, m)$ takes as input a signing key sk_σ and a message $m \in \mathcal{M}$ (where $\mathcal{M}$ is some fixed message space, possibly depending on λ) and outputs a signature σ.
- $\mathsf{Ver}(vk, m, \sigma)$ takes as input a verification key vk, a message $m \in \mathcal{M}$, and a signature σ, and output either 0 or 1.

The standard completeness requirement is that for any $(vk, sk_\sigma) \leftarrow \mathsf{Gen}(1^\lambda)$ and any $m \in \mathcal{M}$, we have $\mathsf{Ver}(vk, m, \mathsf{Sign}(sk_\sigma, m)) = 1$. This can be relaxed to hold with overwhelming probability over the choice of (vk, sk_σ).

We will use a security notion known as *strong* existential unforgeability under an adaptive chosen message attack (SEU-CMA), which is defined using the following game between a challenger and an adversary $\mathcal{A}$ (both given the security parameter 1^λ):

Setup. The challenger generates a key pair $(vk, sk_\sigma) \leftarrow \mathcal{G}(1^\lambda)$, and gives vk to $\mathcal{A}$.

Signature Queries. $\mathcal{A}$ adaptively issues signature queries $m_i \in \mathcal{M}$. The challenger responds by giving $\sigma_i \leftarrow \mathsf{Sign}(sk_\sigma, m)$ to $\mathcal{A}$.

Output. $\mathcal{A}$ outputs a pair (m, σ). The adversary wins if $\mathsf{Ver}(vk, m, \sigma) = 1$ and (m, σ) is not among the pairs (m_i, σ_i) generated during the query phase.

We define the advantage of an adversary $\mathcal{A}$ to be the probability that $\mathcal{A}$ wins the above game, taken over the randomness of the challenger and the adversary. We say that a signature scheme is SEU-CMA if every probabilistic polynomial time adversary $\mathcal{A}$ has only negligible advantage in the above game. In addition, we say that a signature scheme is *one-time* SEU-CMA if any PPT adversary that is limited to a *single* signature query has only negligible advantage.

Strongly unforgeable signatures can be constructed from one-way functions [22], and more efficiently from collision-resistant hash functions [26]. Both primitives are implied by the worst-case hardness of standard lattice problems (see, e.g., [1, 23]).

2.5 Hashing

A family of functions $\mathcal{H} = \{h_i : D \to R\}$ from a domain D to range R is called *pairwise independent* [39] if, for every *distinct* $x, x' \in D$ and every $y, y' \in R$,

$$\Pr_{h \leftarrow \mathcal{H}}[h(x) = y \text{ and } h(x') = y'] = 1/|R|^2 .$$

The family $\mathcal{H}$ is called *universal* if, for every *distinct* $x, x' \in D$, $\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(x')] = 1/|R|$. Pairwise independence is a strictly stronger property than universality. Families satisfying either notion can be efficiently constructed and evaluated.

We say that PPT algorithms $(S_{\text{crh}}, F_{\text{crh}})$ give a collection of *collision-resistant* hash functions from length $\ell(\lambda)$ to length $\ell'(\lambda) < \ell(\lambda)$ if (1) $S_{\text{crh}}(1^\lambda)$ outputs a function index i, (2) $F_{\text{crh}}(i, \cdot)$ computes a (deterministic) function $H_i : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\ell'(\lambda)}$, and (3) for every every PPT adversary $\mathcal{A}$, $\mathcal{A}(1^\lambda, i)$ outputs distinct $x, x' \in \{0,1\}^{\ell(\lambda)}$ such that $H_i(x) = H_i(x')$ with only negligible probability (over the choice of $i \leftarrow S_{\text{crh}}(1^\lambda)$).

2.6 Extracting Randomness

Here we review a few concepts related probability distributions and extracting uniform bits from weak random sources.

The *statistical distance* between two random variables X and Y having the same domain $\mathcal{D}$ is $\Delta(X, Y) = \frac{1}{2}\sum_{v \in \mathcal{D}} |\Pr[X = v] - \Pr[Y = v]|$. We say that two variables are ϵ-close if their statistical distance is at most ϵ.

The min-entropy of a random variable X is $H_\infty(X) = -\lg(\max_x \Pr[X = x])$. In our applications, the variable X will often be correlated with another variable Y whose value will be known to the adversary; this of course decreases the entropy of X. We will find it convenient to use a definition of Dodis *et al.* [14], which captures the remaining unpredictability of X conditioned on the value of Y, called *average min-entropy*:

$$\tilde{H}_\infty(X|Y) := -\lg\left(\mathop{\mathrm{E}}_{y \leftarrow Y}\left[2^{-H_\infty(X|Y=y)}\right]\right).$$

The average min-entropy corresponds exactly to the optimal probability of predicting of X, given knowledge of Y. The following useful lemma was proved in [14]:

Lemma 2.1. *If Y has 2^r possible values and Z is any random variable, then $\tilde{H}_\infty(X|(Y, Z)) \geq H_\infty(X|Z) - r$.*

In our cryptosystems we will need to derive truly uniform bits from a weakly random source X. In general, this can be done using a *randomness extractor* (see Shaltiel's survey for details [37]). However, for our purposes we will not need any of the more sophisticated extractor constructions;

pairwise independent hash functions [39] will suffice, and in fact interact particularly well with the notion of average min-entropy. We will use the following lemma from [14]:

Lemma 2.2. *Let X, Y be random variables such that $X \in \{0,1\}^n$ and $\tilde{H}_\infty(X|Y) \geq k$. Let $\mathcal{H}$ be a family of pairwise independent hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$. Then for the random variable $h \leftarrow \mathcal{H}$, we have*

$$\Delta((Y, h, h(X)), (Y, h, U_\ell)) \leq \epsilon$$

as long as $\ell \leq k - 2\lg(1/\epsilon)$.

3 Lossy and All-But-One Trapdoor Functions

3.1 Lossy TDFs

Here we define lossy trapdoor functions. Define the following quantities as functions of the security parameter: $n(\lambda) = \text{poly}(\lambda)$ represents the input length of the function and $k(\lambda) \leq n(\lambda)$ represents the *lossiness* of the collection (we often omit the dependence on λ). For convenience, we also define the *residual* leakage $r := n - k$. Then a collection of (n, k)-*lossy trapdoor functions* is given by a tuple of (possibly probabilistic) poly-time algorithms $(S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ having the properties below. For notational convenience, we define the algorithms $S_{\text{inj}}(\cdot) := S_{\text{ltdf}}(\cdot, 1)$ and $S_{\text{loss}}(\cdot) := S_{\text{ltdf}}(\cdot, 0)$.

1. *Easy to sample an injective function with trapdoor:* $S_{\text{inj}}(1^\lambda)$ outputs (s, t) where s is a function index and t is its trapdoor, $F_{\text{ltdf}}(s, \cdot)$ computes an injective function $f_s(\cdot)$ over the domain $\{0,1\}^n$, and $F_{\text{ltdf}}^{-1}(t, \cdot)$ computes $f_s^{-1}(\cdot)$.

2. *Easy to sample a lossy function:* $S_{\text{loss}}(1^\lambda)$ outputs $(s, \perp)$ where s is a function index, and $F_{\text{ltdf}}(s, \cdot)$ computes a function $f_s(\cdot)$ over the domain $\{0,1\}^n$ whose image has size $\leq 2^r = 2^{n-k}$.

3. *Hard to distinguish injective from lossy:* the first outputs of $S_{\text{inj}}(1^\lambda)$ and $S_{\text{loss}}(1^\lambda)$ are computationally indistinguishable. More formally, let X_λ denote the distribution of s from $S_{\text{inj}}(1^\lambda)$, and let Y_λ denote the distribution of s from $S_{\text{loss}}(1^\lambda)$. Then $\{X_\lambda\} \overset{c}{\approx} \{Y_\lambda\}$.

Note that we make no explicit requirement that an injective function be hard to invert. As we will see in Lemma 3.1, this is implied by combination of the lossiness and indistinguishability properties.

For our lattice-based constructions we will need to consider a slightly relaxed definition of lossy TDFs, which we call *almost-always* lossy TDFs. Namely, *with overwhelming probability*, the output s of S_{inj} describes an injective function f_s that F_{ltdf}^{-1} inverts correctly on all values. In other words, there is only a negligible probability (over the choice of s) that $f_s(\cdot)$ is not injective or that $F_{\text{ltdf}}^{-1}(t, \cdot)$ incorrectly computes $f_s^{-1}(\cdot)$ for some input. Furthermore, we only require that the lossy function f_s generated by S_{loss} has image size at most 2^r with overwhelming probability. In general, the function sampler cannot check these conditions because they refer to "global" properties of the generated function. The use of almost-always lossy TDFs does not affect security in our applications (e.g., CCA-secure encryption) because the adversary will have no control over the generation of trapdoor/lossy functions. Therefore the potential advantage of the adversary due to sampling an improper function will be bounded by a negligible quantity.

3.2 All-But-One TDFs

For some of our applications, it will be more convenient to work with a richer abstraction that we call *all-but-one* (ABO) trapdoor functions. In an ABO collection, each function has several *branches*. Almost all the branches are injective trapdoor functions (having the same trapdoor value), except for one branch which is lossy. The lossy branch is specified as a parameter to the function sampler, and its value is hidden (computationally) in the resulting function index.

We retain the same notation for n, k, r as above, and also let $\mathcal{B} = \{B_\lambda\}_{\lambda\in\mathbb{N}}$ be a collection of sets whose elements represent the branches. Then a collection of (n,k)-*all-but-one* trapdoor functions with branch collection $\mathcal{B}$ is given by a tuple of (possibly probabilistic) poly-time algorithms $(S_{\text{abo}}, G_{\text{abo}}, G^{-1}_{\text{abo}})$ having the following properties:

1. *Sampling a trapdoor function with given lossy branch:* for any $b^* \in B_\lambda$, $S_{\text{abo}}(1^\lambda, b^*)$ outputs (s,t), where s is a function index and t is its trapdoor.

   For any $b \in B_\lambda$ distinct from b^*, $G_{\text{abo}}(s,b,\cdot)$ computes an injective function $g_{s,b}(\cdot)$ over the domain $\{0,1\}^n$, and $G^{-1}_{\text{abo}}(t,b,\cdot)$ computes $g^{-1}_{s,b}(\cdot)$.

   Additionally, $G_{\text{abo}}(s,b^*,\cdot)$ computes a function $g_{s,b^*}(\cdot)$ over the domain $\{0,1\}^n$ whose image has size at most $2^r = 2^{n-k}$.

2. *Hidden lossy branch:* for any $b_0^*, b_1^* \in B_\lambda$, the first output s_0 of $S_{\text{abo}}(1^\lambda, b_0^*)$ and the first output s_1 of $S_{\text{abo}}(1^\lambda, b_1^*)$ are computationally indistinguishable.

Just as with lossy TDFs, we also need to consider a relaxed definition of ABO trapdoor functions, also called *almost-always.* Specifically, the injective, invertible, and lossy properties need only hold with overwhelming probability over the choice of the function index s. For similar reasons, using an almost-always ABO collection will not affect security in our applications.

3.3 Basic Relations

Lossy and ABO trapdoor functions are equivalent for appropriate choices of parameters; we briefly sketch this equivalence here. First, suppose we have a collection of (n,k)-ABO TDFs having branch set $B = \{0,1\}$ (without loss of generality). Then we can construct a collection of (n,k)-lossy TDFs as follows: S_{inj} samples an ABO function having lossy branch $b^* = 1$ (retaining the trapdoor), and S_{loss} samples an ABO function having lossy branch $b^* = 0$. The evaluation algorithm F_{ltdf} always computes the ABO function on branch $b = 0$. The inverter F^{-1}_{ltdf} can invert any injective function (with the trapdoor) because it is evaluated on a non-lossy branch.

We now sketch the converse implication. Suppose that $(S_{\text{ltdf}}, F_{\text{ltdf}}, F^{-1}_{\text{ltdf}})$ gives a collection of (n,k)-lossy TDFs. We can construct an (n,k)-ABO collection having branch set $B = \{0,1\}$ as follows: the generator $S_{\text{abo}}(1^\lambda, b^*)$ outputs $(s,t) = ((s_0,s_1),t)$ where $(s_0,t_0) \leftarrow S_{\text{ltdf}}(1^\lambda, b^*)$, $(s_1,t_1) \leftarrow S_{\text{ltdf}}(1^\lambda, 1-b^*)$, and $t = t_{1-b^*}$. The evaluation algorithm on index $s = (s_0,s_1)$, branch b and value x outputs $F_{\text{ltdf}}(s_b, x)$. The inversion algorithm on trapdoor t, branch $b = 1-b^*$, and input y outputs $F^{-1}_{\text{ltdf}}(t,y)$. It is straightforward to verify the required properties of this construction.

Additionally, an ABO collection for branch set $B = \{0,1\}$ and input length n having residual leakage $r = n-k$ implies an ABO collection for branch set $B = \{0,1\}^\ell$ and the same input length n having residual leakage $\ell \cdot r$ (this is therefore only interesting when the original leakage $r < n/\ell$). The main idea is to generate, for desired lossy branch $b^* \in \{0,1\}^\ell$, ℓ individual functions having lossy branches b_i^* (i.e., the ith bit of b^*). The evaluation algorithm on branch $b \in \{0,1\}^\ell$ and input

x outputs the value of each corresponding function on branch b_i and x. Then when $b \neq b^*$, the branches differ on some bit i, and x can be recovered from the corresponding function value (given the trapdoor). When $b = b^*$, then all ℓ functions are lossy and the total number of possible outputs is at most $(2^r)^\ell$.

3.4 Primitives Implied by Lossy TDFs

We now show that lossy TDFs (having appropriate parameters) are sufficient for black-box constructions of other important cryptographic primitives, such as standard (injective) trapdoor functions and collision-resistant hash functions.

3.4.1 Trapdoor Functions

Our first result shows that the *injective* functions from a lossy collection are indeed trapdoor functions in the standard sense (i.e., easy to invert with a trapdoor, and hard to invert otherwise).

Lemma 3.1. *Let $(S_{ltdf}, F_{ltdf}, F_{ltdf}^{-1})$ give a collection of (n, k)-lossy trapdoor functions with $k = \omega(\log \lambda)$. Then $(S_{inj}, F_{ltdf}, F_{ltdf}^{-1})$ give a collection of injective trapdoor functions. (The analogous result applies for almost-always collections.)*

Proof. By definition, $f_s(\cdot) = F_{\text{ltdf}}(s, \cdot)$ is injective for any s generated by S_{inj}, and F_{ltdf}^{-1} inverts $f_s(\cdot)$ given the trapdoor t. Therefore the completeness conditions hold.

Suppose by way of contradiction that $\mathcal{I}$ is a PPT inverter, i.e., that $\mathcal{I}(1^\lambda, s, f_s(x))$ outputs x with nonnegligible probability over the choice of $(s, t) \leftarrow S_{\text{inj}}(1^\lambda)$, $x \leftarrow \{0,1\}^n$, and $\mathcal{I}$'s randomness. We use $\mathcal{I}$ to build a distinguisher $\mathcal{D}$ between injective functions (those generated by S_{inj}) and lossy ones (those generated by S_{loss}).

$\mathcal{D}$ works as follows: on input a function index s, choose $x \leftarrow \{0,1\}^n$ and compute $y = F_{\text{ltdf}}(s, x)$. Let $x' \leftarrow \mathcal{I}(s, y)$. If $x' = x$, output "injective," otherwise output "lossy."

We now analyze $\mathcal{D}$. First, if s is generated by $S_{\text{inj}}(1^\lambda)$, then by assumption on $\mathcal{I}$, we have $x' = x$ with nonnegligible probability, and $\mathcal{D}$ outputs "injective." Now, suppose s is generated by $S_{\text{loss}}(1^\lambda)$. Then the probability (over the choice of s and x) that even an unbounded $\mathcal{I}$ predicts x is given by the average min-entropy of x conditioned on $(s, f_s(x))$, i.e., the prediction probability is at most $2^{-\tilde{H}_\infty(x|(s,f_s(x)))}$. Because $f_s(\cdot)$ takes at most 2^{n-k} values, Lemma 2.1 implies $\tilde{H}_\infty(x|(s, f_s(x))) \geq H_\infty(x|s) - (n-k) = n - (n-k) = k$. Because $k = \omega(\lg \lambda)$, the probability that $\mathcal{I}(s, y)$ outputs x, and $\mathcal{D}$ outputs "injective," is $\mathsf{negl}(\lambda)$. We conclude that $\mathcal{D}$ distinguishes injective functions from lossy ones, a contradiction. $\square$

3.4.2 Collision-Resistant Hashing

Our next result shows how to construct collision-resistant hash functions from a lossy TDF collection. The construction is quite simple: the hash function $H(x) := h(f(x))$, where f is an injective function from a lossy TDF collection, and h is selected from a *universal* family of hash functions. For an appropriate output length of the universal family, the function H will shrink its input. For appropriate values of the lossiness, finding collisions will imply the ability to distinguish injective functions from lossy ones. The main idea behind the security proof is the following: in a real instance of the function $H = h \circ f$, all collisions must occur in the "outer" application of h, because f is injective. Now consider an alternate construction in which we replace f by a *lossy* function f'.

Then for appropriate settings of the parameters, the function h will (with overwhelming probability) have *no collisions* on the image of f', because the image is small. Therefore all collisions in the alternate construction must occur in the "inner" application of f'. We can therefore distinguish between injective and lossy functions by whether a given collision of H occurs in its outer or inner part. We now proceed more formally.

Assume without loss of generality that the input length $n(\lambda) = \lambda$ simply equals the security parameter. Let $(S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ give a collection of (n,k)-lossy trapdoor functions $\{f_s : \{0,1\}^n \to \mathcal{R}\}$ having arbitrary range $\mathcal{R}$ and residual leakage $r = n - k \leq \rho n$ for some constant $\rho < 1/2$. (An almost-always family would also suffice.) Let $\mathcal{H} = \{h_i : \mathcal{R} \to \{0,1\}^{\kappa n}\}$ be a *universal* family of hash functions where $\kappa = 2\rho + \delta < 1$ for some appropriate constant $\delta \in (0, 1-2\rho)$.[4] (A pairwise independent family would be sufficient for this purpose, but is not necessary.)

The algorithms for the collection of collision-resistant hash functions are as follows:

- Generator $S_{\text{crh}}(1^\lambda)$ chooses $(s,t) \leftarrow S_{\text{inj}}(1^\lambda)$ and disposes of t. It also chooses $h \leftarrow \mathcal{H}$. The index of the generated hash function is $i = (s,h)$.
- Evaluator $F_{\text{crh}}(i,x)$ on index $i = (s,h)$ and input $x \in \{0,1\}^n$ outputs $h(F_{\text{ltdf}}(s,x)) \in \{0,1\}^{\kappa n}$.

Lemma 3.2. *The algorithms (S_{crh}, F_{crh}) described above give a collection of collision-resistant hash functions from $\{0,1\}^n$ to $\{0,1\}^{\kappa n}$.*

Proof. Let $\mathcal{C}$ be an adversary in the collision-finding game for the collection we described. Specifically, $\mathcal{C}$ takes an index $i = (s,h)$ and outputs a supposed collision $x, x' \in \{0,1\}^n$. Let E be the event that the output x, x' is a valid collision. Let E' be the event that x, x' is a valid collision *and* $F_{\text{ltdf}}(s,x) \neq F_{\text{ltdf}}(s,x')$. In the real game, because $F_{\text{ltdf}}(s,\cdot)$ is injective, the events E and E' are equivalent.[5] Then it will suffice to show that $p_0 = \Pr[E']$ in the real game is negligible, via an alternate game.

The alternate game proceeds as follows: $\mathcal{C}$ is given an index $i = (s,h)$ where s is instead generated by S_{loss}, and h is chosen as above. Then by indistinguishability of lossy and injective functions, $p_1 = \Pr[E']$ in the alternate game is only negligibly different from p_0. We now show that p_1 is negligible (even if $\mathcal{C}$ is *unbounded*).

Fix the s chosen in the alternate game, and let $\mathcal{I} = F_{\text{ltdf}}(s, \{0,1\}^n)$ be the image of the lossy function. By lossiness, $|\mathcal{I}| \leq 2^{\rho n}$. Now consider any fixed distinct pair $y, y' \in \mathcal{I}$: by universality of $\mathcal{H}$, we have $\Pr_h[h(y) = h(y')] \leq 2^{-\kappa n}$. Summing over all the (at most) $2^{2\rho n}$ such pairs via a union bound, we see that

$$\Pr_h[\exists \text{ distinct } y, y' \in \mathcal{I} : h(y) = h(y')] \leq 2^{(2\rho - \kappa)n} = 2^{-\delta n} = \mathsf{negl}(\lambda).$$

Now consider the event E' in the alternate game: for x, x' to be a valid collision and $y = F_{\text{ltdf}}(s,x)$ and $y' = F_{\text{ltdf}}(s,x')$ to be distinct requires $h(y) = h(y')$. By above, the probability of such an event is negligible, and we are done. $\square$

We conclude by noting that an alternate construction, in which s is generated by S_{loss} instead of S_{inj}, also gives a collection of collision-resistant hash functions. This construction might even

[4]Technically, we require one family $\mathcal{H}_\lambda$ of hash functions for each value of the security parameter λ.

[5]In the almost-always case, comparable events are equivalent if we add the constraint that $F_{\text{ltdf}}(s,\cdot)$ is actually injective, which fails with negligible probability.

seem more natural, because $F_{\text{ltdf}}(s,\cdot)$ can be seen as "compressing" its input into a small image (of possibly long strings), followed by a "smoothing" step in which h maps the image to a set of short strings. The proof is symmetric to the one above, with the event E' redefined to require that x, x' be a valid hash collision and that $F_{\text{ltdf}}(s,x) = F_{\text{ltdf}}(s,x')$. Then in the real game (the "lossy" case), events E and E' are equivalent except when h contains a collision on the image $\mathcal{I}$; in the alternate game (the "injective" case), event E' never occurs.

Finally, we note that these constructions do *not* require the trapdoor property of lossy TDFs in either the construction or the security proof. Therefore, it is possible to construct collision-resistant hash functions simply from a collection of *lossy functions*, a weaker primitive whose injective functions need not have trapdoors.

4 Cryptosystems and More

Our main goal is to construct a cryptosystem that is secure against chosen ciphertext attacks, using lossy and ABO trapdoor functions. We start in Section 4.1 with a simple construction of a cryptosystem that is secure against chosen-plaintext attacks, which illuminates some of the main ideas behind the main CCA-secure construction. In Section 4.2 we use the CPA-secure cryptosystem for constructing oblivious transfer against semi-honest adversaries and secure multiparty computation. Finally, we conclude in Section 4.3 with our CCA-secure construction and its proof of security.

4.1 CPA-Secure Construction

We first describe our basic CPA-secure cryptosystem. All of the parameters in the system will depend upon the security parameter λ; for notational convenience we will often omit this explicit dependence.

Let $(S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ give a collection of (n,k)-lossy trapdoor functions. (Almost-always lossy TDFs are also sufficient.) Let $\mathcal{H}$ be a family of pairwise independent hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, where $\ell \leq k - 2\lg(1/\epsilon)$ for some negligible $\epsilon = \mathsf{negl}(\lambda)$. Our cryptosystem will have message space $\{0,1\}^\ell$.

- **Key generation.** $\mathcal{G}(1^\lambda)$ first generates an injective trapdoor function: $(s,t) \leftarrow S_{\text{inj}}(1^\lambda)$. It also chooses a hash function $h \leftarrow \mathcal{H}$.

  The public key $pk = (s,h)$ consists of the injective function index and the hash function. The secret key $sk = (t,h)$ consists of the trapdoor and the hash function.

- **Encryption.** $\mathcal{E}$ takes as input (pk, m) where $pk = (s,h)$ is a public key and $m \in \{0,1\}^\ell$ is the message.

  It first chooses $x \leftarrow \{0,1\}^n$ uniformly at random. The ciphertext is $c = (c_1, c_2)$, where

  $$c_1 = F_{\text{ltdf}}(s,x), \quad c_2 = m \oplus h(x).$$

- **Decryption.** $\mathcal{D}$ takes as input (sk, c) where $sk = (t,h)$ is the secret key and $c = (c_1, c_2)$ is a ciphertext.

  The decryption algorithm computes $x = F_{\text{ltdf}}^{-1}(t, c_1)$ and outputs $c_2 \oplus h(x)$.

Theorem 4.1. *The algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ described above are a cryptosystem secure against chosen plaintext attack.*

Proof. We prove security by defining two hybrid experiments Game_1, Game_2 where Game_1 is the real chosen plaintext game. Then we show that the adversary's views in Game_1 and Game_2 are indistinguishable. Finally we show unconditionally that the adversary's advantage in Game_2 is only negligibly better than 1/2. It follows that the cryptosystem is CPA-secure.

A chosen-plaintext attack game is entirely specified by two algorithms that keep joint state: (1) an algorithm Setup that on input 1^λ, outputs a public key pk, and (2) an algorithm $\mathsf{Challenge}$ that on input of two messages $m_0, m_1 \in \{0,1\}^\ell$ from the adversary, outputs a challenge ciphertext c^*. In both of our games, $\mathsf{Challenge}(m_0, m_1)$ will be the same as in the real game: it chooses a uniform bit $b \leftarrow \{0,1\}$ and outputs $c^* \leftarrow \mathcal{E}(pk, m_b)$.

The only difference between the two games is in Setup. In Game_1, $\mathsf{Setup}(1^\lambda)$ proceeds as in the real CPA game, outputting $pk = (s, z)$ where $(s, t) \leftarrow S_{\text{inj}}(1^\lambda)$ and $z \leftarrow \{0,1\}^d$ is a uniform seed for the extractor. In Game_2, $\mathsf{Setup}(1^\lambda)$ generates a lossy function instead, outputting $pk = (s, z)$ where $(s, \perp) \leftarrow S_{\text{loss}}(1^\lambda)$ and $z \leftarrow \{0,1\}^d$.

It is straightforward to show that the adversary's views in the two games are indistinguishable, using the indistinguishability of injective and lossy functions. We now show that in Game_2, any adversary (even an unbounded one) has only negligible advantage, unconditionally.

Consider the random variable x used by $\mathcal{E}$ in Game_2, which is independent of pk. Because $f_s(\cdot) = F_{\text{ltdf}}(s, \cdot)$ has image size at most 2^{n-k}, by Lemma 2.1 we have $\tilde{H}_\infty(x|(c_1, pk)) \geq H_\infty(x|pk) - (n-k) = k$. Therefore by Lemma 2.2 and the hypothesis that $\ell \leq k - 2\lg(1/\epsilon)$, we have that $h(x)$ is ϵ-close to uniform (conditioned on the rest of the view of the adversary). It follows that $c_2 = m_b \oplus h(x)$ is ϵ-close to uniform, independent of the challenge bit b (and the rest of the view). We conclude that the adversary has negligible advantage in Game_2, and we are done. □

4.2 Interlude: Oblivious Transfer and MPC

One interesting property of our CPA scheme is that it can be used to create an oblivious transfer protocol in a manner that roughly follows the EGL paradigm [18]. Suppose we have a CPA-secure cryptosystem that allows sampling a public key in two different but indistinguishable ways: first, in the normal way so that the corresponding decryption key is known, and second, in an "oblivious" way so that messages encrypted under the public key remain semantically secure even to the sampler. Then the following is an ℓ-out-of-m (semi-honest) oblivious transfer protocol: the receiver generates ℓ public keys normally (with decryption keys) and $m - \ell$ public keys obliviously, and delivers all m public keys to the sender so that the normal public keys correspond to the ℓ desired messages. The sender encrypts each of the m messages under the corresponding public key, and returns the m ciphertexts, of which the receiver can decrypt exactly the desired ℓ.

In our CPA-secure construction, one can sample a public key obliviously by generating a lossy function rather than an injective one, letting $(s, \perp) \leftarrow S_{\text{loss}}(1^\lambda)$. By arguments from the proof of Theorem 4.1, public keys sampled in this way are (computationally) indistinguishable from normal ones, and messages encrypted under such keys are (unconditionally) secret. We remark that in the context of the OT protocol, these security properties are a reversal of the usual case in the EGL paradigm (using, e.g., trapdoor permutations), where the receiver's security is unconditional and the sender's security is computational.

Using zero-knowledge proofs (which can be based on any one-way function [24]) one can obtain an OT protocol secure against malicious adversaries and, more generally, secure multiparty computation [25]. Using our realization of lossy TDFs from Section 6, we obtain OT and MPC based solely on worst-case lattice assumptions.

We remark that this construction of OT from lossy TDFs is primarily of theoretical interest, because semi-honest OT protocols can be based directly on existing lattice-based [2, 33, 34] and DDH-based cryptosystems. In these cryptosystems, one can sample a (malformed) "lossy" public key (indistinguishable from a valid public key), whose ciphertexts carry no information about the encrypted messages.

4.3 CCA-Secure Construction

We now describe our CCA-secure cryptosystem.

Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a strongly unforgeable signature scheme where the public verification keys are in $\{0,1\}^v$. Let $(S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ give a collection of (n,k)-lossy trapdoor functions, and let $(S_{\text{abo}}, G_{\text{abo}}, G_{\text{abo}}^{-1})$ give a collection of (n,k')-ABO trapdoor functions having branches $B_\lambda = \{0,1\}^v$ (which is equal to the space of signature verification keys).[6] We require that the total lossiness $k + k' \geq n + \kappa$ for some positive κ.

Let $\mathcal{H}$ be a family of pairwise independent hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, where $\ell \leq \kappa - 2\lg(1/\epsilon)$ for some negligible $\epsilon = \mathsf{negl}(\lambda)$. Our cryptosystem will have message space $\{0,1\}^\ell$.

- **Key generation.** $\mathcal{G}(1^\lambda)$ first generates an injective trapdoor function: $(s,t) \leftarrow S_{\text{inj}}(1^\lambda)$. It then generates an ABO trapdoor function having lossy branch 0^v: $(s',t') \leftarrow S_{\text{abo}}(1^\lambda, 0^v)$. Finally, it chooses a hash function $h \leftarrow \mathcal{H}$.

  The public key consists of the two function indices and the hash function:

  $$pk = (s, s', h).$$

  The secret decryption key consists of the two trapdoors, along with the public key:

  $$sk = (t, t', pk).$$

  (In practice, the ABO trapdoor t' may be discarded, but we retain it here for convenience in the security proof.)

- **Encryption.** $\mathcal{E}$ takes as input (pk, m) where $pk = (s, s', h)$ is a public key and $m \in \{0,1\}^\ell$ is the message.

  It first generates a keypair for the one-time signature scheme: $(vk, sk_\sigma) \leftarrow \mathsf{Gen}(1^\lambda)$. It then chooses $x \leftarrow \{0,1\}^n$ uniformly at random. It computes

  $$c_1 = F_{\text{ltdf}}(s, x), \quad c_2 = G_{\text{abo}}(s', vk, x), \quad c_3 = m \oplus h(x).$$

  Finally, it signs the tuple (c_1, c_2, c_3) as $\sigma \leftarrow \mathsf{Sign}(sk_\sigma, (c_1, c_2, c_3))$.

  The ciphertext c is output as

  $$c = (vk, c_1, c_2, c_3, \sigma).$$

[6]Almost-always lossy and ABO TDFs are also sufficient.

- **Decryption.** $\mathcal{D}$ takes as input (sk, c) where $sk = (t, t', pk = (s, s', h))$ is the secret key and $c = (vk, c_1, c_2, c_3, \sigma)$ is a ciphertext.

  The decryption algorithm first checks that $\mathsf{Ver}(vk, (c_1, c_2, c_3), \sigma) = 1$; if not, it outputs $\perp$. It then computes $x = F_{\mathrm{ltdf}}^{-1}(t, c_1)$, and checks that $c_1 = F_{\mathrm{ltdf}}(s, x)$ and $c_2 = G_{\mathrm{abo}}(s', vk, x)$; if not, it outputs $\perp$.

  Finally, it outputs $m = c_3 \oplus h(x)$.

Theorem 4.2. *The algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ described above are a cryptosystem secure against adaptive chosen ciphertext attack.*

4.4 Proof

First we argue the correctness of the cryptosystem. Consider decryption of some properly generated ciphertext $c = (vk, c_1, c_2, c_3, \sigma)$ of a message m. By completeness of the one-time signature, $\mathsf{Ver}(vk, (c_1, c_2, c_3), \sigma) = 1$. The function $f_s(\cdot) = F_{\mathrm{ltdf}}(s, \cdot)$ is injective (with overwhelming probability, if the lossy TDF is almost-always), therefore $F_{\mathrm{ltdf}}^{-1}(t, c_1) = x$, where x is the randomness used in the encryption. By construction, $c_1 = F_{\mathrm{ltdf}}(s, x)$ and $c_2 = G_{\mathrm{abo}}(s', vk, x)$. Therefore the decryptor outputs $c_3 \oplus h(x) = m \oplus h(x) \oplus h(x) = m$.

We prove security by first describing a sequence of games $\mathrm{Game}_1, \ldots, \mathrm{Game}_5$, where Game_1 is the real chosen ciphertext attack game. Then we show that for all $i = 1, \ldots, 4$, Game_i and Game_{i+1} are indistinguishable (either computationally, or in some cases statistically). Finally, we make an unconditional argument that even an unbounded adversary must have negligible advantage in Game_5. It follows that the cryptosystem is CCA-secure.

We now define the sequence of games we use to prove security. A chosen ciphertext attack game is entirely specified by three algorithms (which keep joint state) that interact with the adversary in the manner described in the definition of the CCA game:

- Setup. On input 1^λ, outputs a public key pk.
- $\mathsf{Decrypt}$. On input a ciphertext c from the adversary, outputs an $m \in \{0,1\}^\ell \cup \{\perp\}$.
- $\mathsf{Challenge}$. On input two messages $m_0, m_1 \in \{0,1\}^\ell$ from the adversary, outputs a challenge ciphertext c^*.

When referring to an implementation of these algorithms in a specific game i, we use a subscript i, e.g., Setup_1.

Before defining these algorithms for the individual games, we define two "global" aspects of the algorithms that will remain the same in all the games. First, $\mathsf{Setup}(1^\lambda)$ will always first choose a one-time signature keypair $(vk^*, sk_\sigma^*) \leftarrow \mathsf{Gen}(1^\lambda)$, and will then proceed as we define below. Second, the $\mathsf{Challenge}(m_0, m_1)$ will always choose a bit $b \leftarrow \{0,1\}$ and output a ciphertext $c^* \leftarrow \mathcal{E}(pk, m_b)$, but with one small modification to the operation of $\mathcal{E}$: instead of generating a one-time signature keypair (vk, sk_σ) on its own in its first step, it uses $(vk, sk_\sigma) = (vk^*, sk_\sigma^*)$. We stress that $\mathsf{Challenge}$ operates this way in *all* the games we define, hence we will not specify it further.

When making these changes to the real CCA game, the view of the adversary remains identical, because $\mathsf{Challenge}$ is called only once. We make these changes merely for the convenience of having vk^* defined throughout both query phases, which will aid the analysis.

Game$_1$: Algorithms Setup_1 and $\mathsf{Decrypt}_1$ are identical to those in the CCA2 game given in Section 2.3, with the above-noted changes. That is, $\mathsf{Setup}_1(1^\lambda)$ calls $(pk, sk) \leftarrow \mathcal{G}(1^\lambda)$ and outputs pk; $\mathsf{Decrypt}_1(c)$ calls $m \leftarrow \mathcal{D}(sk, c)$ and outputs m.

In particular, note that $\mathcal{G}$ chooses the ABO lossy branch to be 0^v, and $\mathcal{D}$ inverts c_1 using the injective function trapdoor t.

Game$_2$: In this game, $\mathsf{Setup}_2 = \mathsf{Setup}_1$. The only change is to $\mathsf{Decrypt}_2$, which is defined as follows: on input a ciphertext $c = (vk, c_1, c_2, c_3, \sigma)$, if $vk = vk^*$ (as chosen by Setup_2), then output $\perp$. Otherwise return $\mathsf{Decrypt}_1(c)$. (Note that by defining vk^* in Setup, this new rule is well defined during both query phases.)

Game$_3$: In this game, $\mathsf{Decrypt}_3 = \mathsf{Decrypt}_2$. The only change is to Setup_3, in which the ABO function is chosen to have a lossy branch $b^* = vk^*$ rather than 0^v. Formally, in $\mathcal{G}$ we replace $(s', t') \leftarrow S_{\text{abo}}(1^\lambda, 0^v)$ with $(s', t') \leftarrow S_{\text{abo}}(1^\lambda, vk^*)$.

Note that $\mathsf{Decrypt}_3$ still decrypts using the injective function trapdoor t.

Game$_4$: In this game, $\mathsf{Setup}_4 = \mathsf{Setup}_3$. The only change is to $\mathsf{Decrypt}_4$, in which decryption is now done using the ABO trapdoor t'. Formally, in $\mathcal{D}$ we replace $x = F_{\text{ltdf}}^{-1}(t, c_1)$ with $x = G_{\text{abo}}^{-1}(t', vk, c_2)$.

Note that $\mathsf{Decrypt}_4$ still first rejects if $vk = vk^*$ (as in $\mathsf{Decrypt}_2$), and performs all the consistency checks of $\mathcal{D}$.

Game$_5$: In this game, $\mathsf{Decrypt}_5 = \mathsf{Decrypt}_4$. The only change is to Setup_5, in which we replace the injective function with a lossy one. Formally, in $\mathcal{G}$ we replace $(s, t) \leftarrow S_{\text{inj}}(1^\lambda)$ with $(s, t) \leftarrow S_{\text{loss}}(1^\lambda)$.

We now state and prove a sequence of claims that establish the main theorem.

Claim 4.3. *Game$_1$ and Game$_2$ are computationally indistinguishable, given the one-time strong existential unforgeability of the signature scheme.*

Proof. We begin by observing that Game$_1$ and Game$_2$ behave equivalently unless an event F happens, which is that the adversary $\mathcal{A}$ makes a legal (i.e., not equal to c^*) decryption query of the form $c = (vk = vk^*, c_1, c_2, c_3, \sigma)$, where $\mathsf{Ver}(vk, (c_1, c_2, c_3), \sigma) = 1$. We show that event F happens with negligible probability.

Consider a simulator $\mathcal{S}$ that mounts a (one-time) chosen message attack against the signature scheme as follows: on input vk generated by $\mathsf{Gen}(1^\lambda)$, it emulates Setup by letting $vk^* = vk$ and letting $(pk, sk) \leftarrow \mathcal{G}(1^\lambda)$, and gives pk to $\mathcal{A}$. Upon any decryption query from $\mathcal{A}$ of the form $c = (vk = vk^*, c_1, c_2, c_3, \sigma)$ such that $\mathsf{Ver}(vk, (c_1, c_2, c_3), \sigma) = 1$, $\mathcal{S}$ immediately outputs $((c_1, c_2, c_3), \sigma)$ as a forgery and returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{S}$ returns $m \leftarrow \mathcal{D}(sk, c)$ to $\mathcal{A}$.

When $\mathcal{A}$ asks to be challenged on two messages $m_0, m_1 \in \{0,1\}^\ell$, $\mathcal{S}$ chooses $b \leftarrow \{0,1\}$ and creates the challenge ciphertext $c^* = (vk^*, c_1^*, c_2^*, c_3^*, \sigma^*)$ by running $\mathcal{E}(pk, m_b)$ except that the signature σ^* is generated by querying the signing oracle on the message (c_1^*, c_2^*, c_3^*).

It is clear that $\mathcal{S}$ simulates Game$_2$ perfectly to $\mathcal{A}$. We now show that event F happens if and only if $\mathcal{S}$ outputs a valid forgery. If F happens during the first query phase, then $\mathcal{S}$ outputs a valid signature without making any queries, which is a forgery. If F happens during the second query phase on a query $c = (vk^*, c_1, c_2, c_3, \sigma)$, then because $c \neq c^*$ we must have either $(c_1, c_2, c_3) \neq$

(c_1^*, c_2^*, c_3^*) or $\sigma \neq \sigma^*$. In either case, $\mathcal{S}$'s output $((c_1, c_2, c_3), \sigma)$ is unequal to its single signature query $((c_1^*, c_2^*, c_3^*), \sigma^*)$, and hence is a forgery.

Because the signature scheme is one-time strongly unforgeable, we conclude that F happens with negligible probability. □

Claim 4.4. *Game$_2$ and Game$_3$ are computationally indistinguishable, given the security of the ABO TDF collection.*

Proof. We prove this claim by describing a simulator algorithm $\mathcal{S}^{\mathcal{O}}(1^\lambda)$ that has access to an oracle $\mathcal{O}$ that takes two inputs $b_0^*, b_1^* \in B_\lambda$. $\mathcal{S}$ will simulate Game$_2$ (resp., Game$_3$) perfectly, given an oracle that returns an index s' where $(s', t') \leftarrow S_{\text{abo}}(1^\lambda, b_0^*)$ (resp., $(s', t') \leftarrow S_{\text{abo}}(1^\lambda, b_1^*)$). By the hidden lossy branch property, the two oracles return computationally indistinguishable outputs, and hence the claim follows.

The simulator $\mathcal{S}^{\mathcal{O}}(1^\lambda)$ operates by implementing Setup, Decrypt, and Challenge. Setup is implemented in a manner similar to Game$_2$ by choosing $(vk^*, sk_\sigma^*) \leftarrow \mathsf{Gen}(1^\lambda)$, $(s, t) \leftarrow S_{\text{inj}}(1^\lambda)$, $h \leftarrow \mathcal{H}$, but by letting $s' \leftarrow \mathcal{O}(0^v, vk^*)$. The public key is output as $pk = (s, s', h)$. We stress that the s' part of the public key comes from the oracle. We also point out that $\mathcal{S}$ knows the injective trapdoor t, but does not know the trapdoor t' corresponding to s'.

Decrypt is implemented just as in Game$_2$ and Game$_3$. Note that the only secret information Decrypt needs to operate is t, which the simulator knows. Likewise, Challenge is implemented just as in all the games.

One can now verify that $\mathcal{S}$ perfectly simulates Game$_2$ or Game$_3$, depending on $\mathcal{O}$. □

Claim 4.5. *Game$_3$ and Game$_4$ are perfectly equivalent (if the lossy and ABO collections are both perfect) or statistically close (if either the lossy or ABO TDF collection is almost-always).*

Proof. The only difference between Game$_3$ and Game$_4$ is in the implementation of Decrypt. We will show that Decrypt is perfectly equivalent in the two games (with overwhelming probability, if the trapdoor systems are almost-always).

First note that if the trapdoor systems are almost-always, the injective, invertible, and lossy properties hold for all inputs simultaneously, with overwhelming probability over the choice of s and s'. From now on we assume that this is so.

We now analyze Decrypt in both games on an arbitrary query $c = (vk, c_1, c_2, c_3, \sigma)$. Since Decrypt always outputs $\perp$ in both games if $vk = vk^*$, we may assume that $vk \neq vk^*$. Additionally, both implementations check that $c_1 = F_{\text{ltdf}}(s, x) = f_s(x)$ and $c_2 = G_{\text{abo}}(s', vk, x) = g_{s',vk}(x)$ for some x that they compute (in different ways), and output $\perp$ if not. Therefore we may assume that such an x exists. It suffices to show that such an x is unique, and that both implementations of Decrypt find it.

In both games, (s, t) is generated by $S_{\text{inj}}(1^\lambda)$ and (s', t') is generated by $S_{\text{abo}}(1^\lambda, vk^*)$. Therefore $f_s(\cdot)$ and $f_{s',vk}(\cdot)$ are both injective (in the latter case, because $vk \neq vk^*$). Therefore there is a unique x such that $(c_1, c_2) = (f_s(x), f_{s',vk}(x))$. Decrypt$_3$ finds that x by computing $F_{\text{ltdf}}^{-1}(t, c_1)$, while Decrypt$_4$ finds it by computing $G_{\text{abo}}^{-1}(t', c_2)$. □

Claim 4.6. *Game$_4$ and Game$_5$ are computationally indistinguishable, given the security of the lossy TDF collection.*

Proof. We prove this claim by describing a simulator algorithm $\mathcal{S}(1^\lambda, s)$ that simulates Game$_4$ perfectly if s was generated by $S_{\text{inj}}(1^\lambda)$, and that simulates Game$_5$ perfectly if s was generated by $S_{\text{loss}}(1^\lambda)$. By the indistinguishability of injective and lossy functions, the claim follows.

The simulator $\mathcal{S}(1^\lambda, s)$ operates by implementing Setup, $\mathsf{Decrypt}$, and $\mathsf{Challenge}$. Setup is implemented in a manner similar to Game$_4$ by choosing $(vk^*, sk^*_\sigma) \leftarrow \mathsf{Gen}(1^\lambda)$, $(s', t') \leftarrow S_{\text{abo}}(1^\lambda, vk^*)$, and $h \leftarrow \mathcal{H}$. The public key is output as $pk = (s, s', h)$. We stress that the s part of the public key comes from $\mathcal{S}$'s input. We also point out that $\mathcal{S}$ knows the ABO trapdoor t', but does not know the trapdoor t corresponding to s (if it even exists).

$\mathsf{Decrypt}$ is implemented just as in Game$_4$ and Game$_5$. Note that the only secret information $\mathsf{Decrypt}$ needs to operate is t', which the simulator knows. Likewise, $\mathsf{Challenge}$ is implemented just as in all the games.

It is easy to see that $\mathcal{S}$ perfectly simulates Game$_4$ or Game$_5$, depending on whether s is the index of an injective or lossy function (respectively). $\square$

Claim 4.7. *No (unbounded) adversary has more than a negligible advantage in Game$_5$.*

Proof. Fix all the randomness (including the adversary's) in Game$_5$, except for the choice of the hash function h and the randomness x used by $\mathsf{Challenge}$ when producing the challenge ciphertext $c^* \leftarrow \mathcal{E}(pk, m_b)$. We will show that conditioned on the values of the challenge ciphertext components c_1^*, c_2^* and the value of the fixed randomness, the value $h(x)$ is a nearly uniform and independent "one-time pad," and therefore the adversary has negligible advantage in guessing which message m_b was encrypted. (To be completely rigorous, we could define another game in which the component $c_3^* = h(x) \oplus m_b$ is replaced by string chosen uniformly and independent of all other variables, including b.) It follows by averaging over the choice of the fixed randomness that the adversary has negligible advantage in the full game.

We first observe that $f_s(\cdot) = F_{\text{ltdf}}(s, \cdot)$ and $g_{s',vk^*}(\cdot) = G_{\text{abo}}(s', vk^*, \cdot)$ are lossy functions with image sizes at most 2^{n-k} and $2^{n-k'}$, respectively (for an overwhelming fraction of the fixed randomness, in the almost-always case). Therefore the random variable $(c_1^*, c_2^*) = f(x) = (f_s(x), g_{s',vk^*}(x))$ can take at most $2^{2n-(k+k')} \leq 2^{n-\kappa}$ values by our requirement that $k + k' \geq n + \kappa$.

By Lemma 2.2, we have $\tilde{H}_\infty(x | (c_1^*, c_2^*, h)) \geq H_\infty(x|h) - (n - \kappa)$, which is $n - (n-\kappa) = \kappa$ because x and h are independent. By the hypothesis that $\ell \leq \kappa - 2\lg(1/\epsilon)$, we have that $(c_1^*, c_2^*, h, h(x))$ and $(c_1^*, c_2^*, h, U_\ell)$ are within $\epsilon = \mathsf{negl}(\lambda)$ in statistical distance, and we are done. $\square$

4.5 Discussion

We stress that in all the games, the challenge ciphertext c^* is created in the same way (given the fixed pk). The only difference between games is instead in how the public key is formed and how decryption queries are answered. Of course, changing the public key pk changes the distribution of the challenge ciphertext; the important point is that the simulator always knows all the randomness of the challenge ciphertext it produces. Instead, the simulator does *not* know whether the public key is properly formed, i.e., whether it is lossy or injective. This is in constrast to prior constructions, in which the simulator always produces valid public keys, but does not know the randomness of the challenge ciphertext it produces. This difference is what allows our decryption algorithm to test well-formedness of a ciphertext by recovering randomness.

Another important component of our construction is the use of a (one-time) strongly unforgeable signature scheme. This allows the simulator first to choose the verification key vk^* of the challenge ciphertext, and then to make the public key be lossy at branch vk^*. In particular, the lossy branch is not dependent on the challenge messages m_0, m_1 chosen later by the adversary. Our use of a strongly unforgeable signature scheme is similar to that of Cannetti, Halevi, and Katz [10] in their conversion of IBE schemes to CCA-secure schemes. One possible future direction is to create variations of our scheme using ideas similar to those developed in the IBE literature [10, 8, 7, 9].

5 Realization from DDH-Hard Groups

We now present constructions of lossy TDFs and all-but-one TDFs using groups in which the decisional Diffie-Hellman (DDH) problem is hard. The construction will illustrate our core ideas and will also serve as a template for the lattice-based constructions in the next section.

We begin by giving a brief overview of the DDH problem. Then we show how to build lossy TDFs from DDH-hard groups, and how to extend the construction to build all-but-one TDFs.

5.1 Background

Let $\mathcal{G}$ be a an algorithm that takes as input a security parameter λ and outputs a tuple $\mathbb{G} = (p, G, g)$ where p is a prime, G is a cyclic group of order p, and g is a generator of G.

Our construction will make use of groups for which the DDH problem is believed to be hard. The DDH assumption is that the ensemble $\left\{(\mathbb{G}, g^a, g^b, g^{ab})\right\}_{\lambda\in\mathbb{N}}$ is computationally indistinguishable from $\left\{(\mathbb{G}, g^a, g^b, g^c)\right\}_{\lambda\in\mathbb{N}}$, where $\mathbb{G} = (p, G, g) \leftarrow \mathcal{G}(\lambda)$, and $a, b, c \leftarrow \mathbb{Z}_p$ are uniform and independent.

5.2 Preliminary Tools

For the remainder of this section, we implicitly assume that a group description $\mathbb{G} = (p, G, g) \leftarrow \mathcal{G}(1^\lambda)$ is fixed and known to all algorithms. (In our TDF constructions, this group will be generated by the function sampler S_{ltdf} and made part of the function description.)

An ElGamal-like encryption primitive. First we review a (well-known) variant of the ElGamal cryptosystem, which is additively homomorphic. A secret key is chosen as $z \leftarrow \mathbb{Z}_p$, and the public key is $h = g^z$. To encrypt an $m \in \mathbb{Z}_p$, choose an $r \leftarrow \mathbb{Z}_p$ and create the ciphertext $E_h(m; r) = (g^r, h^r \cdot g^m)$. To decrypt a ciphertext $c = (c_1, c_2)$, output $D_z(c) = \log_g(c_2/c_1^z)$; when c encrypts a bit $m \in \{0, 1\}$ (or any small value m) this discrete logarithm may be computed easily by enumeration. It is well-known (and straightforward to prove) that this cryptosystem is semantically secure under the DDH assumption.

Note that the cryptosystem is additively homomorphic in the following way:

$$E_h(m; r) \odot E_h(m'; r') = E_h(m + m'; r + r'),$$

where $\odot$ denotes coordinate-wise multiplication of ciphertexts. Similarly, for $x \in \mathbb{Z}_p$,

$$E_h(m; r)^x = E_h(mx; rx)$$

where exponentiation of a ciphertext is also coordinate-wise. Finally, we note that *without* even knowing the public key under which a ciphertext was created, one can add any scalar value $v \in \mathbb{Z}_p$ to the underlying plaintext (we will need this only for our ABO construction):

$$\text{Let } c = (c_1, c_2) = E_h(m; r). \quad \text{Then } c \boxplus v := (c_1, c_2 \cdot g^v) = E_h(m + v; r).$$

Encrypting matrices. We now describe a special method for encrypting a matrix $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n \times n}$ and generating a corresponding decryption key. First choose n independent secret/public keypairs $z_j, h_j = g^{z_j}$ for $j \in [n]$ (according to the ElGamal variant above), and n independent exponents $r_i \leftarrow \mathbb{Z}_p$ for $i \in [n]$. The encryption of $\mathbf{M}$ consists of the matrix $\mathbf{C} = (c_{i,j})$ of ciphertexts $c_{i,j} = E_{h_j}(m_{i,j}; r_i)$ for all $i, j \in [n]$. (Note that we need not publish the public keys h_j.) The decryption key is the collection of secret keys z_j for $j \in [n]$.

Note that because every ciphertext in row i uses the same randomness r_i, we can equivalently represent the encrypted matrix somewhat more compactly via matrices $\mathbf{C}_1$ and $\mathbf{C}_2$, where

$$\mathbf{C}_1 = \begin{pmatrix} g^{r_1} \\ \vdots \\ g^{r_n} \end{pmatrix} \qquad \mathbf{C}_2 = \begin{pmatrix} h_1^{r_1} \cdot g^1 & h_2^{r_1} \cdot g^0 & \cdots & h_n^{r_1} \cdot g^0 \\ \vdots & & \ddots & \vdots \\ h_1^{r_n} \cdot g^0 & h_2^{r_n} \cdot g^0 & \cdots & h_n^{r_n} \cdot g^1 \end{pmatrix}$$

Lemma 5.1. *The matrix encryption scheme described above produces indistinguishable ciphertexts under the DDH assumption.*

Proof. Intuitively, the lemma follows for the fact that it is secure to reuse randomness when encrypting under several independent public keys, because given only g^r one can still produce a ciphertext having randomness r if one knows the secret key z. We now proceed more formally.

Let $\mathbf{L} = (\ell_{i,j}), \mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n \times n}$ be any two arbitrary matrices. We first define a set of hybrid experiments $H_0, \ldots, H_{n^2}$. In experiment H_k, the output is a matrix $\mathbf{C} = (c_{i,j})$ chosen in the following way: choose secret/public keypairs $z_j \in \mathbb{Z}_p, h_j = g^{z_j}$ for $j \in [n]$ and exponents $r_i \leftarrow \mathbb{Z}_p$ for $i \in [n]$ as above. Then for the first k pairs $(i, j) \in [n]^2$ (where we order the pairs lexicographically), let $c_{i,j} = E_{h_j}(\ell_{i,j}; r_i)$. For the remaining pairs (i, j), let $c_{i,j} = E_{h_j}(m_{i,j}; r_i)$.

Observe that experiment H_0 produces an encryption of the matrix $\mathbf{L}$ and H_{n^2} produces an encryption of the matrix $\mathbf{M}$. Below we argue that for every $k \in [n]$, experiments H_{k-1} and H_k are computationally indistinguishable. Then because $n = \text{poly}(\lambda)$, H_0 and H_{n^2} are also indistinguishable, and the claim follows.

For any $k \in [n]^2$, let (i^*, j^*) be the lexicographically kth pair in $[n]^2$. Consider the following simulator algorithm S: the input is a public key h^* $[= g^{z^*}]$ from the ElGamal variant and a ciphertext $c^* = (c_1^*, c_2^*)$ $[= E_{h^*}(?; r^*) = (g^{r^*}, g^{r^* z^*} \cdot g^?)]$, where c^* is an encryption (under h^*) of either $\ell_{i,j}$ or $m_{i,j}$. S produces an encrypted matrix $\mathbf{C} = (c_{i,j})$ in the following way. First, for every $j \neq j^*$ it chooses secret/public keys $z_j \leftarrow \mathbb{Z}_p, h_j = g^{z_j}$ as above, and for every $i \neq i^*$ it chooses random exponents $r_i \leftarrow \mathbb{Z}_p$.

For rows $i \neq i^*$, S "encrypts normally." That is, for $i < i^*$ and all $j \in [n]$, let $c_{i,j} = E_{h_j}(\ell_{i,j}; r_i)$; similarly for $i > i^*$ and all $j \in [n]$, let $c_{i,j} = E_{h_j}(m_{i,j}; r_i)$.

For row $i = i^*$, S "encrypts using the secret key z_j." That is, for column $j < j^*$, let

$$c_{i,j} = (c_1^*, (c_1^*)^{z_j} \cdot g^{\ell_{i,j}}) = (g^{r^*}, g^{r^* z_j} \cdot g^{\ell_{i,j}}) = E_{h_j}(\ell_{i,j}; r^*),$$

and similarly for $j > j^*$ (encrypting $m_{i,j}$). Finally, for $i = i^*$ and $j = j^*$, let $c_{i,j} = c^*$.

One can see that S's output is distributed according to either H_{k-1} or H_k, depending on whether c^* was an encryption of $\ell_{i,j}$ or $m_{i,j}$ (respectively). Because these two cases are indistinguishable by the security of the ElGamal variant, so are H_{k-1} and H_k, and we are done. $\square$

5.3 Lossy TDF

We now describe the function generation, evaluation, and inversion algorithms for our lossy TDF.

- *Sampling an injective/lossy function.* The injective function generator $S_{\text{inj}}(1^\lambda)$ first selects $\mathbb{G} = (p, G, g) \leftarrow \mathcal{G}(1^\lambda)$. The function index is a matrix encryption $\mathbf{C}$ (as described above) of the identity $\mathbf{I} \in \mathbb{Z}_p^{n \times n}$ (and implicitly the group description $\mathbb{G}$). The trapdoor information t consists of the the corresponding decryption keys $\mathbf{z}_j$ for $j \in [n]$.

  The lossy function generation algorithm $S_{\text{loss}}(1^\lambda)$ likewise selects $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$. The function index is a matrix encryption $\mathbf{C}$ of $\mathbf{0} \in \mathbb{Z}_p^{n \times n}$ (and $\mathbb{G}$'s description). There is no trapdoor output.

- *Evaluation algorithm.* F_{ltdf} takes as input $(\mathbf{C}, \mathbf{x})$, where $\mathbf{C}$ is a function index (a matrix encryption of some $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n \times n}$) and $\mathbf{x} \in \{0,1\}^n$ is an n-bit input interpreted as a vector. The output is the vector of ciphertexts $\mathbf{y} = \mathbf{x}\mathbf{C}$, where the linear product is interpreted in the natural way using the homomorphic operations of the cryptosystem. By construction of $\mathbf{C}$ and the homomorphic properties of the cryptosystem, we have

$$y_j := \bigodot_{i \in [n]} c_{i,j}^{x_i} \quad = \quad E_{h_j}\left((\mathbf{x}\mathbf{M})_j \; ; \; R := \langle \mathbf{x}, \mathbf{r} \rangle\right),$$

  where $\mathbf{r} = (r_1, \ldots, r_n)$ is the vector of random exponents used to construct $\mathbf{C}$.

  Note that if the function index $\mathbf{C}$ was generated by S_{inj} (i.e., $\mathbf{M} = \mathbf{I}$), we have $y_j = E_{h_j}(x_j; R)$, whereas if $\mathbf{C}$ was generated by S_{loss} (i.e., $\mathbf{M} = \mathbf{0}$) we have $y_j = E_{h_j}(0; R)$. Note also that the randomness R inherent in y_j is the same for all $j \in [n]$; therefore, we may represent $\mathbf{y}$ more compactly using $n+1$ group elements in a manner similar to that for matrix encryption.

- *Inversion algorithm.* F_{ltdf}^{-1} takes as input $(t, \mathbf{y})$ where the trapdoor information t consists of the decryption keys $(\mathbf{z}_1, \ldots, \mathbf{z}_n)$. The output is $\mathbf{x} \in \{0,1\}^n$ where $x_j = D_{\mathbf{z}_j}(y_j)$.

Shorter outputs. Our basic construction takes an n-bit input as a binary string and has an output of n ciphertexts (which can be represented compactly using $n+1$ group elements). We note that it is possible to achieve somewhat shorter output size by parsing the input into messages from a space of size 2^α. In this generalization, function outputs will consist of $\lceil n/\alpha \rceil + 1$ group elements. However, there is a trade-off in the inversion time, as the ElGamal decryption algorithm will need to enumerate over the possible 2^α values. Therefore, this generalization works only for small values of α, i.e., $\alpha = O(\log \lambda)$.

Alternatively, we can also realize a more efficient variant of our construction over $\mathbb{Z}_{N^2}$, where N is the product of two primes chosen at random by the setup algorithm. In this variant, we can use the techniques of Paillier [30] to create an inversion routine that will know the factorization of N and thus has access to an efficient trapdoor for an order N subgroup. The evaluation routine can then process the input in blocks of $\alpha = \lceil \lg(N) \rceil$ bits. The resulting ABO system gives rise to a CCA-secure cryptosystem having comparable performance to that of the Cramer-Shoup [12] projective-hash system when realized with Paillier encryption.

Theorem 5.2. *The algorithms described above give a collection of* $(n, n - \lg p)$*-lossy TDFs under the DDH assumption for* $\mathcal{G}$.

Proof. We have shown invertibility for injective functions via the trapdoor information, and indistinguishability between injective and lossy functions follows by Lemma 5.1. It remains to show the lossiness property.

Recall that for a function generated by S_{loss}, for any input $\mathbf{x}$ the output $\mathbf{y}$ is such that $y_j = E_{h_j}(0; R)$ for some fixed $R \in \mathbb{Z}_p$ (dependent on $\mathbf{x}$) and fixed h_j. Therefore the number of possible function outputs is at most p, the residual leakage r is at most $\lg p$, and the lossiness is $k = n - r \geq n - \lg p$. □

5.4 All-But-One TDF

For a cyclic group of order p, the residual leakage of our lossy TDF is at most $\lg p$. For large enough values of n, we can use a generic transformation from lossy to all-but-one TDFs to obtain an ABO collection with many branches, based on the DDH assumption. However, the generic transformation is rather inefficient. Here we demonstrate a more efficient ABO collection where the number of branches can be as large as p. The construction is an extension of our lossy TDF construction.

Let the set of branches $B_\lambda = [q]$, where q is at most the *smallest* value of p produced by $\mathcal{G}(1^\lambda)$ (we often omit the dependence of B_λ on λ). When a cyclic group $\mathbb{G}$ of order p is clear from context, we interpret a branch value $b \in B$ as a distinct element of $\mathbb{Z}_p$.

- *Sampling an ABO function.* The function generator $S_{\text{abo}}(1^\lambda, b^* \in B)$ first selects $\mathbb{G} = (p, G, g) \leftarrow \mathcal{G}(1^\lambda)$. The function index is a matrix encryption $\mathbf{C}$ of the matrix $-(b^*\mathbf{I}) \in \mathbb{Z}_p^{n \times n}$ (and implicitly the group description $\mathbb{G}$). The trapdoor information t consists of the corresponding decryption keys $\mathbf{z}_j$ for $j \in [n]$, along with the lossy branch value b^*.

- *Evaluation algorithm.* G_{abo} takes as input $(\mathbf{C}, b, \mathbf{x})$ where $\mathbf{C}$ is a function index, $b \in B$ is the desired branch, and $\mathbf{x}$ is an n-bit input interpreted as a vector. The output is the vector of ciphertexts $\mathbf{y} = \mathbf{x}(\mathbf{C} \boxplus b\mathbf{I})$, where the homomorphic scalar addition operation $\boxplus$ applies entry-wise to the matrices, and the linear product $\mathbf{x}$ is interpreted in the same way as in the lossy TDF construction.

  By the homomorphic properties of the encryption and the construction of $\mathbf{C}$, the jth coordinate of $\mathbf{y}$ is
  $$y_j = E_{h_j}\left((b - b^*)x_j \; ; \; R := \langle \mathbf{x}, \mathbf{r} \rangle\right),$$
  where $\mathbf{r} = (r_1, \ldots, r_n)$ is the vector of random coefficients used in the creation of $\mathbf{C}$. Note that if $b = b^*$, each $y_j = E_{h_j}(0; R)$. Also note that as before, the output $\mathbf{y}$ can be compactly represented using $n + 1$ group elements.

- *Inversion algorithm.* G_{abo}^{-1} takes as input $(t, b, \mathbf{y})$ where t is the trapdoor information (decryption keys $\mathbf{z}_j$ for $j \in [n]$ and the lossy branch b^*), $b \neq b^*$ is the evaluated branch, and $\mathbf{y}$ is the function output. G_{abo}^{-1} outputs $\mathbf{x}$ where $x_j = D_{\mathbf{z}_j}(y_j)/(b - b^*)$. Note that y_j can be efficiently decrypted because its plaintext is only one of two values (either 0 or $b - b^*$). Note also that the inversion algorithm is defined only for $b \neq b^*$.

Theorem 5.3. *The algorithms described above give a collection of* $(n, n - \lg p)$*-all-but-one TDFs, under the DDH assumption for* $\mathcal{G}$.

Proof. We have shown invertibility above. The hidden lossy branch property follows by Lemma 5.1. The lossiness property follows from the fact that when $b = b^*$, each $y_j = E_{h_j}(0; R)$ is completely determined by a single value $R \in \mathbb{Z}_p$, of which there are only p possibilities. □

6 Realization from Lattices

Our construction of a lossy TDF based on worst-case lattice problems uses the same basic ideas as our construction based on DDH: using an additively homomorphicf cryptosystem, the function computes an encrypted linear product $\mathbf{xM}$, where in the lossy case we will have $\mathbf{M} = \mathbf{0}$. However, we must overcome additional technical challenges stemming chiefly from the fact that in lattice-based cryptosystems, ciphertexts include extra random "noise" terms. This requires careful trade-offs between the lossy and injective cases: in the lossy case, the noise leaks additional information about which homomorphic operations were performed; in the injective case, the noise determines the amount of recoverable information that can "fit into" a ciphertext, and affects the correctness of decryption after performing homomorphic operations.

6.1 Background

We start by introducing the notation and computational problems that are relevant to this section, for the most part following Regev [34].

For any $x, y \in \mathbb{R}$ with $y > 0$ we define $x \bmod y$ to be $x - \lfloor x/y \rfloor y$. For $x \in \mathbb{R}$, $\lfloor x \rceil = \lfloor x + 1/2 \rfloor$ denotes the nearest integer to x (with ties broken upward). We define $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, i.e., the group of reals $[0, 1)$ with modulo 1 addition.

Probability distributions. The *normal distribution* with mean 0 and variance σ^2 (or standard deviation σ) is the distribution on $\mathbb{R}$ having density function $\frac{1}{\sigma \cdot \sqrt{2\pi}} \exp(-x^2/2\sigma^2)$. It is a standard fact that the sum of two independent normal variables with mean 0 and variances σ_1^2 and σ_2^2 (respectively) is a normal variable with mean 0 and variance $\sigma_1^2 + \sigma_2^2$. We will also need a standard tail inequality: a normal variable with variance σ^2 is within distance $t \cdot \sigma$ (i.e., t standard deviations) of its mean, except with probability at most $\frac{1}{t} \cdot \exp(-t^2/2)$. Finally, it is possible to efficiently sample from a normal variable to any desired level of accuracy.

For $\alpha \in \mathbb{R}^+$ we define Ψ_α to be the distribution on $\mathbb{T}$ of a normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$, reduced modulo 1. For any probability distribution $\phi : \mathbb{T} \to \mathbb{R}^+$ and an integer $q \in \mathbb{Z}^+$ (often implicit) we define its *discretization* $\bar{\phi} : \mathbb{Z}_q \to \mathbb{R}^+$ to be the discrete distribution over $\mathbb{Z}_q$ of the random variable $\lfloor q \cdot X_\phi \rceil \bmod q$, where X_ϕ has distribution ϕ.

For an integer $q \geq 2$ and some probability distribution $\chi : \mathbb{Z}_q \to \mathbb{R}^+$, an integer dimension $d \in \mathbb{Z}^+$ and a vector $\mathbf{z} \in \mathbb{Z}_q^d$, define $A_{\mathbf{z},\chi}$ as the distribution on $\mathbb{Z}_q^d \times \mathbb{Z}_q$ of the variable $(\mathbf{a}, \langle \mathbf{a}, \mathbf{z} \rangle + e)$ where $\mathbf{a} \leftarrow \mathbb{Z}_q^d$ is uniform and $e \leftarrow \chi$ are independent, and all operations are performed in $\mathbb{Z}_q$.

Learning with error (LWE). For an integer $q = q(d)$ and a distribution χ on $\mathbb{Z}_q$, the goal of the *learning with error* problem $\mathsf{LWE}_{q,\chi}$ is to distinguish (with nonnegligible probability) between an oracle that returns independent samples from $A_{\mathbf{z},\chi}$ for some uniform $\mathbf{z} \leftarrow \mathbb{Z}_q^d$, and an oracle that returns independent samples from the uniform distribution on $\mathbb{Z}_q^d \times \mathbb{Z}_q$.

The conjectured hardness of LWE is parameterized chiefly by the dimension d. Therefore in this section, we let d be the security parameter (rather than λ as before), and let all other parameters (e.g., q, α, n, and several others) implicitly be functions of this parameter.

Regev [34] demonstrated that for certain discretized normal error distributions, LWE is as hard as solving several standard *worst-case* lattice problems *using a quantum algorithm.* We state a version of his result here:

Proposition 6.1 ([34]). *Let $\alpha = \alpha(d) \in (0,1)$ and let $q = q(d)$ be a* prime *such that $\alpha \cdot q > 2\sqrt{d}$. If there exists an efficient (possibly quantum) algorithm that solves $\mathsf{LWE}_{q,\bar{\Psi}_\alpha}$, then there exists an efficient* quantum *algorithm for solving the following worst-case lattice problems:*

- *SIVP: In any lattice Λ of dimension d, find a set of d linearly independent lattice vectors of length within at most $\tilde{O}(d/\alpha)$ of optimal.*
- *GapSVP: In any lattice Λ of dimension d, approximate the length of a shortest nonzero lattice vector to within a $\tilde{O}(d/\alpha)$ factor.*

We will define our lossy and ABO trapdoor constructions in relation to the LWE problem, without explicitly taking into account the connection to lattices (or their parameter restrictions). Then in Section 6.5, we will instantiate the parameters appropriately, invoking Proposition 6.1 to ensure security assuming the (quantum) hardness of lattice problems.

6.2 Preliminaries

Encrypting based on LWE. Here we construct a cryptosystem based on the hardness of the LWE problem. The cryptosystem itself is *symmetric key* (not public key) and has certain limited homomorphic properties over a small message space, which will be sufficient for our purposes in constructing lossy TDFs. This basic cryptosystem is similar to Regev's *public key* cryptosystem [34] and its multi-bit variant [27].

The message space of our cryptosystem will be $\mathbb{Z}_p$ for some $p \geq 2$. For every message $m \in \mathbb{Z}_p$, define the "shift" for m to be $c_m = \frac{m}{p} \in \mathbb{T}$. We let χ denote an unspecified error distribution, which we instantiate later.

Except where noted, all operations are performed in $\mathbb{Z}_q$ for some integer $q > p$. The secret key is a uniform $\mathbf{z} \leftarrow \mathbb{Z}_q^d$. To encrypt an $m \in \mathbb{Z}_p$, choose uniform $\mathbf{a} \leftarrow \mathbb{Z}_q^d$ and an *error term* $e \leftarrow \chi$. Define the *rounding error* $u = \lfloor qc_m \rceil - qc_m \in [-1/2, 1/2]$. Then the ciphertext is

$$E_{\mathbf{z}}(m, u; \mathbf{a}, e) := (\mathbf{a}, \langle \mathbf{a}, \mathbf{z} \rangle + qc_m + u + e) \in \mathbb{Z}_q^d \times \mathbb{Z}_q.$$

Note that we treat u as an explicit input to the encryption algorithm even though it is normally derived from m, because it will be convenient to treat $E_{\mathbf{z}}(m, u; \mathbf{a}, e)$ as a well-defined expression even for $u \notin [-1/2, 1/2]$. In cases where u is simply derived from m in the manner described, we will often omit it and write

$$E_{\mathbf{z}}(m; \mathbf{a}, e) := (\mathbf{a}, \langle \mathbf{a}, \mathbf{z} \rangle + \lfloor qc_m \rceil + e).$$

For a ciphertext $c = (\mathbf{a}, c')$, the decryption algorithm $D_{\mathbf{z}}(c)$ computes $t = (c' - \langle \mathbf{a}, \mathbf{z} \rangle)/q \in \mathbb{T}$ and outputs an $m \in \mathbb{Z}_p$ such that $t - c_m \in \mathbb{T}$ is closest to 0 modulo 1. Note that for any ciphertext $c = E_{\mathbf{z}}(m, u; \mathbf{a}, e)$, as long as the *absolute total error* $|e + u| < q/2p$, the decryption $D_{\mathbf{z}}(c)$ is correct (outputs m).

The cryptosystem is homomorphic:

$$E_{\mathbf{z}}(m, u; \mathbf{a}, e) + E_{\mathbf{z}}(m', u'; \mathbf{a}', e') \quad = \quad E_{\mathbf{z}}(m + m', u + u'; \mathbf{a} + \mathbf{a}', e + e')$$

Furthermore, even without knowing the secret key under which a ciphertext was created, one can add any scalar value $v \in \mathbb{Z}_p$ to its plaintext (we will need this property only for our ABO construction). Let $c = (\mathbf{a}, c') = E_{\mathbf{z}}(m, u; \mathbf{a}, e)$, and define $u' = \lfloor qc_v \rceil - qc_v \in [-1/2, 1/2]$. Then

$$c \boxplus v := (\mathbf{a}, c' + \lfloor qc_v \rceil) = E_{\mathbf{z}}(m + v, u + u'; \mathbf{a}, e).$$

Encrypting matrices. We now describe a special extension of the encryption scheme to matrices $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{h \times w}$ of an arbitrary height h and width w.

- **Secret key.** For each column $j \in [w]$, choose independent $\mathbf{z}_j \leftarrow \mathbb{Z}_q^d$. The tuple $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_w)$ forms the secret key.
- **Encryption.** To encrypt a matrix $\mathbf{M} \in \mathbb{Z}_p^{h \times w}$, do the following: for each row $i \in [h]$, choose independent $\mathbf{a}_i \leftarrow \mathbb{Z}_q^d$, forming a matrix $\mathbf{A} \in \mathbb{Z}_q^{h \times d}$ whose ith row is $\mathbf{a}_i$. Generate an error matrix $\mathbf{E} = (e_{i,j}) \in \mathbb{Z}_q^{h \times w}$ by choosing independent error terms $e_{i,j} \leftarrow \chi$. Let $\mathbf{U} = (u_{i,j})$ be a matrix of rounding errors, where $u_{i,j} = \lfloor q c_{m_{i,j}} \rceil - q c_{m_{i,j}} \in [-1/2, 1/2]$.

  The matrix encryption of $\mathbf{M}$ is denoted

  $$\mathbf{C} = (c_{i,j}) = E_{\mathbf{Z}}(\mathbf{M}, \mathbf{U}; \mathbf{A}, \mathbf{E}),$$

  where $c_{i,j} = E_{\mathbf{z}_j}(m_{i,j}, u_{i,j}; \mathbf{a}_i, e_{i,j})$. We omit the $\mathbf{U}$ argument when its value is implied.

  Note that each ciphertext uses an independent error term $e_{i,j}$, but that the randomness $\mathbf{a}_i$ is reused across row i and the secret key $\mathbf{z}_j$ is reused across each column j. The encrypted matrix can be represented more compactly as $(\mathbf{A}, \mathbf{C}')$, where $c'_{i,j} = \langle \mathbf{a}_i, \mathbf{z}_j \rangle + q c_{m_{i,j}} + u_{i,j} + e_{i,j}$.
- **Decryption.** An encrypted matrix $\mathbf{C} = (c_{i,j})$ of size $h' \times w$ (whose width w must match the secret key, but whose height h' can be arbitrary) is decrypted as the matrix $\mathbf{M} = (m_{i,j}) = D_{\mathbf{Z}}(\mathbf{C}) \in \mathbb{Z}_p^{h' \times w}$, where $m_{i,j} = D_{\mathbf{z}_j}(c_{i,j})$.
- **Linear operations.** By the homomorphism of the underlying cryptosystem, all linear operations (addition of ciphertexts, multiplication and addition by scalars) extend naturally to linear operations involving encrypted matrices. For example, say $\mathbf{C} = E_{\mathbf{Z}}(\mathbf{M}, \mathbf{U}; \mathbf{A}, \mathbf{E})$ is an encryption of some $\mathbf{M} \in \mathbb{Z}_p^{h \times w}$. Then for any $\mathbf{x} \in \mathbb{Z}_p^h$,

  $$\mathbf{xC} = E_{\mathbf{Z}}(\mathbf{xM}, \mathbf{xU}; \mathbf{xA}, \mathbf{xE}).$$

  Likewise, if $\mathbf{V} \in \mathbb{Z}_p^{h \times w}$ is a matrix of scalars inducing a matrix of rounding errors $\mathbf{U}'$, then

  $$\mathbf{C} \boxplus \mathbf{V} = E_{\mathbf{Z}}(\mathbf{M} + \mathbf{V}, \mathbf{U} + \mathbf{U}'; \mathbf{A}, \mathbf{E}).$$

Lemma 6.2. *For any height and width $h, w = poly(d)$, the matrix encryption scheme described above produces indistinguishable ciphertexts under the assumption that $\mathsf{LWE}_{q,\chi}$ is hard.*

Proof. It will be convenient in this proof to work with the compact representation $(\mathbf{A}, \mathbf{C}')$ of matrix encryptions. It suffices to show that for any $\mathbf{M} \in \mathbb{Z}_p^{h \times w}$, a proper encryption $E_{\mathbf{Z}}(\mathbf{M}; \mathbf{A}, \mathbf{E})$ of $\mathbf{M}$ is indistinguishable from a "uniform" encryption $E_{\mathbf{Z}}(\mathbf{M}; \mathbf{A}, \mathbf{R})$ where the error matrix $\mathbf{R} \leftarrow \mathbb{Z}_q^{h \times w}$ is uniform, because the latter's two components $(\mathbf{A}, \mathbf{C}')$ are uniform and independent.

We define a set of hybrid experiments $H_0, \ldots, H_w$. In experiment H_k, the output is a (compact) encryption $E_{\mathbf{Z}}(\mathbf{M}; \mathbf{A}, \mathbf{E})$ where the entries in the first k columns of $\mathbf{E}$ are chosen independently from χ, and the remainder are uniform and independent. Observe that experiment H_0 produces a proper encryption of $\mathbf{M}$, while experiment H_w produces a uniform encryption. Below we show that experiments H_{k-1} and H_k are computationally indistinguishable. Then because the number of columns $w = \text{poly}(d)$, the claim follows.

For any $k \in [w]$, consider the following simulator algorithm $S^{\mathcal{O}}$, where $\mathcal{O}$ returns samples either from the distribution $A_{\mathbf{z},\chi}$ for some $\mathbf{z} \leftarrow \mathbb{Z}_q^d$, or from the uniform distribution on $\mathbb{Z}_q^d \times \mathbb{Z}_q$. First, for all $j \neq k$, S chooses independent secret keys $\mathbf{z}_j \leftarrow \mathbb{Z}_q^d$. Then for each $i \in [h]$, S queries $\mathcal{O}$, obtaining a sample $(\mathbf{a}_i, b_i)$. S lets $\mathbf{A}$ be the matrix whose ith row is $\mathbf{a}_i$, and lets $c_{i,k} = b_i + \lfloor q c_{m_{i,k}} \rceil$. Then for all columns $j < k$ and for all $i \in [h]$, S chooses independent error terms $e_{i,j} \leftarrow \chi$; for all columns $j > k$ and for all $i \in [h]$, S chooses uniform and independent error terms $e_{i,j} \leftarrow \mathbb{Z}_q$. For all $j \neq k$ and all $i \in [h]$, S lets $c'_{i,j} = \langle \mathbf{a}_i, \mathbf{z}_j \rangle + \lfloor q c_{m_{i,j}} \rceil + e_{i,j}$. The output is $(\mathbf{A}, \mathbf{C}')$.

Observe that if the samples from $\mathcal{O}$ are uniform, S's output is distributed according to H_{k-1} because the b_i values are uniform. If the samples from $\mathcal{O}$ are drawn from $A_{\mathbf{z},\chi}$, S's output is distributed according to H_k. Under the assumption that $\mathsf{LWE}_{q,\chi}$ is hard, the distributions $A_{\mathbf{z},\chi}$ and U are computationally indistinguishable; therefore, so are H_{k-1} and H_k, and we are done. $\square$

We now show a technical lemma that will be needed for both the correctness and lossiness properties of our lossy TDF construction.

Lemma 6.3. *Let $q \geq 4pn$, let $\alpha \leq 1/(16p(n+g))$ for some positive g, and let $\mathbf{E} = (e_{i,j}) \in \mathbb{Z}_q^{n \times w}$ be an error matrix generated by choosing independent error terms $e_{i,j} \leftarrow \chi = \bar{\Psi}_\alpha$. Then except with probability at most $w \cdot 2^{-g}$ over the choice of $\mathbf{E}$,* every *entry of $\mathbf{xE}$ has absolute value less than $\frac{q}{4p}$ for* all $\mathbf{x} \in \{0,1\}^n$.

Proof. It suffices to show that for each column $\mathbf{e}^T$ of $\mathbf{E}$, $|\langle \mathbf{x}, \mathbf{e} \rangle| < q/4p$ for all $\mathbf{x}$ simultaneously except with probability at most 2^{-g} over the choice of $\mathbf{e}$. The lemma follows by a union bound over all w columns of $\mathbf{E}$.

We will show that for any *fixed* $\mathbf{x} \in \{0,1\}^n$,

$$\Pr_{\mathbf{e}}[|\langle \mathbf{x}, \mathbf{e} \rangle| \geq q/4p] \leq 2^{-(n+g)}.$$

Taking a union bound over all $\mathbf{x} \in \{0,1\}^n$, we can conclude that $|\langle \mathbf{x}, \mathbf{e} \rangle| < q/4p$ for *all* $\mathbf{x} \in \{0,1\}^n$ except with probability at most 2^{-g}.

Now by definition, $e_i = \lfloor q s_i \rceil \bmod q$ where s_i are independent normal variables with mean 0 and variance α^2 for each $i \in [n]$. Then $\langle \mathbf{x}, \mathbf{e} \rangle$ is at most $n/2 \leq q/8p$ away from $q(\langle \mathbf{x}, \mathbf{s} \rangle \bmod 1)$. Therefore it suffices to show that $|\langle \mathbf{x}, \mathbf{s} \rangle| < 1/8p$ except with probability at most $2^{-(n+g)}$.

Because the s_i are independent, $\langle \mathbf{x}, \mathbf{s} \rangle$ is distributed as a normal variable with mean 0 and variance at most $n \cdot \alpha^2 \leq (n+g) \cdot \alpha^2$, hence a standard deviation of at most $\sqrt{n+g} \cdot \alpha$. Then by the tail inequality on normal variables and the hypothesis on α,

$$\Pr_{\mathbf{s}}\left[|\langle \mathbf{x}, \mathbf{s} \rangle| \geq \frac{1}{8p}\right] \leq \Pr_{\mathbf{s}}\left[|\langle \mathbf{x}, \mathbf{s} \rangle| \geq 2\sqrt{n+g} \cdot \left(\sqrt{n+g} \cdot \alpha\right)\right] \leq \frac{\exp(-2(n+g))}{2\sqrt{n+g}} < 2^{-(n+g)}. \quad \square$$

6.3 Lossy TDF

Our construction of a lossy TDF based on LWE uses the same ideas as our construction based on DDH. In particular, evaluating the function will involve computing an encrypted linear product $\mathbf{xM}$, and in the lossy case we will have $\mathbf{M} = \mathbf{0}$. However, additional challenges must be addressed, stemming chiefly from the fact that ciphertexts now include extra error terms that can leak information (e.g., about the homomorphic operations that produced them). The main difficulty is to ensure that (in the injective case) the decrypted plaintexts contain more information than might be leaked (in the lossy case) by the error terms. We will accomplish this by exploiting the entire plaintext space $\mathbb{Z}_p$, rather than $\{0,1\}$ as before. However, doing this properly involves some subtleties.

As a first attempt, we could let the input be a vector $\mathbf{x} \in \mathbb{Z}_p^n$, and specify an injective function by an encryption $\mathbf{C}$ of the identity matrix $\mathbf{I} \in \mathbb{Z}_p^{n\times n}$. The output of the function would be an encryption of $\mathbf{xI} = \mathbf{x}$, which would yield up to $\lg p$ bits of the input per output ciphertext. The problem with this construction is that when computing $\mathbf{xC}$ via the homomorphic operations, the *error terms* of $\mathbf{C}$ are also amplified by the entries of $\mathbf{x}$, which can be as large as p. Therefore, in the lossy case, each output ciphertext might *leak* $\lg p$ bits (or more) via its noise. Therefore, this simple construction does not seem to confer any benefit.

Instead, in our actual construction the *output* uses the entire message space $\mathbb{Z}_p$, but the *input* is still interpreted in binary. This will ensure that the homomorphic operations do not amplify the error terms by too much. Our method uses a special (nonsquare) matrix instead of the identity. Let $\ell = \lfloor \lg p \rfloor$, assume without loss of generality that n is divisible by ℓ, and let $m = n/\ell$. Then we define a "tall and skinny" matrix $\mathbf{B} \in \mathbb{Z}_p^{n\times m}$ as follows: in column $j \in [m]$, the $((j-1)\ell + k)$th entry is $2^{k-1} \in [1,p]$ for $k \in [\ell]$. All other entries of $\mathbf{B}$ are zero. Formally, $\mathbf{B}$ is the tensor (or Kronecker) product $\mathbf{I} \otimes \mathbf{b}$, where $\mathbf{I} \in \mathbb{Z}_p^{m\times m}$ is the identity and $\mathbf{b} = (1, 2, \ldots, 2^{\ell-1})^T \in \mathbb{Z}_p^{\ell\times 1}$ is the column vector containing increasing powers of 2.

This choice of $\mathbf{B}$ is motivated by the following fact: break an input vector $\mathbf{x} \in \{0,1\}^n$ into m chunks of ℓ bits each, and interpret the jth chunk as a value $v_j \in \mathbb{Z}_p$ by reading the chunk as a value in binary notation (least significant bit first). Then each $\mathbf{x} \in \{0,1\}^n$ corresponds to a unique $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{Z}_p^m$. Most important, by our definition of $\mathbf{B}$, we have $\mathbf{xB} = \mathbf{vI} = \mathbf{v}$.

Our injective trapdoor function will be described by a matrix encryption of $\mathbf{B}$. Evaluating the function on $\mathbf{x} \in \{0,1\}^n$ will compute an encrypted product $\mathbf{xB} = \mathbf{v}$. This allows us to recover the entire input by decrypting $\mathbf{v}$ and computing the corresponding $\mathbf{x}$. At the same time, the output consists of only $m = n/\ell$ ciphertexts, which means that in the lossy case, less information is leaked overall via their error terms. We obtain a lossy TDF by ensuring that the amount of information recoverable from each ciphertext (namely, $\ell \approx \lg p$ bits) significantly exceeds the amount of information carried by its error term (which is $\approx \lg n$ bits, due to the noise expansion from the n homomorphic operations).

We now describe the lossy TDF generation, evaluation, and inversion algorithms more formally.

- *Sampling an injective/lossy function.* The injective function generator $S_{\text{inj}}(1^d)$ generates a matrix encryption
$$\mathbf{C} = E_{\mathbf{Z}}(\mathbf{B}, \mathbf{U}; \mathbf{A}, \mathbf{E})$$
(with $\mathbf{Z}$, $\mathbf{U}$, $\mathbf{A}$, and $\mathbf{E}$ chosen as described above), and outputs $\mathbf{C}$ as the function index. The trapdoor information t consists of the secret keys $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_w)$.

  The lossy function generator $S_{\text{loss}}(1^d)$ outputs a matrix encryption
$$\mathbf{C} = E_{\mathbf{Z}}(\mathbf{0}, \mathbf{U}; \mathbf{A}, \mathbf{E})$$
of the all-zeros matrix $\mathbf{0}$. There is no trapdoor output.

- *Evaluation algorithm.* F_{ltdf} takes as input $(\mathbf{C}, \mathbf{x})$ where $\mathbf{C}$ is the function index (an encryption of either $\mathbf{M} = \mathbf{B}$ or $\mathbf{M} = \mathbf{0}$) and $\mathbf{x} \in \{0,1\}^n$ is an n-bit input interpreted as a vector. The output is the vector of ciphertexts $\mathbf{y} = \mathbf{xC}$.

By the homomorphic properties, the output $\mathbf{y}$ is

$$\mathbf{y} = E_{\mathbf{Z}}(\mathbf{xM}, \mathbf{xU}; \mathbf{xA}, \mathbf{xE}).$$

Note that every ciphertext y_j is of the form $(\mathbf{xA}, y'_j) \in \mathbb{Z}_q^d \times \mathbb{Z}_q$, so we may represent $\mathbf{y}$ more compactly using a single copy of $\mathbf{xA} \in \mathbb{Z}_q^d$ and n values from $\mathbb{Z}_q$.

- *Inversion algorithm.* F_{ltdf}^{-1} takes as input $(\mathbf{Z}, \mathbf{y})$, where $\mathbf{Z}$ is the trapdoor information. It computes $\mathbf{v} = D_{\mathbf{Z}}(\mathbf{y}) \in \mathbb{Z}_p^m$, and outputs the unique $\mathbf{x} \in \{0,1\}^n$ such that $\mathbf{v} = \mathbf{xB}$.

Theorem 6.4. *Instantiate the parameters of the above scheme as follows: let $p = n^{c_1}$ for some constant $c_1 > 0$, let $q \in [4pn, O(pn^{c_2})]$ for some constant $c_2 > 1$, let $n = d^{c_3}$ for some constant $c_3 > 1$, and let $\chi = \bar{\Psi}_\alpha$ where $\alpha \leq 1/(32pn)$.*

Then the algorithms described above give a collection of almost-always (n,k)-lossy TDFs under the assumption that $\mathsf{LWE}_{q,\chi}$ is hard, where the residual leakage $r = n - k$ is

$$r \leq \left(\frac{c_2}{c_1} + o(1)\right) \cdot n.$$

Proof. First we show that the inversion algorithm F_{ltdf}^{-1} is correct on all inputs $\mathbf{y} = F_{\text{ltdf}}(\mathbf{C}, \mathbf{x})$, with overwhelming probability over the choice of $\mathbf{C}$ by S_{inj}. As observed above, we have

$$\mathbf{y} = E_{\mathbf{Z}}(\mathbf{v} = \mathbf{xB}, \mathbf{xU}; \mathbf{xA}, \mathbf{xE}).$$

Letting $g = n$ in Lemma 6.3, we have $|(\mathbf{xE})_j| < q/4p$ for every $\mathbf{x}$ and $j \in [m]$, except with probability $m \cdot 2^{-n} = \mathsf{negl}(d)$ over the choice of $\mathbf{E}$. Furthermore, $|(\mathbf{xU})_j| \leq n/2 \leq q/8p$ by the size of $\mathbf{U}$'s entries. Therefore the total error in y_j is $|(\mathbf{xE})_j + (\mathbf{xU})_j| < q/2p$ for all j, hence the decryption $D_{\mathbf{Z}}(\mathbf{y})$ outputs $\mathbf{v}$.

We now analyze the lossiness of a lossy function. For any input $\mathbf{x}$,

$$\mathbf{y} = E_{\mathbf{Z}}(\mathbf{0} = \mathbf{x0}, \mathbf{xU}; \mathbf{xA}, \mathbf{xE}).$$

As in the correctness argument, for every $\mathbf{x}$ and $j \in [m]$ the absolute total error $|(\mathbf{xU})_j + (\mathbf{xE})_j| < q/2p$ (with overwhelming probability over $\mathbf{E}$). Therefore for every $j \in [m]$, y_j is a ciphertext $(\mathbf{xA}, y'_j) \in \mathbb{Z}_q^d \times \mathbb{Z}_q$, where $\mathbf{xA}$ is the same randomness for all j and $y'_j = \langle \mathbf{xA}, \mathbf{z}_j \rangle + 0 + (\mathbf{xU})_j + (\mathbf{xE})_j$ can take at most q/p possible values. Then the total number of outputs of the lossy function is at most $q^d \cdot (q/p)^m$. The logarithm of this quantity is a bound on the residual leakage $r = n - k$:

$$\begin{aligned} r &\leq d \cdot \lg q + m \cdot \lg O(n^{c_2}) \\ &\leq O(n^{1/c_3} \lg n) + m \cdot (O(1) + c_2 \lg n) \\ &\leq o(n) + n \cdot \frac{O(1) + c_2 \lg n}{\lfloor c_1 \lg n \rfloor} \\ &\leq n \cdot \left(\frac{c_2}{c_1} + o(1)\right), \end{aligned}$$

where we have crucially used the fact that $m = n/\lfloor \lg p \rfloor = n/\lfloor c_1 \lg n \rfloor$.

Finally, lossy functions are indistinguishable from injective ones by the security of matrix encryption (Lemma 6.2). $\square$

6.4 All-But-One TDF

Our construction of an all-but-one TDF relies on all the ideas from our prior constructions, but also includes some important technical differences. As always, evaluating the ABO function on an input $\mathbf{x} \in \{0,1\}^n$ involves homomorphically computing an encrypted product $\mathbf{vM}$, where $\mathbf{v} \in \mathbb{Z}_p^m$ corresponds to $\mathbf{x}$ in the manner described above, and $\mathbf{M}$ is some matrix that depends on the branch of the function being evaluated. We will require that $\mathbf{M} = \mathbf{0}$ for the lossy branch, and that $\mathbf{v}$ is recoverable from the product $\mathbf{vM}$ for all other branches.

In our prior ABO construction based on DDH, the matrix $\mathbf{M}$ was some multiple $(b - b^*)\mathbf{I}$ of the identity, for $b, b^* \in \mathbb{Z}_p$. Because the matrices $\mathbf{M}$ had entries from an exponentially large group $\mathbb{Z}_p$, the construction supported exponentially many branches.

In the current setting, our matrices $\mathbf{M}$ have entries from a smaller group $\mathbb{Z}_p$, where $p = \text{poly}(d)$. Therefore, simply using multiples of $\mathbf{I}$ will not yield enough branches. Instead, we generalize to matrices $\mathbf{M}$ having full row rank (i.e., all their rows are linearly independent), which suffices for recovering $\mathbf{v}$ from the product $\mathbf{vM}$. We use a family of pairwise independent hash functions to generate the matrix $\mathbf{M}$ for the desired branch, and arrange for $\mathbf{M} = \mathbf{0}$ on the lossy branch. To ensure (with overwhelming probability) that the $\mathbf{M}$s for *all* other branches have full row rank, we use matrices having a few more columns. This decreases the lossiness of the function (because the output consists of more ciphertexts, which leak information via their error terms), but not by a significant amount.

The construction. As above, let $\ell = \lfloor \lg p \rfloor$, assume ℓ divides n and let $m = n/\ell$, and let $\mathbf{b} = (1, 2, \ldots, 2^{\ell-1})^T \in \mathbb{Z}_p^{\ell \times 1}$ be the column vector containing increasing powers of 2. For any $\mathbf{x} \in \{0,1\}^n$ we associate a unique $\mathbf{v} \in \mathbb{Z}_p^m$ (and vice versa) in the manner described in the previous section. Our construction will crucially use the fact that $\mathbf{x}(\mathbf{M} \otimes \mathbf{b}) = \mathbf{vM}$ for any $\mathbf{M} \in \mathbb{Z}_p^{m \times w}$.

Let the branch set $B = B_d = \mathbb{Z}_p^t$ for some sufficiently large t we set later, and let w denote the width of the encrypted matrices, which will depend on the other parameters and the desired lossiness. Let $\mathcal{H}$ denote a family of pairwise independent hash functions from $B = \mathbb{Z}_p^t$ to $\mathbb{Z}_p^{m \times w}$.

- *Sampling an ABO function.* The function generator $S_{\text{abo}}(1^d, b^* \in B)$ first chooses a hash function $h \leftarrow \mathcal{H}$. The function index consists of h and a matrix encryption

  $$\mathbf{C} = E_{\mathbf{Z}}(-h(b^*) \otimes \mathbf{b}, \mathbf{U}; \mathbf{A}, \mathbf{E})$$

  (where $\mathbf{Z}$, $\mathbf{U}$, $\mathbf{A}$, and $\mathbf{E}$ are chosen in the usual way). The trapdoor information consists of the secret keys $\mathbf{Z}$, the lossy branch value b^*, and the hash function h.

- *Evaluation algorithm.* G_{abo} takes as input $((h, \mathbf{C}), b, \mathbf{x})$ where $(h, \mathbf{C})$ is the function index, $b \in B$ is the desired branch, and $\mathbf{x} \in \{0,1\}^n$ is an n-bit input interpreted as a vector. The output is

  $$\mathbf{y} := \mathbf{x}(\mathbf{C} \boxplus (h(b) \otimes \mathbf{b})).$$

  Let $\mathbf{H} = h(b) - h(b^*)$. Then by the homomorphic properties and linearity of $\otimes$, we have

  $$\mathbf{y} = E_{\mathbf{Z}}(\mathbf{vH} = \mathbf{x}(\mathbf{H} \otimes \mathbf{b}), \mathbf{x}(\mathbf{U} + \mathbf{U}') \; ; \; \mathbf{xA}, \mathbf{xE}),$$

  where $\mathbf{U}'$ is the matrix of rounding errors (each in $[-1/2, 1/2]$) induced by the scalar matrix $(h(b) \otimes \mathbf{b})$.

- *Inversion algorithm.* G^{-1}_{abo} takes as input $((\mathbf{Z}, b^*, h), b, \mathbf{y})$, where $(\mathbf{Z}, b^*, h)$ is the trapdoor information, b is the evaluated branch, and $\mathbf{y}$ is the function output. It first decrypts, yielding a vector $\mathbf{m} = D_{\mathbf{Z}}(\mathbf{y}) \in \mathbb{Z}_p^m$. It then computes $\mathbf{H} = h(b) - h(b^*)$, and if possible, solves (via Gaussian elimination, e.g.) for the unique $\mathbf{v} \in \mathbb{Z}_p^m$ such that $\mathbf{vH} = \mathbf{m}$. The output is the $\mathbf{x} \in \{0,1\}^n$ associated with $\mathbf{v}$. (We show below that such a unique $\mathbf{v}$ exists with overwhelming probability.)

Lemma 6.5. *Let $b^* \in B$ be arbitrary and let p be prime. Then with probability at least $1 - p^{m+t-w}$ over the choice of $h \leftarrow \mathcal{H}$, the matrix $\mathbf{H} = h(b) - h(b^*) \in \mathbb{Z}_p^{m \times w}$ has row rank m for* every *$b \in B$, $b \neq b^*$. In particular, a unique solution $\mathbf{v}$ to the system $\mathbf{vH} = \mathbf{m}$ can be found, if it exists.*

Proof. It suffices to show that for any single $b \neq b^*$, $\mathbf{H} = h(b) - h(b^*)$ has row rank m with probability at least $1 - p^{m-w}$ (over the choice of h). The lemma then follows by a union bound over all $p^t - 1$ values of $b \neq b^*$.

We observe that a uniformly random matrix $\mathbf{H} \in \mathbb{Z}_p^{m \times w}$ has row rank $< m$ with probability at most p^{m-w}. This is because for any fixed nonzero $\mathbf{v} \in \mathbb{Z}_p^m$, we have $\Pr_{\mathbf{H}}[\mathbf{vH} = \mathbf{0}] = p^{-w}$ (this is the only place where we use the fact that p is prime). The observation follows by summing over all the $p^m - 1$ nonzero $\mathbf{v} \in \mathbb{Z}_p^m$ using the union bound.

Now conditioned on the value $h(b^*)$, the value $h(b)$ is still uniformly random by pairwise independence. Therefore, $\mathbf{H} = h(b) - h(b^*)$ is uniform, and we are done. □

Theorem 6.6. *Instantiate the parameters of the above scheme as follows: let $p = n^{c_1}$ be* prime *for some constant $c_1 > 0$, let $q \in [4pn, O(pn^{c_2})]$ for some constant $c_2 > 1$, let $n = d^{c_3}$ for some constant $c_3 > 1$, and let $\chi = \bar{\Psi}_\alpha$ where $\alpha \leq 1/(32pn)$. Let the matrix width $w = m + t + t' = m + 2d$, letting (say) $t = t' = d$.*[7]

Then the algorithms described above give a collection of almost-always (n,k)-ABO TDFs with branch set $\mathbb{Z}_p^t = \mathbb{Z}_p^d$ (of size exponential in d) under the assumption that $\mathsf{LWE}_{q,\chi}$ is hard, where the residual leakage $r = n - k$ is

$$r \leq \left(\frac{c_2}{c_1} + o(1)\right) \cdot n.$$

Proof. The proof is very similar to that of Theorem 6.4, adjusted to accommodate the larger matrix width w and the pairwise independent matrices $\mathbf{H}$.

The correctness of the inversion algorithm for all branches $b \neq b^*$ and on all values $\mathbf{y}$ (with overwhelming probability over the choice of function) follows by Lemma 6.5. Specifically, for any output $\mathbf{y}$, the absolute total error in y_j is $< q/2p$ for all $j \in [w]$ (with overwhelming probability), hence the decryption $D_{\mathbf{Z}}(\mathbf{y})$ outputs $\mathbf{vH}$. Furthermore, with all but $p^{m+t-w} = p^{-d} = \mathsf{negl}(d)$ probability, every $\mathbf{H} = h(b) - h(b^*)$ has full row rank, so $\mathbf{v}$ can be recovered from $\mathbf{vH}$ for all branches $b \neq b^*$.

We now analyze the lossiness. All ciphertexts y_j are encryptions of 0 and carry the same randomness $\mathbf{xA} \in \mathbb{Z}_q^d$. By Lemma 6.3, the total error in every y_j has absolute value $< q/2p$ (with overwhelming probability over the choice of the function). Therefore the total number of outputs of the function on lossy branch b^* is at most $q^d \cdot (q/p)^w = q^d \cdot (q/p)^{m+2d}$. A calculation similar to the one from Theorem 6.4 yields the claimed lossiness, where the only difference is an extra additive term in the residual leakage of $2d \lg O(n^{c_2}) = O(n^{1/c_3} \lg n) = o(n)$.

Finally, the hidden lossy branch property follows by the security of matrix encryption. □

[7]More generally, it suffices to let t, t' be any functions of d growing faster than any constant and slower than $n^{1-\delta}$ for some $\delta > 0$.

6.5 Connection to Lattices

We now relate the security of our constructions to the conjectured worst-case (quantum) hardness of lattice problems, rather than simply to the assumed average-case hardness of LWE. The main statement is a connection between any desired constant lossiness rate $K \in (0,1)$ (larger K means more information is lost) and the associated approximation factor for lattice problems. This merely involves a somewhat tedious (but otherwise routine) instantiation of all of the parameters $n, p, q, \ldots$ to satisfy the various hypotheses of the constructions.

Theorem 6.7. *For any constant $K \in (0,1)$, the construction of Section 6.3* with prime q *gives a family of almost-always (n, Kn)-lossy TDFs for all sufficiently large n, assuming that SIVP and GapSVP are hard for quantum algorithms to approximate to within $\tilde{O}(d^c)$ factors, where $c = 2 + \frac{3}{2(1-K)} + \delta$ for any desired $\delta > 0$.*

The same applies for the construction in Section 6.4, with prime q and p, *of almost-always (n, Kn)-all-but-one TDFs.*

Proof. Using the notation from Theorem 6.4 (likewise Theorem 6.6), we let $p = n^{c_1}$ and let $n = d^{c_3}$ for some constants $c_1 > 0, c_3 > 1$ that we set later, and let $\alpha = 1/(32pn)$. In order to invoke Proposition 6.1 (connecting LWE to lattice problems), we will need to use some

$$q > 2\sqrt{d}/\alpha = 64pn\sqrt{d} = 64pn^{1+1/(2c_3)}.$$

Therefore we set $c_2 = 1 + 1/(2c_3)$, so we may take $q = O(pn^{c_2})$.

Now invoking Theorem 6.4, we get that the lossy TDF collection has residual leakage

$$n \cdot \left(\frac{c_2}{c_1} + \epsilon \right) = n \cdot \left(\frac{1 + 2c_3}{2c_1c_3} + \epsilon \right)$$

for any $\epsilon > 0$ and sufficiently large n.

Now by Proposition 6.1, LWE is hard for our choice of parameters, assuming the lattice problems are hard to approximate within $\tilde{O}(d/\alpha) = \tilde{O}(d^{1+c_3(c_1+1)})$ factors for quantum algorithms. With the constraint on the residual leakage as $\frac{1+2c_3}{2c_1c_3} < (1-K)$, we get that $c_1 > \frac{1+2c_3}{2c_3(1-K)}$. This implies that the exponent in the lattice approximation factor may be brought arbitrarily close to $1 + c_3 + \frac{1+2c_3}{2(1-K)}$. Then under the constraint that $c_3 > 1$, the exponent may be brought arbitrarily close to $2 + \frac{3}{2(1-K)}$, as desired. ☐

Acknowledgments

We are very grateful to Dan Boneh for offering important insights in the early stages of our work.

References

[1] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108, 1996.

[2] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, pages 284–293, 1997.

[3] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil P. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In *CRYPTO*, pages 283–298, 1998.

[4] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[5] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, 1988.

[6] Dan Boneh. The decision Diffie-Hellman problem. In *ANTS*, pages 48–63, 1998.

[7] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[8] Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In *CT-RSA*, pages 87–103, 2005.

[9] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In *ACM Conference on Computer and Communications Security*, pages 320–329, 2005.

[10] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[11] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25, 1998.

[12] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.

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

[14] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, pages 523–540, 2004.

[15] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552, 1991.

[16] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[17] Edith Elkind and Amit Sahai. A unified methodology for constructing public-key encryption schemes secure against adaptive chosen-ciphertext attack. Cryptology ePrint Archive, Report 2002/042, 2002. `http://eprint.iacr.org/`.

[18] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[19] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, pages 537–554, 1999.

[20] Yael Gertner, Tal Malkin, and Steven Myers. Towards a separation of semantic and CCA security for public key encryption. In *TCC*, pages 434–455, 2007.

[21] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS*, pages 126–135, 2001.

[22] Oded Goldreich. *Foundations of Cryptography*, volume II. Cambridge University Press, 2004.

[23] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(42), 1996.

[24] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO*, pages 171–185, 1986.

[25] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[26] Qiong Huang, Duncan S. Wong, and Yiming Zhao. Generic transformation to strongly unforgeable signatures. In *ACNS*, pages 1–17, 2007.

[27] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Multi-bit cryptosystems based on lattice problems. In *PKC*, pages 315–329, 2007.

[28] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.

[29] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.

[30] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[31] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1979.

[32] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO*, pages 433–444, 1991.

[33] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.

[34] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[35] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[36] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.

[37] Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.

[38] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[39] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.