Lossy Trapdoor Functions and Their Applications*

Chris Peikert†
SRI International

Brent Waters‡
SRI International

August 29, 2008

Abstract

We propose a general cryptographic primitive called *lossy trapdoor functions* (lossy TDFs), and use it to develop new approaches for constructing several important cryptographic tools, including (injective) trapdoor functions, collision-resistant hash functions, oblivious transfer, and chosen ciphertext-secure cryptosystems (in the standard model). All of these constructions are simple, efficient, and black-box.

We realize lossy TDFs 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 for *quantum* algorithms (alternately, under an *average-case* hardness assumption for classical algorithms).

Taken together, our 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 chosen ciphertext-secure cryptosystem based solely on worst-case complexity assumptions.

*A preliminary version of this work appeared in the 40th ACM Symposium on Theory of Computing (STOC 2008).

†This material is based upon work supported by the National Science Foundation under Grants CNS-0716786 and CNS-0749931. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

‡Supported by NSF Grants CNS-0524252, CNS-0716199, CNS-0749931; the US Army Research Office under the CyberTA Grant No. W911NF-06-1-0316; and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

1 Introduction

A central goal in cryptography is to realize a variety of security notions based on plausible and concrete computational assumptions. Historically, such assumptions have typically been concerned with problems from three broad categories: those related to *factoring integers*, those related to computing *discrete logarithms* in cyclic groups, and more recently, those related to computational problems on *lattices*.

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

In public key cryptography in particular, two important notions are *trapdoor functions* (TDFs) and *security under chosen ciphertext attack* (CCA security) [39, 45, 20]. Trapdoor functions, which (informally) are hard to invert unless one possesses some secret "trapdoor" information, conceptually date back to the seminal paper of Diffie and Hellman [18] and were first realized by the RSA function of Rivest, Shamir, and Adleman [48]. Chosen-ciphertext security, which (again informally) guarantees confidentiality of encrypted messages even in the presence of a decryption oracle, has become the *de facto* notion of security for public key encryption under active attacks.

Unfortunately, it is still not known how to realize TDFs and CCA security (in the standard model) under all types of assumptions described above. For CCA security, the main approach in the existing literature relies on *noninteractive zero-knowledge* (NIZK) proofs [8, 23] (either for general NP statements or specific number-theoretic problems); using NIZKs, cryptosystems have been constructed based on problems related to factoring and discrete logs [39, 20, 50, 16, 17], but not lattices. For trapdoor functions, the state of the art is even less satisfactory: though TDFs are widely viewed as a general primitive, they have so far been realized only from problems related to factoring [48, 44, 40].

In this paper, we make the following contributions:

- We introduce a new general primitive called *lossy trapdoor functions*, and give realizations based on the conjectured hardness of the decisional Diffie-Hellman (DDH) problem in cyclic groups, and the conjectured *worst-case* hardness of certain lattice problems.

- We show that lossy trapdoor functions imply injective (one-to-one) trapdoor functions in the traditional sense. This yields the first known trapdoor functions based on computational problems that are not directly related to integer factorization.

- 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 known to exist only in the random oracle model [6, 24].

  Our approach has two main benefits: first, our construction is black-box, making it more efficient than those that follow the general NIZK paradigm [39, 20, 50].[1] Second, it yields the first known CCA-secure cryptosystem based entirely on (worst-case) lattice assumptions, resolving a problem that had remained open since the seminal work of Ajtai [1] and Ajtai and Dwork [2].

[1] We note that Cramer and Shoup [16, 17] gave *efficient* CCA-secure constructions based on NIZK proofs for *specific* number-theoretic problems.

- We further demonstrate the utility of lossy TDFs by constructing pseudorandom generators, collision-resistant hash functions, and oblivious transfer (OT) protocols, in a black-box manner and with simple and tight security reductions. Using standard (but non-black box) transformations [30, 31], our OT protocols additionally imply general secure multiparty computation for malicious adversaries.

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 any cryptosystem that is secure under a chosen-plaintext attack (CPA-secure) [5]. A 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 potentially 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 [5, 6, 24].

Unfortunately, the technique of encrypting a ciphertext's own randomness has so far met with less success in the standard model, because CPA 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.

While the above is just a contrived counterexample for one particular attempt, Gertner, Malkin, and Reingold [27] demonstrated a *black-box separation* between (poly-to-one) trapdoor functions and CPA-secure encryption. The chief difficulty is that inverting a trapdoor function requires the recovery of its *entire* input, whereas decrypting a ciphertext recovers the input message, but not necessarily the randomness. For similar reasons, there is also some evidence that achieving CCA security from CPA security (in a black-box manner) would be difficult [26].

Perhaps for these reasons, constructions of CCA-secure encryption in the standard model [39, 20, 50, 16, 17] have followed a different approach. As explained in [21], 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. A primary benefit of zero-knowledge is that the decryption algorithm can be sure that a 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 [20], and efficient systems based on specific number theoretic assumptions [16, 17], such as the decisional Diffie-Hellman (DDH) [11] and decisional composite residuosity [40] 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 CPA-secure public key cryptosystems based on *worst-case* lattice assumptions are known [2, 46, 47], there are still no known *CCA-secure* systems, because it is unknown how to realize NIZKs for all of NP (or for appropriate specific lattice problems) under 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 chosen-plaintext 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 ways. The first way matches the usual completeness condition for an (injective) trapdoor function: given a suitable trapdoor for f, the entire input x can be efficiently recovered from $f(x)$. In the second way, f *statistically loses* a significant amount of information about its input, i.e., every output of f has many preimages. Finally, the two behaviors are *indistinguishable*: given just the description of f (i.e., its code), no efficient adversary can tell whether f is injective or lossy.

Using lossy trapdoor functions as a general tool, we develop new techniques for constructing standard trapdoor functions and CCA-secure cryptosystems, and for proving their security. In essence, lossy TDFs allow for proving 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 a chosen-ciphertext attack).

In more detail, the public parameters of our schemes will include some function f that is either injective or lossy. In the injective case (typically corresponding to the real system), the invertibility of f permits recovery of its entire input and ensures correctness of the system. In the lossy case (typically corresponding to a "thought experiment" in the security proof), one typically shows that the loss of information by f implies *statistical* security of the system. 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., by choosing the underlying randomness itself.

In the following, we demonstrate the utility of lossy TDFs by informally sketching constructions of standard TDFs, CPA-secure encryption, and CCA-secure encryption. Later in the paper we also demonstrate simple constructions of pseudorandom generators, collision-resistant hash functions, and oblivious transfer protocols that enjoy tight and elementary security reductions. (Formal definitions, constructions, and proofs are given in Sections 3 and 4.)

1.2.1 Trapdoor Functions and CPA-Secure Encryption

Suppose we have a collection of lossy TDFs having domain $\{0,1\}^n$, where the lossy functions "lose" (say) $k = n/2$ bits of the input. Then the *injective* functions from this collection make up 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 invoke $\mathcal{I}$ on $f(x)$, it must output (with some noticeable probability) the same value x, because f is injective. Now consider the same experiment when f is replaced by a lossy function f'. Observe that $f'(x)$ *statistically* hides the value of x, because there are on average about $2^k = 2^{n/2}$ other values x' such that $f'(x') = f'(x)$, and all are equally likely. Therefore even an *unbounded* inverter $\mathcal{I}$ cannot guess the hidden value x, except with negligible probability. We conclude that no efficient inverter can exist for the injective functions, unless the injective and lossy functions are distinguishable.

Using the fact that lossy TDFs imply standard injective TDFs, we could construct a CPA-secure cryptosystem by standard techniques, e.g., by using a generic Goldreich-Levin hard-core predicate [29] to conceal the message bit and using the trapdoor to recover it. However, it is instructive (and a useful warm-up for our CCA-secure construction) to see that lossy TDFs admit hard-core *functions*, via a very simple and direct

proof of security.

Let $\mathcal{H}$ be a family of pairwise independent hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, where $\{0,1\}^n$ is the domain of the lossy TDFs and $\ell < n$ is essentially the number of bits lost by the lossy functions (we defer details for the purposes of this sketch). Then a hash function h chosen at random from $\mathcal{H}$ acts as a hard-core function for the injective TDFs of the collection. This follows from the fact that h is a *strong randomness extractor*, by an average-case variant of the leftover hash lemma [33, 19].

In more detail, consider an adversary that attempts to distinguish $h(x) \in \{0,1\}^\ell$ from uniform, given h and $f(x)$ for injective f and uniformly random $x \in \{0,1\}^n$. The adversary's advantage must be essentially the same if f is replaced with a lossy function f'. In this case, the value of x is *statistically* well-hidden given $f'(x)$ (more precisely, x has large min-entropy on the average). Because h is a good extractor, it follows that $h(x)$ is statistically close to uniform over $\{0,1\}^\ell$ given $f'(x)$ and h, so even an unbounded adversary has negligible advantage.

1.2.2 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, then the simulator will not be able to answer (even well-formed) decryption queries, because the plaintext information is lost. Therefore, we introduce a richer abstraction called *all-but-one* (ABO) trapdoor functions, which can be constructed from a collection of sufficiently lossy TDFs, or more directly from similar underlying concrete assumptions.

An ABO collection is associated with a large set B that we call *branches*. The generator of an ABO function takes an extra parameter $b^* \in B$, called the *lossy* branch, and outputs 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 lossy branch is hidden (computationally) by the description of g.

Cryptosystem. Our construction of a CCA-secure cryptosystem uses a collection of lossy TDFs and a collection of ABO TDFs, both having domain $\{0,1\}^n$. As before, we 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. For full CCA2 security and non-malleability (as opposed to CCA1, or "lunchtime," security), we also use a strongly unforgeable one-time signature scheme in which verification keys can be interpreted as branches of the ABO function.[2] In this sketch we give only the main ideas, and defer the exact selection of parameters to Section 4.

The key generator of the cryptosystem chooses an injective function f from the lossy TDF collection, along with its trapdoor f^{-1}. Next, it chooses an ABO function g whose lossy branch is arbitrarily set to $b^* = 0$ (the decrypter does not need the trapdoor for g, so it can be discarded). Finally, it selects a hash function h at random from $\mathcal{H}$. The public key is $pk = (f, g, h)$, and the trapdoor f^{-1} is kept as the secret decryption key (along with pk 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 one-time signature scheme. It then chooses an $x \in \{0,1\}^n$ uniformly at random. The ciphertext is generated as

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

[2] Strongly unforgeable one-time signatures are implied by one-way functions, and in particular by lossy trapdoor functions. A variant of our system remains CCA1-secure without the signature, but it is trivially malleable and thus not CCA2-secure.

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

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 relative to vk, and aborts if not. Next, it computes $x' = f^{-1}(c_1)$ using the trapdoor, obtaining an encryption *witness* x'. Then the decrypter "recomputes" the ciphertext to verify 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 $m' = h(x') \oplus c_3$.

Security. The proof of security follows a hybrid two key-type argument, but without zero knowledge (due to the recovery of the encryption witness). The proof involves several hybrid experiments that are indistinguishable to any efficient adversary. In the first hybrid, the ABO lossy branch b^* is instead set to $b^* = vk^*$, where vk^* is a verification key that eventually appears in the challenge ciphertext. In the next hybrid, the decryption oracle decrypts using the trapdoor g^{-1} for the ABO function, rather than f^{-1}. The decryption oracle is thus able to decrypt successfully for *all branches but one*, namely, the vk^* branch — but by unforgeability of the signature scheme, any query involving vk^* has 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 both components $c_1 = f(x)$ and $c_2 = g(vk^*, x)$ of the challenge ciphertext lose information about x. Therefore, $h(x)$ is statistically close to uniform (given the rest of the view of the adversary), so even an unbounded adversary has only negligible advantage in guessing the encrypted message.

We conclude this summary with a few remarks. First, we note that in practice one would likely use our techniques as a public-key key encapsulation mechanism (KEM) where the key would be derived using the hash function as $h(x)$. Second, while our system falls outside the NIZK paradigm, we do rely on some techniques that are reminiscent of previous work. Our construction uses a two-key strategy originally due to Naor and Yung [39], 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 on a *ciphertext* encrypted under the other key. In contrast, our simulation participates in a distinguishing experiment on the *other key itself*. Additionally, our use of one-time signatures for CCA2 security inherits from the work of Dolev, Dwork and Naor [20], and is technically most similar to the method of Canetti, Halevi, and Katz [15] for constructing CCA-secure encryption from identity-based encryption. Finally, we point out that our decryption algorithm does not strictly recover *all* the randomness of the ciphertext, because it does not recover the randomness used to generate the one-time signing key or the signature itself. This is a minor technical point, as the decrypter does recover enough randomness to check validity of the ciphertext (the signature is publicly verifiable). Additionally, for the weaker notion of CCA1 (or "lunchtime") security, the one-time signature is unnecessary; vk can be replaced with a random choice of branch, and the decrypter does recover all of the randomness.

1.3 Realizing Lossy TDFs

We now sketch our basic framework for constructing lossy and all-but-one trapdoor functions. To illuminate the main principles, we assume a generic CPA-secure cryptosystem having a few special (but informally described) properties. We then sketch how to obtain such properties under concrete assumptions.

The first property we assume is that the underlying cryptosystem is *additively homomorphic*. A function f (whether injective or lossy) on $\{0,1\}^n$ is specified by an entry-wise encryption of some $n \times n$ matrix $\mathbf{M}$. To evaluate $f(x)$, view the input $x \in \{0,1\}^n$ as an n-dimensional binary vector $\mathbf{x}$, and compute an

(entry-wise) encryption of the linear product $\mathbf{x} \cdot \mathbf{M}$ by applying the homomorphic operation to the encrypted entries of $\mathbf{M}$.

For an injective function, the encrypted matrix $\mathbf{M}$ is the identity matrix $\mathbf{I}$, and the trapdoor is the decryption key for the cryptosystem. The function f 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.

For a lossy function, the encrypted matrix $\mathbf{M}$ is the all-zeros matrix $\mathbf{M} = \mathbf{0}$. Then for *every* input x, the value $f(x)$ is an entry-wise encryption of the all-zeros vector, so f intuitively "loses" x. 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 of the output ciphertext depends only on the randomness of the input ciphertexts (and not, say, on the public key or the encrypted messages). Many known cryptosystems are even homomorphic with respect to randomness, which certainly 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, and the trapdoor is the set of corresponding decryption keys. Across each row i, we encrypt entry $m_{i,j}$ under key pk_j and the *same randomness* r_i (using fresh randomness for each row). By hypothesis, it is secure to reuse randomness across keys pk_j, so the matrix $\mathbf{M}$ is computationally hidden. Additionally, because the homomorphism isolates randomness, all the ciphertexts in the output vector $f(x)$ are *also* encrypted under the same randomness R (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 ciphertext entry of $f(x)$. On the other hand, 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* (but under a different fixed key). Therefore the number of possible outputs of f is bounded by the number of possible random strings that can arise. By choosing the dimension n so that the number of inputs 2^n is significantly larger than the number of random strings, we can guarantee that the function is lossy.

The construction of all-but-one trapdoor functions is similar, but somewhat more general. Each branch b of the function simply corresponds to a different matrix $\mathbf{M}_b$, whose encryption can be derived from the public description of the function. The function is generated so that $\mathbf{M}_b$ is an invertible matrix (and is computable with the trapdoor) for all the injective branches b, whereas $\mathbf{M}_{b^*} = \mathbf{0}$ for the lossy branch b^*.

Concrete assumptions. Under the decisional Diffie-Hellman assumption, it is relatively straightforward to implement the above framework for constructing lossy and all-but-one TDFs (see Section 5). On the other hand, when instantiating lossy TDFs under worst-case lattice assumptions via known lattice-based cryptosystems [2, 46, 47], many additional technical difficulties arise. In fact, we only know how to construct lossy TDFs based on the "learning with errors" (LWE) problem as defined by Regev [47], which is a generalization of the famous "learning parity with noise" problem to larger moduli. The LWE problem can be seen as an average-case "bounded-distance decoding" problem on a certain family of random lattices, and appears (like the learning parity with noise problem) to be hard. Moreover, Regev gave a reduction showing that LWE is indeed hard on the average if standard lattice problems are hard in the *worst case* for *quantum* algorithms [47]. Quantum algorithms are not known to have any advantage over classical algorithms for the worst-case lattice problems in question. Moreover, it may be that the reduction can be "dequantized," i.e., that the hardness of LWE could be based on a classical worst-case assumption.

There are two main reasons why our lattice-based constructions seem to be limited to LWE. First, the

LWE problem involves some *public* randomness that can be securely reused, whereas the underlying hard problems in the prior lattice-based cryptosystems of Ajtai and Dwork [2] and Regev [46] involve *secret* randomness that is not apparently reusable in a secure way. The second difficulty is that lattice-based cryptosystems involve some non-reusable error terms, which leak additional information in our constructions. The error terms in the problems of [2, 46] come from an *exponentially* large domain, therefore they may leak more bits than we are able to lose via our matrix construction. In contrast, the error terms in the LWE problem come from a polynomial-sized domain, so their leakage can be bounded appropriately (this 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 the use of lossy trapdoors is that security is typically demonstrated via indistinguishability arguments over a scheme's *public parameters*, as opposed to its outputs. Prior constructions of CPA-secure lattice-based cryptosystems [2, 46, 47] (among others) used this style of argument, but to our knowledge it has never been employed in the context of chosen-ciphertext security.

The present approach can be contrasted with the oblivious transfer (OT) construction of Even, Goldreich, and Lempel [22]. They construct (semi-honest) oblivious transfer protocols from any public key cryptosystem in which a public key can be sampled without knowing its corresponding decryption key (or equivalent). In the OT protocol, one of the messages is encrypted under such a public key, thereby hiding it computationally from the receiver. Lossy TDFs can be employed to construct OT in a similar way, but the security properties are reversed: 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.

Another interesting comparison is to the techniques used to construct CCA-secure cryptosystems from identity-based encryption (IBE) [53] that were introduced by Canetti, Halevi, and Katz [15] and improved in later work [13, 14, 12]. Our construction and simulation share some techniques with these works, but also differ in important ways. In the constructions based on IBE, the simulator is able to acquire secret keys for all identities but one special identity ID^*, and can therefore answer decryption queries in the CCA experiment. The special identity ID^* is hidden *statistically* by the public key, while the challenge ciphertext encrypted under ID^* hides its message only *computationally*. In our simulation, the security properties are once again reversed: the lossy branch b^* is hidden only computationally by the public key, but the challenge ciphertext hides its message statistically.

Our concrete constructions of lossy TDFs under the DDH assumption (by reusing randomness across many encryption keys) are technically similar to the ElGamal-like cryptosystems of Bellare *et al.* [4] that reuse randomness for efficiency, and to constructions of pseudorandom *synthesizers* by Naor and Reingold [37]. In particular, indistinguishability of injective and lossy functions follows directly from the pseudorandomness property of the synthesizer. The novelty in our constructions is in the use of additional homomorphic structure to compute encrypted linear products, and to bound the number of possible outputs in the lossy case.

1.5 Subsequent Work

Additional work on lossy trapdoor functions and related concepts has appeared since the initial publication of this work in [43]. Rosen and Segev [49] and Boldyreva, Fehr, and O'Neill [10] independently gave simple and compact constructions of lossy and ABO TDFs under the decisional composite residuosity assumption, based on the natural trapdoor function of Paillier [40]. (The preliminary version of this work [43] constructed somewhat more complex lossy and ABO TDFs under a variant of Paillier's assumption.) Additionally,

Boldyreva *et al.* [10] constructed CCA-secure *deterministic* cryptosystems for high-entropy messages using lossy and ABO TDFs.

Using entirely different techniques, Gentry, Peikert, and Vaikuntanathan [25] recently demonstrated two "natural" collections of trapdoor functions under worst-case lattice assumptions. One collection (which relies on essentially the same quantum worst-case assumption as ours) consists of functions that are injective, but not known to be lossy; the other collection (which is based only on *classical* worst-case assumptions) contains functions that are *many-to-one* (non-injective), and seem best suited for complementary cryptographic applications such as signatures schemes and identity-based encryption.

2 Preliminaries

Here we review some standard notation and cryptographic definitions. We also give relevant background relating to entropy of distributions and extraction of randomness from weakly-random sources.

2.1 Basic Concepts

We let $\mathbb{N}$ denote the natural numbers. For any $k \in \mathbb{N}$, $[k]$ denotes the set $\{1, \ldots, k\}$. Unless described otherwise, all quantities are implicitly functions of a *security parameter* denoted $\lambda \in \mathbb{N}$, except in Section 6, where we use d. We use standard asymptotic notation O, o, Ω, and ω to denote the growth of functions. We say that $f(\lambda) = \tilde{O}(g(\lambda))$ if $f(\lambda) = O(g(\lambda) \log^c \lambda)$ for some constant c. We let $\mathrm{poly}(\lambda)$ denote an unspecified function $f(\lambda) = O(\lambda^c)$ for some constant c.

We let $\mathsf{negl}(\lambda)$ denote some unspecified function $f(\lambda)$ such that $f = o(\lambda^{-c})$ for *every* constant c, saying that such a function is *negligible* (in λ). We say that a probability is *overwhelming* if it is $1 - \mathsf{negl}(\lambda)$. Throughout the paper, a *probabilistic polynomial-time* (PPT) algorithm is a randomized algorithm that runs in time $\mathrm{poly}(\lambda)$, and is implicitly given λ (represented in unary) as an input.

For convenience, we often identify random variables with their probability distributions. Let X and Y be two random variables over some countable set S. The *statistical distance* between X and Y is defined as

$$\Delta(X, Y) = \tfrac{1}{2} \sum_{s \in S} \left| \Pr[X = s] - \Pr[Y = s] \right|.$$

Statistical distance is a metric; in particular, it obeys the triangle inequality.

Let $\mathcal{X} = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ denote two ensembles of random variables indexed by λ. We say that $\mathcal{X}$ and $\mathcal{Y}$ are *statistically indistinguishable* and write $\mathcal{X} \stackrel{s}{\approx} \mathcal{Y}$ if $\Delta(X_\lambda, Y_\lambda)$ is negligible as a function of λ. Given an algorithm $\mathcal{A}$, define its *advantage* in distinguishing between $\mathcal{X}$ and $\mathcal{Y}$ as

$$\left| \Pr[\mathcal{A}(X_\lambda) = 1] - \Pr[\mathcal{A}(Y_\lambda) = 1] \right|,$$

where the probability is taken over the random values X_λ and Y_λ, and the randomness of $\mathcal{A}$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are *computationally indistinguishable* and write $\mathcal{X} \stackrel{c}{\approx} \mathcal{Y}$ if the advantage of any PPT algorithm $\mathcal{A}$ is negligible as a function of λ.[3] It is routine to see that statistical indistinguishability implies computational indistinguishability. When the ensembles are clear from context, we often say that two random variables are statistically/computationally indistinguishable.

[3] For simplicity, throughout the paper we opt to define security against *uniform* adversaries; all our results can be easily adapted to a non-uniform treatment.

It is a standard fact that the outputs of any algorithm (respectively, any PPT algorithm) on two statistically (resp., computationally) indistinguishable variables are themselves statistically (resp., computationally) indistinguishable. Moreover, it is straightforward to prove (via a hybrid argument) that statistical and computational indistinguishability are transitive under polynomially-many steps. More precisely, if $\mathcal{X}_1 \stackrel{s}{\approx} \mathcal{X}_2 \stackrel{s}{\approx} \cdots \stackrel{s}{\approx} \mathcal{X}_k$ (respectively, $\mathcal{X}_1 \stackrel{c}{\approx} \cdots \stackrel{c}{\approx} \mathcal{X}_k$) is any sequence of $k = \text{poly}(\lambda)$ ensembles, then $\mathcal{X}_1 \stackrel{s}{\approx} \mathcal{X}_k$ (resp., $\mathcal{X}_1 \stackrel{c}{\approx} \mathcal{X}_k$).

In the following, we frequently define security notions in terms of interactive experiments (sometimes called "games") involving an adversary algorithm $\mathcal{A}$ (formally, an interactive Turing machine). The *view* of the adversary in such an experiment is the ensemble of random variables, indexed by the security parameter λ, where each variable includes the random coins of $\mathcal{A}$ and all its inputs over the course of the experiment when run with security parameter λ.

2.2 Trapdoor Functions

We recall one 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 as a function of the security parameter. A collection of *injective* trapdoor functions is given by a tuple of PPT algorithms (S, F, F^{-1}) having the following properties:

1. *Easy to sample, compute, and invert with trapdoor:* S outputs (s, t) where s is a function index and t is its trapdoor, $F(s, \cdot)$ computes an injective (deterministic) 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 $\mathcal{I}(s, f_s(x))$ outputs x is negligible, where the probability is taken over the choice of $(s, t) \leftarrow S$, $x \leftarrow \{0,1\}^n$, and $\mathcal{I}$'s randomness.

2.3 Cryptosystems and Security Notions

We recall the definition of a public-key cryptosystem and the standard notions of security, including security under chosen-plaintext attack (CPA) and under chosen-ciphertext attack (CCA). A cryptosystem consists of three PPT algorithms that are modeled as follows:

- $\mathcal{G}$ 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 message space, possibly depending on λ), and outputs a ciphertext c.
- $\mathcal{D}(sk, c)$ takes as input a secret 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.

The standard completeness requirement is that for any $(pk, sk) \leftarrow \mathcal{G}$ and any $m \in \mathcal{M}$, we have $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$. We relax this notion to require that decryption is correct with overwhelming probability over all the randomness of the algorithms.

A basic notion of security for a public key cryptosystem is indistinguishability under a chosen plaintext attack, called *CPA security*. A cryptosystem is said to be CPA-secure if the views of any PPT adversary $\mathcal{A}$ in the following two experiments indexed by a bit $b \in \{0,1\}$ are computationally indistinguishable: a key pair

$(pk, sk) \leftarrow \mathcal{G}$ is generated and pk is given to $\mathcal{A}$. Then $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$, and is given a ciphertext $c^* \leftarrow \mathcal{E}(pk, m_b)$, i.e., an encryption of message m_b.

A much stronger notion of security for a public key cryptosystem is indistinguishability under an adaptive chosen ciphertext attack, or *CCA security*. This notion is similarly defined by two experiments as described above, where the adversary $\mathcal{A}$ is additionally given access to an oracle $\mathcal{O}$ that computes $\mathcal{D}(sk, \cdot)$ during part or all of the game. In a variant called CCA1 (or "lunchtime") security, the oracle $\mathcal{O}$ computes $\mathcal{D}(sk, \cdot)$ *before* the ciphertext c^* is given to $\mathcal{A}$, and outputs $\perp$ on all queries thereafter. In the stronger and more standard notion of CCA2 security, the oracle $\mathcal{O}$ computes $\mathcal{D}(sk, \cdot)$ throughout the entire experiment, with the exception that it returns $\perp$ if queried on the particular challenge ciphertext c^* (this condition is necessary, otherwise the definition is trivially impossible to realize).

2.4 Strongly Unforgeable One-Time Signatures

We now review standard definitions of signature schemes and a security notion called *strong* (one-time) unforgeability. A signature system consists of three PPT algorithms Gen, Sign, and Ver, which are modeled as follows:

- Gen 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 outputs either 0 or 1.

The standard completeness requirement is that for any $(vk, sk_\sigma) \leftarrow \mathsf{Gen}$ and any $m \in \mathcal{M}$, we have $\mathsf{Ver}(vk, m, \mathsf{Sign}(sk_\sigma, m)) = 1$. We relax this notion to require that the verification algorithm succeeds (i.e., outputs 1) with overwhelming probability over the randomness of the algorithms.

The security notion of *strong* existential unforgeability under a one-time chosen message attack is defined in terms of the following experiment between a challenger and a PPT adversary algorithm $\mathcal{A}$: the challenger first generates a key pair $(vk, sk_\sigma) \leftarrow \mathcal{G}$, and gives vk to $\mathcal{A}$. Then $\mathcal{A}$ queries an oracle that computes $\mathsf{Sign}(sk_\sigma, \cdot)$ on a message $m \in \mathcal{M}$ of its choice. Finally, $\mathcal{A}$ outputs a pair (m', σ'), and is said to succeed if $\mathsf{Ver}(vk, m', \sigma') = 1$ and $(m', \sigma') \neq (m, \sigma)$. The advantage of $\mathcal{A}$ is the probability that $\mathcal{A}$ succeeds, taken over all the randomness of the experiment; a signature scheme is strongly unforgeable under a one-time chosen message attack if every PPT adversary $\mathcal{A}$ has only negligible advantage in the above game.

Strongly unforgeable one-time signatures can be constructed from any one-way function [28], and more efficiently from collision-resistant hash functions [34]. As we show later, both primitives have black-box constructions from lossy trapdoor functions.

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* [54] 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 computed [54].

A collection of *collision-resistant* hash functions from length $\ell(\lambda)$ to length $\ell'(\lambda) < \ell(\lambda)$ is modeled by a pair of PPT algorithms $(S_{\text{crh}}, F_{\text{crh}})$, where

1. S_{crh} outputs a function index i,
2. $F_{\text{crh}}(i, \cdot)$ computes a (deterministic) function $H_i : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\ell'(\lambda)}$,
3. for every PPT adversary $\mathcal{A}$, the probability (over the choice of i and the randomness of $\mathcal{A}$) that $\mathcal{A}(i)$ outputs distinct $x, x' \in \{0,1\}^{\ell(\lambda)}$ such that $H_i(x) = H_i(x')$ is negligible in λ.

A collection of *universal one-way hash functions* (UOWHFs) [38] is similarly given by algorithms $(S_{\text{uowhf}}, F_{\text{uowhf}})$, with the following security property. Let $\mathcal{A}$ be a PPT adversary that participates in the following experiment: $\mathcal{A}$ outputs an $x \in \{0,1\}^{\ell(\lambda)}$, then a function index $i \leftarrow S_{\text{uowhf}}$ is chosen and given to $\mathcal{A}$, then $\mathcal{A}$ outputs some $x' \in \{0,1\}^{\ell(\lambda)}$ distinct from x. Then the probability (over all the randomness of the game) that $F_{\text{uowhf}}(i, x) = F_{\text{uowhf}}(i, x')$ is negligible (in λ).

2.6 Extracting Randomness

The min-entropy of a random variable X over a domain S is

$$H_\infty(X) = -\lg(\max_{s \in S} \Pr[X = s]).$$

In many natural settings, the variable X is correlated with another variable Y whose value is known to an adversary. For our purposes, it is most convenient to use the notion of *average min-entropy* as defined by Dodis *et al.* [19], which captures the remaining unpredictability of X conditioned on the value of Y:

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

The average min-entropy is the negative logarithm of the *average predictability* of X conditioned on the random choice of Y; that is, the average maximum probability of predicting X given Y. (See [19] for further discussion and alternate notions.)

Lemma 2.1 ([19, Lemma 2.2])**.** *If Y takes at most 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 applications, we need to derive nearly-uniform bits from a weakly random source X. In general, this can be done using any *strong randomness extractor* (see Shaltiel's survey for details [52]). In our case, pairwise independent hash functions [54] are sufficient, and as a bonus, they interact particularly well with the notion of average min-entropy.

Lemma 2.2 ([19, Lemma 2.4])**.** *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 $h \leftarrow \mathcal{H}$, 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. For convenience, we also define the *residual leakage* $r(\lambda) := n(\lambda) - k(\lambda)$. For all these quantities, we often omit the dependence on λ.

A collection of (n, k)-*lossy trapdoor functions* is given by a tuple of PPT algorithms $(S_{\text{ltdf}}, F_{\text{ltdf}}, F^{-1}_{\text{ltdf}})$ having the properties below. For notational convenience, 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_{inj} outputs (s, t) where s is a function index and t is its trapdoor, $F_{\text{ltdf}}(s, \cdot)$ computes an *injective* (deterministic) function $f_s(\cdot)$ over the domain $\{0,1\}^n$, and $F^{-1}_{\text{ltdf}}(t, \cdot)$ computes $f_s^{-1}(\cdot)$. If a value y is not in the image of f_s, i.e., if $f_s^{-1}(y)$ does not exist, then the behavior of $F^{-1}_{\text{ltdf}}(t, y)$ is unspecified (because of this, the output of F^{-1}_{ltdf} may need to be checked for correctness in certain applications).

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

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

Note that we make no explicit requirement that an injective function be hard to invert. As shown in Lemma 3.3, this property is implied by combination of the lossiness and indistinguishability properties.

For our lattice-based constructions we need to consider a slightly relaxed definition of lossy TDFs, which we call *almost-always* lossy TDFs. Namely, we require that *with overwhelming probability* over the randomness of S_{inj}, the index s of S_{inj} describes an injective function f_s that F^{-1}_{ltdf} inverts correctly on all values in the image of f_s. In other words, there is only a negligible probability (over the choice of s) that $f_s(\cdot)$ is not injective or that $F^{-1}_{\text{ltdf}}(t, \cdot)$ incorrectly computes $f_s^{-1}(\cdot)$ for some input. Furthermore, we require that with overwhelming probability, the lossy function f_s generated by S_{loss} has image size at most 2^r. 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 has no control over the generation of trapdoor/lossy functions. Therefore the potential advantage of the adversary due to sampling an improper function is bounded by a negligible quantity.

3.2 All-But-One TDFs

For our CCA applications, it is convenient to work with a richer abstraction that we call *all-but-one* (ABO) trapdoor functions. In an ABO collection, each function has an extra input called its *branch*. All of 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) by the resulting function description.

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 PPT algorithms $(S_{\text{abo}}, G_{\text{abo}}, G_{\text{abo}}^{-1})$ having the following properties:

1. *Sampling a trapdoor function with given lossy branch:* for any $b^* \in B_\lambda$, $S_{\text{abo}}(b^*)$ 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 (deterministic) function $g_{s,b}(\cdot)$ over the domain $\{0,1\}^n$, and $G_{\text{abo}}^{-1}(t, b, \cdot)$ computes $g_{s,b}^{-1}(\cdot)$. As above, the behavior of $G_{\text{abo}}^{-1}(t, b, y)$ is unspecified if $g_{s,b}^{-1}(y)$ does not exist.

   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 $S_{\text{abo}}(b_0^*)$ and the first output s_1 of $S_{\text{abo}}(b_1^*)$ are computationally indistinguishable.

Just as with lossy TDFs, we also need to consider an "almost-always" relaxation of the ABO definition. 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 does not affect security in our applications.

3.3 Basic Relations

Lossy and ABO trapdoor functions are equivalent for appropriate choices of parameters and degrees of lossiness. We first show an easy equivalence between the two notions for ABOs with binary branch sets.

Lemma 3.1. *A collection of (n, k)-ABO TDFs having exactly two branches is equivalent to a collection of (n, k)-lossy TDFs.*

Proof. Suppose the existence of an (n, k)-ABO collection having branch set $\{0, 1\}$ (without loss of generality). We construct a collection of (n, k)-lossy TDFs as follows: S_{inj} generates an ABO function having lossy branch $b^* = 1$ (retaining the trapdoor), and S_{loss} samples an ABO function having lossy branch $b^* = 0$ (discarding the trapdoor). The evaluation algorithm F_{ltdf} simply always computes the function on branch $b = 0$. It is clear that F_{ltdf}^{-1} can invert any injective function (using the trapdoor) because it is evaluated on a non-lossy branch, whereas S_{loss} generates a lossy function having image size at most 2^{n-k}.

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

We can also construct an ABO collection for larger branch sets from one with just a binary branch set. Our construction involves some degradation in lossiness (i.e., additional leakage) because it invokes several functions on the same input. It is an interesting question whether this can be improved.

Lemma 3.2. *An $(n, n-r)$-ABO collection for branch set $B = \{0, 1\}$ implies an $(n, n-\ell \cdot r)$-ABO collection for branch set $B = \{0, 1\}^\ell$.*

Proof. We construct the claimed ABO collection as follows: S_{abo}, given the desired lossy branch $b^* \in \{0,1\}^\ell$, constructs ℓ individual functions $g^{(i)}$ each having lossy branch $b_i^* \in \{0,1\}$ (the ith bit of b^*) using the hypothesized collection, and keeps all the trapdoors. The evaluation algorithm G_{abo}, on branch $b \in \{0,1\}^\ell$ and input x, outputs b and the tuple of values $g^{(i)}(b_i, x)$. When $b \neq b^*$, the branches differ at some position $j \in [\ell]$, so x can be recovered from the corresponding value $g^{(j)}(b_j, x)$ (using the corresponding trapdoor). When $b = b^*$, then all ℓ functions are evaluated on their lossy branches, and the total number of possible outputs is at most $(2^r)^\ell = 2^{\ell \cdot r}$. □

3.4 Implications of Lossy TDFs

Here we show that lossy TDFs (having appropriate parameters) can be used for simple, black-box constructions of other important cryptographic primitives, including standard (injective) trapdoor functions, pseudorandom generators, and collision-resistant hash functions. We stress that most of the applications in this section *do not require a trapdoor*, but only indistinguishability between injective and lossy functions (the only exception is in obtaining standard trapdoor functions). It seems plausible that realizing this weaker notion of "lossy functions" could be achieved more simply or efficiently than the full definition of lossy TDFs; we leave an investigation of this question to future work.

3.4.1 Trapdoor Functions

First we show 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 3.3. *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 $\mathcal{I}(s, f_s(x))$ outputs x with nonnegligible probability over the choice of $(s,t) \leftarrow S_{\text{inj}}$, $x \leftarrow \{0,1\}^n$, 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_{inj}, 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_{loss}. 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 and the independence of x from s implies that

$$\tilde{H}_\infty(x|(s, f_s(x))) \geq H_\infty(x|s) - (n-k) = n - (n-k) = k.$$

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 of the hypothesis. □

3.4.2 Hard-Core Functions and Pseudorandom Generators

Here we show that lossy TDFs admit simple "hard-core" predicates (and more generally, multibit functions) with tight and elementary security reductions. Informally, a hard-core function for a function $f : \{0,1\}^n \to \{0,1\}^*$ is a function $h : \{0,1\}^n \to \{0,1\}^\ell$, where $h(x)$ is computationally indistinguishable from a uniform and independent string $r \leftarrow \{0,1\}^\ell$, given knowledge of $f(x)$ for $x \leftarrow \{0,1\}^n$.

Our results here can be contrasted with that of Goldreich and Levin [29], who demonstrated a "universal" hard-core predicate for every one-way function. On one hand, their result applies to *any* one-way function (or collection thereof), whereas ours relies crucially on lossy functions. On the other hand, their proof relies on a highly non-trivial security reduction whose running time depends on the success probability of the distinguisher; our proof is entirely elementary and the security reduction is tight (i.e., the running time and success probability of our distinguisher between lossy and injective functions are essentially the same as those of the distinguisher between the value of the hard-core function and a uniform string).

In the following, let $(S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ give a collection of (n,k)-lossy TDFs (in fact, we only need a collection of lossy functions; F_{ltdf}^{-1} is unnecessary). 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 $\epsilon = \mathsf{negl}(\lambda)$. Define the following random variables that are sampled by the experiments described below, which are implicitly indexed by the security parameter λ.

Variable X_0: choose $(s,t) \leftarrow S_{\text{inj}}$, $h \leftarrow \mathcal{H}$, and $x \leftarrow \{0,1\}^n$. The value of X_0 is $(s, h, F_{\text{ltdf}}(s,x), h(x))$.

Variable X_1: choose $(s,t) \leftarrow S_{\text{loss}}$, $h \leftarrow \mathcal{H}$, and $x \leftarrow \{0,1\}^n$. The value of X_1 is $(s, h, F_{\text{ltdf}}(s,x), h(x))$.

Variable X_2: choose $(s,t) \leftarrow S_{\text{loss}}$, $h \leftarrow \mathcal{H}$, $x \leftarrow \{0,1\}^n$, and $r \leftarrow \{0,1\}^\ell$. The value of X_2 is $(s, h, F_{\text{ltdf}}(s,x), r)$.

Variable X_3: choose $(s,t) \leftarrow S_{\text{inj}}$, $h \leftarrow \mathcal{H}$, $x \leftarrow \{0,1\}^n$, and $r \leftarrow \{0,1\}^\ell$. The value of X_3 is $(s, h, F_{\text{ltdf}}(s,x), r)$.

Lemma 3.4. *Let X_0, X_1, X_2, X_3 be as defined above. Then*

$$\{X_0\} \overset{c}{\approx} \{X_1\} \overset{s}{\approx} \{X_2\} \overset{c}{\approx} \{X_3\}.$$

In particular, $\mathcal{H}$ is a family of hard-core functions for the lossy collection.

Proof. The fact that $\{X_0\} \overset{c}{\approx} \{X_1\}$ follows immediately from the indistinguishability of injective and lossy functions: a PPT algorithm, given as input an s generated either by S_{inj} or S_{loss}, can sample $h \leftarrow \mathcal{H}$, $x \leftarrow \{0,1\}^n$, and compute $F_{\text{ltdf}}(s,x), h(x)$ on its own and output $(s, h, F_{\text{ltdf}}(s,x), h(x))$. Because the two distributions of s are computationally indistinguishable by hypothesis, the resulting output distributions X_0 and X_1 are likewise. A virtually identical argument shows that $\{X_2\} \overset{c}{\approx} \{X_3\}$ as well.

It remains to show that $\{X_1\} \overset{s}{\approx} \{X_2\}$. Let s be any fixed function index generated by S_{loss}. Because $f_s(\cdot) = F_{\text{ltdf}}(s,\cdot)$ has at most 2^{n-k} outputs and $x \leftarrow \{0,1\}^n$ is independent of s, by Lemma 2.1 we have

$$\tilde{H}_\infty(x|s, f_s(x)) \geq H_\infty(x|s) - (n-k) = k.$$

Therefore, by Lemma 2.2, the hypothesis that $\ell \leq k - 2\lg(1/\epsilon)$, and the definition of X_1, X_2, we have

$$\Delta(X_1, X_2) \leq \epsilon(\lambda) = \mathsf{negl}(\lambda),$$

as desired. □

A *pseudorandom generator* is a deterministic function $G : \{0,1\}^n \to \{0,1\}^{n'}$ for some $n' > n \geq 1$ such that the uniform distribution over $\{0,1\}^{n'}$ is computationally indistinguishable from $G(x)$, where $x \leftarrow \{0,1\}^n$ is chosen uniformly at random. Hard-core predicates (and hard-core functions) have played an integral role in the construction of pseudorandom generators [9, 55, 33]. In particular, Håstad *et al.* [33] constructed pseudorandom generators from any one-way function; their construction is much simpler (and the security reduction is tighter) when the one-way function is also *injective*. Their approach is first to apply the Goldreich-Levin hard-core predicate of an injective one-way function to construct an object called a *pseudoentropy generator*, which, informally, is a deterministic function G such that $G(x)$ is computationally indistinguishable from some distribution having more entropy than x. They then construct a pseudorandom generator from any pseudoentropy generator; see Sections 4.3 and 4.6 of [33] for details. We observe that because pairwise independent hash functions are hard-core for the injective functions of a lossy TDF collection, they can be used in lieu of the Goldreich-Levin predicate in the construction of [33], yielding a tight security reduction for the resulting construction of pseudoentropy generators.

3.4.3 Universal One-Way and Collision-Resistant Hashing

We now construct UOWHFs and collision-resistant hash functions from lossy TDFs. The construction is quite simple: the hash function H is defined as $H(x) := h(f(x))$, where f is (say) a lossy function, and h is selected from a *universal* family of hash functions. For an appropriate output length of the universal family, H shrinks its input, and for appropriate degrees of lossiness, finding collisions (of the appropriate type) implies the ability to distinguish injective functions from lossy ones.

The main idea behind the security proof (for both UOWHFs and CRHFs) is the following: if the function $H = h \circ f$ is constructed using an injective f, then all collisions in H must occur in the "outer" application of h. Now consider the function $H = h \circ f'$, where f' is lossy. For an appropriate level of lossiness, with overwhelming probability the function h contains *no collisions*, either on the selected target point (for UOWHFs) or over the entire image of f' (for CRHFs). 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. (Interestingly, the proof works equally well regardless of whether the "true" construction uses an injective or lossy inner function.) We now proceed more formally with the construction of CRHFs, which are themselves UOWHFs (see also the discussion following the proof of Lemma 3.5).

Assume without loss of generality that the input length $n(\lambda) = \lambda$ 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 also suffices.) 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 constant $\delta \in (0, 1-2\rho)$.[4]

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

- Generator S_{crh} chooses $(s,t) \leftarrow S_{\text{inj}}$ 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 3.5. *The algorithms (S_{crh}, F_{crh}) described above give a collection of collision-resistant hash functions from $\{0,1\}^n$ to $\{0,1\}^{\kappa n}$.*

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

Proof. Let $\mathcal{C}$ be an adversary that attempts to find collisions 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 suffices 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. □

Discussion. The crucial hypothesis in the above proof is that the residual leakage ρn of the lossy TDF collection is significantly less than $n/2$, so as to circumvent the birthday bound. For UOWHFs, the exact same proof goes through as long as the leakage is bounded by ρn for some constant $\rho < 1$, because we only need to rule out collisions for the specific input chosen by the adversary before the hash function is selected.

We also note that alternate constructions, in which s is generated by S_{loss} instead of S_{inj}, can also yield UOWHFs and CRHFs. These constructions might 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 point out again that the our construction of hash functions does not require a trapdoor, but only a collection of lossy functions.

4 Cryptosystems and Oblivious Transfer

Here we show how to construct cryptosystems enjoying various notions of security 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 sketch how the CPA-secure cryptosystem also implies oblivious transfer and multiparty computation protocols. We conclude in Section 4.3 with our CCA-secure construction and its proof of security.

[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.

4.1 CPA-Secure Construction

We first describe our basic CPA-secure cryptosystem. All of the parameters in the system depend upon the security parameter λ; for notational convenience we 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 has message space $\{0,1\}^\ell$.

- **Key generation.** $\mathcal{G}$ first generates an injective trapdoor function: $(s,t) \leftarrow S_{\text{inj}}$. 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 give a CPA-secure cryptosystem.*

Proof. Correctness of decryption is immediate from correctness of F_{ltdf}^{-1}. (If the lossy TDF collection is almost-always, then decryption may fail with only negligible probability.)

Security under chosen plaintext attack essentially follows immediately from the fact that pairwise independent hash functions are hard-core for lossy TDFs, as established by Lemma 3.4 in Section 3.4.2. We show that the view of the adversary in either of the CPA experiments (in which m_b is encrypted, for $b \in \{0,1\}$) is computationally indistinguishable from a common "hybrid" experiment, from which it follows that the two CPA experiments are themselves computationally indistinguishable.

In more detail, consider a hybrid chosen-plaintext attack experiment in which $pk = (s,h)$ is generated by choosing $(s,t) \leftarrow S_{\text{loss}}$ and $h \leftarrow \mathcal{H}$, and the ciphertext $(c_1, c_2) = (F_{\text{ltdf}}(s,x), r \oplus m_b)$ for $x \leftarrow \{0,1\}^n$ and $r \leftarrow \{0,1\}^\ell$. Note that the view of the adversary is identical for either value of b and any choice of messages m_0, m_1, because r is uniform and independent of all other variables. By Lemma 3.4, this hybrid view is computationally indistinguishable from the view in the CPA experiment when m_b is encrypted. This completes the proof. □

4.2 Interlude: Oblivious Transfer and Multiparty Computation

One interesting property of our CPA-secure scheme is that it can be used to create an oblivious transfer protocol (secure against *semi-honest*, or "honest-but-curious," adversaries) in a manner that roughly follows the approach of Even, Goldreich, and Lempel [22]. This approach relies on a CPA-secure cryptosystem that allows sampling a public key in two different but indistinguishable ways: first, in a "normal" way together with the corresponding decryption key, and second, in an "oblivious" way so that messages encrypted under

the public key remain hidden even given the random coins of the sampler. As shown in [22], 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, where 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 cryptosystem, one can sample a public key obliviously simply by generating a lossy function rather than an injective one, letting $(s, \perp) \leftarrow S_{\text{loss}}$ instead of $(s, t) \leftarrow S_{\text{inj}}$. By 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 statistically hidden. We remark that these security properties are a reversal of those obtained previously in the EGL paradigm (using, e.g., trapdoor permutations), where the receiver's security is statistical and the sender's security is computational.

Oblivious transfer protocol secure against *malicious* adversaries can be constructed using the zero-knowledge "compiler" paradigm of Goldreich, Micali, and Wigderson [30] or using a recent black-box transformation of Haitner [32], and secure multiparty computation can be obtained using the (non-black-box) compilation paradigm of Goldreich, Micali, and Wigderson [31]. However, these constructions are inefficient and primarily of theoretical interest. Recent work by Peikert, Vaikuntanathan, and Waters [42] constructs *efficient* (and "universally composable") OT protocols against malicious adversaries under a variety of assumptions, including those used in this work to instantiate lossy TDFs. We also point out that semi-honest OT protocols are implicit in the existing literature on lattice-based cryptosystems [2, 46, 47]; these cryptosystems are proved secure by showing that it is possible to sample a (malformed) public key that is indistinguishable from a valid public key, whose ciphertexts statistically hide 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 one-time 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 contains the set of signature verification keys).[6]

We require that the total residual leakage over the lossy and ABO collections is

$$r + r' = (n - k) + (n - k') \leq n - \kappa, \tag{1}$$

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

- **Key generation.** $\mathcal{G}$ generates an injective trapdoor function via $(s, t) \leftarrow S_{\text{inj}}$, an ABO trapdoor function having lossy branch 0^v via $(s', t') \leftarrow S_{\text{abo}}(0^v)$, and 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:

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

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

(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 generates a keypair for the one-time signature scheme via $(vk, sk_\sigma) \leftarrow \mathsf{Gen}$, 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

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

- **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 $x = F^{-1}_{\text{ltdf}}(t, c_1)$, and checks that $c_1 = F_{\text{ltdf}}(s, x)$ and $c_2 = G_{\text{abo}}(s', vk, x)$; 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 give a CCA2-secure cryptosystem.*

4.4 Proof of Theorem 4.2

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 $f_s(\cdot) = F_{\text{ltdf}}(s, \cdot)$ is injective (with overwhelming probability over the choice of s, in the almost-always case), therefore $F^{-1}_{\text{ltdf}}(t, c_1) = x$, where x is the randomness used in the encryption. By construction, $c_1 = F_{\text{ltdf}}(s, x)$ and $c_2 = G_{\text{abo}}(s', vk, x)$. Therefore the decryption algorithm outputs $c_3 \oplus h(x) = m \oplus h(x) \oplus h(x) = m$.

We prove CCA-security by describing a sequence of experiments $\text{Game}_1, \ldots, \text{Game}_6$, where Game_1 is the real chosen ciphertext attack experiment in which m_b is encrypted, for an arbitrary choice of $b \in \{0,1\}$. Then we show that for all $i = 1, \ldots, 5$, the adversary's views in Game_i and Game_{i+1} are indistinguishable (either computationally or statistically). Finally, it follows immediately from the definition of Game_6 that the adversary's view is identical for either value of $b \in \{0,1\}$. It follows by transitivity that the two chosen-ciphertext attack experiments for $b \in \{0,1\}$ are computationally indistinguishable, hence the cryptosystem is CCA-secure.

We now define the sequence of games we use to prove security. An experiment 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 experiment:

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

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

Before defining these algorithms for the individual experiments, we define two "global" aspects of the algorithms that remain the same in all the games. First, Setup always first chooses a one-time signature keypair $(vk^*, sk^*_\sigma) \leftarrow \mathsf{Gen}$, and then proceeds as we define below. Second, whenever $\mathsf{Challenge}(m_0, m_1)$ produces a challenge ciphertext c^* by calling $\mathcal{E}(pk, m_b)$, instead of generating a one-time signature keypair (vk, sk_σ) on its own, it uses $(vk, sk_\sigma) = (vk^*, sk^*_\sigma)$ as generated in the first step of Setup. We stress that Challenge operates this way in *all* the games we define.

When making these changes to the real CCA game (Game_1), the view of the adversary remains identical, because Challenge is invoked exactly once. We make these changes merely for the convenience of having vk^* defined throughout both query phases, which aids the analysis.

Game_1: Algorithms Setup_1, $\mathsf{Decrypt}_1$, and $\mathsf{Challenge}_1$ are identical to those in the CCA2 experiment described in Section 2.3, with the above-noted changes. That is, Setup_1 calls $(pk, sk) \leftarrow \mathcal{G}$ 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, $\mathsf{Setup}_2 = \mathsf{Setup}_1$ and $\mathsf{Challenge}_2 = \mathsf{Challenge}_1$. The only change is in $\mathsf{Decrypt}_2$, 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, $\mathsf{Decrypt}_3 = \mathsf{Decrypt}_2$ and $\mathsf{Challenge}_3 = \mathsf{Challenge}_2$. The only change is in Setup_3, in which the ABO function is chosen to have a lossy branch $b^* = vk^*$ rather than $b^* = 0^v$. Formally, in $\mathcal{G}$ we replace $(s', t') \leftarrow S_{\text{abo}}(0^v)$ with $(s', t') \leftarrow S_{\text{abo}}(vk^*)$.

Note that $\mathsf{Decrypt}_3$ still decrypts using the injective function trapdoor t, and that the ABO function trapdoor t' is never used in this experiment.

Game_4: In this game, $\mathsf{Setup}_4 = \mathsf{Setup}_3$ and $\mathsf{Challenge}_4 = \mathsf{Challenge}_3$. The only change is in $\mathsf{Decrypt}_4$, in which decryption is now done using the ABO trapdoor t'. Formally, in $\mathcal{D}$ we replace $x = F^{-1}_{\text{ltdf}}(t, c_1)$ with $x = G^{-1}_{\text{abo}}(t', vk, c_2)$.

Note that $\mathsf{Decrypt}_4$ still first rejects if $vk = vk^*$ (as in $\mathsf{Decrypt}_2$), and performs all the consistency checks of $\mathcal{D}$. Also note that the injective function trapdoor t is never used in this experiment.

Game_5: In this game, $\mathsf{Decrypt}_5 = \mathsf{Decrypt}_4$ and $\mathsf{Challenge}_5 = \mathsf{Challenge}_4$. The only change is in Setup_5, in which we replace the injective function with a lossy one. Formally, in $\mathcal{G}$ we replace $(s, t) \leftarrow S_{\text{inj}}$ with $(s, \perp) \leftarrow S_{\text{loss}}$.

Game_6: In this game, $\mathsf{Setup}_6 = \mathsf{Setup}_5$ and $\mathsf{Decrypt}_6 = \mathsf{Decrypt}_5$. The only change is in $\mathsf{Challenge}_6$, where the c_3 component of its output ciphertext c^* is replaced by a uniform and independent value $c_3 = r \leftarrow \{0,1\}^\ell$. Formally, in the call to $\mathcal{E}$ by $\mathsf{Challenge}_6$, we replace $c_3 = m_b \oplus h(x)$ with $c_3 = r \leftarrow \{0,1\}^\ell$.

First observe that, as desired, the adversary's view in Game_6 is identical for either choice of $b \in \{0,1\}$, because b is never used in the experiment. We now state and prove a sequence of indistinguishability results that establish the main theorem.

Claim 4.3. *The adversary's views in Game$_1$ and Game$_2$ are computationally indistinguishable, assuming the strong one-time existential unforgeability of the signature scheme.*

Proof. We begin by observing that the views in Game$_1$ and Game$_2$ are identical unless a certain 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 Gen, it emulates Setup by letting $vk^* = vk$ and letting $(pk, sk) \leftarrow \mathcal{G}$, 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 $m_0, m_1 \in \{0,1\}^\ell$, $\mathcal{S}$ 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 $\mathcal{S}$'s signing oracle on the message (c_1^*, c_2^*, c_3^*), instead of running Sign.

It is clear by construction 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 (before $\mathcal{A}$ is challenged on c^*), then $\mathcal{S}$ outputs a valid signature without making any queries, which is a forgery. If F happens during the second query phase (after $\mathcal{A}$ receives c^*) via 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 $((c_1, c_2, c_3), \sigma)$ differs from 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 event F happens with negligible probability, as desired. □

Claim 4.4. *The adversary's views in Game$_2$ and Game$_3$ are computationally indistinguishable, assuming the hidden lossy branch property of the ABO TDF collection.*

Proof. We show that the adversary's views in Game$_2$ and Game$_3$, *conditioned on any fixed value of the signature keypair* $(vk^*, sk_\sigma^*) \leftarrow \mathsf{Gen}$, are computationally indistinguishable. Because the distribution of (vk^*, sk_σ^*) is identical in both Game$_2$ and Game$_3$, it follows by averaging over the choice of (vk^*, sk^*) that the experiments in their entirety are indistinguishable.

Fix any (vk^*, sk_σ^*). We define a PPT simulator $\mathcal{S}$ whose input is (vk^*, sk_σ^*) and an ABO function index s' which was generated as either $(s', t') \leftarrow S_{\text{abo}}(0^v)$ or $(s', t') \leftarrow S_{\text{abo}}(vk^*)$. Because the two types of inputs are computationally indistinguishable, the output of $\mathcal{S}$ in the two cases will be as well. We construct $\mathcal{S}$ so that its output is identical to Game$_2$ or Game$_3$, respectively.

$\mathcal{S}$ operates by implementing Setup, $\mathsf{Decrypt}$, and $\mathsf{Challenge}$ to create a view. To implement Setup, it executes the remainder of $\mathcal{G}$, i.e., it lets $(s, t) \leftarrow S_{\text{inj}}$ and $h \leftarrow \mathcal{H}$, and outputs $pk = (s, s', h)$. It implements $\mathsf{Decrypt}$ and $\mathsf{Challenge}$ exactly as in Game$_2$ and Game$_3$ (which are identical in these respects). Note that $\mathcal{S}$ can do so because it is given the signing key sk^* as input and it generates the injective function trapdoor t itself.

One can verify by construction that the view generated by $\mathcal{S}$ is exactly Game$_2$ when s' is generated by $S_{\text{abo}}(0^v)$, and is exactly Game$_3$ when s' is generated by $S_{\text{abo}}(vk^*)$, thus the proof is complete. □

Claim 4.5. *The adversary's views in Game$_3$ and Game$_4$ are identical (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 show that Decrypt is equivalent in the two games (with overwhelming probability, if the lossy or ABO collections are almost-always).

First recall that if the trapdoor systems are almost-always, then 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 need only consider the case in which such x exists. It suffices to show that this x is unique, and that both implementations of Decrypt find it.

In both games, (s, t) is generated by S_{inj} and (s', t') is generated by $S_{\text{abo}}(vk^*)$. Therefore $f_s(\cdot)$ and $g_{s',vk}(\cdot)$ are both injective (in the latter case, because $vk \neq vk^*$). Therefore there is a unique x such that $(c_1, c_2) = (f_s(x), g_{s',vk}(x))$. Decrypt_3 finds that x by computing $F^{-1}_{\text{ltdf}}(t, c_1)$, while Decrypt_4 finds it by computing $G^{-1}_{\text{abo}}(t', c_2)$, and the proof is complete. □

Claim 4.6. *The adversary's views in Game_4 and Game_5 are computationally indistinguishable, assuming the indistinguishability of injective and lossy functions of the lossy TDF collection.*

Proof. We prove this claim by describing a PPT simulator algorithm $\mathcal{S}$ that, on input s, simulates Game_4 perfectly if s was generated by S_{inj}, and that simulates Game_5 perfectly if s was generated by S_{loss}. By the indistinguishability of injective and lossy functions, the claim follows.

The simulator $\mathcal{S}(s)$ operates by implementing Setup, Decrypt, and Challenge. It implements Setup is implemented in a manner similar to Game_4 by choosing $(vk^*, sk^*_\sigma) \leftarrow \text{Gen}$, $(s', t') \leftarrow S_{\text{abo}}(vk^*)$, and $h \leftarrow \mathcal{H}$, and outputting a public key $pk = (s, s', h)$. 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).

Decrypt and Challenge are implemented just as in Game_4 and Game_5, which are identical in these respects. Note that the only secret information Decrypt needs to operate is t', which the simulator generated itself. By construction, $\mathcal{S}$ therefore perfectly simulates Game_4 or Game_5, depending on whether s is the index of an injective or lossy function (respectively), as desired. □

Claim 4.7. *The adversary's views in Game_5 and Game_6 are statistically indistinguishable.*

Proof. Fix all the random coins (including the adversary's) in Game_5 and Game_6, except for the choice of the hash function h and the randomness x used by Challenge when producing the ciphertext c^*. We show that for every such value of the fixed coins, the views in Game_5 and Game_6 are statistically indistinguishable; the claim follows by averaging over the choice of the random coins.

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. (When the lossy and/or ABO collections are almost-always, then the claim holds with overwhelming probability over the choice of s, s'.) Therefore the random variable $(c^*_1, c^*_2) = (f_s(x), g_{s',vk^*}(x))$ can take at most $2^{r+r'} \leq 2^{n-\kappa}$ values by our hypothesis in (1).

By Lemma 2.1 and the independence of x from s, s', we have

$$\tilde{H}_\infty(x | c^*_1, c^*_2, s, s') \geq H_\infty(x | s, s') - (n - \kappa) = n - (n - \kappa) = \kappa.$$

Now by the hypothesis that $\ell \leq \kappa - 2\lg(1/\epsilon)$ and Lemma 2.2, we have

$$\Delta((c^*_1, c^*_2, h, h(x)),\ (c^*_1, c^*_2, h, r')) \leq \epsilon = \mathsf{negl}(\lambda),$$

where $r' \leftarrow \{0,1\}^\ell$ is uniform and independent of all other variables. In Game_5, we have $c_3 = h(x) \oplus m_b$, whereas in Game_6, we have $c_3 = r \leftarrow \{0,1\}^\ell$, which is identically distributed to $r' \oplus m_b$ (because r' is uniform and independent). Therefore the two games are statistically indistinguishable, and this completes the proof. □

4.5 Discussion and Alternate Constructions

We stress that in all the games except the last (in which m_0 and m_1 are never used), the challenge ciphertext c^* is created in the same way by "honestly" running the encryption algorithm $\mathcal{E}(pk, m_b)$. The only difference between games is instead in how the public key is formed and how decryption queries are answered. This is in contrast to prior constructions, in which the hybrid experiments always involve valid public keys, but the simulator does not know the underlying 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.

The use of a one-time strongly unforgeable signature scheme for full CCA2 security (and in particular, non-malleability) dates back to the work of Dolev, Dwork, and Naor [20], and is technically similar to its use in the work of Canetti, Halevi, and Katz [15] in their construction of CCA2-secure encryption from identity-based cryptosystems. We point out that for weaker CCA1 ("lunchtime") security, the one-time signature in our encryption algorithm is not needed, and vk can simply be replaced by a uniformly random value in $\{0,1\}^v$ that specifies the branch on which the ABO function is to be evaluated. The proof of security remains essentially the same, where Game_1 and Game_2 now become statistically indistinguishable because the value of vk^* is statistically hidden (it is uniform and independent of the adversary's view) during the query phase, before the challenge ciphertext c^* is produced.

Finally, we point out that the choice of the hash function $h \leftarrow \mathcal{H}$ can be deferred from the key generation algorithm to the encryption algorithm, with a fresh choice of h chosen for (and included in) each ciphertext, with no change in the proof. (The same holds for our basic CPA-secure construction.) Because in most systems it is typical to encrypt many messages under a single public key, this alternate construction is less efficient in terms of communication (but it may have other applications).

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 conjectured 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 $\mathbb{G} = (p, G, g) \leftarrow \mathcal{G}$ 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 represent the encrypted matrix somewhat more compactly via matrices $\mathbf{C}_1$ and $\mathbf{C}_2$, where

$$\mathbf{C}_1 = \begin{pmatrix} g^{r_1} \\ \vdots \\ g^{r_n} \end{pmatrix} \qquad \mathbf{C}_2 = \begin{pmatrix} h_1^{r_1} \cdot g^{m_{1,1}} & h_2^{r_1} \cdot g^{m_{1,2}} & \cdots & h_n^{r_1} \cdot g^{m_{1,n}} \\ \vdots & & \ddots & \vdots \\ h_1^{r_n} \cdot g^{m_{n,1}} & h_2^{r_n} \cdot g^{m_{n,2}} & \cdots & h_n^{r_n} \cdot g^{m_{n,n}} \end{pmatrix}$$

We point out that if we ignore the message terms $g^{m_{i,j}}$, the matrix $\mathbf{C}_2$ (consisting solely of the $h_j^{r_i}$ terms) is a *synthesizer*, as defined by Naor and Reingold [37]. Specifically, the n^2 values $h_j^{r_i}$ are indistinguishable from n^2 uniform and independent elements of G, under the DDH assumption. This is the essential reason why the matrix encryption remains semantically secure.

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. (A tighter security reduction is also possible using the random self-reducibility of DDH; see, e.g., [37]. We give a looser reduction for self-containment and simplicity.)

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. □

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_{inj} first selects $\mathbb{G} = (p, G, g) \leftarrow \mathcal{G}$. 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_{loss} likewise selects $\mathbb{G} \leftarrow \mathcal{G}$. 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^{-1}_{ltdf} 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 consist of $\lceil n/\alpha \rceil + 1$ group elements. However, there is a trade-off in the inversion time, as the ElGamal decryption algorithm needs to enumerate over the 2^α possible values. Therefore, this generalization is polynomial-time only for small values of α, i.e., $\alpha = O(\log \lambda)$.

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 $\lg p$ bits. For large enough values of n, we can use the generic transformation (see Section 3.3) 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 $\mathcal{G}$ (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 $S_{\text{abo}}(b^* \in B)$ first selects $\mathbb{G} = (p, G, g) \leftarrow \mathcal{G}$. 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

Here we construct lossy and all-but-one TDFs based on the hardness of the *learning with errors* (LWE) problem, as defined by Regev [47]. The LWE problem is a generalization to larger moduli of the *learning parity with noise* problem (see, e.g., [7]). It can be viewed as an (average-case) "bounded-distance decoding" problem on a certain family of random lattices under a natural error distribution, and is conjectured (along with learning parity with noise) to be hard on the average. Very interestingly, Regev showed that LWE is indeed hard on the average if standard lattice problems (like approximating the shortest vector problem) are hard in the *worst case* for *quantum* algorithms [47]. No efficient (or even subexponential-time) quantum algorithms are known for the associated worst-case lattice problems, despite significant research efforts. Our results rely solely on the conjectured average-case hardness of LWE, and inherit Regev's worst-case connection as a "black box." We stress that although the underlying worst-case lattice assumption relates to quantum algorithms, the LWE problem and our constructions based on it are entirely classical.

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

By calibrating the parameters appropriately, we can obtain lossy TDFs that lose any desired *constant fraction* (e.g., 99%) of the input. Unfortunately, this is not strong enough to obtain all-but-one TDFs having more than a constant number of branches via the parallel black-box construction of Section 3.3, because the residual leakage of the parallel construction is multiplied by the the logarithm of the number of branches. Fortunately, by generalizing our lossy TDF construction and using some additional ideas, we are able to construct ABO TDFs directly from the LWE assumption (see Section 6.4 for details).

6.1 Background

We start by introducing the notation and computational problems that are relevant to this section, for the most part following [47].

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.

Lattices. A (full-rank) d-dimensional *lattice* $\Lambda \subset \mathbb{R}^d$ can be defined as the set of all integer linear combinations of some set of d linearly independent *basis* vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_d\} \subset \mathbb{R}^d$:

$$\Lambda = \left\{ \sum_{i \in [d]} c_i \mathbf{b}_i \ : \ c_1, \ldots, c_d \in \mathbb{Z}^d \right\}.$$

A given lattice has infinitely many bases (when $d \geq 2$), which are related to each other by unimodular transformations over the integers. Computationally, a lattice is typically represented to an algorithm by some choice of basis.

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 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 sample efficiently 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 $\mathbb{Z}_q$ given by $\lfloor q \cdot \phi \rceil \bmod q$.

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 errors (LWE). For an integer $q = q(d)$ and a distribution χ on $\mathbb{Z}_q$, the goal of the *learning with errors* problem $\mathsf{LWE}_{q,\chi}$ is to distinguish (with nonnegligible probability) between the distribution $A_{\mathbf{z},\chi}$ for some secret uniformly random $\mathbf{z} \leftarrow \mathbb{Z}_q^d$, and the uniform distribution on $\mathbb{Z}_q^d \times \mathbb{Z}_q$, given access to any $\mathrm{poly}(d)$ number of samples from the unknown distribution. The assumption that "LWE is hard" can be restated more succinctly as "$A_{\mathbf{z},\chi}$ is pseudorandom" (for $\mathbf{z} \leftarrow \mathbb{Z}_q^d$ chosen uniformly at random).

The conjectured hardness of LWE is parametrized 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 showed that for certain normal error distributions, LWE is as hard as several standard *worst-case* lattice problems, *for quantum algorithms*. We state a version of the main theorem here:

Proposition 6.1 ([47])**.** *Let $\alpha = \alpha(d) \in (0, 1)$ and let $q = q(d)$ be a* prime *such that $\alpha \cdot q > 2\sqrt{d}$. There is a* quantum *polynomial-time reduction from solving either of the two lattice problems in the* worst-case *to solving $\mathsf{LWE}_{q,\bar{\Psi}_\alpha}$ on the average:*

- *SIVP: In any lattice of dimension d (represented by a basis), find a set of d linearly independent lattice vectors, the longest of which has length (in the Euclidean ℓ_2 norm) within at most an $\tilde{O}(d/\alpha)$ factor of optimal.*
- *GapSVP: In any lattice of dimension d (represented by an arbitrary basis), approximate the length (in the Euclidean ℓ_2 norm) of a shortest nonzero lattice vector to within a $\tilde{O}(d/\alpha)$ factor.*

The SIVP and GapSVP problems appear to be quite hard in the worst case: to obtain some $\mathrm{poly}(d)$ approximation factor, known classical (and quantum) algorithms require time and space that are exponential in d [3]; known polynomial-time algorithms obtain approximation factors that are only slightly subexponential in d [36, 51]. We add that Proposition 6.1 was recently strengthened by Peikert [41] to apply to the SIVP and GapSVP problems in any ℓ_p norm, $2 < p \leq \infty$, for essentially the same $\tilde{O}(d/\alpha)$ approximation factors.

We first define our lossy and ABO functions in terms of the LWE problem, without explicitly taking into account the connection to lattices (or the hypotheses on the parameters required by Proposition 6.1). Then in Section 6.5, we instantiate the parameters appropriately, invoking Proposition 6.1 to obtain a quantum worst-case hardness guarantee.

6.2 Basic Tools

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 is sufficient for our purposes in constructing lossy TDFs. This basic cryptosystem is similar to, but somewhat simpler than, Regev's *public key* cryptosystem [47] and a multibit variant [35].

The message space of our cryptosystem is $\mathbb{Z}_p$ for some $p \geq 2$. For every message $m \in \mathbb{Z}_p$, define the "offset" 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 determined by m), because it is 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 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.

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 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 $u_{i,j} = \lfloor qc_{m_{i,j}} \rceil - qc_{m_{i,j}} \in [-1/2, 1/2]$.

  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 it is determined by $\mathbf{M}$.

  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 $c'_{i,j} = \langle \mathbf{a}_i, \mathbf{z}_j \rangle + qc_{m_{i,j}} + u_{i,j} + e_{i,j}$.

- **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 $h, w = \text{poly}(d)$, the matrix encryption scheme described above produces indistinguishable ciphertexts under the assumption that $\mathsf{LWE}_{q,\chi}$ is hard.*

Proof. It is most convenient 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 $\mathcal{O}$ produces 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 $c_{i,k} = b_i + \lfloor qc_{m_{i,k}} \rceil$. 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 $c'_{i,j} = \langle \mathbf{a}_i, \mathbf{z}_j \rangle + \lfloor qc_{m_{i,j}} \rceil + e_{i,j}$. 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. □

We now show a technical lemma that is 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 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, $e_i = \lfloor qs_i \rceil \bmod q$ 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,

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

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 involves computing an encrypted linear product $\mathbf{xM}$, and in the lossy case we 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 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 $\mathbf{xI} = \mathbf{x}$,

and each output ciphertext would correspond to $\log p$ bits of the input. 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 are values in $\mathbb{Z}_p$ that can be as large as p. Therefore, in the lossy case, the errors term in each output ciphertext might *leak* $\lg p$ bits (or more) due to the amplified error. Therefore, this simple construction does not seem to work.

Instead, in our construction the *output* uses the entire message space $\mathbb{Z}_p$, but the *input* is still interpreted in binary. This lets us recover the entire input from a relatively small number of output ciphertexts, but also ensures that the error terms are not amplified too much by the homomorphic operations. Our method uses a special (nonsquare) matrix instead of the identity matrix. 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 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 importantly, by our definition of $\mathbf{B}$, we have $\mathbf{xB} = \mathbf{vI} = \mathbf{v}$.

Our injective trapdoor function is described by a matrix encryption of $\mathbf{B}$. Evaluating the function on $\mathbf{x} \in \{0,1\}^n$ corresponds to computing an encrypted product $\mathbf{xB} = \mathbf{v}$. This permits recovery of the entire input by decrypting $\mathbf{v}$ and producing 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 accumulated error 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_{inj} 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_{loss} 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 (for any fixed $\mathbf{A}$ and $\mathbf{x}$). 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). □

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 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 $\mathbf{I}$ does 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 simultaneously have full row rank, we use matrices having a few more columns. This increases the leakage of the lossy branch of the function (because the output consists of more ciphertexts, which each have 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 crucially uses 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 depends on the other parameters and the desired lossiness. Let $\mathcal{H}$ denote a family of pairwise independent functions from $B = \mathbb{Z}_p^t$ to $\mathbb{Z}_p^{m \times w}$ (note that these functions actually expand a branch value into a large matrix in $\mathbb{Z}_p^{m \times w}$).

- *Sampling an ABO function.* The function generator $S_{\text{abo}}(b^* \in B)$ 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_{abo}^{-1} 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) 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 for all $\mathbf{H}$ 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. □

6.5 Worst-Case Connection

We now relate the security of our constructions to the conjectured worst-case (quantum) hardness of lattice problems. 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.

[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 $\delta > 0$.

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 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

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

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 to 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, to Cynthia Dwork and Salil Vadhan for insightful comments, and to the anonymous reviewers for many helpful comments on the presentation.

References

[1] Miklós Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

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

[3] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.

[4] Mihir Bellare, Alexandra Boldyreva, K. Kurosawa, and Jessica Staddon. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Transactions on Information Theory*, 53(11):3927–3943, 2007.

[5] 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.

[6] 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.

[7] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[8] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991. Preliminary version in STOC 1998.

[9] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[10] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *CRYPTO*, pages 335–359, 2008.

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

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

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

[14] 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.

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

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

[17] 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.

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

[19] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008. Preliminary version in EUROCRYPT 2004.

[20] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. Preliminary version in STOC 1991.

[21] 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/`.

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

[23] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999. Preliminary version in FOCS 1990.

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

[25] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[26] 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.

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

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

[29] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989.

[30] 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.

[31] 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.

[32] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In *TCC*, pages 412–426, 2008.

[33] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

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

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

[36] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.

[37] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.*, 58(2):336–375, 1999.

[38] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.

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

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

[41] Chris Peikert. Limits on the hardness of lattice problems in ℓ_p norms. *Computational Complexity*, 17(2):300–351, May 2008. Preliminary version in CCC 2007.

[42] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[43] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.

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

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

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

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

[48] 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.

[49] Alon Rosen and Gil Segev. Efficient lossy trapdoor functions based on the composite residuosity assumption. Cryptology ePrint Archive, Report 2008/134, 2008. `http://eprint.iacr.org/`.

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

[51] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.

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

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

[54] 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.

[55] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91, 1982.